NAIST-IS-MT0651132

修士論文

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

山口 明彦

2008年 2月 7日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である.

山口 明彦

審査委員：

小笠原 司 教授	（主指導教員）
川人 光男 教授	（副指導教員）
杉本 謙二 教授	（副指導教員）
神谷 之康 准教授	（副指導教員）
松本 吉央 客員准教授	（副指導教員）

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

山口 明彦

内容梗概

運動学習によって動作の目的のみからそれを実現する動作を自律的に獲得するとき，動作がジャンピング・サーブのように複雑になると膨大な学習コスト（転倒によるダメージなど）や学習時間が掛かる．これを人間のように効率的に行うには，過去に学習した動作知識の「再利用」が不可欠である．本研究では行動価値関数の分離学習によって強化学習に再利用メカニズムを導入することを提案する．これによりタスク非依存だと考えられる転倒回避などの「回避行動」が本来のタスクから分離して獲得され，回避行動を学習した行動価値関数が新たなタスクの学習で再利用されるようになる．さらにこの手法をロボットの全身運動学習に応用する．手法の有効性を示すために格子状空間・2 次元連続平面での移動タスク，及び 4 リンクロボットによるサーブ動作・跳躍の学習に適用し，シミュレーション実験を行った．この結果，移動タスクでは回避行動の再利用により新たなタスクの学習で取るべきでない行動が大幅に避けられるようになり，全身運動学習では跳躍で分離学習した転倒回避行動を再利用してサーブを学習することで，再利用しない場合に比べて学習中の転倒ダメージの合計が半減した．

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

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 修士論文, NAIST-IS-MT0651132, 2008 年 2 月 7 日.

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

Akihiko Yamaguchi

Abstract

In acquiring a motion only from its objective by learning, large cost such as damage from falling over and a large number of trials are required if the motion is a complex one, such as a jumping serve. Reusing the knowledge already learnt is an essential mechanism to learn such motions efficiently, like humans do. In this study, we propose to use a decomposition of action-value functions as a reusing mechanism for reinforcement learning. *Avoidance actions* that are assumed invariant across different tasks (e.g. avoiding to fall over) are learnt separately from primary actions assumed to be task specific. Then the action-value function for the avoidance actions is reused in learning new tasks. Furthermore, we extend the method for multi-link robots to learn whole body motions. This learning method has been applied for moving tasks both in discrete and continuous planes, and has been also applied for a tennis-serve and a jump tasks of a 4-link robot. The simulation results have demonstrated that in the moving tasks, reusing avoidance actions enables the agent to avoid the bad actions effectively. In learning whole-body motions, reusing avoidance actions obtained in learning a jump reduces to half the total falling damage in learning a serve compared to learning without reusing.

Keywords:

motion learning, reinforcement learning, reusing, avoidance actions, jump, tennis serve

*Master's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT0651132, February 7, 2008.

目　　次

1. 序論 **1**

2. 分離学習による回避行動の再利用 **4**

2.1 強化学習における本来のタスクの報酬と回避行動の報酬 4

2.2 再利用の対象となる報酬 5

2.3 行動価値関数の分離学習と再利用 5

2.3.1 タスクの最適行動価値関数 6

2.3.2 再利用可能な行動価値関数 6

2.3.3 分離学習と再利用の流れ 6

2.4 分離した行動価値関数からの行動生成 7

2.5 分離学習と再利用のための行動価値関数学習手法 7

2.5.1 Sarsa, Q-learning: 分離学習と再利用における問題点 8

2.5.2 ΔSarsa: Sarsa をもとにした再利用のための分離学習 9

2.5.3 DQL+: Q-learning をもとにした再利用のための分離学習 . 11

2.6 格子状空間における移動タスク 12

2.6.1 分離学習の検証 13

2.6.2 回避行動の再利用の検証 15

3. 連続状態空間における分離学習と再利用 **17**

3.1 行動価値関数の関数近似 17

3.2 エリジビリティ・トレースを用いた行動価値関数の学習 18

3.3 2 次元連続平面における移動タスク 19

3.3.1 分離学習の検証 20

3.3.2 回避行動の再利用の検証 22

4. 多関節ロボットの全身運動学習への応用 **24**

4.1 運動学習アーキテクチャ 24

4.2 運動学習の流れ 25

4.3 シミュレーション実験 25

4.3.1 運動学習における回避行動 27

4.3.2 サーブと回避行動の分離学習 27

4.3.3 遅いサーブと速いサーブの間の再利用 31

5. 分離学習・再利用のための関数近似器修正法 **33**
5.1 分離学習と再利用における関数近似器の問題 33
5.2 「分離誤差」に基づく関数近似器修正法 34
5.3 関数近似器修正法を組み込んだ分離学習と再利用の流れ 35
5.4 2 次元連続平面における移動タスク（2） 35
5.4.1 分離学習の検証 37
5.4.2 回避行動の再利用の検証 39
5.5 運動学習のシミュレーション 41
5.5.1 跳躍 41
5.5.2 跳躍とサーブの間の再利用 45
5.5.3 遅いサーブと速いサーブの間の再利用（2） 46
6. 結論 **48**
謝辞 **50**
付録 **53**
A. 式（*2.17*）⇒ 式（*2.11*）の証明 **53**
B. エリジビリティ・トレースを用いた更新則 **54**
B.1 ΔSarsa(λ_0, λ_1) 54
B.2 DQL+(λ_0, λ_1) 54
C. ダイナミクスの学習 **55**
D. ロボットの行動 **56**
E. 分離誤差に基づく基底関数修正法の詳細 **57**
E.1 分離誤差を評価する状態集合 57
E.2 基底関数の追加方法 57
E.3 実行のタイミング 58

図　目　次

1.1 本来のタスクと回避行動の報酬分離による再利用 2
2.1 格子状空間における移動タスク 13
2.2 ————実験結果：各アルゴリズムによる分離学習 14
2.3 ————実験結果：各アルゴリズムによる再利用 16
3.1 2 次元連続平面における移動タスク 19
3.2 ————実験結果：各アルゴリズムによる分離学習 21
3.3 ————実験結果：各アルゴリズムによる再利用 23
4.1 全身運動を学習するアーキテクチャ 25
4.2 シミュレーションに用いた 4 リンク 3 関節のロボット 26
4.3 正規化された力の和 d とそれに対するダメージ D の変化の一例 . . 27
4.4 サーブと回避行動の分離学習結果 29
4.5 獲得されたサーブ 30
4.6 遅いサーブと速いサーブの間の再利用結果 32
5.1 分離学習における基底関数の問題と改善方法の例 33
5.2 2 次元連続平面における移動タスク（2） 36
5.3 基底関数修正法を適用して得られた基底関数 36
5.4 基底関数の修正法による分離学習結果（2 次元連続平面） 38
5.5 基底関数の修正法による再利用結果（2 次元連続平面） 40
5.6 土台リンクの水平制御を行った場合の跳躍の分離学習結果（基底関数修正法を導入） 42
5.7 獲得された跳躍 43
5.8 土台リンクの水平制御を行わない場合の跳躍の分離学習結果（基底関数修正法を導入） 44
5.9 跳躍とサーブの間の再利用結果 45
5.10 遅いサーブと速いサーブの間の再利用結果（基底関数修正法を適用） 47

表　目　次

5.1 遅いサーブと速いサーブの間の再利用結果 47

1. 序　　論

ヒューマノイドに代表される高機能な多関節ロボットは設計者によって“作り込まれた”歩行や跳躍などの様々な動作を身に付けているが，設計者の想定を超える例外的な事態に対応したり組み込まれた機能以上の行動をするために“目的から実現方法を自律的に獲得する技術”が不可欠である．そのような技術のひとつとして強化学習が研究されており，ロボットの運動学習にも応用されている [1]〜[5].

強化学習の最大の課題のひとつとして複雑な行動を獲得させようとしたり高次元の状態空間を扱ったときに学習に膨大なコスト（転倒によるダメージなど）や時間が掛かることがあげられ，ロボットの運動学習に適用されている強化学習の多くはロボットやタスクに特化したモデルを用いたり [3]〜[5]，階層構造を導入したり [1] [6] [7] 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.

これらの手法は状態空間が高次元であることへの対処であるが，複雑な行動の獲得には既に学習した動作知識を新たなタスクの学習で「再利用」することによってコスト軽減や学習時間の短縮を図るアプローチも有効である．例えば人間がテニスのジャンピング・サーブを学習するときには転倒の回避や跳躍と言った動作知識は既に獲得されており，それらを再利用しながら効率的に学習を行っていると考えられる．仮にこのように複雑な動作をゼロから学習すれば膨大なコストや学習時間を必要とすると考えられ，再利用のメカニズムは不可欠であると言える．しかしながら強化学習によるロボットの運動学習において，再利用に焦点を絞った研究（例えば [3] [8]）はそれほど多くない．この理由としてはどの動作知識をいつどのように再利用するか推論する必要があり，多くの場合手法がタスク依存になって強化学習が持つ汎用性を損なうからなどが考えられる.

しかしながら再利用される動作知識を

Type A. すべての目的に共通の動作知識．例えば転倒回避や人を傷つけないと言った「回避行動」など.

Type B. 再利用する場面が限定される動作知識．例えば走行や跳躍など.

の 2 種類に分けたとき，再利用するために困難な推論が必要であるのは Type B の動作知識である．そこで本研究では汎用性を損なわずに実現できてかつ再利用され

ることが重要である「回避行動」の再利用メカニズムを強化学習に導入する．

具体的には報酬を本来のタスクの報酬と回避行動の報酬のふたつに分けて別々に行動価値関数を学習し，新たなタスクの学習で回避行動の行動価値関数を再利用するというアプローチを取る（図 1.1）．したがって回避行動を適切に分離することが重要であり，特にロボットの運動学習のように連続状態空間では関数近似器の構成が回避行動の分離に関わる最大の要素のひとつとなる．そこで本研究では（1）強化学習において分離学習により回避行動を再利用するメカニズムを提案，(2）ロボットの運動学習に応用し，(3）回避行動の分離学習及び再利用のための関数近似器修正法を提案する．さらに（4）これらの手法を格子状空間及び 2 次元連続平面における移動タスク，運動学習の例としてサーブと跳躍に適用し，回避行動の再利用がどの程度汎用的に実現できるかを検証する．

Fernández らは過去の学習で獲得した方策を新たなタスクの学習で確率的に再利用する方法を提案しており [8]，これは“確率的再利用アプローチ”であると言える．これに対し本研究の手法は分離学習した回避行動がほかのタスクでも常に使用できることを想定した“決定論的再利用アプローチ”である．この想定が正しい場合は後者が前者より優れていると考えられる．Zhang ら は 2 本指のグリッパによる複雑な物体の把持タスクを把持する方向の学習器と把持する位置の学習器に分解して学習し，方向の学習器が物体が変わったときにも共通に使えるとして再利用した [3].

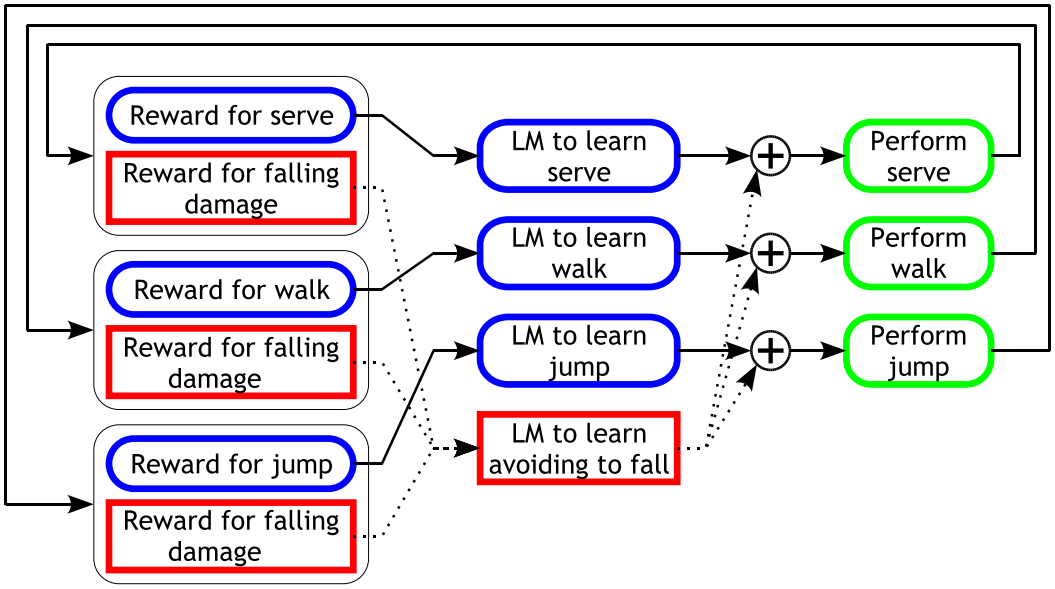


図 1.1 本来のタスク（サーブ，歩行，跳躍）と回避行動（転倒）の報酬分離による再利用． LM は学習器 (Learning Module) を表す．

さらに新しく把持しようとする物体の形状と既に学習した物体の形状を比較することで既存の位置の学習器を再利用する方法も述べており，これは前述した推論の一種であると見られる．この研究はタスクを把持に特化しているものの，実ロボットにおいて学習した知識の再利用を試みた研究として興味深い．

本研究で実現する再利用はすべての目的に共通の動作知識（Type A）の再利用であるが，これに再利用する場面が限定される動作知識（Type B）の再利用メカニズムが加わることで“学習すればするほど知識が蓄積されて知的に行動するようになるロボット”が実現される．これは今後，要求や状況が多様に変化する我々の生活の中にロボットが入って行動を共にするような場合や宇宙空間や地球外惑星などにおいて自律的に行動するような場合に不可欠な技術であり，提案手法はその基礎となるものである．

本論文の構成

2章では離散状態・離散行動の学習システムを想定して回避行動の分離学習と再利用の目的を定式化し，それを実現するための学習手法を提案する．ロボットの状態空間は一般的に連続であり行動価値関数を近似する必要があるため，3章では提案手法を連続状態に拡張する方法について述べ，ロボットの運動学習に比べて簡単な2次元連続平面の移動タスクに適用し有効性を検証する．この結果を踏まえて4章では多関節ロボットの全身運動学習へ応用する手法を示し，サーブのシミュレーション実験による比較を行う．さらに5章ではより難易度の高いタスクに適用する上での関数近似器の問題を述べ，これを改善するための関数近似器修正法を提案，跳躍などのタスクで有効性を検証する．最後に6章でまとめる．

2. 分離学習による回避行動の再利用

本章では有限の離散状態・離散行動の学習システムを想定して回避行動の分離学習と再利用の目的を定式化，これらの目的を達成する上で既存の学習手法が問題となることを示し，分離学習と再利用のための学習手法を提案する．さらに格子状空間の移動タスクにおいて提案手法と既存の学習手法を比較し，有効性を検証する．

2.1 強化学習における本来のタスクの報酬と回避行動の報酬

強化学習 [9] [10] とは「報酬」と「状態」を入力,「行動」を出力とする学習システム（以下エージェントと呼ぶ）が，ある状態において期待される「割引収益」を最大化する行動を選択できるようになることを目的とする学習の枠組である．エージェントの行動選択は状態 $s \in \mathcal{S}$（有限の離散集合とする）から行動 $a \in \mathcal{A}$（同様）への写像であり，これを「方策」と呼んで $a = \pi(s)$ で表す．報酬 $R(s,a) \in \mathbb{R}$ とはエージェントがある状態 s で行動 a を取ったときに与えられる行動の評価（例えば跳躍なら重心の高さなど）であり，割引収益とは将来期待される報酬の総和を割引率 $\gamma\,(0 \leqslant \gamma < 1)$ で割り引いたもの（$\sum_{k=1}^{\infty} \gamma^{k-1} R(s_{t+k}, a_{t+k})$，$t$ は現在時刻）である．したがってこの割引収益がエージェントの方策に対する評価関数であり，これを最大にするような方策を獲得する汎関数最適化が強化学習の目的である．方策の最適解を「最適方策」と呼ぶ．なお以下では最適方策の存在を保証するためにタスクのマルコフ性を仮定する [9]．$\mathcal{S}$ と $\mathcal{A}$ を有限の離散集合としているのも同様の理由による．

このように強化学習ではタスクの目的は報酬によって与えられる．この報酬設計として，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．この場合報酬関数 $R(s,a)$ 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ている．回避行動の報酬関数がほかのタスクでも共通であるなら，これを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．以下では報酬の分離学習によって回避行動を再利用するメカニズムを強化学習に導入する．なおここで想定しているタスクは跳躍，サーブなどのエピソード的タスク（開始と終了があるタスク）である．

2.2 再利用の対象となる報酬

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 [3]）ことができなくなる．しかしこのような報酬は適当なオフセットを加えるなどして正の報酬による表現に切替えることが容易な場合が多く，強化学習の持つ汎用性を大きく損なわせるものではないと考えられる．さらに正負の報酬表現には

（1） 2.5.3 項で述べる Q-learning をもとにした行動価値関数の分離学習アルゴリズム（DQL+）の設計が容易になる．
（2） 行動価値関数を正または負に拘束できるため，次章で述べる連続状態空間における学習の数値解析的な安定さを増すことができる．
（3） 行動選択に関わるパラメタ（2.4 節で述べるボルツマン選択の温度パラメタ τ など）の決定が比較的容易になる．

などの利点があり回避行動の再利用がしやすくなるため，以下では正負の報酬表現を前提とする．

ただし本来のタスクの成功に与えられる正の報酬だけでは解が多数存在することが多く，このため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本来のタスクの報酬に加算することが望ましい．これにより本来のタスクの報酬が負になることがあるが，正の報酬がステップコストに比べて十分大きい場合 2.5 節で述べる更新則を用いれば問題は生じない．

2.3 行動価値関数の分離学習と再利用

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を $R_0(s,a)$，回避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．具体的にはこのようなアプローチのうち最もシンプルなものとして，個々の報酬関数に対して行動価値関数 $Q_i(s,a)$, $i \in \{0, 1\}$ を用意し（Q_0 は本来のタスク R_0 を学習し Q_1 は回避行動 R_1 を学習する），その総和 $Q(s,a) \triangleq \sum_i Q_i(s,a)$ によって学習器の合成を実現するという方法を検討する．

2.3.1 タスクの最適行動価値関数

行動価値関数とはエージェントがある方策 π に従っているときに，ある状態 s においてある行動 a を取った場合の割引収益の期待値である．合成した行動価値関数 Q はベルマンの方程式

$$Q^{\pi}(s,a) = \sum_{i\in\{0,1\}} R_i(s,a) + \gamma \sum_{s'\in\mathcal{S}} P_{ss'}(a) Q^{\pi}(s', \pi(s')) \tag{2.1}$$

によって再帰的に定義される．ここで $P_{ss'}(a)$ 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，$^{\pi}$ は行動価値関数が方策 π に従った場合のものであることを表す．「グリーディ方策」$\pi(s) = \arg\max_{a\in\mathcal{A}} Q(s,a)$ が最適方策であるような行動価値関数を，以下では「タスクの最適行動価値関数」と呼ぶことにし $Q^{\star}$ で表す：

$$Q^{\star}(s,a) = \sum_{i\in\{0,1\}} R_i(s,a) + \gamma \sum_{s'\in\mathcal{S}} P_{ss'}(a) \max_{a'\in\mathcal{A}} Q^{\star}(s', a'). \tag{2.2}$$

