計測自動制御学会東北支部 第206回研究集会 (2002.12.10)
資料番号 206-7

局所重み付き回帰手法を用いた強化学習

Reinforcement learning using the locally weighted regression method

○小池 健太，李 海妍，阿部 健一

○Kenta Koike, Haeyeon Lee, Kenichi Abe

東北大学工学研究科

School of Engineering, Tohoku University

キーワード： 関数近似(function approximation)，局所重み付き回帰(locally weighted regression)，強化学習(reinforcement learning)，Q学習(Q-learning)，行動価値関数(action value function)

連絡先： 〒980-8579 仙台市青葉区荒巻字青葉 東北大学 大学院 工学研究科 電気・通信工学専攻 阿部研究室
小池健太，Tel.: (022)217-7074，Fax.: (022)263-9290，E-mail: koike@abe.ecei.tohoku.ac.jp

1. はじめに

近年，人間が制御則を組み込んだロボットに関しては，高度な動作を行うことも可能となってきている．しかし，将来人間と共存し，人間の代わりとなって働くようなシステムを実現するためには，固定された制御則を用いるだけでなく，動的に変化する環境の中で，ロボット自身が学習によって制御則を獲得することが要求される．このような要求に対するひとつのアプローチとして，強化学習[1)2)]が注目を集めている．強化学習とは，模範となる出力系列が与えられていなくても，最終的な課題の達成度に応じた評価信号から，望ましい制御則を獲得する学習の枠組みである．

動的な制御タスクは，強化学習手法の典型的な応用例である．しかしながら，これらのタスクの多くは本来，連続的な状態・行動空間を有している．それに対して，既存の多くの強化学習アルゴリズムは離散状態，離散行動を仮定しており，単純にこれらのタスクに応用することができない．多くの場合，連続空間は離散化され，新たに離散バージョンの問題として捉えられる．しかしながら，適切な離散化が行われなかった場合，問題の中に隠れ状態を作り出し，最適政策の学習を不可能にする可能性がある．また，細かく離散化しすぎた場合，一般化の能力を失い，学習に要するトレーニングデータを増大させることになる．これは特に，状態が高次元の環境において顕著であり，状態数は状態の次元に対して指数関数的に増加する．単にルックアップテーブルを更新する手法では，比較的単純な問題でさえ学習の失敗を招く例が報告されている[3)]．

本研究では，連続な状態空間をもつ問題に対して，離散化を行わずに効率よく学習を行うことを目的とする．離散化を行う方法以外のアプローチとしては，関数近似手法が挙げられる．線形関数

や，ニューラルネットワーク（NN）などの非線形関数を用いたものなど様々な手法が提案されている．線形関数近似は一定の条件のもとで収束性が証明されているが，単一の線形関数のみでは能力に限界がある．一方，NNを用いた場合については，活性化関数としてシグモイド関数を用いると，各パラメータの変化が出力に大域的な影響を及ぼすため，環境を探査しながらオンラインで状態空間を構築しながら学習を行うのに適していない．そこで動径基底関数(RBF)のような局所的な基底関数を用いることが考えられるが，基底関数がカバーしている範囲外の領域では汎化が行われないため，高次元空間での学習には非常に多くの素子と学習サンプルが必要とされるなどの問題を含む．

本研究では，局所ごとに関数近似を行い，複数の局所モデルの重み付き平均を出力とする回帰手法に着目した．まず，この回帰手法そのものの性能を評価し，さらに実際に連続状態を有する制御タスクにおける強化学習に適用を行い，シミュレーションにより検証を行った．

2. 局所重み付き回帰手法(LWR)

2.1 アルゴリズム

連続な状態をもつ環境下では，無限にある状態全てを実際に訪問することは不可能である．そこで，以前に経験した状態から，まだ経験していない状態へと一般化する手法が必要となる．この場合の一般化手法は関数近似と呼ばれ，線形・非線形関数近似，決定木など種々の手法が利用可能である．本研究では局所重み付き回帰手法(Locally Weighted Regression)[4]と呼ばれる手法を採用する．この手法は，複数の局所モデルを重複を許して配置し(Fig. 1(a))，その領域の活性度に応じた重み付き平均を予測値とするものである(Fig. 1(b))．局所モデルとしてはRBFを用いることもできるが，一般化性能と計算量，取り扱いの容易さの点から

