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強化学習

小林　高彰

システム情報工学研究科

筑波大学

2015年　3月

目 次

図 目 次

第1章

序論

1.1　はじめに

今日，我々の身の回りには単純なものから複雑なものまで，さまざまな工学的システムが満ちあふれている．特に近年では，ペットロボットやロボット掃除機のように，複雑で目的も多様なシステムが増えており，これらの制御系をいかに設計し調整していくかは重要な問題である．

従来のように，使用環境や目的があらかじめ分かっているシステムであれば，設計者が制御系の設計・調整を行うことができるが，使用環境や目的があらかじめ想定できないシステムにおいては，人間が制御系を作りこむことには限界がある．ロボット掃除機を例に取れば，部屋の広さや家具の配置，段差の有無，人がどのくらいの頻度で出歩くのかといった環境の多様さが制御系の設計・調整を困難にする．

一方で，我々人間をはじめとする動物は，膨大な情報にあふれる現実世界において，経験したことのない状況であってもほぼ常に適切な行動を取ることができる．これは，あらかじめ決められた通りに振る舞うのではなく，経験や学習を通じて得た知識を基に，柔軟に環境に適応しているからだと考えられる．

このように，システムが複雑化・多目的化するなかで，システム自体が自律的かつ適応的に制御則を学習していくような手法が，今後ますます重要になってくると考えられるのである．

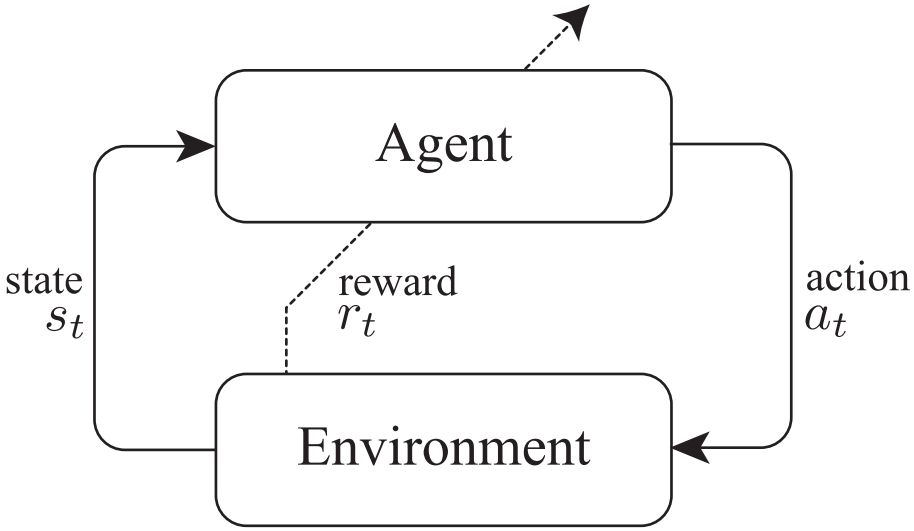


図 1.1: 強化学習における信号の流れ

1.2 研究の背景

1.2.1 強化学習

前述のような，自律的かつ適応的な制御手法の一つに強化学習 [1] がある．強化学習は，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．

強化学習における処理の流れを図 1.1 に示す．ある時刻 t において，エージェントは環境の状態 s_t を観測し，なんらかの行動選択方策に基づいて行動 a_t を選択する．選択した行動 a_t に応じて，環境の状態が s_{t+1} に遷移するとともに，報酬 r_{t+1} がエージェントに与えられる．エージェントの目的は，得られる報酬の総量が最大となるような方策を獲得することである．

強化学習にはさまざまな手法が存在するが，行動選択に用いられる 2 つの関数 : 方策（Actor）と価値関数（Critic）を修正するかによって，次の 3 つの手法に大

別することができる [2].

Critic-only の手法

価値関数を修正することで，選択行動を改善していく手法であり，方策の直接的な更新は行わない．そのため，方策には ε-Greedy 方策や Softmax 方策など，価値の大小関係にしたがって行動を選択するものが用いられる．代表的なアルゴリズムとして，Q 学習（方策オフ型 TD 学習） [3,4] や SARSA（方策オン型 TD 学習） [5] などがある．

Critic-only の手法はアルゴリズムが単純であり，数理的な解析が容易であったことから，比較的早くから研究が進められてきた．しかし，行動が連続な場合，価値関数の更新や行動選択に膨大な計算コストが必要になるという問題がある（詳しくは **1.2.2** で述べる）．

Actor-only の手法

方策を直接（価値関数を用いずに）更新して選択行動を改善する手法で，一般的に勾配法によって方策パラメータの更新を行う．具体的なアルゴリズムとしては，Williams の REINFORCE [6] や SRV [7]，確率的傾斜法 [8] などがある．

この手法には，方策に連続な確率分布（例えば正規分布）を用いることで，連続な行動を簡単かつ比較的小さな計算コストで扱えるという利点がある．しかし，方策の改善に用いる推定勾配方向の分散が大きいため，価値関数を用いた手法と比べて，一般に学習が遅いことが知られている [2,9].

Actor-Critic 手法

前述の 2 つの手法を組み合わせたもので，方策の更新に価値関数の値を利用する方法である（価値関数の更新は，Critic-only の手法と同様に行う）．この手法に関する最初期の研究は，Willen [10] によってなされ，その後 Barto ら [11] によって進められた．その後も，方策勾配に自然勾配法を用いる Natural Actor-Critic(NAC) [12] や木村の手法 [13]，NAC を方策オフ型に拡張した off-NAC [14] など様々なアルゴリズムが提案されている．これらのアル

ゴリズムの多くは，Sutton ら [9] が証明した方策勾配および局所解への収束性についての定理に基づいている．

Actor-Critic 手法の利点は，まず Actor-only の手法と同様に連続行動を扱いやすい点にある．加えて，方策の学習に価値関数を利用することで，推定勾配方向の分散が Actor-only の手法よりも小さくなり，学習が速く進むことが知られている．しかし，方策と価値関数の学習がお互いに影響しあうため，一般に方策の学習が価値関数の学習よりもゆっくりと進むようにパラメータを調整する必要がある．このため，Critic-only の手法と比べると学習が遅くなる可能性があり，さらに調整すべきパラメータも多くなるという問題がある．

これまで，実問題のように状態行動空間が連続な問題に強化学習を適用した例では，多くの場合 Actor-Critic 手法が用いられてきた．しかし，これは Critic-only の手法が Actor-Critic 手法よりも劣るということではなく，Critic-only の手法において連続な行動を扱う現実的な手法がこれまでなかったことが原因だと考えられる．上述のように，Critic-only の手法はアルゴリズムが単純であり，Actor と Critic が相互に影響しあう Actor-Critic 手法よりも学習が速い可能性がある．

実際に Actor-Critic 手法と Critic-only の手法を比較した例は少ないが，木村の研究 [13] では，代表的な Critic-only の手法である Q 学習と Actor-Critic 手法（natural gradient actor-critic）の比較を行っており，一部の実験条件では Q 学習の方が優れた行動選択方策を学習することが示されている．

したがって，Critic-only の手法において，連続な行動を扱う現実的な手法を開発することは，強化学習の適用範囲を広げることにつながると考えられる．Critic-only の手法には，さまざまなアルゴリズムが存在するが，本研究では最も代表的なアルゴリズムの一つである Q 学習を扱う（ただし，本研究で提案する手法は Q 学習以外のアルゴリズムにも容易に適用することができる）．

1.2.2 Q学習

Q学習は強化学習アルゴリズムの一つで，エージェントは行動価値（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重み付き期待値．以下ではQ値と呼ぶ）に基づいて行動を選択する．また，状態・行動からQ値への写像（行動価値関数）を式 (1.1) にしたがって逐次的に更新することで選択行動の改善を行う．

$$Q(\boldsymbol{s}_t, a_t) \leftarrow (1-\alpha)Q(\boldsymbol{s}_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a' \in \mathcal{A}(\boldsymbol{s}_{t+1})} Q(\boldsymbol{s}_{t+1}, a') \right] \tag{1.1}$$

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，およ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すべての行動の集合，$\alpha \in [0,1]$ は学習率，$\gamma \in [0,1)$ は割引率と呼ばれるパラメータである．

Q学習のアルゴリズムは，状態行動空間が離散的な場合（厳密には環境がエルゴード性を有する離散有限マルコフ決定過程である場合），全ての状態行動を十分な回数経験すること，および学習率 α に関する条件が満たされることで，確率1で最適な行動価値関数に収束することが証明されている [4]．

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，解決すべきいくつかの問題がある．まず，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およ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・行動についてのQ値を近似すること，つまり行動価値関数の近似が必要になると考えられる．

また，式 (1.1) からわかるように，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すべての行動についてQ値を求める必要がある．一般的な価値

状態行動に対する汎化：	ある状態・行動についての経験から，似た状態・行動に対して汎化（Q 値の近似）が生じること．
行動選択：	行動選択が容易であること．特に，Q 値が最大である行動を少ない計算コストで求められること．
状態・行動評価：	各状態における Q 値の計算が容易であること．また，更新式 (1.1) 中で用いられる，その状態における最大の Q 値が容易に求められること．
モデルフリー：	環境のモデルや価値関数の形状といった事前知識を必要としないこと．

図 1.2: 連続状態行動空間における Q 学習で必要となる性質 (I)
（Gaskett *et al.* [15] より改変）

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すべ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 学習を用いるうえで大きな制約となる．

1.2.3 連続な状態行動空間における Q 学習

前述の問題を踏まえ，Gaskett ら [15] は，連続な状態行動空間における Q 学習を実現する上で，必要となる 8 つの性質を指摘した．図 1.2 は，その中で特に重要と考えられる性質を整理したものである．

関数表現能力：	高い関数表現能力を持つこと．また，連続関数だけでなく不連続な関数も表現できることが望ましい．
冗長入力に対するロバスト性：	冗長な状態変数を含む環境でも，学習性能が大きく悪化しないこと．
高次元空間での計算コスト：	高次元空間における計算コスト（次元数に対する計算コストのオーダー）が小さいこと．
オンライン学習：	オンライン学習が可能な手法であること．

図 1.3: 連続状態行動空間における Q 学習で必要となる性質 (II)

状態行動に対する汎化は，状態・行動数の増加による学習効率の悪化を抑えるために必要な性質であり，関数近似の定義そのものである．また，**行動選択**と**状態・行動評価**は，どちらも行動数の増加による計算コストの増加を抑えるために必要な性質である．

一方で**モデルフリー**であることは，連続状態行動空間における Q 学習を実現する上で必須の性質とはいえない．しかし，環境のモデル等が未知の問題であっても適切な制御を実現できることは強化学習の利点の一つであるから，モデルフリーであることは実用上，重要な性質である．

これらの性質に加え，筆者は図 1.3 に示す性質も重要であると考える．

Critic-only の手法では，価値関数の近似精度が行動選択の精度に直結する．一方で，近似しようとする価値関数の形状・特徴は事前には分からないので，できるだけ高い**関数表現能力**をもつ手法が望ましい．また，強化学習問題では報酬や状態遷移がしばしば不連続性を有する．そのため，そこから導かれる価値関数も不連続な境界を含むと考えられる（例えば，10 ステップ後に報酬が得られるゴール状態に到達できる状態 $\boldsymbol{s}$ と，値は $\boldsymbol{s}$ に非常に近いが，ゴール状態まで 100 ステップかかるような状態 $\boldsymbol{s}'$ とでは，対応する Q 値は大きく異る）．したがって，関数

近似器は不連続な関数も表現できることが望ましい.

前述のモデルフリーであることに関連するが，一般に制御に必要かつ十分な状態変数は未知である．そのため，状態空間はある程度冗長に設計せざるを得ない．このことは状態空間のさらなる高次元化をまねくが，冗長な変数に関して強力な汎化が生じるような手法であれば，高次元化にともなう学習効率の低下を抑えることが期待できる．したがって，**冗長入力に対するロバスト性**を持つ手法が望ましい．同様の理由から，**高次元空間での計算コスト**の増加が小さい（オーダーが小さい）ことも重要である．

また，Q 学習を始めとする TD 学習では，毎ステップ価値関数を更新するため，常に最新の価値関数の推定値が必要となる．したがって，関数近似器は**オンライン学習**が可能でなければならない．

しかし，強化学習や機械学習の分野で一般的に用いられる関数近似手法の中で，図 1.2，1.3 に示した性質をすべて満たすものはこれまでになかった．以下では，代表的な関数近似手法について，その問題点を述べる（これ以外の個別の手法については **6.2** 節において検討する）.