分離した行動価値関数の総和 $Q(s,a) = \sum_i Q_i(s,a)$ が最適行動価値関数に一致するように学習を行うことができれば，分離学習においてタスクを実現する最適方策を獲得したことになる．

2.3.2 再利用可能な行動価値関数

回避行動を学習する行動価値関数 Q_1 が「再利用可能である」とは，Q_1 が報酬 R_1 にのみ依存し R_0 に依存しないことを意味する．したがって

$$Q_1^{\pi_1}(s,a) = R_1(s,a) + \gamma \sum_{s'\in\mathcal{S}} P_{ss'}(a) Q_1^{\pi_1}(s', \pi_1(s')) \tag{2.3a}$$

$$\pi_1(s) = \arg\max_{a\in\mathcal{A}} Q_1^{\pi_1}(s,a) \tag{2.3b}$$

を再利用可能な行動価値関数 Q_1 の定義とする．

2.3.3 分離学習と再利用の流れ

このとき回避行動を学習する Q_1 は以下のような流れで新たなタスクの学習において再利用される．

（1） あるタスクの学習（タスク固有の報酬関数 R_0，回避行動の報酬関数 R_1）で Q_0, Q_1 を分離学習する．

（2） 新たなタスクの学習（タスク固有の報酬関数 R_0'，回避行動の報酬関数 R_1）では Q_0' はすべてのパラメタが何も学習していない初期状態から開始し，Q_1 は（1）の学習結果を初期値として学習する．

2.4 分離した行動価値関数からの行動生成

エージェントの行動選択は合成した行動価値関数 $Q(s,a) = \sum_i Q_i(s,a)$ に基づいて行われる．再利用時には本来のタスクを学習する行動価値関数 Q_0 はゼロに初期化されるが，既に回避行動を学習した Q_1 はそのまま再利用されるため合成した行動価値関数の回避行動は負の値を示す．この情報をもとにエージェントは新たなタスクを達成する行動を探索するため，負の行動価値を示す行動が避けられるような行動選択の方法が必要である．このような要求を満たす行動選択の方法のひとつに，ある状態における各行動価値に基づいて選択確率を決定するボルツマン選択 [9] がある．すなわち確率

$$P(a|s) = \frac{\exp\left(\frac{1}{\tau}Q(s,a)\right)}{\sum_{a' \in \mathcal{A}} \exp\left(\frac{1}{\tau}Q(s,a')\right)} \tag{2.4}$$

で行動 a を選択する．ただし温度パラメタ τ は，学習が進むとグリーディ方策に近付くように $\tau = \tau_0 \mathrm{e}^{-\delta_\tau N_{\mathrm{eps}}}$ によって減少させている[1]．τ_0 は初期値，δ_τ は逆時定数，N_{eps} はエピソード数である．再利用時に効率的に回避行動を避けるため，τ_0 の決定は重要であり，設計した回避行動の報酬から概算する．

2.5 分離学習と再利用のための行動価値関数学習手法

行動価値関数の学習方法には Q-learning [12] や Sarsa [13] があり， Russell ら [14] は複数の報酬に対して行動価値関数を分離して学習する場合 Q-learning だと $R_0(s,a)$ と $R_1(s,a)$ が同時に値を持つような状況下などでは最適な行動が学習できず，そのような状況下でも最適な行動が学習できる Sarsa を用いるべきだと主張，Sarsa による分離学習を Q-decomposition として提案した．しかし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ため，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．このように Q-learning や Sarsa による学習はいずれも分離学習または再利用において問題があるため，以下ではまずこれらの問題を具体的に述べ，続いて Sarsa をもとにした分離学習手法と Q-learning をもとにした分離学習手法を提案する．

[1] これは強化学習でしばしば用いられるテクニックで， Sarsa が最適行動価値関数を獲得するために必要であったり，連続状態空間で Q-learning を行う際に関数近似器の使用による発散 [11] を抑えるのに有効 [9] であるという利点がある．

2.5.1 Sarsa, Q-learning: 分離学習と再利用における問題点

Sarsa の問題点

Sarsa を個々の報酬関数 $R_i(s,a)$, $i \in \{0, 1\}$ に対する行動価値関数の更新に用いる Russell らの Q-decomposition の場合，その更新則は

$$Q_{i,t+1}(s_t, a_t) \leftarrow Q_{i,t}(s_t, a_t) + \alpha\big\{R_i(s_t, a_t) + \gamma Q_{i,t}(s_{t+1}, a_{t+1}) - Q_{i,t}(s_t, a_t)\big\} \tag{2.5}$$

となる．ただし γ は割引率，α はステップサイズパラメタである．この更新則を用い，状態・行動が十分に経験されてかつ方策 $\pi(s)$ が合成された行動価値関数 $Q(s,a) = \sum_i Q_i(s,a)$ のグリーディ方策に収束する場合，$Q(s,a)$ はベルマンの最適方程式

$$Q^\star(s,a) = \sum_{i\in\{0,1\}} R_i(s,a) + \gamma \sum_{s'\in\mathcal{S}} P_{ss'}(a) Q^\star(s', \pi(s')) \tag{2.6}$$

で定義される行動価値関数に収束することが証明されている [14]．これはタスクの最適行動価値関数（式（*2.2*））に一致するため，式（*2.5*）による分離学習によってタスクの報酬 $\sum_i R_i(s,a)$ に対する最適な行動が獲得される．

合成した行動価値関数が式（*2.6*）に収束する証明には，個々の行動価値関数が

$$Q_i^\pi(s,a) = R_i(s,a) + \gamma \sum_{s'\in\mathcal{S}} P_{ss'}(a) Q_i^\pi(s', \pi(s')), \quad \text{for } i \in \{0, 1\} \tag{2.7}$$

に収束するという定理が利用されている[2]．これより Q_i はタスクの方策 π に依存したものであり式（*2.3*）に一致せず，回避行動を学習する Q_1 はほかのタスクと共通でないため再利用することはできないことがわかる．以上より Sarsa による分離学習（式（*2.5*））は，タスクの学習は可能だが再利用には適さない．

Q-learning の問題点

Q-learning を個々の報酬関数 $R_i(s,a)$, $i \in \{0, 1\}$ に対する行動価値関数の更新に用いた場合，その更新則は

$$Q_{i,t+1}(s_t, a_t) \leftarrow Q_{i,t}(s_t, a_t) + \alpha\big\{R_i(s_t, a_t) + \gamma \max_{a\in\mathcal{A}} Q_{i,t}(s_{t+1}, a) - Q_{i,t}(s_t, a_t)\big\} \tag{2.8}$$

[2]式（*2.7*）を i について足し合わせると式（*2.6*）に等しくなる．

となる．この更新則を用い，状態・行動が十分に経験された場合，個々の行動価値関数はベルマンの最適方程式

$$Q_i^\star(s,a) = R_i(s,a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a) \max_{a' \in \mathcal{A}} Q_i^\star(s',a'), \quad \text{for } i \in \{0,1\} \tag{2.9}$$

に収束する [15]．式（2.9）を $i = 0, 1$ について足し合わせると

$$\sum_{i \in \{0,1\}} Q_i^\star(s,a) = \sum_{i \in \{0,1\}} R_i(s,a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a) \sum_{i \in \{0,1\}} \max_{a' \in \mathcal{A}} Q_i^\star(s',a') \tag{2.10}$$

が得られる．ここから

$$\forall s \in \mathcal{S}: \quad \sum_{i \in \{0,1\}} \max_{a \in \mathcal{A}} Q_i^\star(s,a) = \max_{a \in \mathcal{A}} \sum_{i \in \{0,1\}} Q_i^\star(s,a) \tag{2.11}$$

が成立するとき式（2.10）は

$$Q^\star(s,a) = \sum_{i \in \{0,1\}} R_i(s,a) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a) \max_{a' \in \mathcal{A}} Q^\star(s,a') \tag{2.12}$$

となって，合成された行動価値関数 $Q(s,a) = \sum_i Q_i(s,a)$ がタスクの報酬 $\sum_i R_i(s,a)$ に対する最適行動価値関数に収束することがわかる．しかしながら式（2.11）は一般的な報酬に対して成立しないため，Q-learning による分離学習式（2.8）はタスクの学習を行えない（合成した行動価値関数が式（2.2）に一致しない）．ただしこの分離学習によって得られる回避行動の行動価値関数 Q_1（式（2.9））は再利用可能な行動価値関数（式（2.3））に一致するという利点がある．

2.5.2 ΔSarsa: Sarsa をもとにした再利用のための分離学習

Q-learning によって獲得される行動価値関数は再利用可能だが合成したものはタスクの最適行動価値関数へ収束せず，Sarsa によって獲得される行動価値関数は合成したものがタスクの最適行動価値関数へ収束するがタスク依存で再利用できない．そこで回避行動の行動価値関数を “Q-learning の更新則（式（2.8））に従う要素” と “Sarsa で得られる行動価値関数と Q-learning で得られる行動価値関数の差を表す要素” に分解して学習する．すなわち $Q_1 - \Delta Q_1$ は Sarsa の更新則に従うが，Q_1 は Q-learning の更新則に従うように ΔQ_1 の更新則を定める．一方 Q_0 は Sarsa で更新する．このような学習を行うことで合成した行動価値関数 $Q_0 + Q_1 - \Delta Q_1$ はタス

クの最適行動価値関数に収束することが保証され，かつ回避行動を Q-learning で学習する Q_1 は再利用可能となる．

$Q_1 - \Delta Q_1$ をひとつの行動価値関数だと考えて Sarsa の更新則を適用すると

$$\begin{aligned}(Q_{1,t+1}(s_t,a_t) - \Delta Q_{1,t+1}(s_t,a_t)) \leftarrow (Q_{1,t}(s_t,a_t) - \Delta Q_{1,t}(s_t,a_t)) \\ + \alpha\big\{R_1(s_t,a_t) + \gamma(Q_{1,t}(s_{t+1},a_{t+1}) - \Delta Q_{1,t}(s_{t+1},a_{t+1})) \\ - (Q_{1,t}(s_t,a_t) - \Delta Q_{1,t}(s_t,a_t))\big\}\end{aligned} \tag{2.13}$$

となる．これを Q_1 が Q-learning の更新則（式（2.8））に従うように変形することで，ΔQ_1 の更新則が

$$\begin{aligned}\Delta Q_{1,t+1}(s_t,a_t) \leftarrow \Delta Q_{1,t}(s_t,a_t) + \alpha\big\{\gamma \max_{a\in\mathcal{A}} Q_{1,t}(s_{t+1},a) \\ - \gamma(Q_{1,t}(s_{t+1},a_{t+1}) - \Delta Q_{1,t}(s_{t+1},a_{t+1})) - \Delta Q_{1,t}(s_t,a_t)\big\}\end{aligned} \tag{2.14}$$

と導出される．なお Q_0 には Sarsa の更新則（式（2.5））を適用する．

このようにして学習される ΔQ_1 は，回避行動の報酬 R_1 に対する行動価値関数からタスク依存の要素を切り離して学習していることになる．この学習法を用いる利点は，分離学習によってタスクの最適行動価値関数を得られることと回避行動の行動価値関数を再利用できることである．逆にこの学習法の欠点は ΔQ_1 の学習が増えることによるメモリ量及び計算コストの増加である．しかしながらタスクに依存する Q_0 と ΔQ_1 をひとつの行動価値関数とみなすことでメモリ量の増加は解決される．すなわち

$$Q_{\text{task}}(s,a) \triangleq Q_0(s,a) - \Delta Q_1(s,a) \tag{2.15}$$

と定義すると，Q_{task} の更新則は式（2.5），（2.14）から

$$\begin{aligned}Q_{\text{task},t+1}(s_t,a_t) \leftarrow Q_{\text{task},t}(s_t,a_t) + \alpha\Big\{R_0(s_t,a_t) + \gamma Q_{\text{task},t}(s_{t+1},a_{t+1}) \\ - \gamma\big(\max_{a\in\mathcal{A}} Q_{1,t}(s_{t+1},a) - Q_{1,t}(s_{t+1},a_{t+1})\big) - Q_{\text{task},t}(s_t,a_t)\Big\}\end{aligned} \tag{2.16}$$

となる．この Q_{task} を改めて Q_0 とおき直すことで，分離学習の達成（合成した行動価値関数がタスクの最適行動価値関数となる）と再利用可能な行動価値関数の獲得（Q_1 がタスクに依存しない）の両者が実現できる．この更新則を以下では ΔSarsa と呼ぶ．

2.5.3 DQL+: Q-learning をもとにした再利用のための分離学習

前述したように Q-learning で個々の報酬を学習した場合，学習される行動価値関数が式（*2.11*）を満たさない限り合成した行動価値関数がタスクの最適行動価値関数とならない．式（*2.11*）が成立するのはある状態における個々の行動価値が同時に 0 以外の値を持たない場合などであるから（詳細は後述），逆に式（*2.11*）が成り立たないのは本来のタスクの報酬と回避行動の報酬の間に「トレードオフ」が存在するような場合である．ここではトレードオフが存在するような場合には回避行動を優先するように行動価値関数の更新則を定め，式（*2.11*）が近似的に満たされるようにする．例えば転倒ダメージが大きいが高く跳べるような行動は転倒ダメージの大きさを優先して選択しない．これによってトレードオフが存在する場合にも，合成した行動価値関数を用いてタスクを近似的に達成する行動が生成できるようになる．

トレードオフが存在しない場合の行動価値関数は，具体的には

$$\begin{aligned} &{}^{\forall}(s,a) \in \mathcal{S} \times \mathcal{A} : \\ &\prod_{i \in \{0,1\}} Q_i^\star(s,a) = 0 \ \wedge \ Q_0^\star(s,a) \geqslant 0 \ \wedge \ Q_1^\star(s,a) \leqslant 0 \end{aligned} \tag{2.17}$$

と書かれる．これは回避行動が本来のタスクを実現する行動と完全に分離される場合，すなわち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 0 以外の値を持たない場合である．この条件を満たす行動価値関数に対しては式（*2.11*）が成立するので（証明は付録 A），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．

行動価値関数が式（*2.17*）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) 回避行動の報酬を受け取り，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 $\max_{a \in \mathcal{A}} Q_{0,t}(s_{t+1}, a) > 0$ の場合，式（*2.8*）によって $Q_{0,t+1}(s_t, a_t)$ は正に更新される一方で $Q_{1,t+1}(s_t, a_t)$ 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．

そこで式（*2.8*）の更新に使う報酬を

$$R'_0(s_t, a_t) \leftarrow R_0(s_t, a_t) + wR_1(s_t, a_t) \tag{2.18a}$$

$$R'_1(s_t, a_t) \leftarrow R_1(s_t, a_t) \tag{2.18b}$$

のように変更し（w は適当な正の定数[3]），さらに式（*2.8*）の更新を行った後に

$$\texttt{if} \quad Q_{0,t+1}(s_t, a_t) < 0: \quad Q_{0,t+1}(s_t, a_t) \leftarrow 0 \tag{2.19a}$$

$$\texttt{if} \quad Q_{1,t+1}(s_t, a_t) > 0: \quad Q_{1,t+1}(s_t, a_t) \leftarrow 0 \tag{2.19b}$$

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（*2.18a*）を用いると $|wR_1|$ が $|R_0|$ より十分に大きい場合 $Q_{0,t+1}(s_t, a_t) \leqslant 0$ となり，式（*2.19a*），（*2.19b*）によって式（*2.17*）が満たされるようになる．これはトレードオフが存在する場合には回避行動を優先する，すなわち $Q_{0,t+1}(s_t, a_t) = 0$ となるように学習していることになる．なお $R_0(s_t, a_t) \geqslant 0$, $R_1(s_t, a_t) = 0$ の場合は行動価値関数の学習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（*2.11*）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転倒などの回避されるべき行動は優先的に避けられることが望ましいため，上記更新則で実用に耐え得ると考えられる．この更新則を以下では DQL+と呼ぶ．

2.6 格子状空間における移動タスク

以上に述べた更新則 ΔSarsa, DQL+が分離学習及び回避行動の再利用において有効に機能することを確認するため，図 2.1 に示す 7×10 の格子状空間における移動タスクに適用する．このタスクはロボットの運動学習への応用を意識したもので，行動は上下左右，スタート状態（“start”）からゴール状態（“goal”）に移動すると本来のタスクの報酬 $R_0 = 1$ が与えられるというタスクである．一方で転倒などに相当する回避すべき状態（“bad state”）があり，この状態に行くと大きな負の報酬 $R_1 = -10$ が与えられる．さらに「風」（“wind”）がエージェントの行動を変化させ（小さい矢は $+1$，大きい矢は $+2$ だけ行動を変化させる），必ず “bad state” に遷移してしまう状態が存在する．これは転倒しかけの状態を想定している．さらに “bad state” の上部（“trap”）は $R_1 = -10$ と同時に $R_0 = +10$ も与えられる状態であり，2.5.3 項で述べたトレードオフが存在する（2.5.3 項の（a）に該当）．このため単純に Q-learning で分離学習を行うと格子上部をうろつく局所解に収束してしまう．局所解に陥った場合を考慮して，エージェントの行動数が 1000 を超えた場合はエピソードを終了する．なおエージェントの 1 回の行動につき -0.005 のステップコストを R_0 に加えている．エピソードはスタートからはじまり，ゴールに到達するか “bad state” に遷移した時点で終了とする．

[3] $|wR_1|$ が $|R_0|$ より大きくなるように選択．$|R_1|$ が $|R_0|$ より大きい場合 1 でよい．

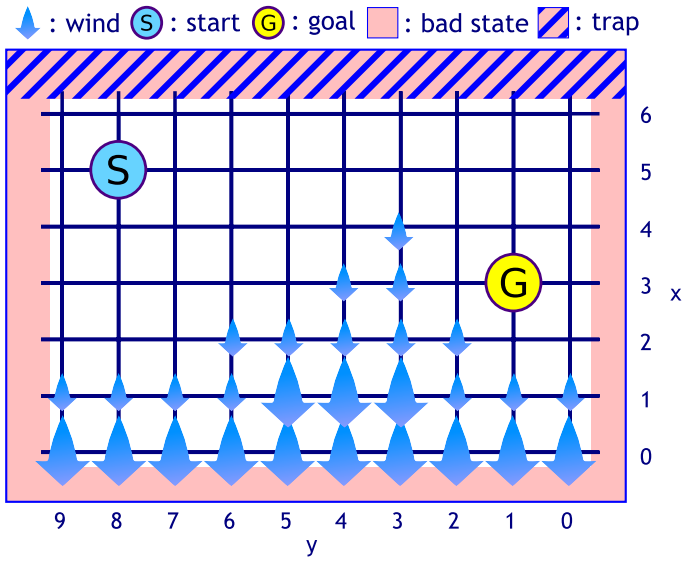


図 2.1　格子状空間における移動タスク.

2.6.1　分離学習の検証

各アルゴリズムによって分離学習が可能か検証するため，行動価値関数がすべての状態でゼロである初期状態から学習を行った．比較したアルゴリズムは

QL	（報酬を分離しない）通常の Q-learning
Sarsa	（報酬を分離しない）通常の Sarsa
DSarsa	分離した報酬に Sarsa を適用（Russell らの Q-decomposition [14] [4]）
ΔSarsa	ΔSarsa を用いた分離学習
DQL+	DQL+を用いた分離学習

である．ボルツマン選択の温度パラメタの初期値はすべて $\tau_0 = 1$ とした．なお分離した報酬に通常の Q-learning を適用した場合，前述したように格子上部をうろつく解に収束したため除外してある．結果のエピソード–報酬曲線を図 2.2 に示す．縦軸の報酬は 1 エピソードの合計報酬（R_0, R_1 を可算しエピソード内で合計したもの）である．

Sarsa, DSarsa, ΔSarsa に違いは見られないが，これらはいずれも合成した行動価値関数が Sarsa と同等の更新則に従っているからである．QL は Sarsa と比較して学習の初期段階ではわずかに学習が遅く，後半ではやや速い．これは QL が遷移先の状態 s' の最大行動価値 $\max_{a'} Q^\star(s', a')$ を使って更新しているため，学習の初期では “bad state” からの負の報酬が伝搬しにくいからであると考えられる．DQL+で

[4] 以下の図では Russell らの Q-decomposition を DSarsa と表記する．これは Q-decomposition には行動価値関数を分離するという意味しかなく，他手法との違いが不明瞭であるためである．

一度報酬が下がっているのは，学習の初期（0 ～ 100 エピソード）では探索のために “trap” に訪れて $R_0 = +10$ と $R_1 = -10$ を受け取るが，中期（100 エピソード ～ ）では DQL+の学習性質（R_1 を優先する）により “trap” に行く割合が Sarsa や学習初期と比較して少なくなっているためである（$R_0 = +10$ が受け取られない）．ただし “trap” に行かない分速く収束している．

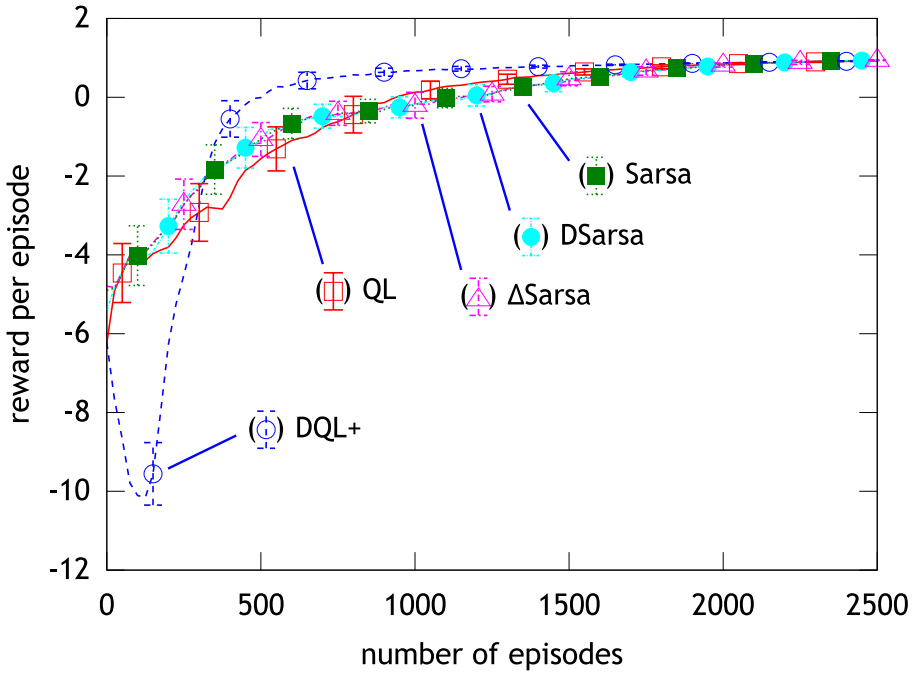


図 2.2 格子状空間における実験結果：分離学習におけるエピソード–報酬曲線．100 回の実験における平均と ±1 の標準偏差．ただし学習の平均的な進捗を把握しやすくするために，時定数 50 試行の低域通過フィルタを適用したデータから平均と標準偏差を計算した．QL:（報酬を分離しない）通常の Q-learning，Sarsa:（報酬を分離しない）通常の Sarsa，DSarsa: 分離した報酬に Sarsa を適用（Russell らの Q-decomposition），ΔSarsa: ΔSarsa を用いた分離学習，DQL+: DQL+を用いた分離学習．いずれも再利用を行わない．

2.6.2 回避行動の再利用の検証

次にスタートの位置を $(3,0)$ に，ゴールの位置を $(6,8)$ に変えて 2.6.1 項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い，以下のアルゴリズムを比較する：

QL　（報酬を分離しない）通常の Q-learning，再利用なし
Sarsa　（報酬を分離しない）通常の Sarsa，再利用なし
DSarsa　分離した報酬に Sarsa を適用，Russell らの Q-decomposition，Q_1 を再利用
ΔSarsa　ΔSarsa を用いた分離学習，Q_1 を再利用
DQL+　DQL+を用いた分離学習，Q_1 を再利用

ボルツマン選択の温度パラメタの初期値はすべて $\tau_0 = 0.1$ とした．2.4 節で述べたように回避行動を効率的に避けるために低く設定してある．結果のエピソード–報酬曲線を 図 2.3 に示す．

QL と Sarsa には目だった違いは見られない．回避行動の報酬 R_1 の合計（$0 \sim 1500$ エピソード）はそれぞれ -4799，-5237 で若干 QL の方が受け取った負の報酬の量が少ないが大差ない．残りのアルゴリズムは Q_1 を再利用したためこれらよりも負の報酬を受け取る量が少なく，ΔSarsa：-257.1，DQL+：-33.10，DSarsa：-1932 であった．DSarsa が ΔSarsa や DQL+よりも大きいのは “trap” に行く局所解に陥っている場合があるからである．また DSarsa の最終的な分散が大きくなっているが，これはゴールに行かずスタート付近に留まる局所解や “trap” に行く局所解に陥っている場合があるからである．2.5.1 項で述べたように Sarsa で学習される行動価値関数 Q_1 は一般的に再利用には不向きであり，この格子状空間における移動タスクはそれを示す例となっている．この問題はボルツマン選択の温度パラメタの初期値 τ_0 を大きくして探索を増やすことで解決できるが，その分より多くの R_1 を受け取ってしまい再利用の効果が薄れる．一方 ΔSarsa, DQL+は正常に収束しており，両者を比較すると DQL+の方がやや収束が速い．これは DQL+が R_1 を優先するため “trap” に探索に行く割合が少ないからである．

この格子状空間の例では DQL+の再利用性能が最も優れていたが，トレードオフが存在するようなタスクには対応できないことを考慮すると ΔSarsa が（2.5.2 項で述べた通り）分離学習・再利用ともに最適なアルゴリズムであると考えられる．本節のタスクでもトレードオフが存在するが，この場合は回避行動を優先して避けてもタスクを達成できるため DQL+は最適な行動の獲得に成功している．しかし例えば

“trap” で受け取る報酬が $R_0 = +12$, $R_1 = -10$ のような場合，最適な行動はゴールではなく “trap” に行く行動になるが，DQL+は回避行動を優先するため “trap” ではなくゴールを目指す解に収束する．

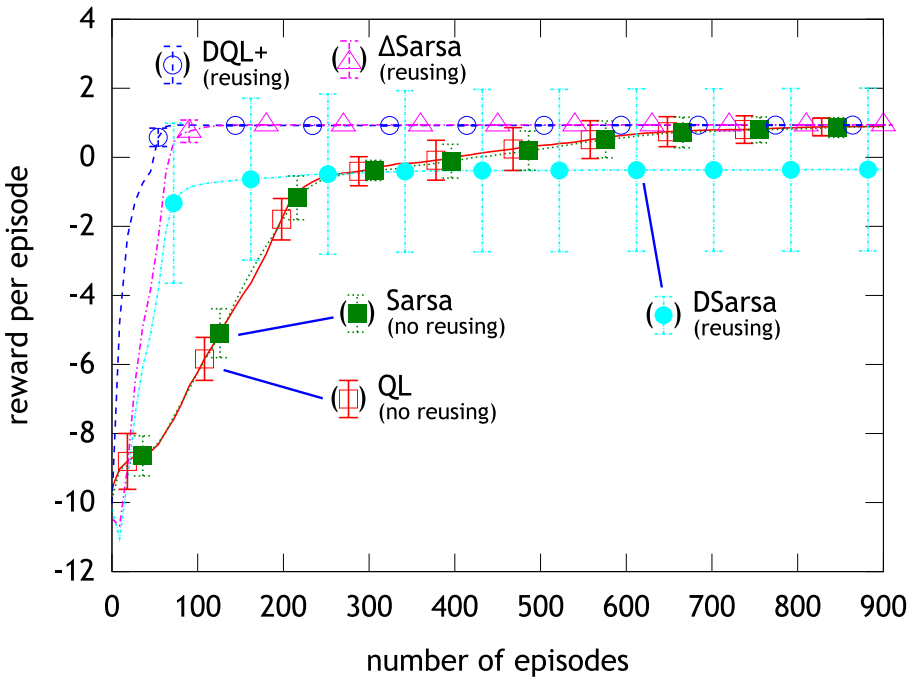


図 2.3 格子状空間における実験結果：回避行動を再利用した場合のエピソード–報酬曲線．100 回の実験における平均と ±1 の標準偏差．時定数 50 試行の低域通過フィルタを適用したデータから計算．QL:（再利用なし，報酬を分離しない）通常の Q-learning，Sarsa:（再利用なし，報酬を分離しない）通常の Sarsa，DSarsa: 分離した報酬に Sarsa を適用（Russell らの Q-decomposition，Q_1 を再利用），ΔSarsa: ΔSarsa を用いた分離学習（Q_1 を再利用），DQL+: DQL+を用いた分離学習（Q_1 を再利用）．

3. 連続状態空間における分離学習と再利用

ロボットの状態などのように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．本章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 [1]）正規化ガウシアンネットワーク [16]（Normalized Gaussian Network，以下 NGnet）を用いて連続状態空間で学習を行う方法を述べる．さらに学習を速めるためにしばしば用いられるエリジビリティ・トレースを分離学習及び再利用に適した形で導入し，2 次元連続平面における移動タスクで各アルゴリズムを比較する．

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

$$y \approx f(x) \triangleq \sum_s N_s(x)\left(W_s x + b_s\right) \tag{3.1}$$

$$N_s(x) \triangleq \frac{G(x;\mu_s,\Sigma_s)}{\sum_{s'} G(x;\mu_{s'},\Sigma_{s'})} \tag{3.2}$$

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．

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

$$Q_i^\star(x,a) \approx \tilde{Q}_i(x,a) \triangleq \sum_{s\in\mathcal{S}} N_s(x) Q_i(s,a) \tag{3.3}$$

によって行動価値関数が近似されているとする（$\mathcal{S}$ はユニットの集合）．$Q_i(s,a)$ は実数のテーブルであり，これが学習対象のパラメタとなる．

3.2 エリジビリティ・トレースを用いた行動価値関数の学習

Sarsa や Q-learning の更新則（式（2.5），（2.8））は 1 時間ステップ前の行動価値関数のみを更新するが，行動の履歴を保持しておいて行動価値関数を一度に更新する方法を「エリジビリティ・トレース」と呼ぶ．エリジビリティ・トレースには過去の履歴をどの程度更新するかを調整するパラメタ λ が用意されており，$\lambda = 0$ で 1 ステップ前のみを更新する方法に一致する． Sarsa にエリジビリティ・トレースを導入した更新則を Sarsa(λ) [13]， Q-learning にエリジビリティ・トレースを導入した更新則を（Peng らの） Q(λ)-learning [17] と呼ぶ．

本来のタスクの報酬と比較して回避行動の報酬が大きい場合にエリジビリティ・トレースを用いると，あるエピソードで回避すべき行動を選択した場合にそのエピソード中の履歴すべてが更新されてしまうため，履歴に含まれている回避する必要のない行動も負の方向に更新される．十分に状態・行動を経験すれば問題はないが，ロボットの運動学習のように状態行動空間が大きくなりタスクが複雑になると状態・行動を十分経験できず，

（1） 学習の途中で回避する必要のない行動をボルツマン選択（式（2.4））などによって積極的に避けてしまい，学習を逆に遅くする場合がある．

（2） 再利用した場合に新たなタスクの学習でも（1）を引き起こす．

などの問題が生じる．そこでエリジビリティ・トレースの更新を制御するパラメタを各行動価値関数に対して設け（$\lambda_i,\ i \in \{0, 1\}$），回避行動の行動価値関数に使われる λ_1 を低く設定（0 でも構わない）することでこの問題に対処する．

ΔSarsa の $\tilde{Q}_0$ の学習には Sarsa(λ) と同じエリジビリティ・トレースを，$\tilde{Q}_1$ の学習には Peng らの Q(λ)-learning を適用し，DQL+の $\tilde{Q}_0$, $\tilde{Q}_1$ の学習にはいずれも Peng らの Q(λ)-learning を適用する．これらの具体的な更新則は付録 B に記載する．なお ΔSarsa で $\tilde{Q}_0$ の更新に Sarsa(λ) と同じエリジビリティ・トレースを適用するのは正確ではないが，ΔSarsa の $\tilde{Q}_0$ の更新は Q-learning の要素と Sarsa の要素を含み（2.5.2 項参照）適切なエリジビリティ・トレースの導出が困難であるため，Sarsa(λ) と同じもので近似している．

3.3 2次元連続平面における移動タスク

連続状態における分離学習及び再利用の性能を検証するために，図3.1に示す2次元連続平面での移動タスクに適用する．基本的には2.6節で述べた格子状空間でのシミュレーション実験を2次元連続平面に拡張したもので，スタート状態（“start”）からゴール状態（“goal”）に移動するエピソード的タスクである．エピソードはスタートからはじまり，ゴールに到達するか“bad state”に遷移した時点で終了とする．

状態は位置 (x_1, x_2)，制御入力は速度（上限を持つ）であり，“wind”はこの入力値を矢の方向に増加させる．“wall”は通り抜けられず，ぶつかった場合“wall”に沿って移動する．エージェントは“start”から“goal”に移動できれば本来のタスクの報酬 $R_0 = 1$ が与えられ（ただしゴール時のずれに応じて若干減少させる），“bad state”に遷移すると発生する -10 から -5 の負の報酬（大きさは“bad state”に留まった時間で決まる）が R_1 として与えられる．行動価値関数を近似するNGnetのパラメタ（中心 μ と分散 Σ）は図3.1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．この行動は次章で述べるロボットの運動学習への応用方法を踏まえている．

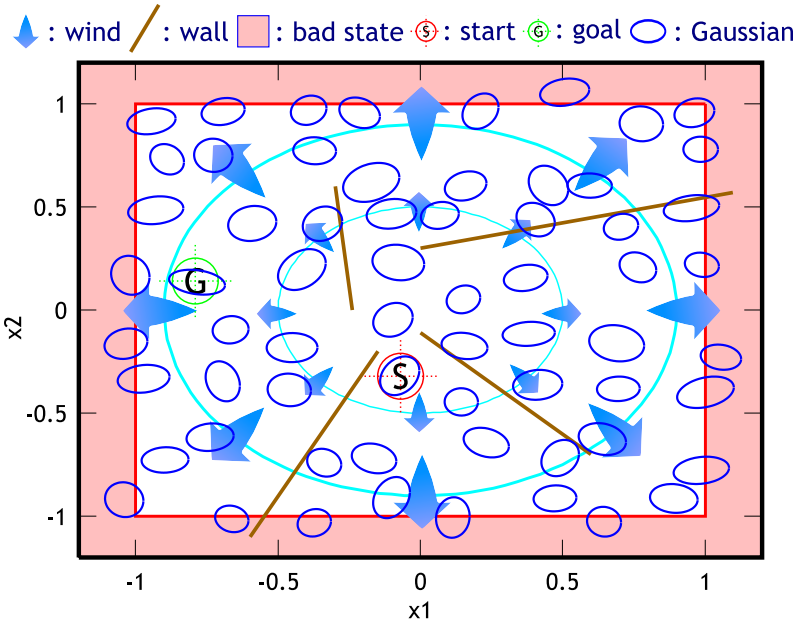


図 3.1 2次元連続平面における移動タスク．楕円は基底関数（正規化ガウス関数）を表す．

3.3.1 分離学習の検証

各アルゴリズムによって分離学習が可能か検証するため，行動価値関数がすべての状態でゼロである初期状態から学習を行った．比較したアルゴリズムは

QL(λ)　（報酬を分離しない）通常の Q($\lambda=0.7$)-learning

Sarsa(λ)　（報酬を分離しない）通常の Sarsa($\lambda=0.7$)

DSarsa(λ,0)　$\tilde{Q}_0$ に Sarsa($\lambda_0 = 0.7$)，$\tilde{Q}_1$ に Sarsa($\lambda_1 = 0$) を用いる分離学習（Russell らの Q-decomposition）

ΔSarsa(λ,0)　ΔSarsa($\lambda_0=0.7, \lambda_1=0$) を用いた分離学習

DQL+(λ,0)　DQL+($\lambda_0=0.7, \lambda_1=0$) を用いた分離学習

である．ボルツマン選択の温度パラメタの初期値はすべて $\tau_0 = 5$ とした．結果のエピソード–報酬曲線を図 3.2 に示す．縦軸の報酬は 1 エピソードの合計報酬（R_0, R_1 を可算しエピソード内で合計したもの）である．

Sarsa(λ) よりも QL(λ) がわずかに収束が速い．しかし 3 つの分離学習はいずれも Sarsa(λ) や QL(λ) に比べて収束が遅い．この原因は $\lambda_1 = 0$ としているからであると考えられ，Q_0 に Sarsa($\lambda_0 = 0.7$)，Q_1 に Sarsa($\lambda_1 = 0.7$) を用いて分離学習を行ったところ Sarsa(λ) と同等の結果が得られた．しかしながら 3.2 節で予想した通り，このアルゴリズムは次項で述べる再利用の実験でゴールに行かずある状態に留まるなどの局所解に陥る場合があり，比較対象からは外している．DSarsa(λ,0) は DQL+(λ,0) より，ΔSarsa(λ,0) は DSarsa(λ,0) よりわずかに収束が速い．DSarsa(λ,0) と ΔSarsa(λ,0) の収束の速さが異なるのは，エリジビリティ・トレースの導入により合成した行動価値関数が従う更新則が等しくなくなっているからである．