まず，行動価値関数がどのような形をしているか事前にはわからないので，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 [16,17] や，カーネル関数を用いる手法 [18] 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（multilayer perceptron, MLP）がある [19, 20] 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 [21]．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 [22] 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）[1,23] 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ず，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ができず，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．

1.2.4 選択的不感化ニューラルネットによる関数近似

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．そのような関数近似手法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 [24].

SDNN は，層状ニューラルネットの一種である並列パーセプトロン [25] の入力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．これまでの研究から，SDNN は関数近似器として高い表現能力と汎化能力を兼ね備え，冗長変数にも高いロバスト性をもつなど，連続状態行動空間における Q 学習に適した数々の性質（詳細は **6.1** を参照）を持つことが示されている [24, 26, 27]

新保ら [28] は，SDNN を用いて行動価値関数を行動ごとに近似する方法（詳細は **2.1** を参照）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．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ており，行動ごとに別々の近似器（SDNN）を用意する必要がある．そのため行動次元が連続な場合，計算コストが大きく増加する．また，行動に関する汎化が起こらないため，学習効率も低下する．彼らの実験では，行動の種類を 2 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，新保らが提案した SDNN による価値関数近似器を改良し，状態行動空間がともに連続な環境に Q 学習を適用する実用的な価値関数近似手法

を開発することを目的とする．

1.3　本論文の構成

本論文の第 2 章以降の構成は以下のとおりである．

第 2 章「SDNN を用いた連続状態行動空間における価値関数近似」では，まず，SDNN を用いた一般的な関数近似について説明し，その後，本研究で提案する価値関数近似器の構成と学習法について説明する．また，提案した関数近似器が行動数の増加に伴う計算コストの増加や学習効率の低下を，どのようにして抑えているのかについても合わせて説明する．

第 3 章「離散行動と連続行動の比較」では，第 2 章で提案した手法が，連続な状態行動空間をもつ環境において，適切な行動選択を学習できるかについてシミュレーション実験により検証する．実験には，中間的な行動が重要となるようにゴール条件を修正したアクロボットの振り上げ課題を用い，従来手法との比較を行う．

実問題では，どのセンサ入力が学習に重要であるかが分からない場合が多い．そのため，状態空間は冗長な変数を含み，結果として学習効率が低下する可能性がある．そこで，第 4 章「冗長な状態変数を含む環境での学習」では，アクロボットの振り上げ課題において学習には直接必要ではないが，意味のある値を状態変数に追加した場合において，冗長入力に対する提案手法のロバスト性を検証する．また，入力次元の増加に対する計算コストの増加についても，従来手法との比較を行う．

細やかな制御を行うには，行動を連続化する以外に，制御周期を短くするというアプローチも存在する．しかし，現実には制御対象や計算機などの制約から，制御周期を十分に短くできないような状況も存在する．そこで，第 5 章「制御周期を短くできない環境での学習」では，行動選択が特定のタイミングでしか行えないという条件を加えた環境において，従来手法と比較を通して，提案手法の有効性を検討する．

第 6 章では，第 5 章までの結果をふまえて，提案手法の特徴について改めて考察するとともに，連続な状態行動空間における Q 学習についての先行研究との比

較を行う．

最後に，第 7 章で本研究の結果をまとめるとともに，今後の課題や展望について述べる．

第2章

SDNNを用いた連続状態行動空間における価値関数近似

第1章で述べたように，新保らは，状態変数 $\boldsymbol{s}$ を入力とし，ある行動値 a_i に対応するQ値 $Q(\boldsymbol{s}, a_i)$ を出力するSDNN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（本研究では，これをSDNN-Dと呼ぶ）を構成した．

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，$Q(\boldsymbol{s}, a)$ 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，行動選択の際にすべての行動に関する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，状態変数の次元数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恐れがある．

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値を近似する関数近似器（以下SDNN-C）を提案する [29].

2.1 SDNNによる関数近似

本節では，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 [24] 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，およびそれを連続状態・離散行動空間における価値関数近似に適用したSDNN-Dについて説明する．

2.1.1 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 $\bar{\boldsymbol{x}}_1, \ldots \bar{\boldsymbol{x}}_n$ 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 $\bar{\boldsymbol{s}} = (\bar{s}_1, \ldots, \bar{s}_m)^\top$ および $\bar{\boldsymbol{c}} = (\bar{c}_1, \ldots, \bar{c}_m)^\top$ 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．

不感化する素子の選び方として最も単純なのは，

$$\bar{s}_i \leftarrow \frac{1 + \bar{c}_i}{2} \bar{s}_i \tag{2.1}$$

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，2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．2 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．

2.1.2 SDNN による関数近似器の構成

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 2.1 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．

コードパターンの作り方には，$+1/-1$ の素子の固まりをスライドしていく方法 [24] や代表パターンの間を補間していく方法 [28]，ランダムに選択した素子の出力を反転していく方法 [26] などがある．本研究では，角度のように変域の両端が同じ状態を表す変数に対応したパターンを作りやすいことから，代表パターンの間を補間していく方法を用いることとした．第 3 章以降の実験で，実際に用いたコードパターンは以下のようにして構成した．

1. m 次元の 2 値パターンをランダムに 9 個作成し，$\bar{\boldsymbol{p}}_1, \bar{\boldsymbol{p}}_2, \ldots, \bar{\boldsymbol{p}}_9$ 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．
2. $\bar{\boldsymbol{p}}_i$ 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，$\bar{\boldsymbol{p}}_1, \bar{\boldsymbol{p}}_2, \ldots, \bar{\boldsymbol{p}}_9$ と合わせて 361 個のコードパターンを作成する．
3. このうち $\bar{\boldsymbol{p}}_9$ を除いた 360 個のパターンを，入力変数の変域を 360 等分した区間に順番通り割り当てる．

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 個の変数のすべてについて共通のものを用いることとした．

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

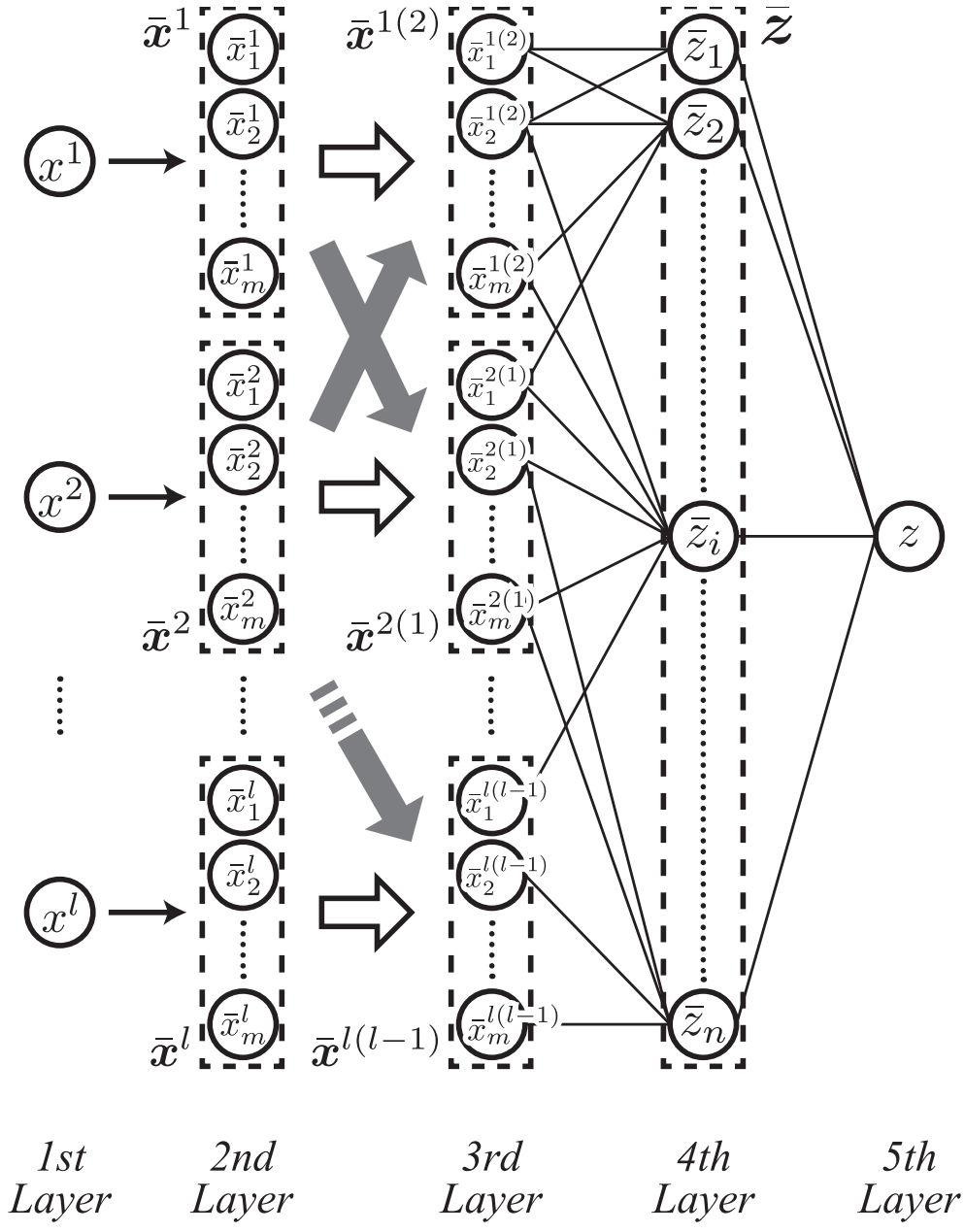


図 2.1: SDNN を用いた l 変数関数近似器の構成

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．以降の表記を簡略化するため，すべての μ, ν の組み合わせ（ただし $\mu \neq \nu$）について $\bar{\boldsymbol{x}}^{\mu(\nu)}$ を並べたもの（第 3 層の出力パターン）を $\bar{\boldsymbol{x}}^{SD}$ と表すこととする．

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 (2.1) ではなく，

$$\bar{x}_k^{\mu(\nu)} = \frac{1 + (\sigma^{\mu\nu}\bar{\boldsymbol{x}}^{\nu})_k}{2}\bar{x}_k^{\mu} \tag{2.2}$$

にしたがって行う（$\bar{x}_k^{\mu(\nu)}$ 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替えるための 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 および 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

$$\begin{aligned}\bar{z}_i &= \phi\left(\boldsymbol{w}_i^{\top}\bar{\boldsymbol{x}}^{SD} - w_{i0}\right) \\ &= \phi\left(\sum_{j=1}^{ml(l-1)} w_{ij}\bar{x}_j^{SD} - w_{i0}\right)\end{aligned} \tag{2.3}$$

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

$$z = g\left(\sum_{i \in I} \bar{z}_i\right) \tag{2.4}$$

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 [25]）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

$$w_{ij} \leftarrow w_{ij} + c\,\mathrm{sgn}\,(\hat{u} - u)\,\bar{x}_j^{SD} \tag{2.5}$$

$$w_{i0} \leftarrow w_{i0} - c\,\mathrm{sgn}\,(\hat{u} - u) \tag{2.6}$$

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．

このようなSDNNを行動数の分だけ用意したものが，新保らの提案したSDNN-Dである．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 (1.1) 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（結合荷重としきい値の更新）を行う．

5. 1. に戻る．

2.2 提案する関数近似器の構成

本研究で提案するSDNN-Cは，SDNN-Dの第4層においてQ値が多数のしきい値素子で分散表現されていることに着目し，ある状態におけるすべての行動に関する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 **2.1.2** 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（図 2.2）．なお，第 1 層から第 4 層については **2.1.2** の関数近似器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 (2.1) にしたがって行い，i 番目の素子の出力 $\bar{z}_i(\bar{y}_i)$ は

$$\bar{z}_i(\bar{y}_i) = \frac{1 + \bar{y}_i}{2} \bar{z}_i \tag{2.7}$$

となる．

第 6 層は，第 5 層のすべ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．

$$\begin{aligned} z &= f(\boldsymbol{x}, y) \\ &= g\left(\sum_{i \in I} \bar{z}_i(\bar{y}_i) \right) \end{aligned} \tag{2.8}$$