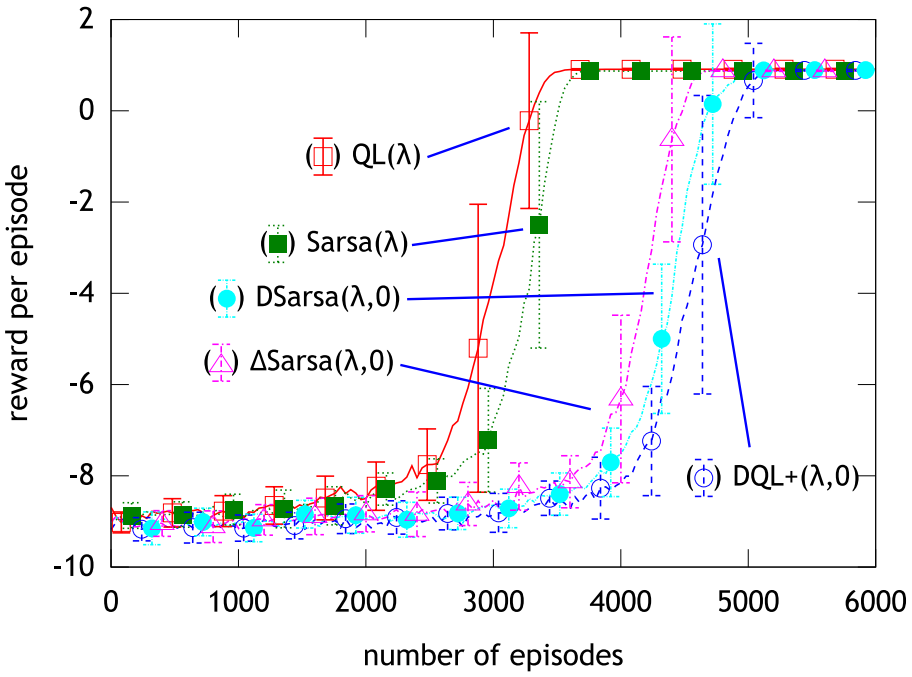


図 3.2 2 次元連続平面における実験結果：分離学習におけるエピソード–報酬曲線．25 回の実験における平均と ±1 の標準偏差．時定数 50 試行の低域通過フィルタを適用したデータから計算．QL(λ):（報酬を分離しない）通常の Q($\lambda = 0.7$)-learning，Sarsa(λ):（報酬を分離しない）通常の Sarsa($\lambda = 0.7$)，DSarsa(λ,0): $\tilde{Q}_0$ に Sarsa($\lambda_0 = 0.7$)，$\tilde{Q}_1$ に Sarsa($\lambda_1 = 0$) を用いる分離学習（Russell らの Q-decomposition），ΔSarsa(λ,0): ΔSarsa($\lambda_0 = 0.7, \lambda_1 = 0$) を用いた分離学習，DQL+($\lambda$,0): DQL+($\lambda_0 = 0.7, \lambda_1 = 0$) を用いた分離学習．いずれも再利用を行わない．

3.3.2 回避行動の再利用の検証

次にスタートと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．比較したアルゴリズムは

QL(λ)　（報酬を分離しない）通常の Q($\lambda=0.7$)-learning，再利用なし

Sarsa(λ)　（報酬を分離しない）通常の Sarsa($\lambda=0.7$)，再利用なし

DSarsa(λ,0)　$\tilde{Q}_0$ に Sarsa($\lambda_0 = 0.7$)，$\tilde{Q}_1$ に Sarsa($\lambda_1 = 0$) を用いる分離学習，Russell らの Q-decomposition，$\tilde{Q}_1$ を再利用

ΔSarsa(λ,0)　ΔSarsa($\lambda_0=0.7, \lambda_1=0$) を用いた分離学習，$\tilde{Q}_1$ を再利用

DQL+(λ,0)　DQL+($\lambda_0=0.7, \lambda_1=0$) を用いた分離学習，$\tilde{Q}_1$ を再利用

である．ボルツマン選択の温度パラメタの初期値はすべて $\tau_0 = 1$ とした．2.4 節で述べたように回避行動を効率的に避けるために低く設定してある．結果のエピソード–報酬曲線を図 3.3 に示す．

QL(λ) が Sarsa(λ) よりも収束が速いのは先の実験と同様である．回避行動の報酬 R_1 の合計（0 ～ 1500 エピソード）はそれぞれ QL(λ)：-6541，Sarsa(λ)：-7810 で，QL(λ) の方が収束が速い分少なくなっている．回避行動の行動価値関数を再利用した 3 つのアルゴリズムはこれらよりも負の報酬 R_1 を受け取る量が少なく，それぞれ DSarsa(λ,0)：-3061，ΔSarsa(λ,0)：-5622，DQL+(λ,0)：-2708 で，$\tilde{Q}_1$ の再利用による効果が現れていることがわかる．その分収束も速い．また DSarsa(λ,0) よりも DQL+(λ,0) の方が受け取る負の報酬がより少なくなっており，2.5.1 項で述べた Sarsa で $\tilde{Q}_1$ を学習した場合の再利用における問題が影響していると考えられる．しかし再利用にも適しているはずの ΔSarsa(λ,0) は再利用による効果がより少ない．この理由としては，関数近似器を用いて Q-learning を行ったときに生じる学習の不安定さ [11] などが考えられ，実際，再利用した行動価値関数 $\tilde{Q}_1$ を調べたところ R_1 は常にゼロ以下であるにも関わらず正の値を示す行動価値が存在していた． DQL+(λ,0) も基本的には Q-learning を用いているが，$\tilde{Q}_1$ を正にならないように強制しているためそのような問題を回避できていると考えられる．

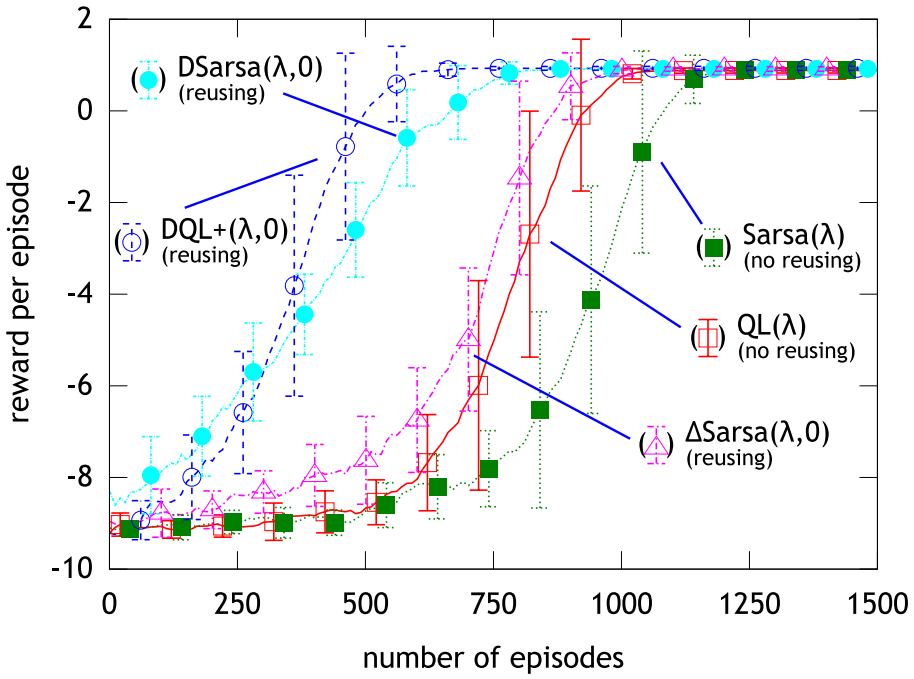


図 3.3 2 次元連続平面における実験結果：回避行動を再利用した場合のエピソード–報酬曲線．25 回の実験における平均と ±1 の標準偏差．時定数 50 試行の低域通過フィルタを適用したデータから計算．QL(λ):（再利用なし，報酬を分離しない）通常の Q($\lambda = 0.7$)-learning, Sarsa(λ):（再利用なし，報酬を分離しない）通常の Sarsa($\lambda = 0.7$), DSarsa(λ,0): $\tilde{Q}_0$ に Sarsa($\lambda_0 = 0.7$), $\tilde{Q}_1$ に Sarsa($\lambda_1 = 0$) を用いる分離学習（Russell らの Q-decomposition, $\tilde{Q}_1$ を再利用），ΔSarsa(λ,0): ΔSarsa($\lambda_0 = 0.7, \lambda_1 = 0$) を用いた分離学習（$\tilde{Q}_1$ を再利用），DQL+(λ,0): DQL+($\lambda_0 = 0.7, \lambda_1 = 0$) を用いた分離学習（$\tilde{Q}_1$ を再利用）．

4. 多関節ロボットの全身運動学習への応用

本章では前章までの手法を多関節ロボットの全身運動学習に応用する方法について述べる．ここでは全身運動を「ロボットの一般化座標に相当する物理量の軌道によって表現可能な運動」と定義し，扱うロボットは土台が非固定の多関節ロボットを想定する．例えば跳躍や（ボールの初期位置を固定した）サーブ，起き上がり運動 [1] [18] 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をあらかじめ与えることはしない．このときの課題は状態空間が高次元になるため学習時間が膨大になること，土台が非固定のため劣駆動系でかつ力学的な拘束条件が変化するため制御問題としても難しいことである．

4.1 運動学習アーキテクチャ

この問題に対処するために以下のアプローチを取る（図 4.1）：

（1） ロボットの非線形ダイナミクスを各ユニット（以下シンボルと呼ぶ）が線形ダイナミクスを持つ NGnet で学習し（詳細は付録 C），このとき得られた基底関数を行動価値関数の推定にも使う．

（2） 行動は NGnet からひとつの目標シンボル s を選びその基底関数の中心 μ_s を目標とした制御を行うというものである．このときの制御は関節角 q の躍度のノルムを時間積分したものが最小になるように関節角軌道を計算し [19]（付録 D），PD 制御器で追従することで実現する．

このような手法を採用した裏には（a）ダイナミクスの非線形性が強くなるほどロボットの制御が困難になり，行動価値関数を推定するためにより多くの基底関数を配置する必要がある（逆にい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限定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

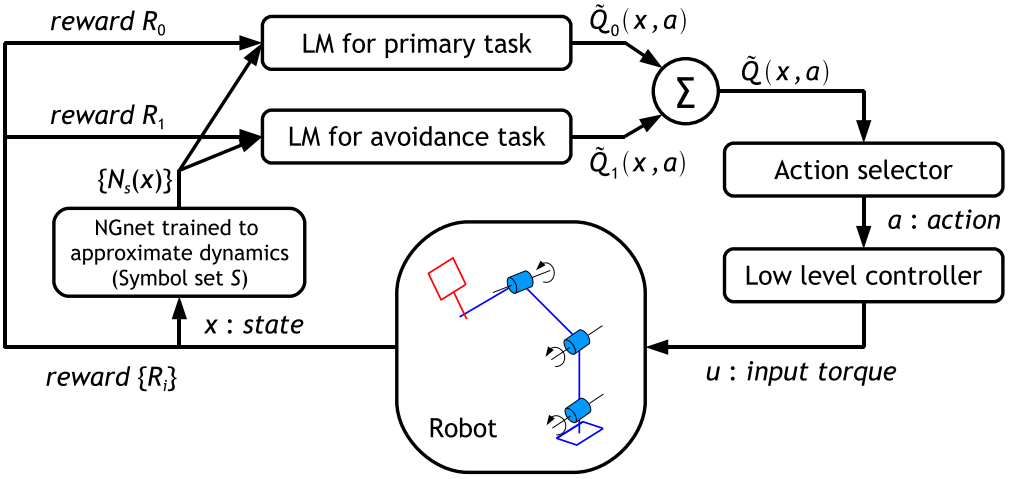


図 4.1　全身運動を学習するアーキテクチャ．LM は学習器を表す．

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．

4.2　運動学習の流れ

このとき運動学習の学習の流れは以下のようになる：

（1）　ロボットにランダムな入力（トルク）を与えて状態変化を観測し，NGnet でそのダイナミクスを学習する．

（2）　あるタスクの学習（タスク固有の報酬関数 R_0，回避行動の報酬関数 R_1）で $\tilde{Q}_0, \tilde{Q}_1$ を分離学習する．

（3）　新たなタスクの学習（タスク固有の報酬関数 R'_0，回避行動の報酬関数 R_1）では $\tilde{Q}'_0$ はすべてのパラメタが何も学習していない初期状態から開始し，$\tilde{Q}_1$ は（2）の学習結果を初期値として学習する．

4.3　シミュレーション実験

以上の手法を 4 リンク 3 関節土台非固定型ロボット（図 4.2）の全身運動学習に適用し，回避行動の分離学習と再利用が行えるか検証する．以下で学習させる全身運動はテニスのサーブであり，ラケットが手先に固定されている．各アクチュエータの軸はすべて同じ向き（y 軸方向）であるためロボットの動作は 2 次元平面内に限られる．このためロボットの状態は位置と姿勢で計 3 次元，関節角 3 次元，及びこれらの速度項 6 次元の合計 12 次元 $(x, z, \theta, q_0, q_1, q_2, \dot{x}, \dot{z}, \dot{\theta}, \dot{q_0}, \dot{q_1}, \dot{q_2})$ で表現される（これらはセンサから観測できるとする）．ロボットの行動はダイナミクスを学習し

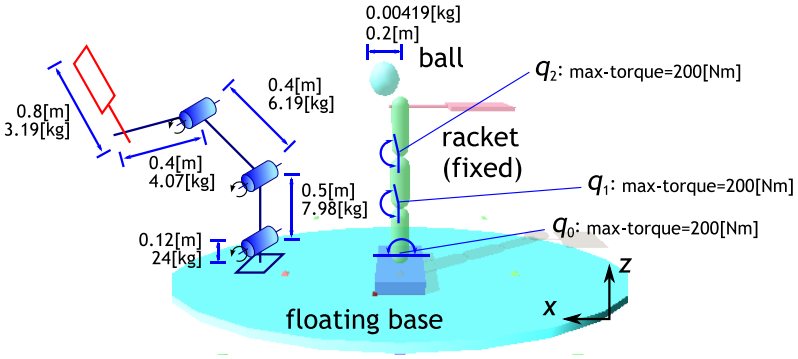


図 4.2 シミュレーションに用いた 4 リンク 3 関節のロボット．手先にはテニスラケットが固定されている．各関節には軸がすべて同じ向き（y 軸方向）のアクチュエータが取り付けられており，動作は 2 次元平面内に限られる．状態は 12 次元 $(x, z, \theta, q_0, q_1, q_2, \dot{x}, \dot{z}, \dot{\theta}, \dot{q_0}, \dot{q_1}, \dot{q_2})$.

た NGnet のシンボル集合 $\mathcal{S}$ から選んだ目標シンボル s と，関節角軌道の計画に使う s の中心 μ_s までの到達時間 $T_f \in \mathcal{I}$（$\mathcal{I}$ は離散集合）から成る．すなわち行動集合 $\mathcal{A} = \mathcal{S} \times \mathcal{I}$（離散集合）から選択される（付録 D）．ひとつの行動は，計画された関節角軌道を PD 制御器で追従し μ_s に達するか T_f が経過することで終了する（シンボル遷移）．なおロボットのダイナミクスのシミュレーションは Open Dynamics Engine [20] を用いて行っている．

土台リンクの水平制御

ロボットの運動学習でいくつかの予備実験を行ったところ，ロボットの土台リンクの接触条件が変化するときに制御が不安定になり，学習に悪影響を与えることがわかった．ロボットの土台リンクがわずかに浮いた状態と接地している状態では行動価値関数を近似する関数器（NGnet）の基底関数の出力 $\{N_s(x)|s \in \mathcal{S}\}$ にほとんど差がないが，それぞれの状態における行動の結果はダイナミクスの違いから大きく異なることが多い．この問題は事前に予測できたためロボットのダイナミクスを学習させた基底関数を用いる方法や分離誤差に基づく基底関数の修正法（5 章）による解決を図ったが，一度土台リンクが浮くと明示的に土台を接地させる制御をしない限り接触状態の切り替わりが続く不安定な状態に陥るため，解決には至らなかった．

そこで土台リンクを水平に保つための制御を明示的に行うことで接触状態の切り替わりが続くことを阻止し，この問題を緩和している．すなわち前述の行動における制御信号（各関節のトルク）のうち土台リンクの関節に入力されるものに，土台リンクを水平に保つ PD 制御信号を上乗せする．

4.3.1 運動学習における回避行動

再利用される回避行動は転倒などのボディに大きな力が加わる行動であり，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．

$$d \triangleq \sum_j \frac{[\|f_j\| - Th_j]_+}{Th_j} \tag{4.1}$$

$$\text{for } d > 0, D = 0\text{: } D' = D + 1 + (1 - D)\left\{\frac{2}{1 + e^{-\tau_{\text{dmg}} d}} - 1\right\} \tag{4.2}$$

$$\text{for } d > 0, D \neq 0\text{: } D' = D + (2 - D)\left\{\frac{2}{1 + e^{-\tau_{\text{dmg}} d}} - 1\right\} \tag{4.3}$$

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} は適当な正の定数，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．d と D の変化の一例を図 4.3 に示す．

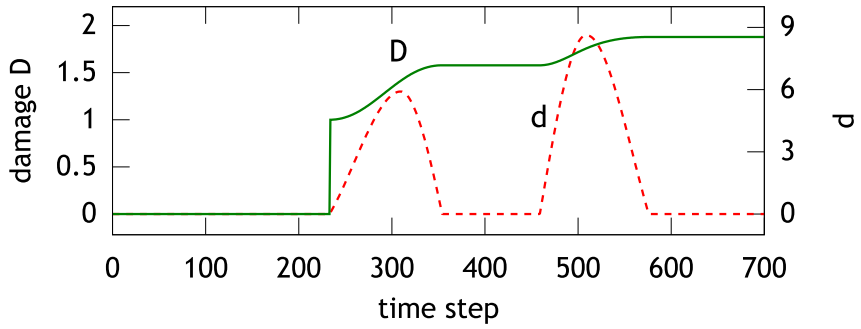


図 4.3 あるシンボル遷移での正規化された力の和 d とそれに対するダメージ D の変化の一例．

4.3.2 サーブと回避行動の分離学習

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

$$R_0 = \begin{cases} 10\left[v_x - \frac{\sqrt{v_z^2}}{2} - r_{0\max}\right]_+ & \text{for } \theta_{\text{ball}} < 15^\circ \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

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

ひとつのエピソードはロボットが直立姿勢・静止状態かつボールが初期位置・静止状態からはじまり，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） ボールが指定された方向以外に飛び去った．
（d） ボールが指定された方向に十分飛び（$2m$ 以上），ロボットが静止している．

のいずれかを満たしていれば終了とする．（d）のみがタスクの成功である．

行動価値関数がすべての状態でゼロである初期状態から分離学習を行った．比較したアルゴリズムは

QL(λ)　（報酬を分離しない）通常の Q($\lambda=0.1$)-learning
DSarsa(λ,0)　$\tilde{Q}_0$ に Sarsa($\lambda_0 = 0.8$)，$\tilde{Q}_1$ に Sarsa($\lambda_1 = 0$) を用いる分離学習（Russell らの Q-decomposition）
ΔSarsa(λ,0)　ΔSarsa($\lambda_0=0, \lambda_1=0$) を用いた分離学習
DQL+(λ,0)　DQL+($\lambda_0=0.8, \lambda_1=0$) を用いた分離学習

である．ボルツマン選択の温度パラメタの初期値はすべて $\tau_0 = 50$ とした．エリジビリティ・トレースの重み λ は予備実験を行い各アルゴリズムに最適なものを選択している．結果のエピソード–報酬曲線を図 4.4 に示す．

DQL+(λ,0) は QL(λ) と同程度の収束の速さであり，分離学習アルゴリズムの中では最もよい性能を発揮している．これは Q(λ)-learning が持つ収束の速さ，3.2 節で述べた λ_1 を低く設定することや 3.3 節で述べた $\tilde{Q}_1$ を拘束することによる DQL+の学習の安定さなどが効果的に機能したからであると考えられる．DQL+(λ,0) で最終的に（3000 エピソード後）得られたサーブの一例を図 4.5 に示す．