また，式 (2.8) 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．

$$z = g\left(\sum_{j \in J(y)} \bar{z}_j \right) \tag{2.9}$$

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こととした．

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

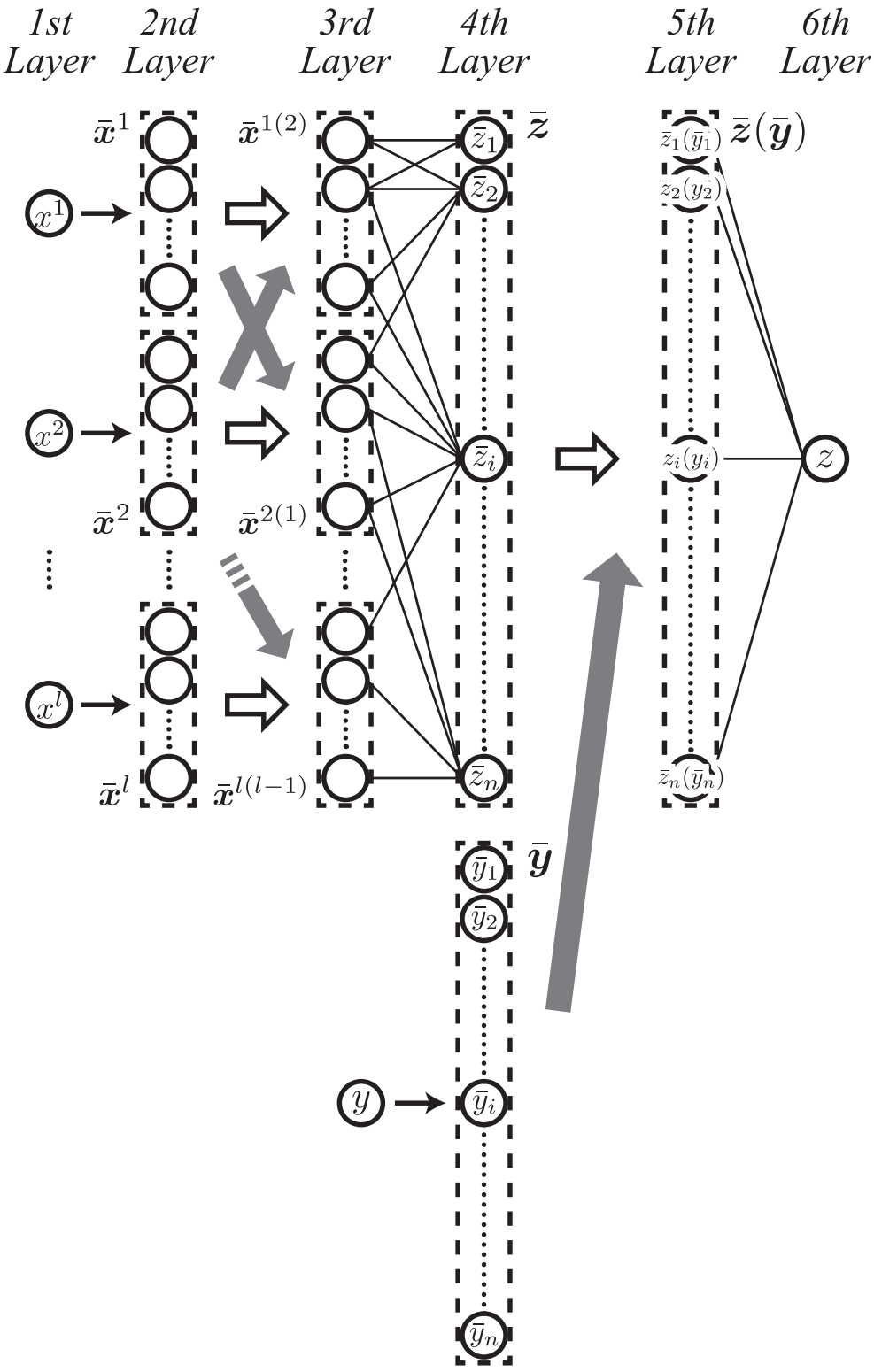


図 2.2: SDNN-C の構造

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

$$\bar{q}_{k,i} = \begin{cases} +1 & (\text{if } k \leq i < k + n') \\ -1 & (\text{otherwise}) \end{cases} \tag{2.10}$$

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．

$$J(y) = \{k, k+1, \ldots, k+n'-1 : y \in Y_k\} \tag{2.11}$$

2.3 モデルの学習則

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閾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，入力 y に対する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いくつか考えられるが，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，学習する入力 y に対応したコードパターンで不感化されなかった出力素子 $\bar{z}_j$ $(j \in J(y))$ 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）1 つに対して，その結合荷重と閾値を式 (2.5)，(2.6) にしたがって修正する．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 1 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．

2.4 行動次元におけ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 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 (2.10) のコードパターンを用いた場合，素子 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 値すべてが変化する．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，行動値が離れるにしたが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．**1.2.2** で見たように，Q 学習ではある状態において，選択可能なすべての行動の中で 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，$Q(\boldsymbol{s}, a_1), \ldots, Q(\boldsymbol{s}, a_q)$ の値をすべ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すべての組み合わ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，すべての 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 $O(p^2+q) < O(p^2q)$，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わかる．

第3章

離散行動と連続行動の比較

3.1 問題設定

第 2 章では，SDNN-D を連続行動空間に拡張した関数近似器 SDNN-C を構成し，これを用いて連続な状態行動空間での Q 学習における価値関数を近似する手法を提案した．

提案手法が既存手法と比べて有意義なのは，状態空間およ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．そこで本章では，上記のような状況を用意して数値実験を行い，離散行動と連続行動の違い，および提案手法の学習性能と計算コストについて検討する [29].

3.2 数値実験

3.2.1 学習課題

実験には，アクロボットの振り上げ課題 [1] を用いる．アクロボットは図 3.1 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ベンチマーク課題として，強化学習を始めとした制御分野においてしばしば用いられている [16,30,31].

一般的な振り上げ課題での制御目的は，両リンクが吊り下がって静止した初期

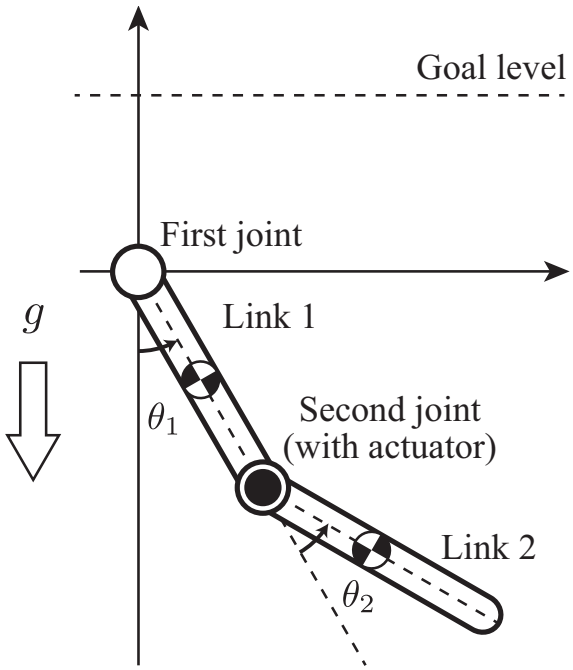


図 3.1: アクロボット

状態（$\theta_1 = \theta_2 = 0$ [rad], $\dot{\theta_1} = \dot{\theta_2} = 0$ [rad/s]）から開始し，できるだけ早く，第 2 リンクの先端を一定の高さ以上まで振り上げるようにアクチュエータを制御することである．見方を変えると，できるだけ効率よく力学的エネルギーを大きくするように行動を選択する問題と捉えることができる．この条件では時計回り方向と反時計回り方向でトルクが最大となる 2 行動をタイミングよく切り替えることが最適な行動選択となるため，それ以外の中間的な行動を選択することは，実質的に無意味である．そこで，本実験ではリンクを振り上げた際に，ゴールラインよりも上でリンクを一瞬静止する（$\dot{\theta_1}, \dot{\theta_2}$ の絶対値がある値未満になる）という条件を加えることとした．

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，学習エージェントが受け取る状態変数 $\boldsymbol{s} = (s_1, s_2, s_3, s_4)$ は，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 $[0, 1]$ に正規化したものとした．式で表すと以下のようになる．

$$\begin{cases} s_1 & = \theta_1/2\pi + 0.5 \\ s_2 & = \theta_2/2\pi + 0.5 \\ s_3 & = h(\dot{\theta_1}/6\pi + 0.5) \\ s_4 & = h(\dot{\theta_2}/10\pi + 0.5) \end{cases} \tag{3.1}$$

ここで関数 $h(x)$ は次式で表される飽和演算を行う関数である．

$$h(x) = \begin{cases} 1 & (1 \leq x) \\ x & (0 \leq x < 1) \\ 0 & (x < 0) \end{cases} \tag{3.2}$$

エージェントは，行動として第 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．行動を離散化する際には，最小値および最大値を含み，その間を等間隔に分けることとした（例えば行動数 $|\mathcal{A}| = 3$ 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 $|\dot{\theta_1}| < 3\pi\lambda$, $|\dot{\theta_2}| < 5\pi\lambda$ 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．

$$\lambda = \begin{cases} 10^{-\left(1+\frac{k-1}{10000-1}\right)} & (k \leq 10000) \\ 10^{-2} & (k > 10000) \end{cases} \tag{3.3}$$

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，および角速度 $\dot{\theta_1}$，$\dot{\theta_2}$ の値が正規化範囲から外れた場合は -5 を与え，それ以外の場合はすべて 0 とした．

3.2.2 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．制御（行動選択）の周期は，学習性能の差が明確に現れるように 1 秒間隔と長めに設定した．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．

実験では，提案手法と従来手法（SDNN-D および RBFN）を比較する．SDNN-D のパラメータは新保らの実験設定に倣うこととし，SDNN-C は，構成する素子の数を SDNN-D と揃えて必要なメモリ量を同程度にした．具体的には，SDNN-D のパラメータは $m = 200, n = 300, c = 0.1$，SDNN-C のパラメータは $m = 200, n = 900, n' = 300, c = 0.1$ とした．このとき，SDNN-D が扱う行動数が $|\mathcal{A}| = 3$ 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 (3.4) とした．

$$g(u) = \begin{cases} +20 & (280 < u) \\ \frac{2}{13}u - \frac{300}{13} & (20 \leq u \leq 280) \\ -20 & (u < 20) \end{cases} \tag{3.4}$$

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 4 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，すべ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，

$$Q(\boldsymbol{s}, a) = \sum_j w_j \exp\left[-\frac{\|\boldsymbol{s} - \hat{\boldsymbol{s}}_j\|^2 + (a - \hat{a}_j)^2}{\sigma^2}\right] \tag{3.5}$$

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，**1.2.2** 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 (3.5) 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を若干低減できるが，それでも行動数にほぼ比例して計算量が増加する（**2.4** の表記法で表すと，RBFN の計算コストは $O\left(N^p + Nq\right)$ 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．

以上の方針にしたがって 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，

表 3.1: 強化学習 1 ステップの平均計算時間

関数近似器	行動数 $\lvert\mathcal{A}\rvert$	計算時間 [ms]
SDNN-C	601	28.4
SDNN-D	3	25.5
RBFN	21	919

初期状態において，すべての状態・行動に対して $Q(\boldsymbol{s}, a) \equiv 0$ とした．RBFN の場合，これはすべ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 および SDNN-D の場合，結合荷重をすべて 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うえで，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（偶数番目の素子）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．

実験は，関数近似器と行動数 $\lvert\mathcal{A}\rvert$ のみが異なる 3 条件，SDNN-C（$\lvert\mathcal{A}\rvert = 601$），SDNN-D（$\lvert\mathcal{A}\rvert = 3$），RBFN（$\lvert\mathcal{A}\rvert = 21$）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．

3.2.3 結果

まず各手法の計算時間を表 3.1 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わかる．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．

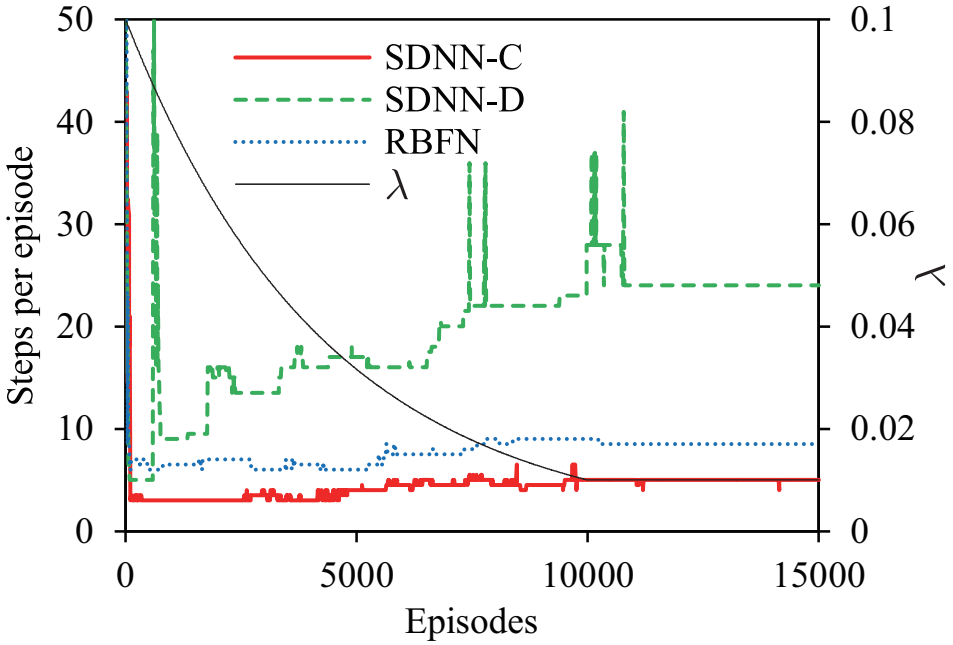
次に，評価エピソードの継続ステップ数，すなわちゴールに到達するまでのステップ数（到達しない場合は 50）の変化を図 3.2 に示す．グラフの横軸は学習エピソード数，左の縦軸はその際の評価エピソードの継続ステップ数であり，SDNN-C, SDNN-D，RBFN のそれぞれについて，10 試行の中央値をプロットしている．また，ゴール時の角速度の許容範囲を表す変数 λ の値（右の縦軸）も黒い細線でプロットした．(a) は全 15000 エピソードについて示したグラフ，(b) はその最初の 500 エピソードの部分を拡大したものである．

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 の学習効率が他の手法と比べて低いとは言えない．

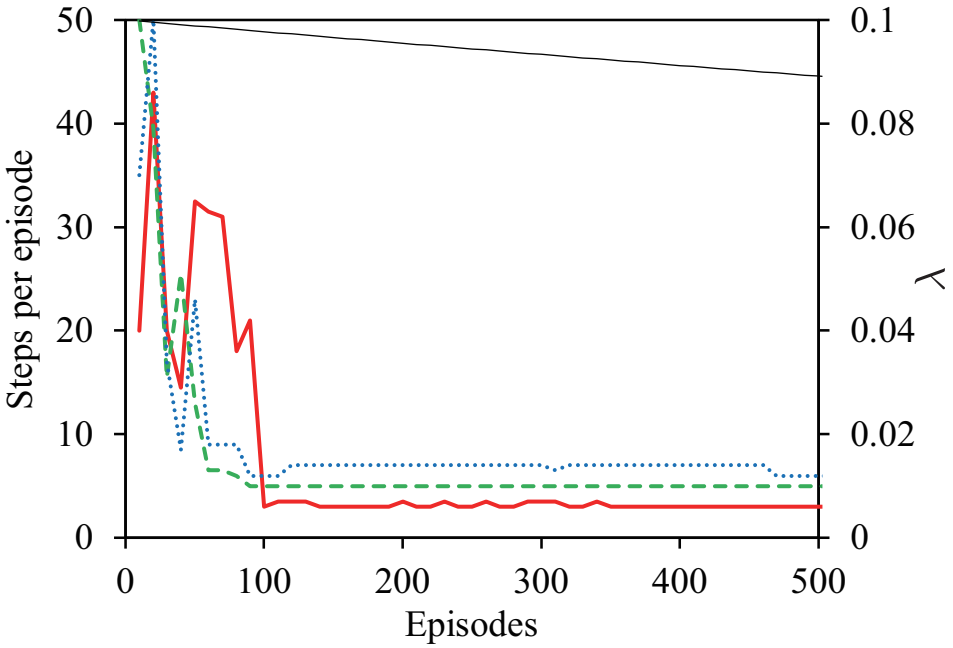
エピソードが進んでゴールの条件が厳しくなると，SDNN-D では継続ステップ数が急激に大きくなることがしばしばあった．これは，それまでに獲得した行動系列ではゴールに到達することができなくなり，ゴールに到達可能な全く別の行動系列を獲得したためと考えられる．一方，SDNN-C と RBFN は，継続ステップ数が急激に変化することはなく，最終的なゴール条件でも 10 ステップ以下で振り上げる行動が学習されている．これは，行動を少しずつ変えることによってゴール条件の変化に対応できたことを示している．

最終的な継続ステップ数（中央値）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．

実際に各エージェントが学習した行動の例を図 3.3，3.4，3.5 に示す．グラフは，



(a) Episode 1 - 15000



(b) Episode 1 - 500

図 3.2: 各関数近似器の学習過程

ある実験試行の最後のテストエピソードにおける状態遷移の様子をプロットしたもので，横軸は物理シミュレーション上の時刻，縦軸はアクロボットの各状態変数の値および行動変数の値である．また，$\dot{\theta}_1, \dot{\theta}_2$ のグラフ中の網掛けの領域は負の報酬が与えられる状態，角速度 0[rad/s] 付近の細線で囲まれた領域はゴール状態を表す.

SDNN-C および RBFN は，実際に中間的な行動を選択できており，角速度の値（特に $\dot{\theta}_2$）も，比較的小さな範囲に収まっている．一方，3 行動の SDNN-D は，角速度がかなり大きな値をとっている点が，何ヶ所か見られる．

3.3 まとめ

ゴール条件を追加したアクロボットの振り上げ課題において，提案手法と従来手法の比較，および連続行動と離散行動の比較を行った．

その結果，SDNN-C は SDNN-D と同程度の計算コストで，約 200 倍の数の行動を扱うことができた．両者の計算コストの差は 3[msec] 程度であるから，**2.4** で述べたように，第 5 層以降の計算コストの割合はかなり小さいと言える．一方，RBFN は行動数が SDNN-C の 1/30 程度であったにもかかわらず，計算に SDNN-C の 30 倍以上の時間を必要とした．

また，学習が収束するまでのエピソード数の比較から，SDNN-C は行動次元において十分な汎化能力を持っているといえる．加えて，行動が難しくなる後半のエピソードにおいても継続ステップ数が大きく悪化していないことから，(行動数が異なるため，単純な比較はできないが）RBFN と同程度かそれ以上の関数表現能力を持っているといえる．

さらに，離散行動の SDNN-D は課題が難しくなるに連れて，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のに対し，連続行動の SDNN-C と RBFN では，このような傾向は見られなかった．この結果から，連続行動は離散行動よりも環境の（連続的な）変化に追従しやすいことが示唆される．実問題では，環境が非定常なものも多いため，変化に追従しやすいという性質は，連続行動を用いる大きな利点であると考えられる．