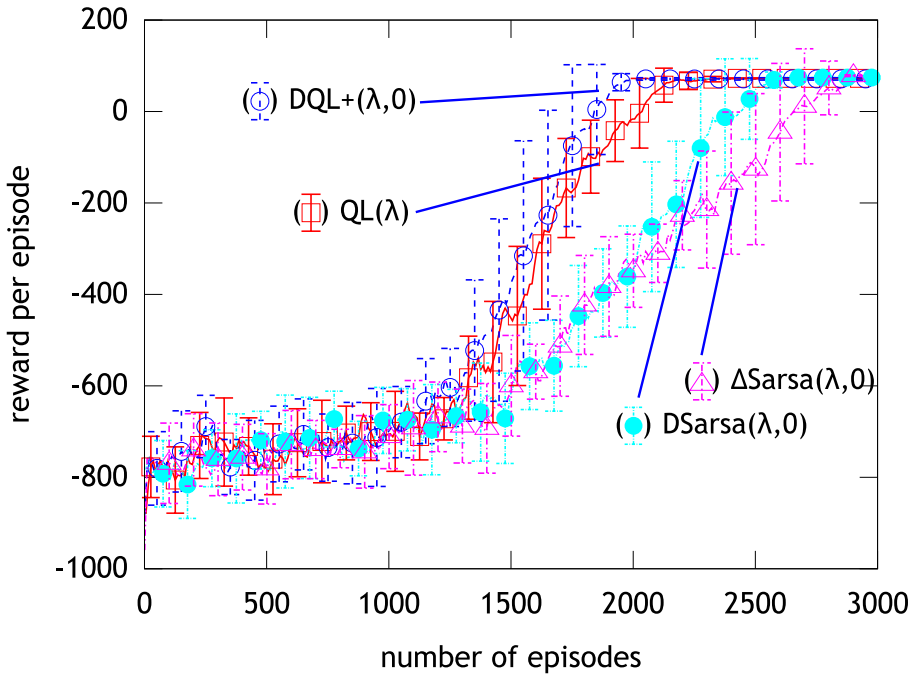


図 4.4 サーブと回避行動の分離学習結果：エピソード–報酬曲線．14 回の実験における平均と ±1 の標準偏差．時定数 50 試行の低域通過フィルタを適用したデータから計算．QL(λ): （報酬を分離しない）通常の Q($\lambda = 0.1$)-learning，DSarsa(λ,0): $\tilde{Q}_0$ に Sarsa($\lambda_0 = 0.8$)，$\tilde{Q}_1$ に Sarsa($\lambda_1 = 0$) を用いる分離学習（Russell らの Q-decomposition），ΔSarsa(λ,0): ΔSarsa($\lambda_0 = 0, \lambda_1 = 0$) を用いた分離学習，DQL+($\lambda$,0): DQL+($\lambda_0 = 0.8, \lambda_1 = 0$) を用いた分離学習．いずれも再利用を行わない．

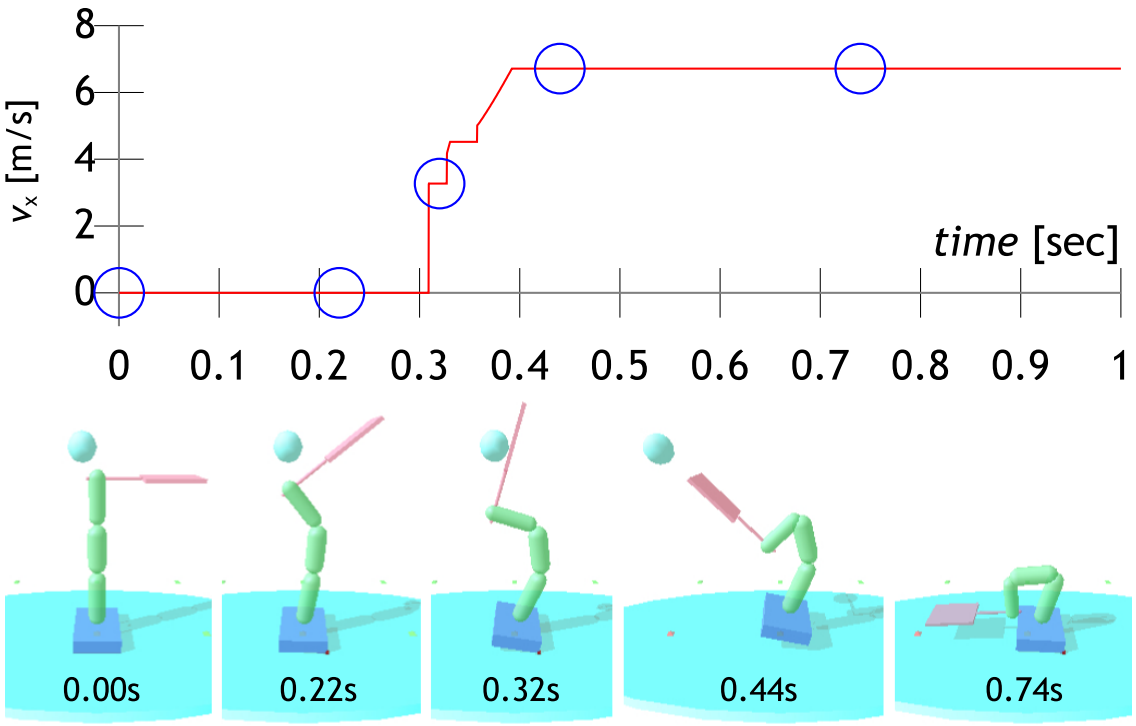


図 4.5 DQL+(λ,0) で獲得されたサーブにおけるボールの水平速度 v_x の軌道とスナップショット．軌道上の円の時刻はスナップショットの時刻と一致する．

4.3.3 遅いサーブと速いサーブの間の再利用

次に，ロボットの運動学習における回避行動の再利用がどの程度有効かを検証するために,「遅いサーブ」を学習し，回避行動の行動価値関数を再利用して「速いサーブ」を学習，再利用しない場合と比較する．ここで速いサーブとは 4.3.2 項で定義したサーブと同様のタスク，遅いサーブとは式（*4.4*）の報酬をボールの速度の水平成分 v_x がある値（5[m/s]）を超えた場合にはゼロとするタスクである（それ以外は速いサーブと同様の報酬設計）．

学習には 4.3.2 項の実験で用いた各アルゴリズムと，4.3.2 項の実験で最も収束が速かった DQL+を用いた．具体的には

QL(λ)	（報酬を分離しない）通常の Q($\lambda=0.1$)-learning，再利用なし
no reusing DQL+	DQL+($\lambda_0=0.8, \lambda_1=0$) を用いた分離学習，再利用なし
DSarsa(λ,0)	$\tilde{Q}_0$ に Sarsa($\lambda_0=0.8$)，$\tilde{Q}_1$ に Sarsa($\lambda_1=0$) を用いる分離学習，Russell らの Q-decomposition，$\tilde{Q}_1$ を再利用
ΔSarsa(λ,0)	ΔSarsa($\lambda_0=0, \lambda_1=0$) を用いた分離学習，$\tilde{Q}_1$ を再利用
DQL+(λ,0)	DQL+($\lambda_0=0.8, \lambda_1=0$) を用いた分離学習，$\tilde{Q}_1$ を再利用

を比較した．再利用される $\tilde{Q}_1$ にはあらかじめ遅いサーブをそれぞれのアルゴリズムで学習させ，その結果得られた回避行動の行動価値関数を用いた．上記アルゴリズムで速いサーブの学習を行ったところ，図 4.6 のような結果を得た．ただし速いサーブの学習におけるボルツマン選択の温度パラメタの初期値はいずれも $\tau_0=10$ としている．

回避行動の報酬 R_1 のみ抽出して 0 ～ 1000 エピソードの間で合計すると，QL(λ) は -2.28×10^5，no reusing DQL+は -2.22×10^5，DSarsa(λ,0) は -4.54×10^5，ΔSarsa(λ,0) は -2.22×10^5，DQL+(λ,0) は -2.08×10^5 となった．負の報酬を受け取る量は DQL+(λ,0) が最も小さくなっており，再利用による効果が現れていることが確認される．図 4.6 のグラフでも DQL+(λ,0) の収束が最も速い．ΔSarsa(λ,0) は no reusing DQL+と同程度に負の報酬を受け取っているが，再利用なしで速いサーブを学習させた 4.3.2 項の実験で ΔSarsa(λ,0) が no reusing DQL+よりも収束が遅かったことを考慮すれば，再利用によって ΔSarsa(λ,0) 自体の性能は向上していると言える．一方で DSarsa(λ,0) はほかのアルゴリズムの倍程度負の報酬を受け取っている．この原因としては，2.5.1 項で述べたように Sarsa で学習した $\tilde{Q}_1$ が再利用に適さないことなどが考えられる．

このようにロボットの運動学習においても DQL+(λ,0) や ΔSarsa(λ,0) による再

利用の有効性が確認できた．しかしながらこれらの再利用の効果は，格子状空間や2次元連続平面における移動タスクの実験結果に比べるとわずかである．この原因としては，

（1） 状態行動空間が大きいため回避行動の行動価値関数が十分に学習されていない．

（2） 連続状態空間へ拡張する際の関数近似器が本来のタスクと回避行動を適切に分離していない．

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

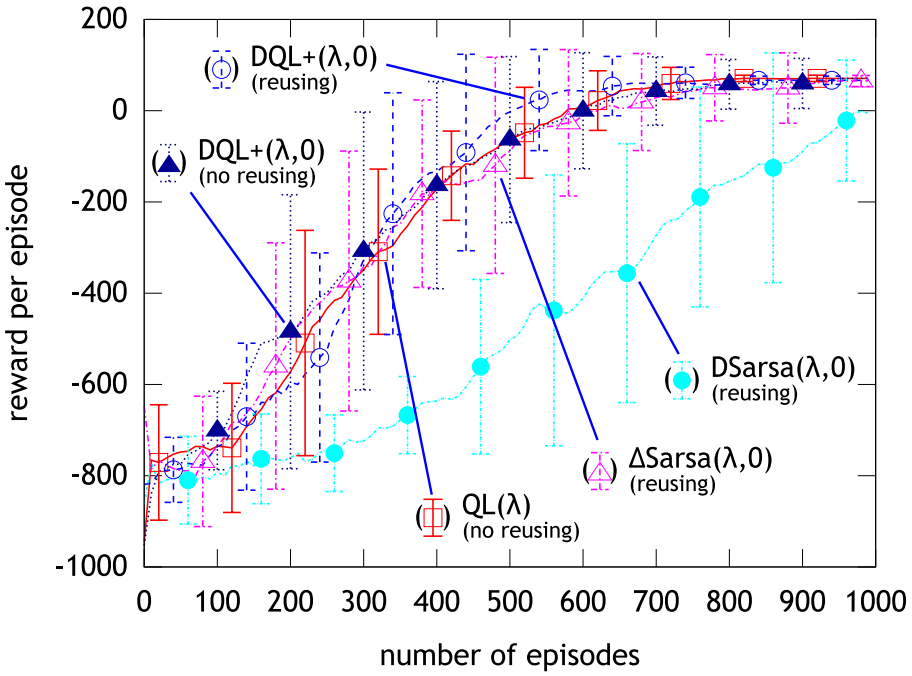


図 4.6 遅いサーブと速いサーブの間の再利用結果：エピソード–報酬曲線．25 回の実験における平均と ±1 の標準偏差．時定数 50 試行の低域通過フィルタを適用したデータから計算．
QL(λ): （報酬を分離しない）通常の Q($\lambda=0.1$)-learning，再利用なし，no reusing DQL+: DQL+($\lambda_0 = 0.8, \lambda_1 = 0$) を用いた分離学習，再利用なし，DSarsa($\lambda$,0): $\tilde{Q}_0$ に Sarsa($\lambda_0 = 0.8$)，$\tilde{Q}_1$ に Sarsa($\lambda_1 = 0$) を用いる分離学習，Russell らの Q-decomposition，$\tilde{Q}_1$ を再利用，ΔSarsa(λ,0): ΔSarsa($\lambda_0 = 0, \lambda_1 = 0$) を用いた分離学習，$\tilde{Q}_1$ を再利用，DQL+(λ,0): DQL+($\lambda_0 = 0.8, \lambda_1 = 0$) を用いた分離学習，$\tilde{Q}_1$ を再利用．再利用される $\tilde{Q}_1$ は遅いサーブの学習で得られた回避行動の行動価値関数．

5. 分離学習・再利用のための関数近似器修正法

本章では連続状態空間へ拡張する際の関数近似器が本来のタスクと回避行動を適切に分離していないことによって生じる問題を考察し，関数近似器の修正による改善を試みる．さらに跳躍などのシミュレーションで分離学習・再利用における効果を調べる．なお本章で提案する修正法は格子状空間・2 次元連続平面での移動タスク，及びロボットの運動学習のいずれにおいても安定して高い性能を示している DQL+ に対して開発し，対象とする関数近似器は 3 章で導入した NGnet とする．

5.1 分離学習と再利用における関数近似器の問題

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

一例として図 5.1(a) に示すような，基底関数 s 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を考える．この基底関数 s に対応付けられた行動価値関数のパラメタ $Q_1(s,a)$ は状態 x_2 から行動 a を選択することによって負の方向に更新される．回避すべき行動を何度も選択しないように，負の報酬やステップサイズパラメタはある程度大きく設定されるので，この後状態 x_1 で同じ行動 a を選択する確率はボルツマン選択などにより小さくなる．結果として行動 a の選択が x_1 における最適方策の場合などでは，学習に時間が掛かるかほかの局所最適解に陥る可能性が高くなる．

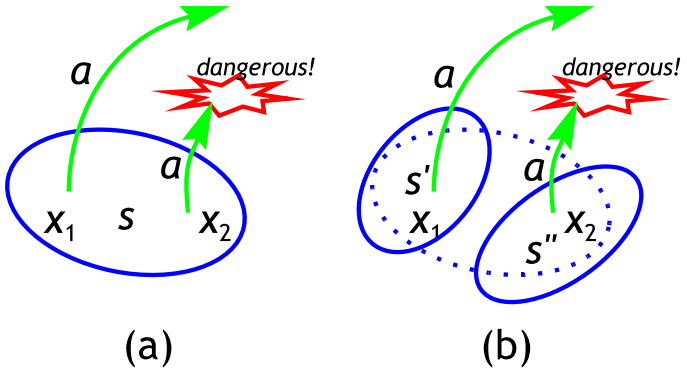


図 5.1 学習に失敗する可能性のある基底関数と行動の例 (a)，及び基底関数の分割による改善 (b)．楕円は基底関数（正規化ガウス関数），a は行動をそれぞれ表す．

これは関数近似器を用いると一般的に起こり得る問題で，当然分離学習でも起こり再利用においても問題となる．同じ例で $Q_1(s,a)$ が状態 x_2 から行動 a を選択したことによって負の方向に更新され新たなタスクの学習で再利用された場合，その付近の状態（x_1 など）から行動 a を安全であるにもかかわらず選択する確率が低くなってしまう．

5.2 「分離誤差」に基づく関数近似器修正法

この問題が状態空間のどこで起きているか調べられれば，図 5.1(a) のような場合には基底関数を図 5.1(b) のように分割することで対処できる．DQL+の場合，分離学習が可能である条件式（*2.11*）が満たされなくなることと，この条件が分離学習した行動価値関数を使って評価できることに注意すると，次式で定義される「分離誤差」を評価することで問題となる基底関数が調べられる：

$$\tilde{C}(x) \triangleq \left(\sum_{i\in\{0,1\}} \max_{a\in\mathcal{A}} \tilde{Q}_i(x,a) - \max_{a\in\mathcal{A}} \sum_{i\in\{0,1\}} \tilde{Q}_i(x,a) \right)^2 . \qquad (5.1)$$

ただし分離誤差による評価が可能なのは DQL+に限られる．ここで定義した $\tilde{C}(x)$ は R_0 と R_1 が同時に値を持つなどのトレードオフが存在する場合でも $\tilde{C}(x) > 0$ となる可能性があり，このような場合に図 5.1(b) のような分割を行っても $\tilde{C}(x)$ はほとんど変化しないと考えられるからである．DQL+では 2.5.3 項の更新則でトレードオフによる $\tilde{C}(x)$ の発生を抑えているため関数近似器の精度による $\tilde{C}(x)$ の発生に限定でき，ここで述べた使用方法が可能となる．

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

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

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

ただし ∗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り，以下のシミュレーション実験では“分離誤差を評価する状態集合”は各基底関数の中

心状態及び2つの基底関数の中間状態からランダムに選択し，実行のタイミングはエピソードの終了直後とした．詳細は付録 E に記載する．

5.3 関数近似器修正法を組み込んだ分離学習と再利用の流れ

関数近似器修正法を用いた場合の分離学習と再利用は以下のような流れで実行される：

（1） 初期の基底関数の集合を用意する（運動学習なら NGnet にロボットのダイナミクスを学習させて生成）．

（2） あるタスクの学習（タスク固有の報酬関数 R_0，回避行動の報酬関数 R_1）で $\tilde{Q}_0, \tilde{Q}_1$ を分離学習する．このとき（1）の基底関数は関数近似器修正法によって変更される．

（3） 新たなタスクの学習（タスク固有の報酬関数 R'_0，回避行動の報酬関数 R_1）では $\tilde{Q}'_0$ はすべてのパラメタが何も学習していない初期状態から開始し，$\tilde{Q}_1$ は（2）の学習結果を初期値として学習する．ただしこれらの関数近似に使われる基底関数は（2）の結果を初期値として用い，関数近似器修正法によってさらに変更される．

5.4 2次元連続平面における移動タスク（2）

基底関数の修正法が有効に機能するか検証するため，3.3 節で行った 2 次元連続平面における移動タスクの設定を図 5.1(a) のような問題が生じやすくなるように変更する（図 5.2）．“start”，“goal”，“bad state”，“wind”，“wall” の定義は 3.3 節と同じだが，負の報酬 R_1 は “bad state” に遷移するか “wall” にぶつかった場合に -200 から -100 の範囲で与えられるように変更する．この大きさは “bad state” に留まった時間や “wall” に接した時間で決まる．

さらに NGnet のパラメタ（中心 μ と分散 Σ）を図 5.2 の楕円に示すように与える．これらの基底関数はスタート付近など，図 5.1(a) のような問題が生じやすいように設定されている．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しその微分を制御入力とする離散行動で，3.3 節の場合と同じである．ただしこの行動集合 $\mathcal{A}$ は基底関数の修正法を適用する際に学習パラメタの大幅な増加を防ぐため，また基底関数の修正法を使わない場合との比較における条件を同じにするために基底関数の修正に対して変化させない．