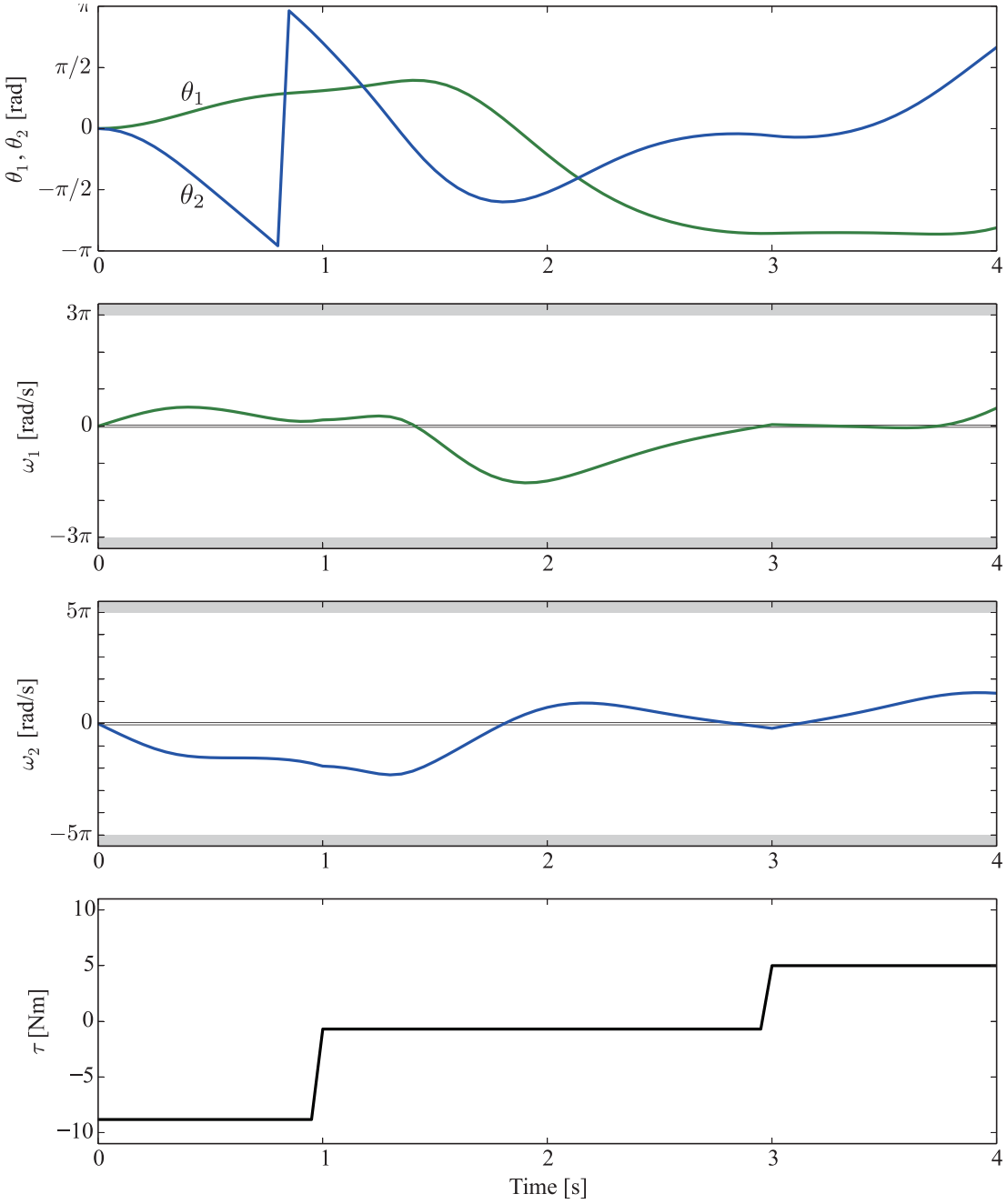


図 3.3: 学習した行動と状態遷移（SDNN-C）

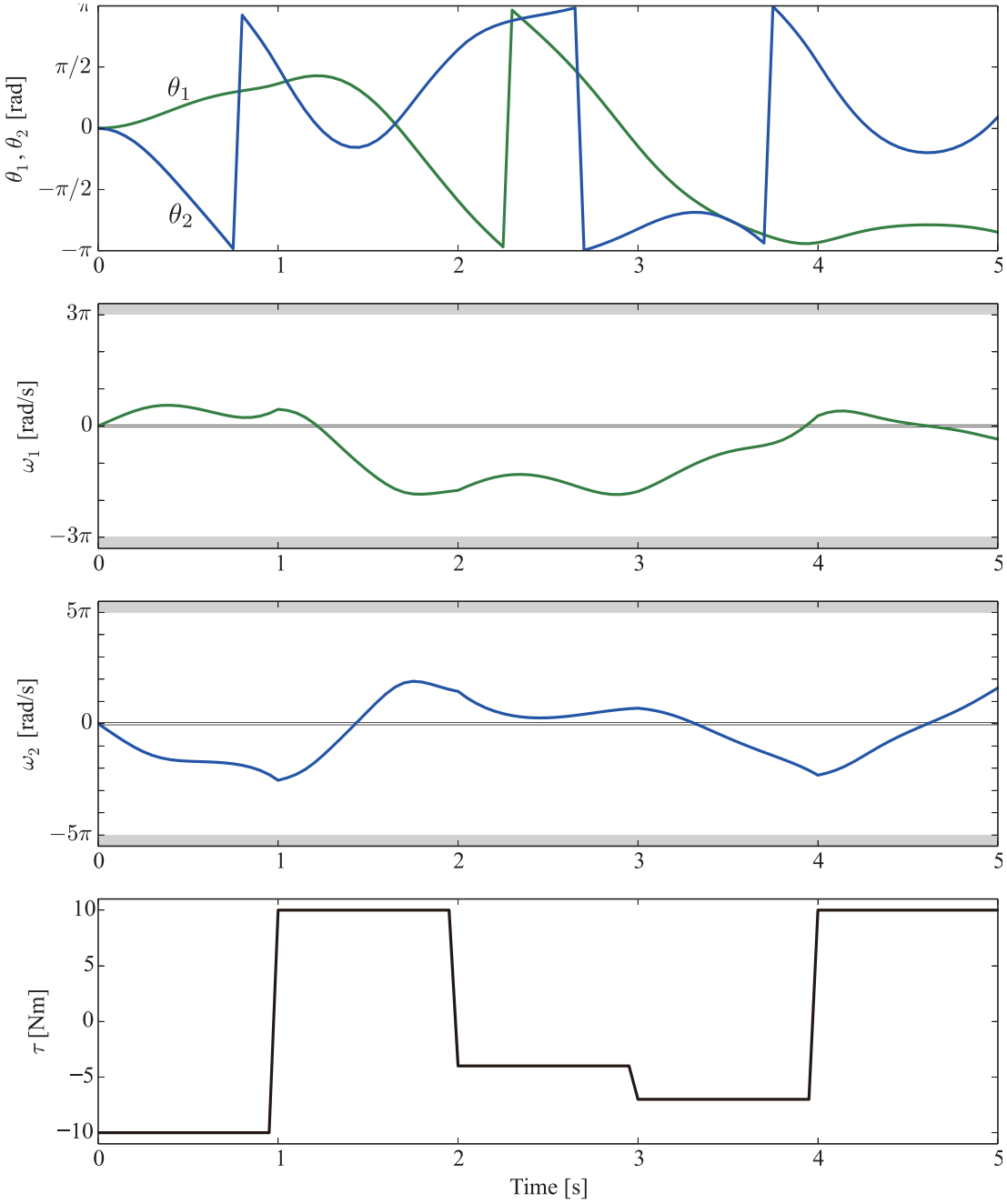


図 3.4: 学習した行動と状態遷移（RBFN）

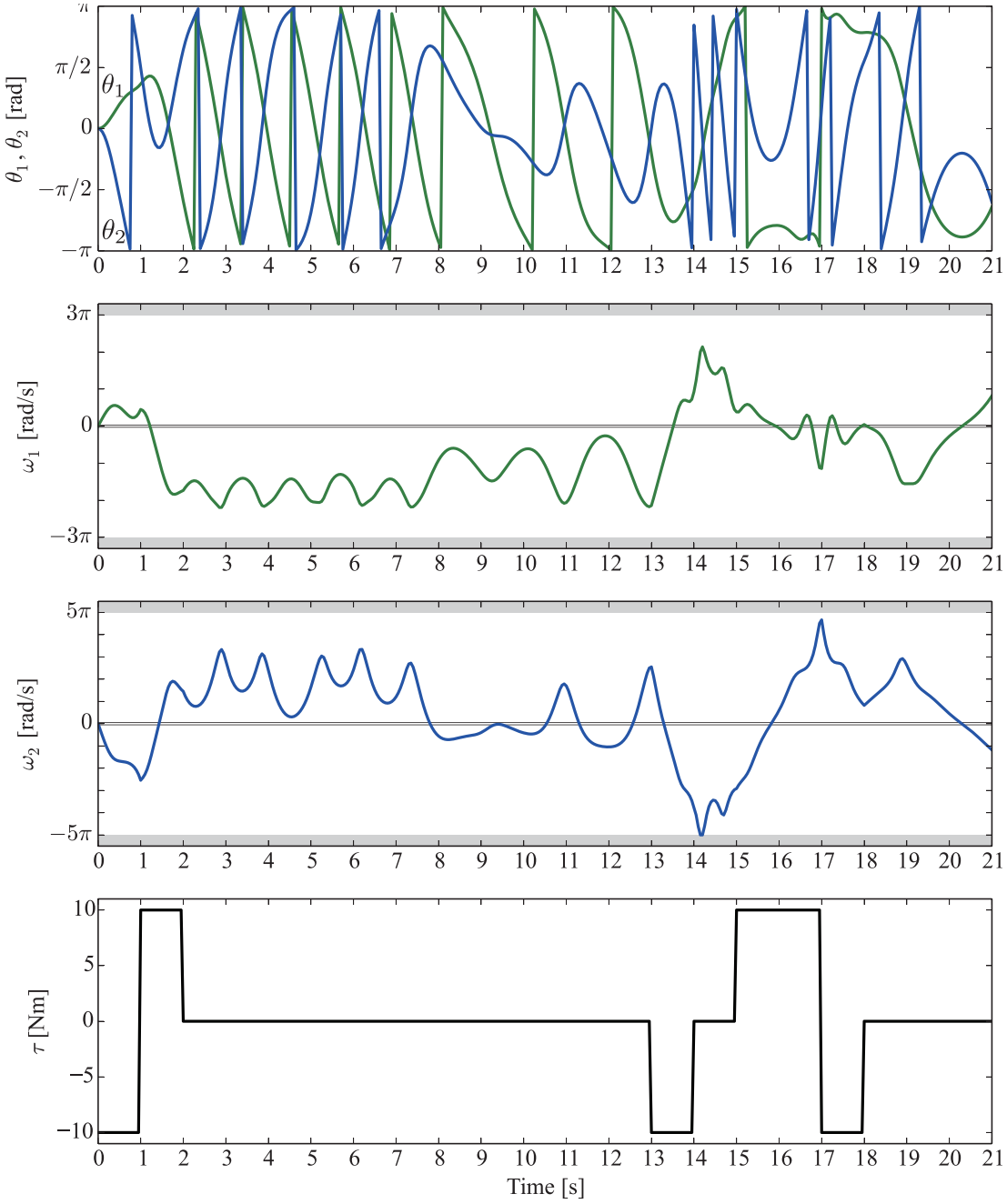


図 3.5: 学習した行動と状態遷移（SDNN-D）

ただし，本章で行ったアクロボットの振り上げ課題は，連続な行動を用いる代わりに，行動数は少ないが制御周期を十分短くすることでもゴールに到達することができる（実際に数値実験を行ったところ，制御周期が 250[ms] ならば，3 行動でも短時間でゴールに到達することができた）．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．つまり，本章の結果から，SDNN-C を用いると制御周期を長くすることができるとは言えるが，SDNN-C を用いることが不可欠だとは言えない．この点に関しては第 5 章で検討する．

第4章

冗長な状態変数を含む環境での学習

4.1 問題設定

第3章ではアクロボットの振り上げ課題を用いて，提案手法が連続な状態行動空間において適切な行動価値関数を学習できることを確認した．

本章では，より実問題に近い状況として，制御目的を達成するために必要（かつ十分）な状態変数が未知であるような場合を考える．このような場合に，状態空間を設計する最も簡単な方法は，制御目的に関係のありそうな変数やセンサ出力をすべて利用することである．しかし，この方法では状態空間の次元数が増えるため，学習効率が低下したり，関数近似器の計算コストが増加するといった問題が起こる（次元の呪い）．

ただし，上記のように選択した状態変数が，すべて重要ということは少なく，一部だけが重要で残りは冗長という可能性が高い．例えば，森本ら [32] は3つのリンクからなるロボットの起き上がり課題において，獲得された関数近似器の基底関数の配置を主成分分析したところ，6つの状態変数のうち2つが学習にほとんど寄与しない（次元が縮退している）ことを示している．このため，実問題における強化学習では，冗長次元に対して高いロバスト性をもつ関数近似器が適していると考えられる．

新保らはSDNN-Dの入力に，4つの状態変数以外に無関係な変数（一様乱数やステップ数に依存した三角関数）をいく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 [28]．また，強化学習の例ではないが，Horieらの研究もSDNNが冗長な入力に対して高いロバスト性を持っていることを示唆し

ている [27]．提案手法はSDNN による関数近似器の出力層に行動パターンによる不感化操作を加えたものであるから，同様に冗長な入力に対して高いロバスト性を持つことが期待される．

そこで本章では，状態空間に冗長だが意味のある変数が含まれる場合に，提案手法の学習効率がどのように変化するかについて検証を行う [33]．

4.2 数値実験

4.2.1 学習課題

数値実験には，第 3 章と同様のアクロボットの振り上げ課題を用いる．アクロボットの初期状態やゴール条件，物理パラメータ，報酬関数の実験設定は全て **3.2.1** と同じものである．

状態変数は，式 (3.1) の s_1, s_2, s_3, s_4（正規化した両リンクの角度と角速度）に加え，2 種類の冗長変数 s_5, s_6 を用いる．s_5, s_6 はそれぞれ，第 1 関節からみた第 2 リンクの先端の高さ，およびアクロボットの近似的な運動エネルギーを $[0, 1]$ の範囲に正規化したもので，式 (4.1) と式 (4.2) で与えられる．

$$s_5 = -\frac{1}{2}\left(\frac{l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)}{l_1 + l_2} + 1\right) \tag{4.1}$$

$$s_6 = 2\left(I_1(s_3 - 0.5)^2 + I_2(s_4 - 0.5)^2\right) \tag{4.2}$$

ここで，l_1，l_2 は各リンクの長さ，I_1，I_2 は慣性モーメントであり，$l_1 = l_2 = 1[\mathrm{m}]$，$I_1 = I_2 = 1[\mathrm{kg \cdot m^2}]$ である．

一方，行動変数 $a = (\tau)$ は，関数近似器の種類や冗長変数の有無によらず，行動集合 $\mathcal{A} = \{-10, -10 + 1/300, -10 + 2/300, \ldots, 0, \ldots, 10 - 1/300, 10\}$ から選択されるものとした．したがって，行動数 $|\mathcal{A}|$ は 601 となる．

4.2.2 方法

実験は第 3 章と同様に，学習エピソード評価エピソードとを分けて行った．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，それ以

表 4.1: 実験条件

条件名	エージェントが観測する状態変数の構成
Case I	(s_1, s_2, s_3, s_4)
Case II	$(s_1, s_2, s_3, s_4, s_5)$
Case III	$(s_1, s_2, s_3, s_4, s_5, s_6)$

降は $\varepsilon = 0.1$ の ε-greedy 方策にしたがって行動を選択し，価値関数の学習を行う（強化学習のパラメータは，$\alpha = 0.5, \gamma = 0.9$ とした）．一方，評価エピソードでは，エージェントは常に greedy 方策に従って行動を選択し，価値関数の学習は行わない．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．

1 回の実験試行は，学習エピソードを 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．

SDNN-C のパラメータは，第 3 章と同様のもの（$m = 200, n = 900, n' = 300, c = 0.1$）とした．出力値の範囲は $[-20, 20]$ で，スケール変換関数 $g(u)$ も同様（式 (3.4)）である．

RBFN は，**3.2.2** と同様に，各状態行動次元に同数のガウス関数を等間隔で配置して構成する．各次元に配置するガウス関数の個数は，状態空間が 4 次元 $\boldsymbol{s} = (s_1, s_2, s_3, s_4)$ のときの計算コストが SDNN-C と同程度になるように決定した．その上で，学習効率が最も良くなるように，ガウス関数の広がりを表すパラメータ σ を設定した．

以上の方針にしたがって RBFN に関する予備実験を行った結果，基底関数は各状態行動次元に 4 個，全体で 4^d 個を格子状に配置した．ここで d は状態行動空間の次元数である．また，σ は 0.2 とした．

SDNN-C，RBFN ともに初期状態において，すべての状態・行動で $Q(\boldsymbol{s}, a) \equiv 0$ となるように初期化した（初期値の設定方法については **3.2.2** 参照）．

計算コストの比較実験は，表 4.1 に示す 3 つの条件について行った．Case I は

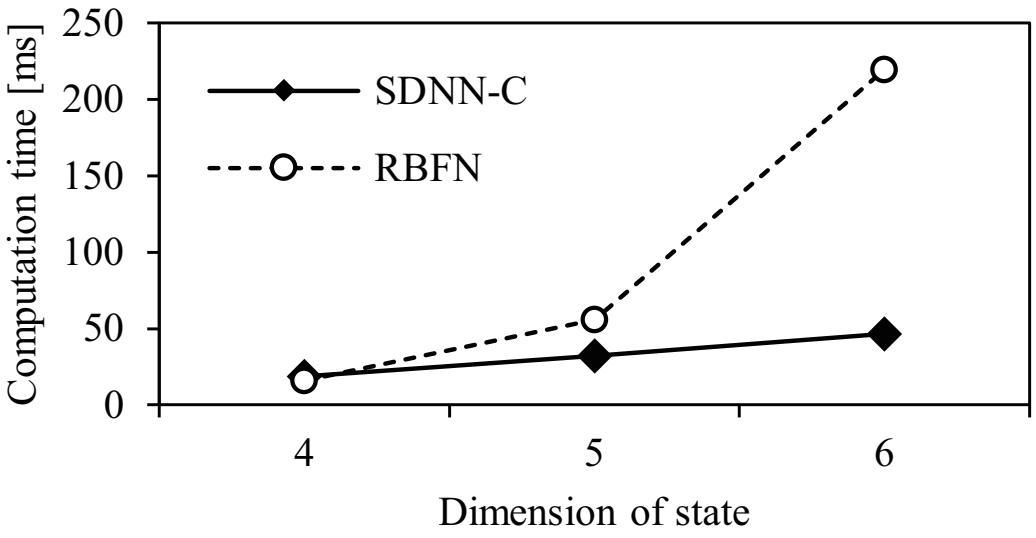


図 4.1: 強化学習 1 ステップの計算にかかった平均時間

冗長な変数を含まない場合（第 3 章の実験条件に相当），Case II と Case III は冗長な変数を含む場合である．各関数近似器，実験条件につき，それぞれ 1 回の実験試行を行い，その中から無作為に抽出した 1000 回の強化学習ステップ（学習エピソード）の平均計算時間を比較した．

また，冗長変数に対する学習効率の実験は，SDNN-C は表 4.1 の 3 つの条件について行った．一方，RBFN は計算コストの問題から Case I のみ行った．各関数近似器，実験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い，継続ステップ数（エピソード終了までにかかったステップ数）の中央値を比較した．

4.2.3 結果

まず，計算時間の結果を図 4.1 に示す．計算環境は，Amphis BTO MD800iCi7G TYPE-SR インテル core i7-3970X (6-core) プロセッサ，16GB SDRAM メモリ (DDR3-1600 4GB × 4) である．横軸は状態空間の次元数（左から順番に Case I, Case II, Case III にあたる），縦軸は平均計算時間である．

図より，状態空間の次元数の増加に対して，RBFN の計算コストは指数的に増加している．これは，基底関数を各次元に等間隔に配置するため，基底関数の数が状態次元の増加に対して指数的に増えるからである．一方で，SDNN-C の計算

コストはより小さな増加に抑えられている．図からは，具体的なオーダーまでは分からないが，SDNN の第 3 層の素子数は状態変数の数に対して 2 次のオーダーで増える（詳細は **2.4** を参照）ため，SDNN-C の計算コストは 2 次のオーダーになっていると考えられる．したがって，状態空間がより高次元の場合，計算コストの差はさらに大きくなると考えられる．

次に，各実験条件における学習効率を図 4.2 に示す．各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，右の縦軸はゴール状態の許容範囲を表す変数 λ の値である．

図 4.2(a) より，SDNN-C エージェントは，約 150 エピソードで継続ステップ数が収束しており，最終的にも 10 ステップ未満でゴールする行動を学習できている．さらに，SDNN-C は冗長次元を加えた Case II および Case III でも同程度のエピソード数で継続ステップ数が収束し，最終的にも 10 ステップ未満でゴールする行動を学習している．

一方で，RBFN エージェントは 4 次元の状態空間でも，15000 エピソードで振り上げ行動を学習できないことが分かる．行動数 $|\mathcal{A}|$ はどちらの関数近似器も 601 であるから，これは RBFN の関数近似能力が不十分であったためと考えられる．

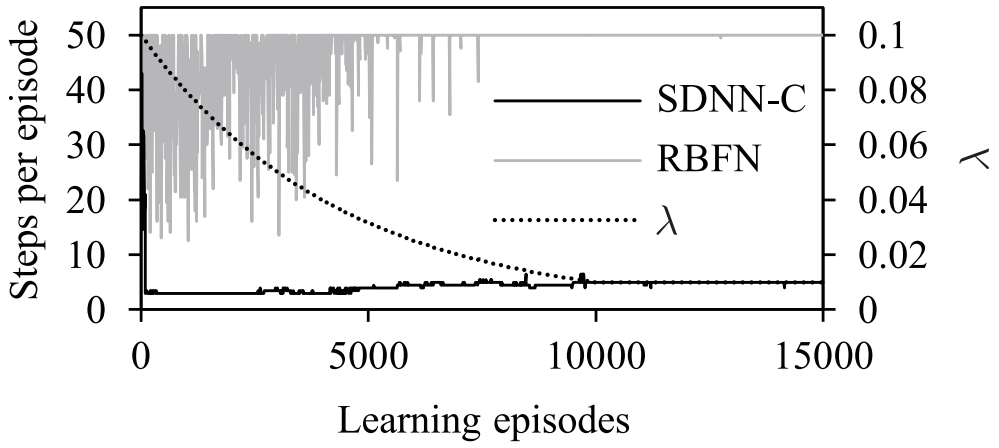
4.3 まとめ

状態空間に冗長だが意味のある変数が含まれる場合について，SDNN-C と RBFN の計算コストと学習効率の比較を行った．

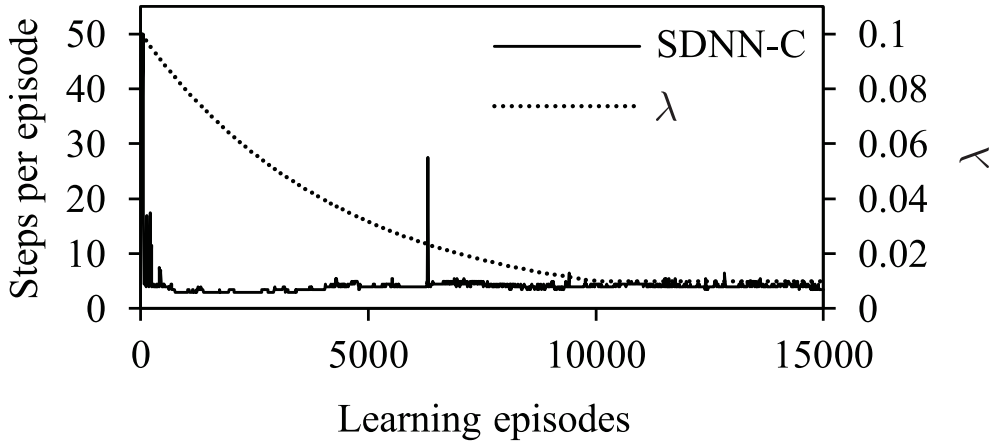
その結果，SDNN-C は冗長変数を追加した場合でも，学習性能の悪化が見られなかった．先に述べた新保らの実験結果などとあわせて考えると，SDNN-C も冗長な状態変数に対して高い冗長性を持っているといえる．

この性質は，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と考えられる．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すべ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．

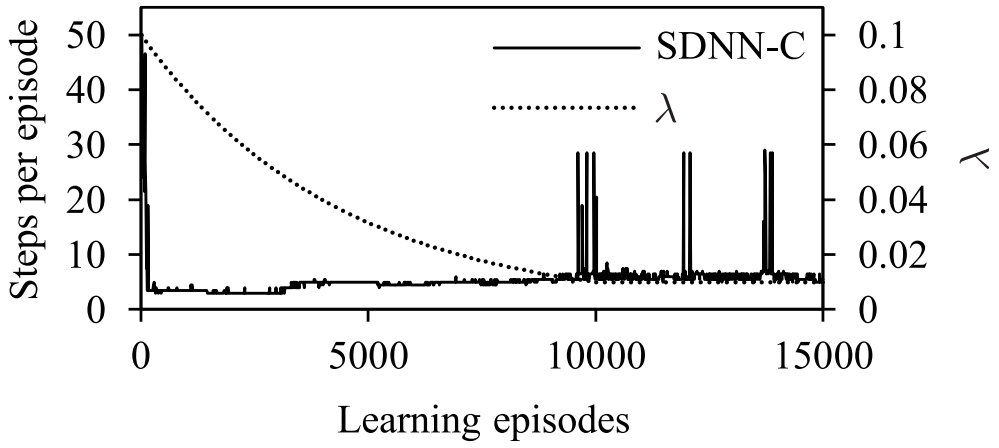
一方，RBFN は計算コストの増加が大きく，計算コストが同程度の SDNN と比



(a) Case I.



(b) Case II.



(c) Case III.

図 4.2: 各実験条件における学習過程

べ，関数近似能力が劣ることが示された．第 3 章の実験では同様の条件（ただし，行動数は 21）で，10 ステップ未満で振り上げる行動を学習できていることから，ガウス関数の数を増やし，空間中により稠密に配置すれば，RBFN でも課題を解くことは可能だと考えられる．しかし，その分だけ計算コストが増加してしまう．計算コストが次元数に対して指数的に増加することとあわせると，RBFN が有効な問題は，状態次元が比較的小さく（4 次元程度），かつ価値関数の形状がかなり単純なものに限られると思われる．

この問題に対し，ガウス関数の数や配置，広がりを適応的に変化させることで，高次元空間でも，比較的少数のガウス関数で価値関数を近似するというアプローチがある [32,34]．しかし，これらの操作は近似結果（関数形状）に変化を引き起こすため，変化が大き過ぎたり早過ぎたりすると，学習が不安定になる可能性がある [34]．また，変数にノイズが乗るような環境では，ノイズの影響で本来は必要ないガウス関数が追加される可能性もある．このように，ガウス関数を適応的に変化させる方法は，計算コストの低減に一定の効果があると考えられるが，必ずしも使いやすい方法とは言えない．

第5章

制御周期を短くできない環境での学習

5.1 問題設定

これまでの振り上げ課題は，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，計算量およ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，第 3 章の結果から，SDNN-C を用いると制御周期を長くすることができるとは言えるが，SDNN-C を用いることが不可欠だとは言えない．

そこで本章では，制御周期を短くすることができない状況における制御を考える．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．

以下では，まずアクロボットの振り上げ課題に，上記のような制御周期に関する条件を加えた問題を定義し，提案手法と SDNN-D，RBFN の学習性能の比較を通して，提案手法の有効性を検討する [29].