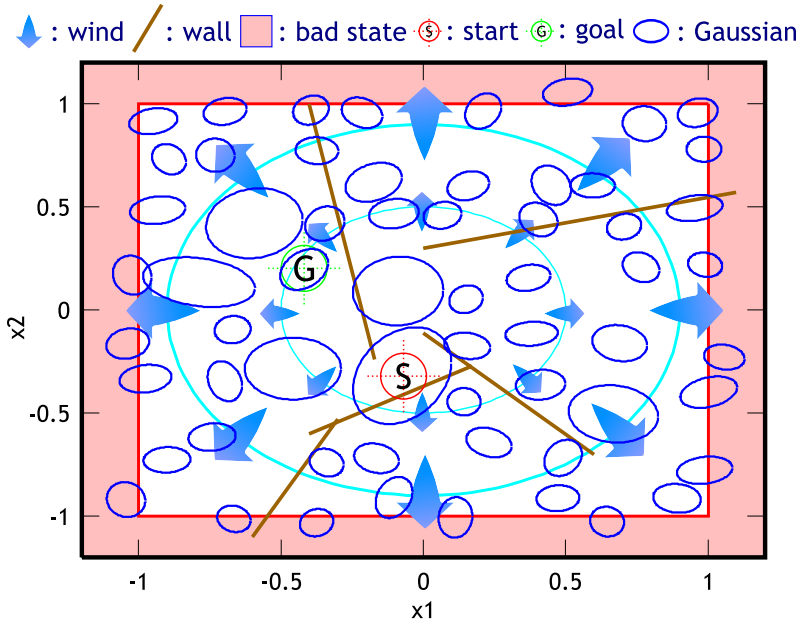


図 5.2　2 次元連続平面における移動タスク（2）．楕円は基底関数（正規化ガウス関数）を表す．

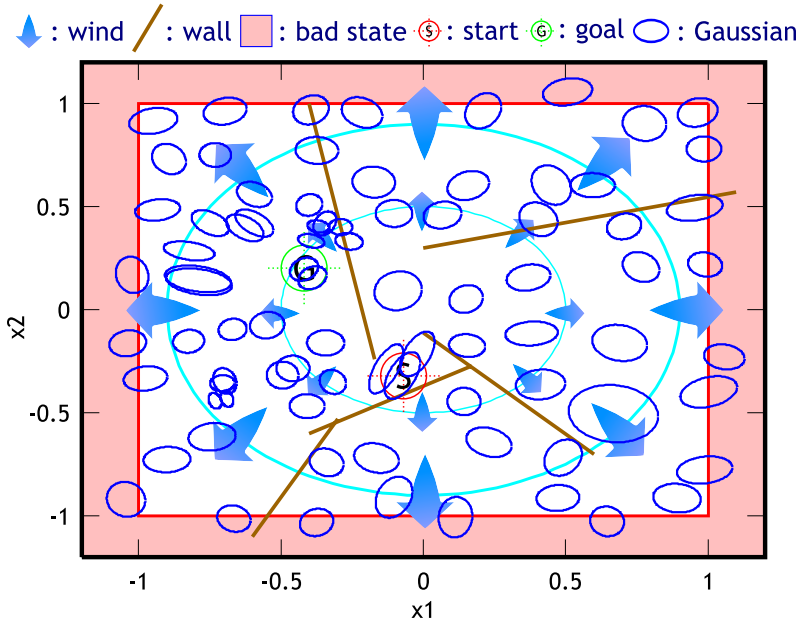


図 5.3　基底関数修正法（5.2 節のアルゴリズム）を適用して得られた基底関数．

5.4.1 分離学習の検証

まず分離学習について以下のアルゴリズムを比較する：

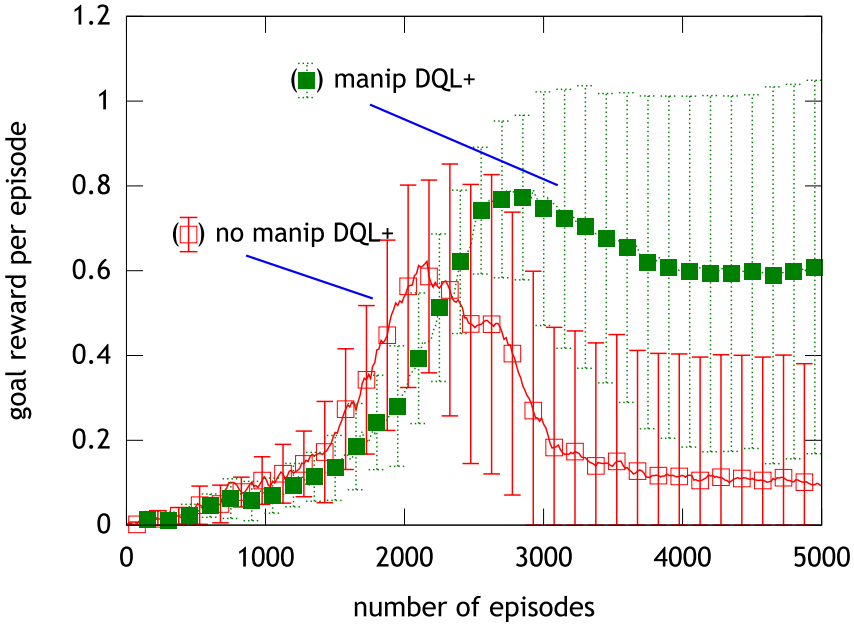
no manip DQL+　DQL+($\lambda_0 = 0.9, \lambda_1 = 0$) を用いた分離学習，基底関数修正法は使わない

manip DQL+　DQL+($\lambda_0 = 0.9, \lambda_1 = 0$) を用いた分離学習，5.2 節で述べた基底関数修正法を毎エピソード終了後に適用

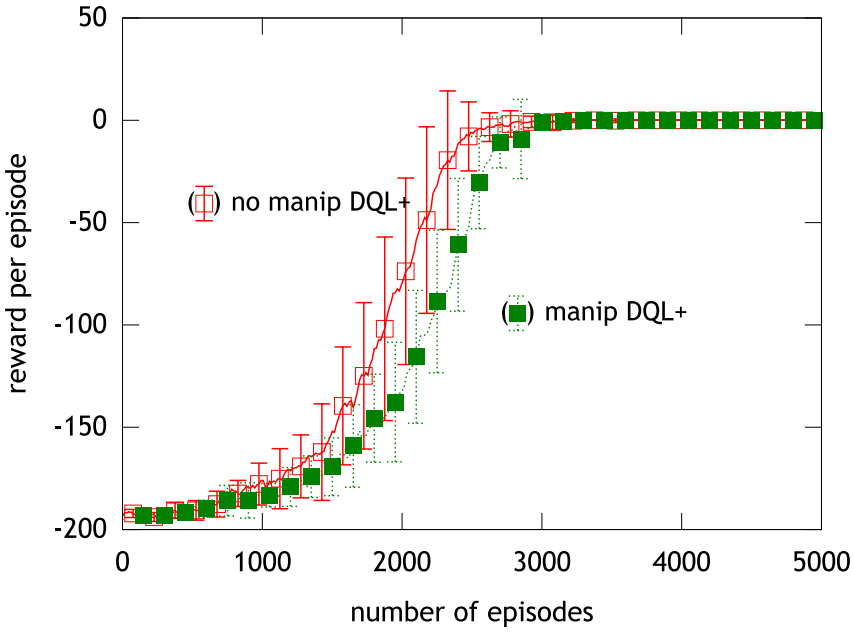
ボルツマン選択の温度パラメタの初期値はいずれも $\tau_0 = 10$ とした．manip DQL+で最終的に（5000 エピソード後）得られた NGnet の基底関数の一例は図 5.3 の楕円に示すようになっており，スタート付近などで基底関数が生成されていることがわかる．図 5.4(a) はゴールによって与えられる報酬[5]をエピソードに対してプロットしたものであり，manip DQL+は no manip DQL+よりも平均で 6 倍程度多くの報酬が獲得できるようになっている．実際 no manip DQL+の場合 20 回の実験中 2 回しかゴールに到達する解を獲得できていないのに対し，manip DQL+の場合 20 回中 12 回まで向上している．ゴールに到達する解を獲得できなかった場合はスタート付近に留まるなどの局所解に陥っており，これは図 5.1(a) のような問題が発生しやすいタスク設定であることが原因である．このため manip DQL+も no manip DQL+も最終的な分散が大きくなっている．

一方図 5.4(b) はエピソードあたりの R_1 の合計をエピソードに対してプロットしたもので，manip DQL+の方が no manip DQL+よりも若干多くの負の報酬を受け取っていることがわかる．R_1 の合計（0 ～ 5000 エピソード）では manip DQL+：-4.01×10^5，no manip DQL+：-3.46×10^5 ほどの違いである．これは基底関数の修正によって学習すべきパラメタが増加したことに起因するが，タスクのパフォーマンスの向上及び次項で述べる再利用性能の向上を考えれば大きな弊害ではない．

[5]スタート付近に留まる解などに陥った場合ステップコストの累積が非常に大きくなるので，R_0 ではなくゴールしたときに与えられる報酬のみをプロットしている．



(a) エピソード–ゴール報酬曲線.



(b) エピソード–回避行動の報酬曲線.

図 5.4 基底関数の修正法による分離学習結果（2 次元連続平面）．いずれも 20 回の実験における平均と ±1 の標準偏差を表示．時定数 50 試行の低域通過フィルタを適用したデータから計算した．no manip DQL+: DQL+($\lambda_0 = 0.9, \lambda_1 = 0$) を用いた分離学習（基底関数修正法は使わない），manip DQL+: DQL+($\lambda_0 = 0.9, \lambda_1 = 0$) を用いた分離学習（基底関数修正法を毎エピソード終了後に適用）．いずれも再利用を行わない．

5.4.2 回避行動の再利用の検証

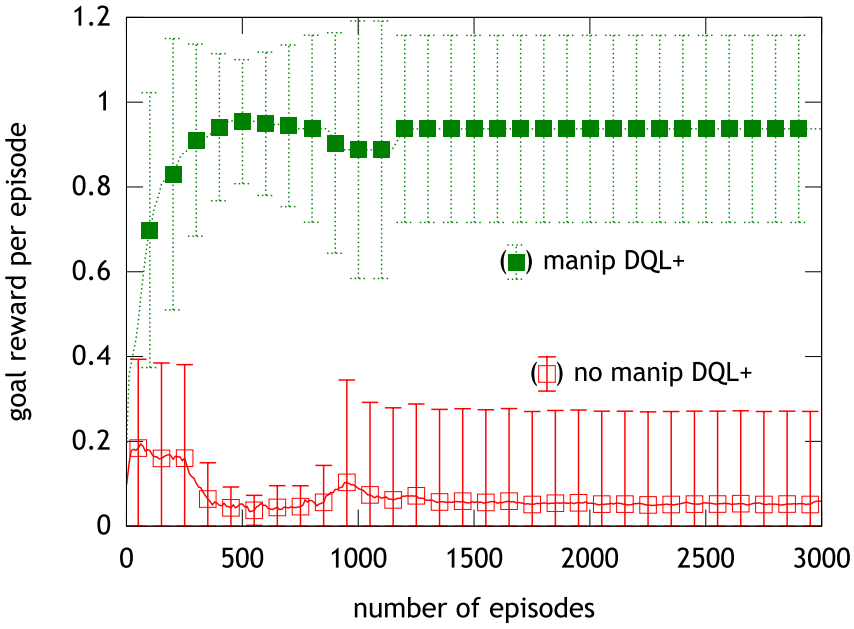
次にゴールの位置を変更して回避行動の行動価値関数を再利用し学習を行った．比較したアルゴリズムは

no manip DQL+　DQL+($\lambda_0 = 0.9, \lambda_1 = 0$) を用いた分離学習，5.4.1 項の no manip DQL+で学習した $\tilde{Q}_1$ を再利用

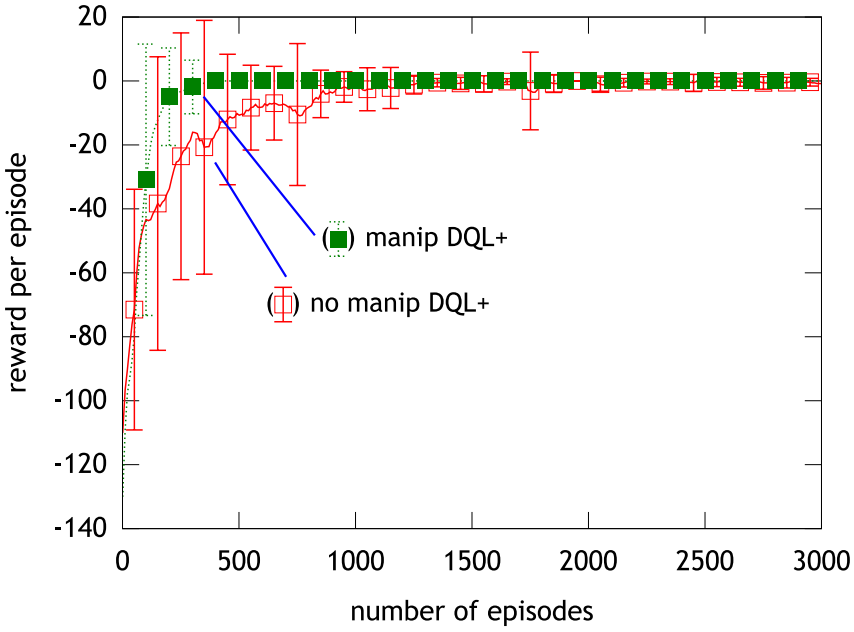
manip DQL+　DQL+($\lambda_0 = 0.9, \lambda_1 = 0$) を用いた分離学習，5.4.1 項の manip DQL+で学習した $\tilde{Q}_1$ を再利用（基底関数も再利用される）

である．ボルツマン選択の温度パラメタの初期値はいずれも $\tau_0 = 2$ とした．なお manip DQL+では 5.4.1 項の manip DQL+で獲得した基底関数がこのタスクに汎化することを示すため，新たな修正を行っていない．図 5.5(a) はゴールによって与えられる報酬をエピソードに対してプロットしたものであり，no manip DQL+ではほとんどの場合でゴールできていないことがわかるが manip DQL+ではそのような場合はわずかである．実際 no manip DQL+の場合 20 回の実験中 1 回しかゴールに到達する解を獲得できていないが，manip DQL+の場合 20 回中 19 回に改善されている．これは基底関数の修正を行わずに再利用した場合だと，学習の初期段階から既に図 5.1(a) のような問題に陥っているためであると考えられる．

図 5.5(b) はエピソードあたりの R_1 の合計をエピソードに対してプロットしたもので，前項の結果と違い manip DQL+の方が no manip DQL+より負の報酬を受け取る量が少なくなっていることがわかる．R_1 の合計（0 ～ 3000 エピソード）は manip DQL+：-6.78×10^3，no manip DQL+：-1.89×10^4 である．この理由は基底関数の修正法によって，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再利用の効果が大きかったからであると考えられる．



(a) エピソード–ゴール報酬曲線.



(b) エピソード–回避行動の報酬曲線.

図 5.5 基底関数の修正法による再利用結果（2 次元連続平面）．いずれも 20 回の実験における平均と ±1 の標準偏差を表示．時定数 50 試行の低域通過フィルタを適用したデータから計算した．no manip DQL+: DQL+($\lambda_0 = 0.9, \lambda_1 = 0$) を用いた分離学習（5.4.1 項の no manip DQL+で学習した $\tilde{Q}_1$ を再利用），manip DQL+: DQL+($\lambda_0 = 0.9, \lambda_1 = 0$) を用いた分離学習（5.4.1 項の manip DQL+で学習した $\tilde{Q}_1$ を再利用，基底関数も再利用される）．

5.5 運動学習のシミュレーション

5.5.1 跳　　躍

本章で提案する関数近似器修正法が運動学習においても有効に機能することを検証するため，比較的困難で，5.1 節で述べたような関数近似器による問題が生じている可能性のある跳躍に適用する．なおこの跳躍で学習される回避行動の行動価値関数は次項でサーブの学習に再利用されるため，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ラケットは跳躍においても取り付けられたままである．

ここでの跳躍はロボットの重心の垂直成分 z_{com} をできるだけ高い位置に移動させることを目的としたものであり，あるシンボル遷移（行動）に対する報酬 R_0 は次式で定義される：

$$R_0 = 200\left[z_{\text{com-max}} - z^{\text{old}}_{\text{com-max}}\right]_+. \tag{5.2}$$

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シンボル遷移内で時間積分したものをステップコストとして R_0 から引いている．

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

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

のいずれかを満たしていれば終了とする．(c) のみがタスクの成功である．

土台リンクの水平制御を行う場合

これまでの運動学習の実験と同様に土台リンクの水平制御（4.3 節）を行った状態で跳躍を学習させ，基底関数の修正法を用いた場合と用いない場合を比較した：

no manip DQL+　DQL+($\lambda_0 = 0.8, \lambda_1 = 0$) を用いた分離学習，基底関数修正法は使わない

manip DQL+　DQL+($\lambda_0 = 0.8, \lambda_1 = 0$) を用いた分離学習，5.2 節で述べた基底関数修正法を毎エピソード終了後に適用

ただしボルツマン選択の温度パラメタの初期値はいずれも $\tau_0 = 50$ とした．図 5.6 は各エピソードで得られる本来のタスクの報酬 R_0 の合計をエピソードに対してプロットしたものである．

基底関数の追加によるパラメタの増加があるにもかかわらず no manip DQL+よりも manip DQL+の方が学習が速く収束していることから，跳躍では 5.1 節で述べた関数近似器の問題による学習の遅延が生じており，関数近似器の修正法によってそれが改善されたと考えられる．なお no manip DQL+で報酬が一度降下しているのは，重心が下にあるため比較的安全な“地面を這い回る局所解”に陥ったことによって生じているステップコストが原因である．manip DQL+で最終的に（10000 エピソード後）獲得された跳躍の一例を図 5.7 に示す．

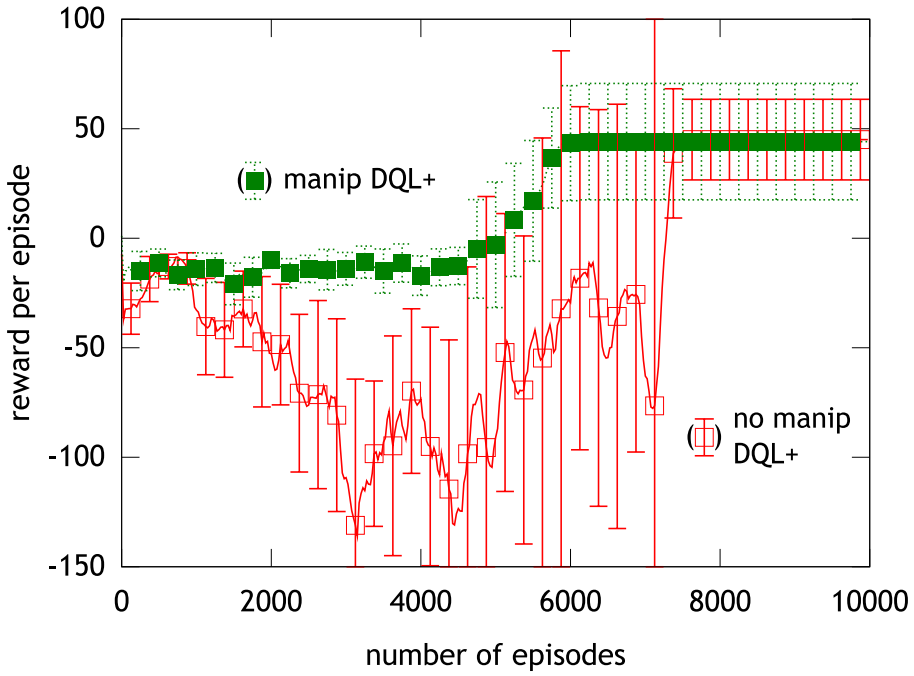


図 5.6 土台リンクの水平制御を行った場合の跳躍の分離学習結果（基底関数修正法を導入）：エピソード–本来のタスクの報酬曲線．7 回の実験における平均と ±1 の標準偏差．時定数 250 試行の低域通過フィルタを適用したデータから計算．no manip DQL+: DQL+($\lambda_0 = 0.8, \lambda_1 = 0$) を用いた分離学習（基底関数修正法は使わない），manip DQL+: DQL+($\lambda_0 = 0.8, \lambda_1 = 0$) を用いた分離学習（基底関数修正法を毎エピソード終了後に適用）．いずれも再利用を行わない．