5.2 数値実験

5.2.1 学習課題

実験課題として，これまでと同様のアクロボットの振り上げ課題を用いる．ただし，制御周期が短くできないような課題とするため，前述のブランコの例と同様にして，エージェントは第 1 リンク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条件を設ける（実際のプログラム上では，θ_1 の値の符号が直前のステップから切り替わったタイミングで判定する）．アクチュエータは次の行動選択のタイミングまで，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，第 1 リンク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）は常に同じ値となるので，状態空間は実質的に s_2, s_3, s_4 の 3 次元となる．

報酬は，ゴール状態に到達した際に $+10$ を，角速度 $\dot{\theta_1}, \dot{\theta_2}$ が正規化範囲を外れた場合は -5 を与える．それ以外の場合は，直前の強化学習ステップ中に $|\theta_1| < 5$[deg] であった時間の割合に応じた大きさの罰を与える．具体的には，時刻 t における報酬 r_t は，式 (5.1) のように定義した．

$$r_t = \begin{cases} +10 & (\text{if} \quad \text{height} \geq 1.5[\text{m}], |\dot{\theta_1}| < 3\pi\lambda, |\dot{\theta_2}| < 5\pi\lambda) \\ -5 & (\text{if} \quad |\dot{\theta_1}| > 3\pi \quad \text{or} \quad |\dot{\theta_2}| > 5\pi) \\ -5 \ \frac{steps_{penalty}}{steps} & (\text{otherwise}) \end{cases} \tag{5.1}$$

ここで，$steps$ は時刻 $t-1$ における強化学習ステップが終了するまでにかかった物理シミュレーションのステップ数，$steps_{penalty}$ はその強化学習ステップ中で $|\theta_1| < 5$[deg] であった物理シミュレーションのステップ数である．

5.2.2 方法

実験はこれまでと同様に，学習エピソードと評価エピソードを分けて行った．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 1000 ステップと

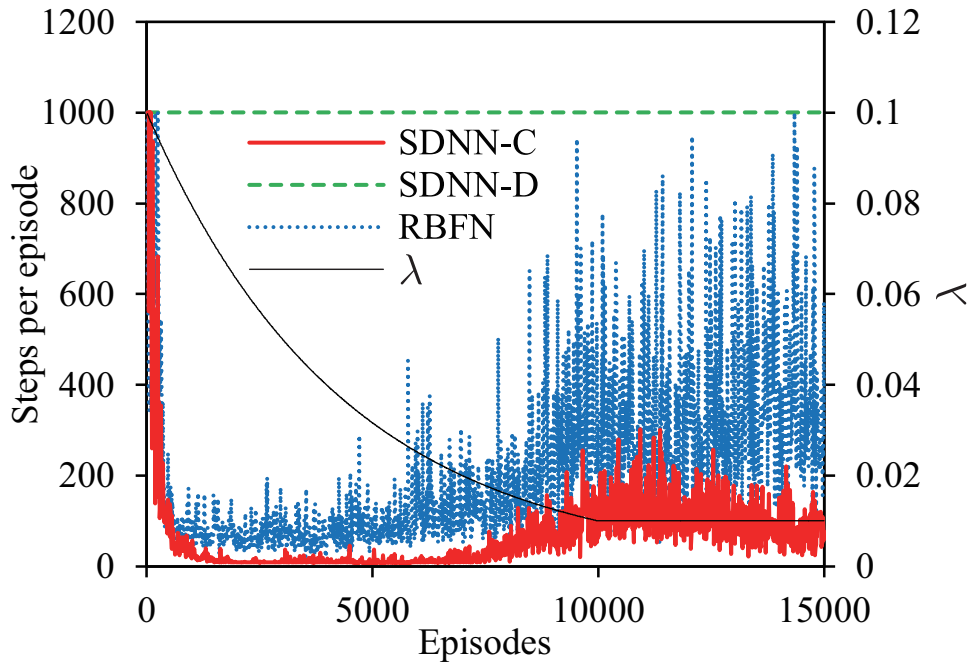


図 5.1: 各関数近似器の学習過程

し，関数近似器に SDNN-C（$|\mathcal{A}| = 601$），SDNN-D（$|\mathcal{A}| = 3$）および RBFN（$|\mathcal{A}| = 601$）を用いて実験を行った．ここで，SDNN-C と SDNN-D は第 3 章の実験と同様の設定（ただし，入力変数は 3 次元）とした．

一方，RBFN は SDNN-C と行動数およ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 としたうえで，強化学習 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 **3.2.2** と共通である．

5.2.3 結果

実験結果を図 5.1 に示す．図 3.2 や図 4.2 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言える．

これに対してSDNN-Cの場合，1000〜2000エピソードで一旦継続ステップ数が収束した後，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わかる．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エピソードで一旦収束し，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．

実際にSDNN-Cを用いたエージェントが学習した行動選択の例を図5.2に示す．グラフは，ある実験試行の最終評価エピソードにおける状態遷移の様子をプロットしたもので，横軸は物理シミュレーション上の時刻，縦軸はアクロボットの各状態変数の値および行動変数の値である．また，$\dot{\theta}_1, \dot{\theta}_2$のグラフ中の網掛けの領域は負の報酬が与えられる状態，角速度0[rad/s]付近の細線で囲まれた領域はゴール状態を表す．グラフは最後の強化学習ステップが終了するまで（$\theta_1 = 0$となるまで）プロットしているが，実際にゴール条件が満たされたのは，6.56[s]の時点であった．グラフより，最初の数ステップでは比較的トルクが大きな行動を選択してリンク2の振れを大きくし，最後のステップで，上手くリンク1の振れに変換して振り上げに成功していることが分かる．

5.3 まとめ

制御周期が短くできないような条件を加えたアクロボットの振り上げ課題おいて，提案手法とSDNN-DおよびRBFNの学習効率を比較した．その結果，この課題は離散行動では解けない（あるいは非常に困難な）課題であり，価値関数の

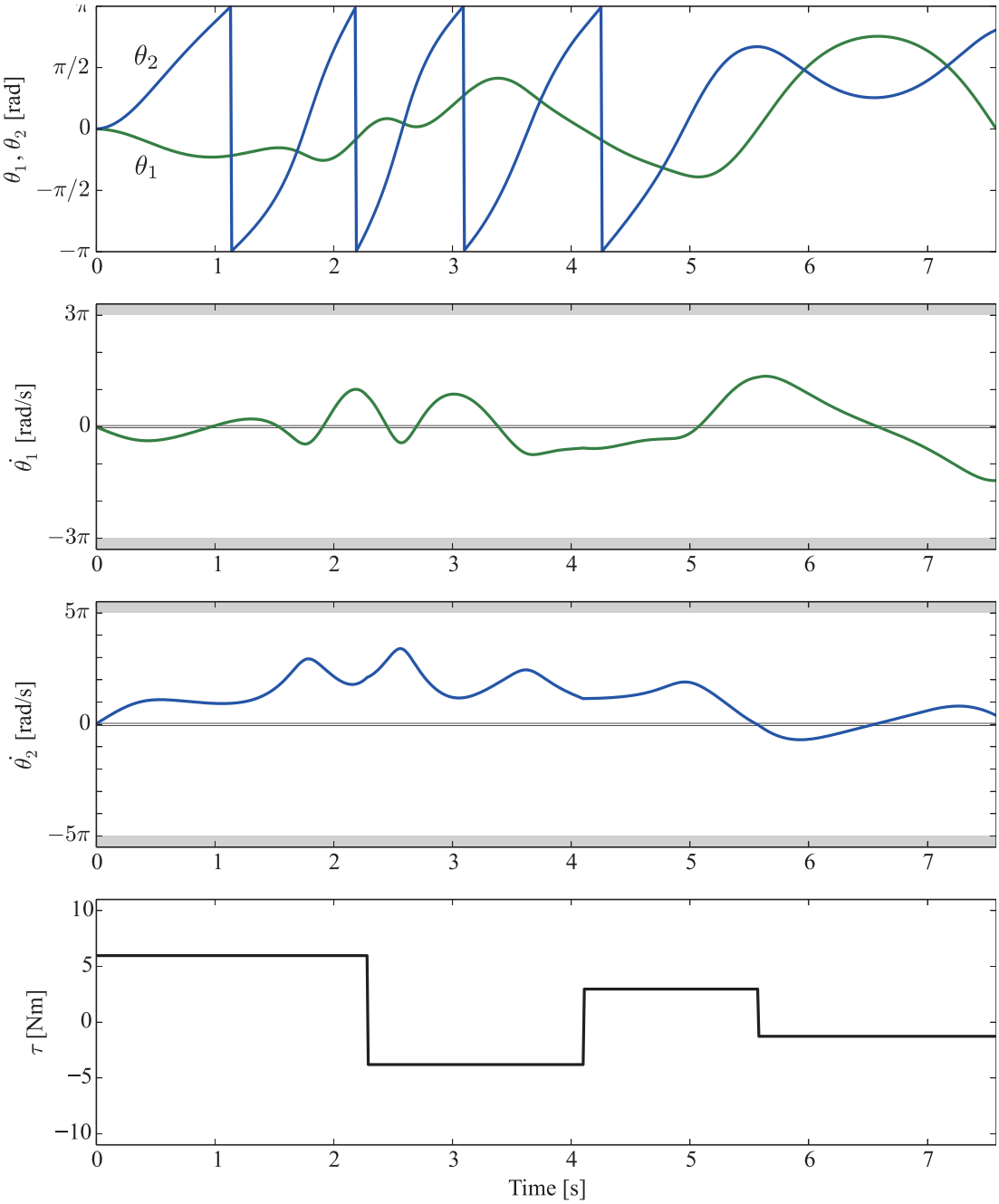


図 5.2: 学習した行動と状態遷移（SDNN-C）

近似器に高い関数表現能力が求められる課題であることが分かった．そして，このような課題において，SDNN-C は従来手法よりも適した関数近似手法であることが示された．

このように制御周期を短くできなかったり，できても計算資源の問題などから限界があるような問題は現実にも数多く存在する．また，制御周期を長くできるということは，ゴール状態までの状態遷移経路を短くできるということを意味する．Q 学習を始めとする TD 学習では，報酬に関する情報が状態遷移を逆向きにたどるように広がっていくため，状態遷移が短いということは，それだけ学習が早まる可能性がある（ただし，連続行動では 1 ステップあたりの遷移先は増えるため，実際にどちらの方が効率が良くなるかは，今後の検討課題である）．

第6章

考察

6.1 提案手法の特徴

本節では，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による関数近似器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[25]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，不連続な境界を含むなど，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[26] この際，近似精度を上げるためには，素子数を増やし表現能力を高める必要があるが，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（**状態行動に対する汎化，関数表現能力**）．

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（**モデルフリー**）．

さらに，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[28]．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，オンライン学習が必要となるQ学習で用いるうえでの利点の一つである（**オンライン学習**）．

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（**高次元空間での計算コスト**）．

これに加え，冗長次元の影響を受けにくい（**冗長入力に対するロバスト性**）．計算コストの増加があまり大きくないこととあわせて，SDNNを用いることで状態空間の設計はかなり容易になると考えられる．

以上はすべて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言えば，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Q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，第5章の実験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Q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[26]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，全く同じ状態で行動値が微妙に変わるとQ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．

6.2 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 [15] は，各状態においていくつかの行動の Q 値を多層パーセプトロン (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，過去のサンプルをすべて保存しておいて適宜再学習しなければならないし（**オンライン学習**の問題），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．さらに，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恐れがあるので，不連続点付近の誤差を減らすことも困難である（**関数表現能力**の問題）．

Millán ら [35] は，状態空間を適応的に離散化したうえで，いく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（**状態行動に対する汎化**，**関数表現能力**の問題）．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 [13,36] 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）の一種と見なせるが，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（実際に，野中らの 2 変数関数の近似課題 [26] に適用してみたところ，計算コストが同程度の SDNN よりも，かなり大きな近似誤差が生じた（**関数表現能力**の問題））．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（**行動選択**の問題）．行動選択にかかる計算コストの問題に対して，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-C にも適用可能である．

Pazis と Parr [37] は，状態価値関数およ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（**状態行動に対する汎化**の問題など）．

Carden [38] 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したが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（**状態行動に対する汎化**，**関数表現能力**の問題）．また，行動数に対する計算コストの増加を抑える点についても考慮されていない（**行動選択**，**行動評価**の問題）．

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言えよう．

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 手法がある．**1.2.1** で説明したように，この手法は Actor によって行動方策が明示的に表現できるため，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，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場合がある．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，実験した範囲ではすべて学習に失敗した．ただし，Actor の設計やパラメータの設定が不適切であったため失敗した可能性も十分に考えられるため，これだけをもって手法自体の問題とは言えない．しかしながら，少なくとも使いやすさの点では SDNN-C を用いた Q 学習の方が優れていると思われる．

第7章

結論

7.1 本研究のまとめ

本研究では，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，連続な状態行動空間におい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，複数の数値実験を行い，提案手法の有効性を検証した．

第3章では連続な状態行動空間を持つ課題において数値実験を行い，提案手法の計算コストおよび学習性能を従来手法と比較した．その結果，提案手法は同程度の関数近似能力を持つと考えられるRBFNよりもはるかに小さい計算コストで，行動値を十分細かく離散化できることが分かった．また，同様にSDNNを用いた新保らの手法と比べても，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がなかった．

加えて，連続行動と離散行動の比較から，連続行動を用いることで環境の変化に追従しやすくなる可能性が示された．このことは，非定常な問題に対して強化学習を適用する上で，大きな利点になると考えられる．