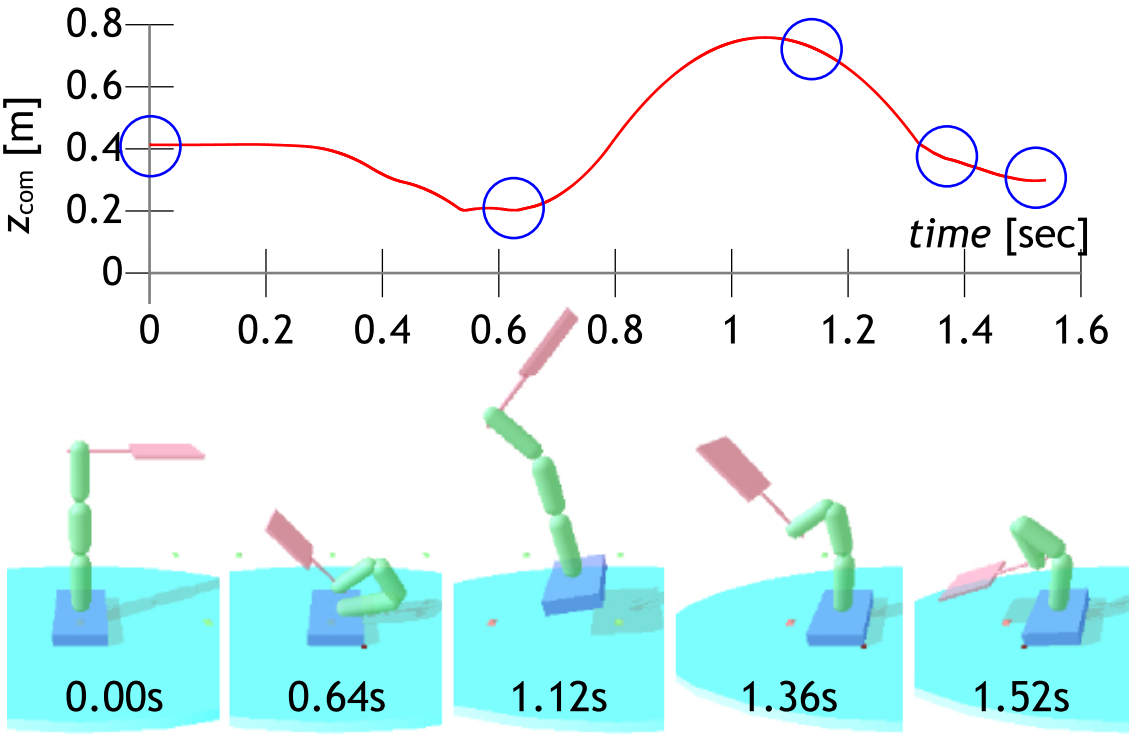


図 5.7 manip DQL+で獲得された跳躍における重心の垂直位置 z_{com} の軌道とスナップショット．軌道上の円の時刻はスナップショットの時刻と一致する．

土台リンクの水平制御を行わない場合

人間が跳躍するときには膝を伸ばすだけでなく足で地面を蹴ることによって加速させており，土台の水平制御はそのような動作を制限している．そこでここでは土台リンクの水平制御を行わずに跳躍を学習させ，関数近似器修正法の効果を調べる．

この実験でも基底関数の修正法を用いた場合（manip DQL+）と用いない場合（no manip DQL+）を比較した．ただしボルツマン選択の温度パラメタの初期値はいずれも $\tau_0 = 100$ とした．結果の図 5.8 は各エピソードで得られる本来のタスクの報酬 R_0 の合計をエピソードに対してプロットしたものである．manip DQL+の方が no manip DQL+よりも収束が速く，最終的に得られた報酬も 1.6 倍程度に向上している．no manip DQL+の収束が遅いのは 5.1 節で述べた関数近似器の問題による学習の遅延のためだと考えられ，分散が大きいのは同様の問題のために低い位置で安全に跳ぶなどの局所解に収束しているためである．いずれの問題も基底関数の修正によって改善されており，このタスクでは修正法が極めて効果的に機能していると言える．

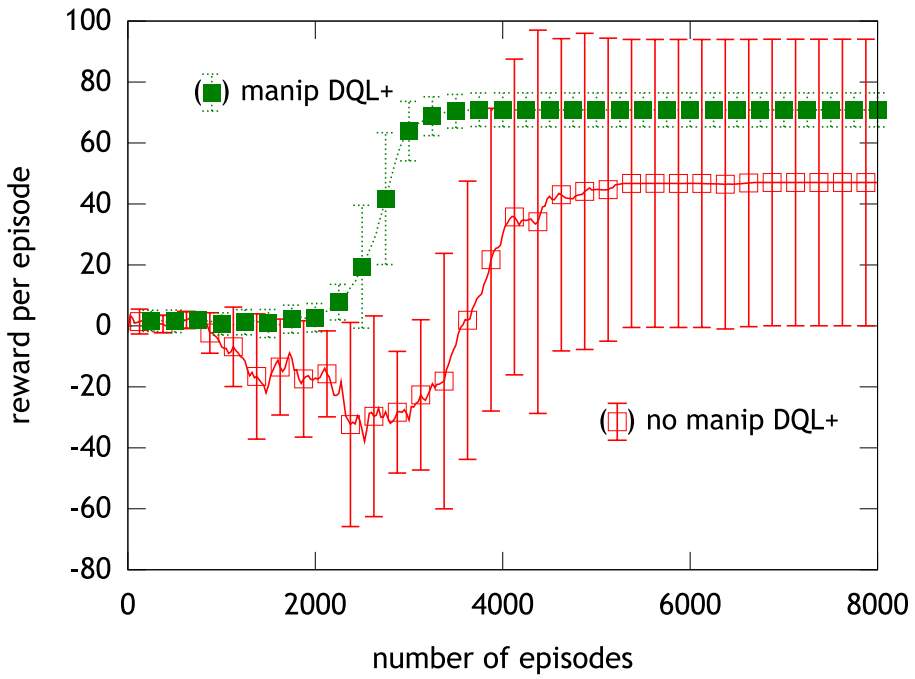


図 5.8 土台リンクの水平制御を行わない場合の跳躍の分離学習結果（基底関数修正法を導入）：エピソード–本来のタスクの報酬曲線．15 回の実験における平均と ±1 の標準偏差．時定数 250 試行の低域通過フィルタを適用したデータから計算．no manip DQL+: DQL+($\lambda_0 = 0.8, \lambda_1 = 0$) を用いた分離学習（基底関数修正法は使わない），manip DQL+: DQL+($\lambda_0 = 0.8, \lambda_1 = 0$) を用いた分離学習（基底関数修正法を毎エピソード終了後に適用）．いずれも再利用を行わない．

5.5.2 跳躍とサーブの間の再利用

次に本研究で提案する再利用メカニズムがより多くのタスク間で適用可能であることを示すために，前項の跳躍で学習した回避行動の行動価値関数がサーブの学習に再利用できるか検証した．ただし 4.3 節で述べた問題のため，土台リンクの水平制御を行った場合の跳躍を再利用しサーブの学習でも同様の制御を行う．reusing DQL+: 跳躍の結果を再利用した DQL+，no reusing DQL+: 再利用しない DQL+（いずれも関数近似器修正法を適用）を比較したところ，図 5.9 に示す報酬–エピソード曲線が得られた．ただしボルツマン選択の温度パラメタの初期値はいずれも $\tau_0 = 10$ とした．

reusing DQL+の方が no reusing DQL+よりも収束が速く，回避行動の報酬 R_1 の合計（0 ～ 1500 エピソード）はそれぞれ -1.98×10^5，-3.73×10^5 で再利用によって負の報酬を受け取る量が減少していることがわかる．したがって回避行動の再利用が，ロボットの運動学習における異なるタスク間でも実現できたと言える．

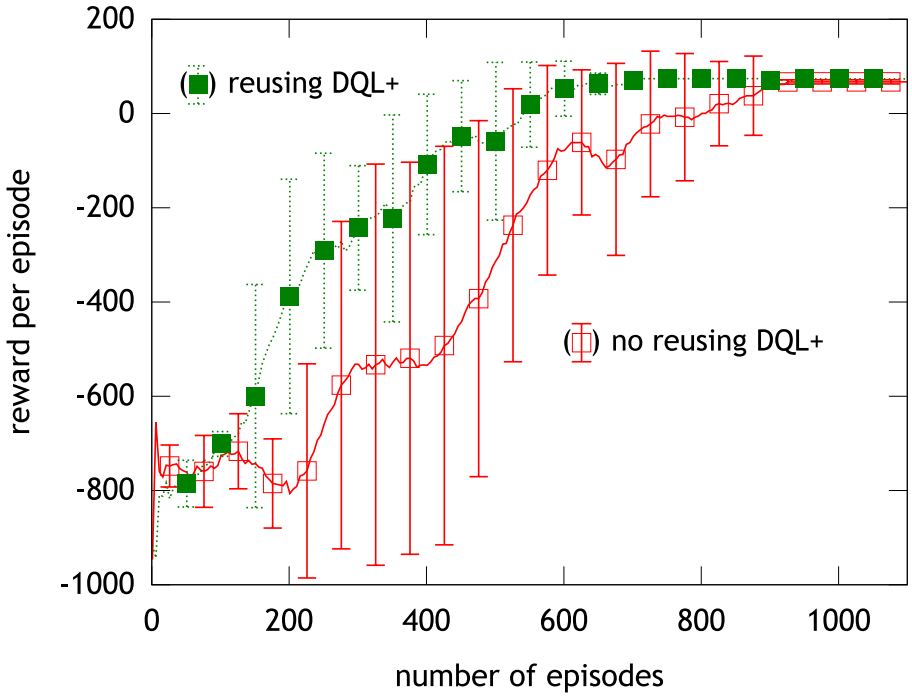


図 5.9 跳躍とサーブの間の再利用結果：エピソード–報酬曲線．8 回の実験における平均と ±1 の標準偏差．時定数 50 試行の低域通過フィルタを適用したデータから計算．reusing DQL+: 跳躍の結果を再利用した DQL+，no reusing DQL+: 再利用しない DQL+（いずれも関数近似器修正法を適用）．

5.5.3 遅いサーブと速いサーブの間の再利用（2）

4.3.3 項で述べた遅いサーブで得た回避行動を学習する行動価値関数を速いサーブの学習で再利用する実験に関数近似器修正法を適用し，手法の効果を検証する．実験は以下の流れで行った．

（1） DQL+に関数近似器修正法を適用して遅いサーブを学習．

（2） reusing DQL+ (manip): （1）の結果の $\tilde{Q}_1$ を再利用した DQL+，no reusing DQL+ (manip): 再利用しない DQL+（いずれも関数近似器修正法を適用）のそれぞれに速いサーブを学習させる．

ただし（2）ではボルツマン選択の温度パラメタの初期値はいずれも $\tau_0 = 10$ とした．この結果，図 5.10 に示すエピソード–報酬曲線グラフを得た．比較として関数近似器修正法を適用しない場合の実験結果を，再利用ありの DQL+: reusing DQL+ (no manip) と再利用なしの DQL+: no reusing DQL+ (no manip) として表示してある．

reusing DQL+ (manip) の方が no reusing DQL+ (manip) よりも収束が速く，回避行動の報酬 R_1 の合計（0 ～ 1500 エピソード）はそれぞれ -2.46×10^5，-3.52×10^5 で再利用によって負の報酬を受け取る量が減少していることがわかる．しかしこれらは関数近似器修正法を用いない場合に比べると収束が遅くなっている．これはサーブのように比較的達成が容易なタスクだと 5.1 節で述べたような関数近似器の問題による学習の遅延が起こらないだけでなく，基底関数の修正で生じるパラメタの増加が逆に学習時間の増加につながっているためであると考えられる．

そこでより幅広い比較を行うために上記（2）で再利用はするが新たに基底関数の修正を行わない設定の実験を追加し（reusing DQL+ (gen)），ほかの実験との結果を比較した．reusing DQL+ (gen) は遅いサーブの学習で生成された基底関数が，どの程度速いサーブの学習に対する基底関数に汎化するかを調べていることになる．表 5.1 にそれぞれのアルゴリズムにおける基底関数の数の変化と回避行動の報酬 R_1 を全エピソードにわたって合計したものを示す．なお基底関数の数はそれぞれの実験で平均したものであり，再利用しない場合の初期状態は 101 個である．

この結果から以下がわかる．

- 同じアルゴリズムなら再利用によって R_1 を受け取る量は減少する．
- 速いサーブの場合は関数近似器の修正法の適用によって R_1 を受け取る量は増加する．
- 速いサーブで関数近似器の修正法を適用しない場合，遅いサーブの学習で関数近似器の修正法を用いて得られた $\tilde{Q}_1$ の再利用（reusing DQL+ (gen)）は用いずに得られた $\tilde{Q}_1$ の再利用（reusing DQL+ (no manip)）と比べて R_1 の合計が同程度である．ただし前者

の方が学習すべきパラメタが多いためある程度基底関数が汎化していると考えられる．これより 5.1 節で述べたような問題が生じている困難なタスクの学習では基底関数の修正法を適用し，比較的簡単なタスク（速いサーブなど）の学習では基底関数の修正法を適用しないというアプローチが妥当であると考えられる．

表 5.1　遅いサーブと速いサーブの間の再利用結果

アルゴリズム	基底関数の数の変化	R_1 の合計（全エピソード）
no reusing DQL+ (manip)	101 → 103.3	-3.52×10^5
reusing DQL+ (manip)	118 → 122.8	-2.46×10^5
no reusing DQL+ (no manip)	101 → 101	-2.32×10^5
reusing DQL+ (no manip)	101 → 101	-1.95×10^5
reusing DQL+ (gen)	118 → 118	-1.93×10^5

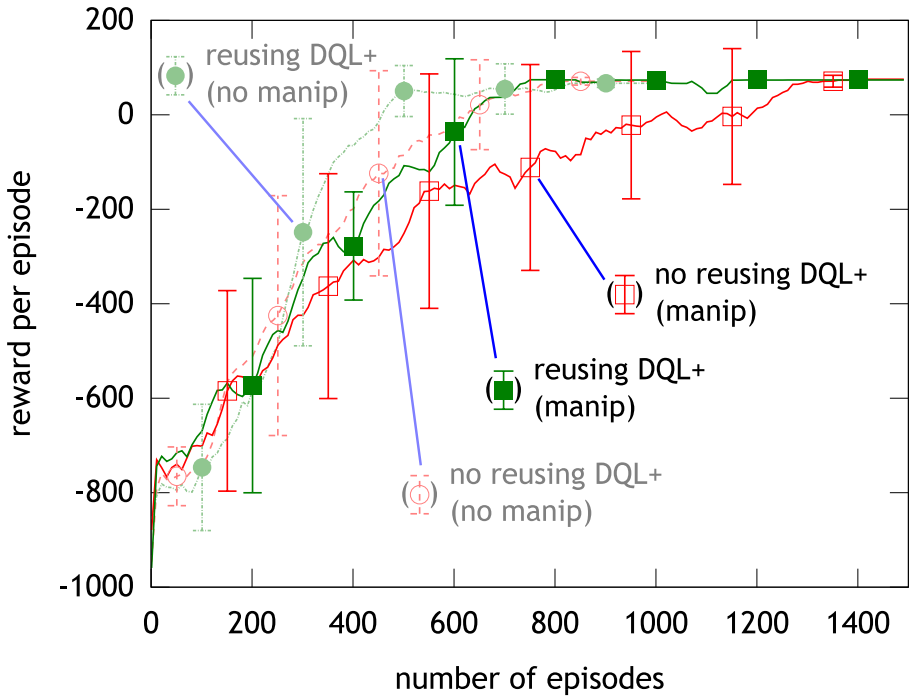


図 5.10　遅いサーブと速いサーブの間の再利用結果（基底関数修正法を適用）：エピソード–報酬曲線．9 回の実験における平均と ±1 の標準偏差．時定数 50 試行の低域通過フィルタを適用したデータから計算．reusing DQL+ (manip): DQL+に関数近似器修正法を適用して遅いサーブを学習，その結果を再利用した DQL+，no reusing DQL+ (manip): 再利用しない DQL+（いずれも関数近似器修正法を適用）．reusing DQL+ (no manip), no reusing DQL+ (no manip): 関数近似器修正法を用いない場合の実験結果（4.3.3 項と同条件）．

6. 結　　論

本研究では回避行動の行動価値関数を分離学習し新たなタスクの学習において再利用するメカニズムを備えた強化学習手法を提案し，多関節ロボットの全身運動学習に応用する手法を示した．さらに連続状態空間の学習で関数近似器の近似精度によって生じる学習の遅延や局所解への収束を改善するために,「分離誤差」に基づく基底関数修正法を提案した．一連の手法は格子状空間及び 2 次元連続平面での移動タスク，サーブと跳躍に適用され，有効に機能することが確認された．

既存の手法との比較として，回避行動の分離学習と再利用のための行動価値関数学習手法として提案した ΔSarsa と DQL+を Russell らが提案する行動価値関数の分離学習手法 [14] とともにシミュレーション実験で比較した．結果 Russell らの手法は分離学習は適切に行えるが再利用には適当でないという 2.5.1 項の分析を裏付けるものであった．これに対し提案手法の DQL+は格子状空間，2 次元連続平面，運動学習のいずれのタスクでも高い再利用の性能を示し，一部の実験では再利用なしの分離学習においても Russell らの手法よりも優れた結果となることもわかった．DQL+ではトレードオフが存在する場合には回避行動を優先するため最適な行動が獲得されないこともあるが，回避すべき行動が本研究で想定しているように必ず避けられるべきものである場合には有効に機能する．一方 ΔSarsa は格子状空間では分離学習・再利用ともに最も優れているという理論を支持する結果となったが，2 次元連続平面や運動学習ではほかの手法よりもやや劣っていた．これはエリジビリティ・トレースが適切でないこと（3.2 節）や関数近似器を用いた行動価値関数の Q-learning による学習の不安定さ（3.3.2 項）などが原因であると考えられる．しかしこれらの問題を解決できれば分離学習及び再利用のための手法として最も優れたものになる可能性があり，今後検討したい．

運動学習における回避行動の再利用では，格子状空間や 2 次元連続平面における再利用ほど効率的ではなかった．すなわち再利用による転倒ダメージの減少がそれほど大きくなかった．この原因としては高次元状態空間における学習の遅さへの対応（関数近似器を “粗く” するなど）とタスク達成に必要な関数近似器の精度の間のバランスを取ることの困難さなどが考えられる．しかし 4 章で述べた運動学習における基底関数をロボットのダイナミクスから生成する方法や 5 章で提案した関数近似器修正手法はこのような問題の解決を試みたものであり，5 章の実験で示され

ているように一部の事例では大幅な改善に成功している．これをより安定に実現することは，今後の課題としたい．

全身運動のシミュレーション実験では土台リンクを水平に保つ制御を用いており，これはロボット依存の方法であることや，土台リンクを積極的に使った跳躍（人間が跳ぶときに足で地面を蹴るような跳躍）ができなくなると言った問題があるなど汎用的とは言いがたい．この問題を解決するには図 4.1 に示したロボットの運動学習への適用方法のうち，本研究で特に重点をおいた基底関数の構成以外にも行動の定義などを改善する必要があると考えられ，今後検討を深めたい．

本研究で提案した DQL+や ΔSarsa はすべての目的に共通の動作知識（Type A）の再利用を実現するものであり，転倒などの回避行動，より一般的に言うと状態行動の制約となるようなタスクに対して適用可能である．逆にジャンピング・サーブの学習で跳躍やサーブを再利用すると言った，再利用する場面が限定される動作知識（Type B）の再利用に適用することは困難である．これを実現するためには学習した動作知識をいつどのように使うか推論する必要があり，今後提案手法を発展させ実現して行く予定である．

人間やマウスなどの哺乳類においても異なる報酬のそれぞれに対して個別の学習器を持つ場合があることが，生理学的な実験から示唆されている．例えば Kobayakawa ら [21] は後天的に学習されると考えられていた匂いに対する行動には先天的に備わっているものもあり，それらは別々の神経回路によって処理されていることを実験的に示した．このような知見は本論文で述べた手法の妥当性を支持すると考えられ，さらに “先天的に備わっている学習器” は類似の構造を持つロボット間で回避行動を再利用することに相当するなど，生理学的な知見からロボットの運動学習メカニズムを発展させるというアプローチも期待される．

謝　　辞

本研究を進めるにあたってご指導頂きました，ATR 脳情報研究所 杉本 徳和 研究員に深く感謝致します．川人 光男 教授 にはミーティング等を通してさまざまな角度からご助言，ご指導を頂きました．深く感謝致します．

小笠原 司 教授，ATR 脳情報研究所 森本 淳 研究員にはたびたび論文のチェックをして頂き，数多くの重要な意見を頂きました．深く感謝致します．

英文のチェックをして頂いた Ganesh Gowrishankar 氏，数多くの議論をして頂いた 松原 崇充 氏に深く感謝致します．

たびたびご助言を頂いた ATR 脳情報研究所 春野 雅彦 主任研究員，日々お世話になった野中 晶子さんを始め ATR 脳情報研究所のスタッフの方々に深く感謝致します．

ロボティクス講座 栗田 雄一 助教，博士後期過程の先輩方には本論文のチェックをして頂きました．深く感謝致します．

計算神経科学講座，ロボティクス講座の皆様にも，日々の研究生活の中でさまざまな助言，支援をして頂きました．深く感謝致します．

最後になりましたが，正・副指導教員を引き受けて頂いた小笠原 司 教授，川人 光男 教授，杉本 謙二 教授，神谷 之康 准教授，松本 吉央 客員准教授に改めて感謝致します．

参 考 文 献

[1] J. Morimoto and K. Doya: “Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning”, Robotics and Autonomous Systems, **36**, 1, pp. 37–51 (31 July 2001).

[2] 木村, 山下, 小林：“強化学習による 4 足ロボットの歩行動作獲得”, 電気学会電子情報システム部門誌, **122-C**, 3, pp. 330–337 (2002).

[3] 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).

[4] 小林, 藤井, 細江：“一次元空間への低次元化写像を用いた対象物操作の強化学習”, 計測自動制御学会論文集, **42**, 7, pp. 814–821 (2006).

[5] T. Matsubara, J. Morimoto, J. Nakanishi, S.-H. Hyon, J. G. Hale and G. Cheng: “Learning to acquire whole-body humanoid CoM movements to achieve dynamic tasks”, ICRA, pp. 2688–2693 (2007).

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

[7] 高橋, 浅田：“階層型学習機構における状態行動空間の構成”, 日本ロボット学会誌, **21**, 2, pp. 38–45 (2003).

[8] F. Fernández and M. Veloso: “Probabilistic policy reuse in a reinforcement learning agent”, AAMAS ’06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, New York, NY, USA, ACM Press, pp. 720–727 (2006).

[9] R. Sutton and A. Barto: “Reinforcement Learning: An Introduction”, MIT Press, Cambridge, MA (1998).

[10] 銅谷, 森本, 鮫島：“強化学習と最適制御（「システム制御理論の新領域特集号」）”, システム/制御/情報 : システム制御情報学会誌, **45**, 4, pp. 186–196 (2001).

[11] L. C. Baird: “Residual algorithms: Reinforcement learning with function approximation”, International Conference on Machine Learning, pp. 30–37 (1995).

[12] C. J. C. H. Watkins: "Learning from Delayed Rewards", PhD thesis, Cambridge University (1989).

[13] G. A. Rummery and M. Niranjan: "On-line Q-learning using connectionist systems", Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department (1994).

[14] 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).

[15] C. J. C. H. Watkins and P. Dayan: "Technical note: Q-learning", Machine Learning, **8**, 3-4, pp. 279–292 (1992).

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

[17] J. Peng and R. J. Williams: "Incremental multi-step Q-learning", International Conference on Machine Learning, pp. 226–232 (1994).

[18] Y. Kuniyoshi, , Y. Ohmura, K. Terada, T. Yamamoto and A. Nagakubo: "Exploiting the global dynamics structure of whole-body humanoid motion - getting the "knack" of roll-and-rise motion", The Eleventh International Symposium on Robotics Research, pp. 385–394 (2005).

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

[20] R. Smith: "Open dynamics engine (ODE)" (2006). http://www.ode.org.

[21] K. Kobayakawa, R. Kobayakawa, H. Matsumoto, Y. Oka, T. Imai, M. Ikawa, M. Okabe, T. Ikeda, S. Itohara, T. Kikusui, K. Mori and H. Sakano: "Innate versus learned odour processing in the mouse olfactory bulb", Nature, **450**, 7169, pp. 503–508 (2007).

[22] J. N. Tsitsiklis and B. V. Roy: "An analysis of temporal-difference learning with function approximation", IEEE Transactions on Automatic Control, **42**, 5, pp. 674–690 (1997).

[23] R. S. Sutton: "Learning to predict by the methods of temporal differences", Machine Learning, **3**, 1, pp. 9–44 (1988).

[24] N. Ueda, R. Nakano, Z. Ghahramani and G. E. Hinton: "SMEM algorithm for mixture models", Neural Computation, **12**, 9, pp. 2109–2128 (2000).

付　　録

A. 式（2.17）⇒式（2.11）の証明

すべての $s \in \mathcal{S}$ について以下の議論が成り立つ.

(1) $\max_{a \in \mathcal{A}} Q_0^\star(s,a) > 0$ の場合

式（2.17）から

$$^\forall a \in \arg\max_{a' \in \mathcal{A}} Q_0^\star(s,a') : \ Q_1^\star(s,a) = 0 \tag{A.1}$$

が成立し，さらに $^\forall a \in \mathcal{A} : Q_1^\star(s,a) \leqslant 0$ なので $\max_{a \in \mathcal{A}} Q_1^\star(s,a) = 0$ となる．よって式（2.11）の左辺は

$$\sum_{i \in \{0,1\}} \max_{a \in \mathcal{A}} Q_i^\star(s,a) = \max_{a \in \mathcal{A}} Q_0^\star(s,a) \tag{A.2}$$

となる．一方式（A.1）かつ $^\forall a \in \mathcal{A} : Q_1^\star(s,a) \leqslant 0$ より式（2.11）の右辺は

$$\max_{a \in \mathcal{A}} \sum_{i \in \{0,1\}} Q_i^\star(s,a) = \max_{a \in \mathcal{A}} Q_0^\star(s,a) \tag{A.3}$$

となって式（2.11）が成立.

(2) $\max_{a \in \mathcal{A}} Q_0^\star(s,a) = 0$ の場合

式（2.11）の左辺は

$$\sum_{i \in \{0,1\}} \max_{a \in \mathcal{A}} Q_i^\star(s,a) = \max_{a \in \mathcal{A}} Q_1^\star(s,a) \tag{A.4}$$

となる．一方 $^\forall a \in \mathcal{A} : Q_0^\star(s,a) \geqslant 0$ なので $^\forall a \in \mathcal{A} : Q_0^\star(s,a) = 0$ が成立．よって式（2.11）の右辺は

$$\max_{a \in \mathcal{A}} \sum_{i \in \{0,1\}} Q_i^\star(s,a) = \max_{a \in \mathcal{A}} Q_1^\star(s,a) \tag{A.5}$$

となって式（2.11）が成立.

以上より式（2.11）が成立することが示された.

B. エリジビリティ・トレースを用いた更新則

B.1 ΔSarsa(λ_0, λ_1)

行動価値関数を近似した式（*3.3*）のパラメタ更新に ΔSarsa にエリジビリティ・トレースを導入した ΔSarsa(λ_0, λ_1) を用いる場合の更新則を以下に示す．これは $\tilde{Q}_0$ の学習には Sarsa(λ) と同じエリジビリティ・トレースを，$\tilde{Q}_1$ の学習には Peng らの Q(λ)-learning を適用したものである．なお 3.2 節で述べたように $\tilde{Q}_0$ の更新に Sarsa(λ) と同じエリジビリティ・トレースを適用するのは正確では無く，ここで用いている Sarsa(λ) と同じ更新則は近似である．

Algorithm 2: ΔSarsa(λ_0, λ_1)

$i \leftarrow 0$ /* primary task */

$e_{i,t} \leftarrow R_0(x_t, a_t) + \gamma \tilde{Q}_{i,t}(x_{t+1}, a_{t+1}) - \gamma\big(\max_{a \in \mathcal{A}} \tilde{Q}_{1,t}(x_{t+1}, a) - \tilde{Q}_{1,t}(x_{t+1}, a_{t+1})\big)$
$\quad - \tilde{Q}_{i,t}(x_t, a_t)$

for each $s \in \mathcal{S}$ **do**

$Tr_i(s, a_t) \leftarrow Tr_i(s, a_t) + N_s(x_t)$

end for

for each $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**

$Q_{i,t+1}(s, a) \leftarrow Q_{i,t}(s, a) + \alpha Tr_i(s, a) e_{i,t}$

$Tr_i(s, a) \leftarrow (\gamma \lambda_i) Tr_i(s, a)$

end for

$i \leftarrow 1$ /* avoidance task */

$e_{i,t} \leftarrow R_i(x_t, a_t) + \gamma \tilde{V}_{i,t}(x_{t+1}) - \tilde{V}_{i,t}(x_t)$

$e'_{i,t} \leftarrow R_i(x_t, a_t) + \gamma \tilde{V}_{i,t}(x_{t+1}) - \tilde{Q}_{i,t}(x_t, a_t)$

for each $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**

$Tr_i(s, a) \leftarrow (\gamma \lambda_i) Tr_i(s, a)$

$Q_{i,t+1}(s, a) \leftarrow Q_{i,t}(s, a) + \alpha Tr_i(s, a) e_{i,t}$

end for

for each $s \in \mathcal{S}$ **do**

$Q_{i,t+1}(s, a_t) \leftarrow Q_{i,t+1}(s, a_t) + \alpha e'_{i,t} N_s(x_t)$

$Tr_i(s, a_t) \leftarrow Tr_i(s, a_t) + N_s(x_t)$

end for

ここで $\tilde{V}_i(x)$ はグリーディ方策における状態価値関数の近似であり，$\tilde{V}_i(x) = \max_{a \in \mathcal{A}} \tilde{Q}_i(x, a)$ で与えられる．

B.2 DQL+(λ_0, λ_1)

行動価値関数を近似した式（*3.3*）のパラメタ更新に DQL+にエリジビリティ・トレースを導入した DQL+(λ_0, λ_1) を用いる場合，その更新則は以下のようになる（こ

のアルゴリズムは [17] [22] [23] を参考に，Peng らの Q(λ)-learning を式（*3.3*）の更新に適用しさらに 2.5.3 項で述べた DQL+の更新則を適用して導出した）：

Algorithm 3: DQL+(λ_0, λ_1)

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

ここで $Tr_i(s, a)$ は activity trace と呼ばれ，行動の履歴を表す．$\tilde{V}_i(x)$ はグリーディ方策における状態価値関数の近似であり，$\tilde{V}_i(x) = \max_{a \in \mathcal{A}} \tilde{Q}_i(x, a)$ で与えられる．

C. ダイナミクスの学習

ロボットのダイナミクスは，

$$A_s(\xi)\ddot{\xi} + b_s(\xi, \dot{\xi}) = \begin{bmatrix} \mathbf{0} \\ u \end{bmatrix} \qquad (C.1)$$

によって表されるとする．ここで $\xi \in \mathbb{R}^{D_X \times 1}$ はベースリンク（土台）の位置・姿勢と関節角 $q \in \mathbb{R}^{D_Q \times 1}$ をまとめたベクトル（一般化座標），$u \in \mathbb{R}^{D_Q \times 1}$ は関節トルク，$A_s(\xi)$ は慣性行列，$b_s(\xi, \dot{\xi})$ はコリオリ力・遠心力項や重力項をまとめたベクトルである．ロボットは直接トルクを指定して駆動できるものとする[6]．このダイナミ

[6]ただし u を関節に取り付けられたモータの電圧と見ることもできる．状態が $(\xi, \dot{\xi})$ としてセンシングでき，入力が u でかつそれらの間のダイナミクスが式（*C.1*）で表されるシステムなら，本手法はそのまま適用可能である．

クスを

$$\frac{\dot{q}(t+\Delta t)-\dot{q}(t)}{\Delta t} = B\begin{bmatrix}\xi\\ \dot{\xi}\end{bmatrix} + d + Au \qquad (C.2)$$

のように線形化・離散化したもの ($A \in \mathbb{R}^{D_Q \times D_Q}$, $B \in \mathbb{R}^{D_Q \times 2D_X}$, $d \in \mathbb{R}^{D_Q \times 1}$) を NGnet の各基底関数に対応させて，ロボットの全状態空間に対応した非線形ダイナミクスを学習させる．式（C.1）の非線形項は ξ と $\dot{\xi}$ のみに依存するから，ロボットの状態は $x = [\xi^\top \ \dot{\xi}^\top]^\top \in \mathbb{R}^{2D_X \times 1}$ である．

NGnet のパラメタは適当にロボットを動かしてサンプルを取り，それをもとに EM アルゴリズムで最適化する．EM アルゴリズムだけだとユニット数が決められない，局所解から脱出できない，と言った問題があるため Ueda らの SMEM [24] をロボットのダイナミクス学習用に変更したものを用いている．なお 4 章，5 章の運動学習の実験では獲得されるユニット数が 100 個前後になるようにパラメタを設定した．

D. ロボットの行動

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

$$\int_0^{T_\mathrm{f}} \left\| \frac{\mathrm{d}^3 q}{\mathrm{d}t^3} \right\|^2 \mathrm{d}t \to \min \qquad (D.1)$$

の解であり，変分法を用いて解くと 5 次方程式が得られる．すなわち関節 j の関節角軌道を $q_j(t) = \sum_{k=0}^{5} c_{jk} t^k$ として境界条件を $q_j(0) = q_j^\mathrm{i}$, $\dot{q}_j(0) = \dot{q}_j^\mathrm{i}$, $\ddot{q}_j(0) = \ddot{q}_j^\mathrm{i}$, $q_j(T_\mathrm{f}) = q_j^\mathrm{f}$, $\dot{q}_j(T_\mathrm{f}) = \dot{q}_j^\mathrm{f}$, $\ddot{q}_j(T_\mathrm{f}) = \ddot{q}_j^\mathrm{f}$ とすると，

$$c_{j0} = q_j^\mathrm{i}, \quad c_{j1} = \dot{q}_j^\mathrm{i}, \quad c_{j2} = \frac{1}{2}\ddot{q}_j^\mathrm{i} \qquad (D.2a)$$

$$A_1 \triangleq \frac{1}{T_\mathrm{f}^3}\left(q_j^\mathrm{f} - (c_{j0} + c_{j1}T_\mathrm{f} + c_{j2}T_\mathrm{f}^2)\right) \qquad (D.2b)$$

$$A_2 \triangleq \frac{1}{T_\mathrm{f}^2}\left(\dot{q}_j^\mathrm{f} - (c_{j1} + 2c_{j2}T_\mathrm{f})\right) \qquad (D.2c)$$

$$A_3 \triangleq \frac{1}{T_\mathrm{f}}\left(\ddot{q}_j^\mathrm{f} - 2c_{j2}\right) \qquad (D.2d)$$

$$c_{j3} = 10A_1 - 4A_2 + \frac{A_3}{2} \qquad (D.2e)$$

$$c_{j4} = \frac{1}{T_\mathrm{f}}\left(-15A_1 + 7A_2 - A_3\right) \tag{D.2f}$$

$$c_{j5} = \frac{1}{T_\mathrm{f}^2}\left(6A_1 - 3A_2 + \frac{A_3}{2}\right) \tag{D.2g}$$

と計算される．具体的には境界条件として現在の関節状態 $(q_j^\mathrm{i},\ \dot{q}_j^\mathrm{i})$ 及び目標シンボルの関節状態 $(q_j^\mathrm{f},\ \dot{q}_j^\mathrm{f})$ を用い，加速度はゼロ $\ddot{q}_j^\mathrm{i} = \ddot{q}_j^\mathrm{f} = 0$ として求める．行動の時間幅 T_f はその大きさによって動作の俊敏さが変化するため，

$$\mathcal{I} = \{0.2\|q^\mathrm{f} - q^\mathrm{i}\|,\ 0.4\|q^\mathrm{f} - q^\mathrm{i}\|,\ 0.8\|q^\mathrm{f} - q^\mathrm{i}\|\} \tag{D.3}$$

の3段階から選択することにした．目標関節角と現在の関節角の差のノルム $\|q^\mathrm{f} - q^\mathrm{i}\|$ を掛けているのは目標シンボルの違いによる動作の俊敏さの差異（T_f が同じなら遠いシンボルほど俊敏な動作になる）を軽減するためである．したがって強化学習エージェントの行動集合は $\mathcal{A} = \mathcal{S} \times \mathcal{I}$ であり，離散行動である．なお $\mathcal{I}$ を連続にすることも可能だが，本稿では簡単のため離散とした．

E. 分離誤差に基づく基底関数修正法の詳細

5.2 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．

E.1 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 $\{\mu_s | s \in \mathcal{S}\}$ と2つの基底関数の中間 $\{(\mu_{s_1} + \mu_{s_2})/2 | s_1,\ s_2 \in \mathcal{S}\}$ からランダムに選択することにする．

E.2 基底関数の追加方法

基底関数を追加する際に4章や3.3節で述べたように行動が基底関数の中心状態 μ に依存する場合，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5.1(b)のような分割として Sato ら [16] の “unit division” を参考にし

た分割

$$\mu'_s = \mu_s, \quad \mu_{s_{\text{new}}} = \mu_s + \sqrt{\lambda_{s,1}} u_{s,1} \tag{E.1a}$$

$$\Sigma'^{-1}_s = \Sigma^{-1}_{s_{\text{new}}} = 4\lambda^{-1}_{s,1} u_{s,1} u^\top_{s,1} + \sum_{n=2} \lambda^{-1}_{s,n} u_{s,n} u^\top_{s,n} \tag{E.1b}$$

を用いる（$\lambda_{s,n}$, $u_{s,n}$ 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 $\mathcal{S}_{\text{prn}} = \{s_i\}$ をもとにして新たな基底関数の生成を行う：

$$\mu'_{s_i} = \mu_{s_i}, \quad \Sigma'^{-1}_{s_i} = \sum_{n=1} \lambda^{-1}_{s_i,n} \beta^{-2}_{s_i,n} u_{s_i,n} u^\top_{s_i,n} \tag{E.2a}$$

$$\beta_{s_i,n} \triangleq 1 - \frac{1}{|\mathcal{S}_{\text{prn}}|} \left| u^\top_{s_i,n} \frac{x - \mu_{s_i}}{\|x - \mu_{s_i}\|} \right| \tag{E.2b}$$

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

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

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（*E.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（*E.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

$$\texttt{for each } a \in \mathcal{A}:$$
$$\texttt{if} \prod_{i \in \{0,1\}} Q_i(s,a) \neq 0: \;\; Q_0(s,a) \leftarrow 0, \;\; Q_1(s,a) \leftarrow 0$$

によって式（*2.17*）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．

E.3 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．