第4章では，より実問題に近い状況として，冗長な状態変数に対するロバスト性について検証した．また合わせて高次元空間での計算コストの増加についても検証を行った．その結果，提案手法は冗長な状態変数が追加された場合でも，学習効率が悪化せず，高いロバスト性を持つことが分かった．また，状態次元数の増加に対して，RBFNの計算コストは指数オーダーで増加するが，SDNN-Cの計算コストはより低いオーダー（2次のオーダー）に抑えられることが示された．

第5章では制御周期を短くすることができないような課題において数値実験を行い，提案手法の有効性について検証した．その結果，この課題は離散行動では解くことができず，また高い関数近似能力を必要とする課題であることが分かった．そして，このような課題においても，SDNN-Cは従来手法よりも適した（高い関数近似能力をもつ）手法であることが示された．

以上のように，本手法は，連続な状態行動空間におけるQ学習に必要と考えられる要素（**1.2.3**参照）をほぼ全て兼ね備えている．Q学習のアルゴリズムの単純さとあわせて，SDNN-Cを用いたQ学習は，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言える．具体的には，環境の変化が速く，Actor-Critic手法などでは学習が追従できないような課題でも，SDNN-Cを用いたQ学習ならば学習できる可能性がある．また，パラメータ調整にあまりコストをかけられないような課題に対しても，Actor-Critic手法よりパラメータ数が少ないことから，有用だと考えられる．

7.2 今後の課題と展望

本研究で提案したSDNN-Cは，連続状態行動空間におけるQ学習を実現する実用的な価値関数近似器であるが，まだ数多くの問題が残されている．最後に，これらの問題のうち主要なものを列挙するとともに，今後の展望について述べることとする．

第一に，SDNN-Cをより多くの課題に適用し，どのような場合に特に有効なのか明らかにすることが挙げられる．**6.1**で述べたように，SDNN-Cは状態変数が全く同じ場合，行動次元に関する不連続性は表現できないため，このような不連続性が重要になる課題がもし存在するならば，苦手とする可能性がある．一方で，環境の変化が速いような課題では，既存手法やActor-Critic手法などよりも有効な可能性がある．

また，第4章では冗長変数についてのロバスト性についての検証を行ったが，さらに複雑な冗長変数を加えた場合（例えば，高さ情報にホワイトノイズを加えたものなど）や，冗長変数の代わりに主要な変数を除いた場合（例えば，高さ情報

を加える代わりに，θ_1 を除く）など，より実問題に近い条件について，詳細な実験を行いたいと考えている．

第二に，行動空間が多次元の場合へのモデルの拡張が必要である．原理的には行動次元が増えても，出力層の素子群を行動に対応したコードパターンで不感化するという，現在の方法をほぼそのまま適用できると考えられる．しかし，出力層素子を行動空間にあわせて超立方体状に配置すると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，Q 値の最大値や対応する行動を効率よく計算する方法について検討する必要がある．案としては，Gibbs サンプリングを用いる [36, 39] 手法との併用や，分割統治法の利用などが考えられる．

第三に，Actor-Critic 手法など Q 学習以外の方法との比較も重要な課題である．これまで，連続行動を扱える現実的な手法は Actor-Critic しかなかったが，本手法を用いることで，連続行動が必要だが Actor-Critic も苦手とするような問題（前述のように，環境の変化が速く，Actor-Critic では学習が追いつかないような問題）でも制御できる可能性がある．

この他，強化学習の枠からは外れるが，SDNN の関数近似能力自体の解析も重要な課題である．SDNN は関数近似器として優れた性質を多く持っているが，これらの性質がどのように実現されているかは，まだ不明な点が多い．この点に関して，筆者はパターンコーディングと選択的不感化がタイルコーディングの非常に特殊な例に当たり，出力層の分散表現はアンサンブル学習の一手法であるバギング [40] と関連しているのではないかと考えている．SDNN は，もともと神経力学系のモデルに関する研究から発展したものであるが，上記のような機械学習や統計的な手法と関連があるとすれば興味深い．

さらに，将来的には，動物の脳内における強化学習モデルとの比較や，生理学的な知見の検討を行うことで，より動物に近い優れた強化学習システムの構築や，動物の学習モデルの解析に繋がれば興味深い．

本研究が強化学習の進展に貢献し，ひいては制御工学や脳科学等の発展に寄与することを期待して，論文の結びとする．

謝辞

本研究を進めるにあたり，始終熱心な御指導，御鞭撻を賜りました筑波大学システム情報系森田昌彦先生に深く感謝いたします．生体情報処理研究室に配属して以来6年間に渡り，研究のみならず，ものの見方や考え方といった様々なことを学ばせて頂きました．本研究が，完成を迎えることができたのも森田先生の多大なるお力添えあってのことであり，大変感謝しております．

折にふれ，非常に有益なご助言を頂きました，同系安信誠二教授，掛谷英紀准教授に厚く感謝いたします．田中文英准教授には，研究室ゼミでの御指導や，議論などを通じ，多くの知識や示唆を頂戴しました．澁谷長史助教には，ゼミの時間のみならず，講義後などにも議論の時間を設けて頂き，非常に多くの御指導，御助言を賜りました．先生方に心より感謝いたします．さらに，研究や日々の学校生活でお世話になりました，全ての先生方にお礼申し上げます．

研究室の先輩方には，大変お世話になりました．特に，山根健先輩と桑原昭之先輩には，折にふれて多くの御助言を頂きました．皆様に厚く感謝申し上げます．そして，日常の議論を通じて多くの知識や示唆を下さいました，研究室の皆様に感謝いたします．

最後になりましたが，常日頃より著者を支えてくださった家族，友人たちに心より感謝いたします．

参考文献

[1] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[2] V.R. Konda and J.N. Tsitsiklis. On actor-critic algorithms. *SIAM journal on Control and Optimization*, Vol. 42, No. 4, pp. 1143–1166, 2003.

[3] C.J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989.

[4] C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine Learning*, Vol. 8, No. 3-4, pp. 279–292, May 1992.

[5] G.A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report, CUED/F-INFENG/TR 166, 1994.

[6] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pp. 229–256, 1992.

[7] V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, Vol. 3, No. 6, pp. 671–692, 1990.

[8] 木村元, 山村雅幸, 小林重信. 部分観測マルコフ決定過程下での強化学習：確率的傾斜法による接近. 人工知能学会誌, Vol. 11, No. 5, pp. 761–768, Sep. 1996.

[9] R.S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pp. 1057–1063. MIT Press, 2000.

[10] I.H. Witten. An adaptive optimal controller for discrete-time Markov environments. *Information and Control*, Vol. 34, No. 4, pp. 286–295, 1977.

[11] A.G. Barto, R.S. Sutton, and C.W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *Systems, Man and Cybernetics, IEEE Transactions on*, Vol. SMC-13, No. 5, pp. 834–846, Sept 1983.

[12] J. Peters, S. Vijayakumar, and S. Schaal. Natural actor-critic. In *Machine Learning: ECML 2005*, Vol. 3720 of *Lecture Notes in Computer Science*, pp. 280–291. Springer Berlin Heidelberg, 2005.

[13] H. Kimura. Natural gradient actor-critic algorithms using random rectangular coarse coding. In *SICE Annual Conference 2008*, pp. 2027–2034, Aug 2008.

[14] T. Mori, Y. Nakamura, and S. Ishii. Off policy natural actor-critic. *NAIST-IS-TR2005007*, 2005.

[15] C. Gaskett, D. Wettergreen, and A. Zelinsky. Q-learning in continuous state and action spaces. In *Advanced Topics in Artificial Intelligence*, Vol. 1747 of *Lecture Notes in Computer Science*, pp. 417–428. Springer Berlin Heidelberg, 1999.

[16] G.D. Konidaris, S. Osentoski, and P.S. Thomas. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, pp. 380–385, August 2011.

[17] A. Geramifard, M. Bowling, and R.S. Sutton. Incremental least-squares temporal difference learning. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pp. 356–361. AAAI Press, 2006.

[18] G. Taylor and R. Parr. Kernelized value function approximation for reinforcement learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 1017–1024, New York, USA, 2009. ACM.

[19] B. Irie and S. Miyake. Capabilities of three-layered perceptrons. In *Neural Networks, 1988., IEEE International Conference on*, pp. 641–648 vol.1, July 1988.

[20] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, Vol. 2, No. 3, pp. 183–192, 1989.

[21] R.S. Sutton. Reinforcement Learning FAQ: Frequently Asked Questions about Reinforcement Learning. `http://webdocs.cs.ualberta.ca/~sutton/RL-FAQ.html` (2015-01-03 参照).

[22] M. McCloskey and N.J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H. Bower, editor, *Psychology of Learning and Motivation*, Vol. 24, pp. 109–165. Academic Press, 1989.

[23] J. Park and I. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, Vol. 3, No. 2, pp. 246–257, June 1991.

[24] 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. 電子情報通信学会論文誌. D-II, 情報・システム, II-パターン処理, Vol. 87, No. 12, pp. 2242–2252, Dec. 2004.

[25] 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, 2008.

[26] 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. 電子情報通信学会論文誌. D, 情報・システム, Vol. 94, No. 12, pp. 2114–2125, Dec 2011.

[27] K. Horie, A. Suemitsu, and M. Morita. Direct estimation of hand motion speed from surface electromyograms using a selective desensitization neural network. *Journal of Signal Processing*, Vol. 18, No. 4, pp. 225–228, 2014.

[28] 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. 電子情報通信学会論文誌. D, 情報・システム, Vol. 93, No. 6, pp. 837–847, Jun 2010.

[29] 小林高彰, 澁谷長史, 森田昌彦. 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. 電子情報通信学会論文誌. D, 情報・システム, Vol. 98, No. 2, pp. 287–299, Feb. 2015.

[30] M.W. Spong. The swing up control problem for the acrobot. *Control Systems, IEEE*, Vol. 15, No. 1, pp. 49–55, Feb 1995.

[31] G. Boone. Minimum-time control of the acrobot. In *1997 IEEE International Conference on Robotics and Automation*, Vol. 4, pp. 3281–3287. IEEE, 1997.

[32] 森本淳, 銅谷賢治. 強化学習を用いた高次元連続状態空間における系列運動学習：起き上がり運動の獲得. 電子情報通信学会論文誌. D-II, 情報・システム, II-パターン処理, Vol. 82, No. 11, pp. 2118–2131, Nov. 1999.

[33] T. Kobayashi, T. Shibuya, and M. Morita. Q-learning in continuous state-action space with redundant dimensions by using a selective desensitization neural network. In *Proceedings of SCIS&ISIS 2014*, pp. 801–806, Dec. 2014.

[34] 鮫島和行, 大森隆司. 強化学習における適応的状態空間構成法. 日本神経回路学会誌, Vol. 6, No. 3, pp. 144–154, 1999.

[35] José del R. Millán, Daniele Posenato, and Eric Dedieu. Continuous-action Q-learning. *Machine Learning*, Vol. 49, No. 2-3, pp. 247–265, 2002.

[36] 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.

[37] J. Pazis and M.G. Lagoudakis. Reinforcement learning in multidimensional continuous action spaces. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, pp. 97–104, April 2011.

[38] S. Carden. Convergence of a Q-learning variant for continuous states and actions. *J. Artif. Intell. Res.(JAIR)*, Vol. 49, pp. 705–731, 2014.

[39] B. Sallans and G.E. Hinton. Reinforcement learning with factored states and actions. *J. Machine Learning Research*, Vol. 5, pp. 1063–1088, Dec 2004.

[40] L. Breiman. Bagging predictors. *Machine Learning*, Vol. 24, No. 2, pp. 123–140, 1996.

著者論文

査読付き論文

1. 小林高彰，澁谷長史，森田昌彦，“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，”電子情報通信学会論文誌 (D), Vol.J98-D, No.2, pp.287–299, 2015.

2. 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.

技術報告

3. 丹野智博，堀江和正，小林高彰，森田昌彦，“ニューラルネットによるパターン分類におけるパターンコーディングの効果，”電子情報通信学会技術研究報告 (NC)，vol.113, no.111, pp.139–143, 2013.

4. 小林高彰，澁谷長史，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連続状態行動空間における Q 学習，”電子情報通信学会技術研究報告 (NC)，vol.111, no.96, pp.119–123, 2011.