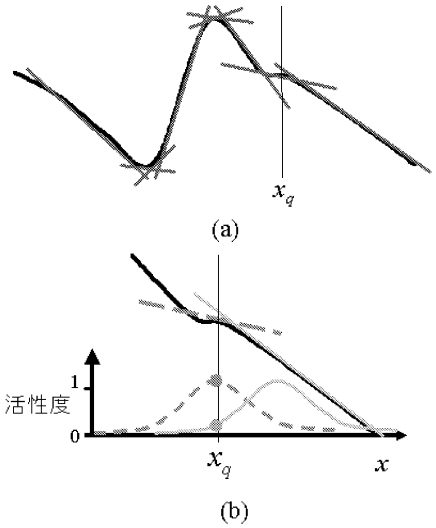

Fig. 1　LWRの基本的な考え方

線形関数を採用することにする．

問題の統計的モデルを以下のように定義する．

$$y = f(\mathbf{x}) + \epsilon \tag{1}$$

$\mathbf{x}$はn次元入力ベクトル，yは出力を表す．ϵは入出力に対し独立で，平均0，分散σ^2のノイズである．ここでの回帰とは，入出力系列$\mathbf{x}$，$\mathbf{y}$からその写像関数$f(\cdot)$を推定，近似することを意味する．

入力$\mathbf{x}$における推定出力$\hat{y}$は，個々のモデルからの出力を$\hat{y}_k$とすると，

$$\hat{y} = \frac{\sum_{k=1}^{K} w_k \hat{y}_k}{\sum_{k=1}^{K} w_k} \tag{2}$$

のように重み付き平均で表される．w_kはその点におけるモデルの活性度，Kは局所モデルの総数である．この活性度w_kはガウスカーネル関数

$$w_k = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_k)^T \mathbf{D}_k (\mathbf{x}-\mathbf{c}_k)\right) \tag{3}$$

に従う．ここで$\mathbf{c}_k$は活性度の中心，$\mathbf{D}_k$は活性度の広がりを決定するパラメータである．一方，各局所モデルの出力$\hat{y}_k$は次式で与えられる．

$$\hat{y}_k = \tilde{\mathbf{x}}^T \beta_k, \qquad \tilde{\mathbf{x}} = ((\mathbf{x}-\mathbf{c}_k)^T, 1)^T \tag{4}$$

ここでβ_kは局所線形モデルのパラメータである．

次に各パラメータの更新についてであるが，本手法では３つのパラメータの更新を行うことで，

近似精度を高めていく．１つ目として，β_kは最急降下法を用いて導かれた逐次更新式を用いて，各モデルごと独立に更新を行う[5]．

$$\beta_{k,t+1} = \beta_{k,t} + w_k \mathbf{P}_{k,t+1}\tilde{\mathbf{x}} + e_k^T \qquad (5)$$

ここで，

$$\begin{aligned} \mathbf{P}_{k,t+1} &= \frac{1}{\lambda}\Big(\mathbf{P}_{k,t} - \frac{\mathbf{P}_{k,t}\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\mathbf{P}_{k,t}}{\lambda/w + \tilde{\mathbf{x}}^T\mathbf{P}_{k,t}\tilde{\mathbf{x}}}\Big) \\ e_k &= y - \beta_{k,t}^T\tilde{\mathbf{x}} \end{aligned}$$

式中のe_kは現在の予測値と目標値との誤差を表す．

２つ目として活性度の中心$\mathbf{c}$については，活性度最大のユニットkに対してのみ，入力の方向へ近づけるよう更新を行う[6]．具体的には，次式に従う．

$$\mathbf{c}_{k,t+1} = \mathbf{c}_{k,t} + \eta \min\Big(1, \frac{|e_{k,t+1}|}{e_{max}}\Big)(\mathbf{x} - \mathbf{c}_{k,t}) \qquad (6)$$

入力方向への移動量は，ステップサイズパラメータηと誤差e_kに依存する．これにより，現在のトレーニング点における予測値と目標値との間の誤差が大きな場合ほど，より大きくトレーニング点へ近づけることになる．また，e_{max}は外れ値の影響を抑える働きをもつ．

３つ目として，活性度の広がり$\mathbf{D}$についてである．これについては，近似精度の良いものほど，活性度の広がりを広げるように更新を行う．活性度最高のユニットが誤差最小のユニットでなかった場合（すなわち，活性度の広がりの更新により，近似精度の向上が期待される場合），活性度の高いほうからM個のユニットを取り出し，絶対誤差でソートする．誤差最小のものを$R=-1$，誤差最大のものを$R=1$とし，M個のユニットへ線形にRを割り振る．このM個のユニットに対してのみ，以下の更新を行う．

$$\mathbf{D}_{k,t+1} = \mathbf{D}_{k,t} + T\mathbf{D}_{k,t}(\mathbf{x} - \mathbf{c}_k)(\mathbf{x} - \mathbf{c}_k)^T\mathbf{D}_{k,t} \qquad (7)$$

ここで，

$$T = \frac{((1+\rho)^R - 1)}{(\mathbf{x} - \mathbf{c}_k)^T\mathbf{D}_{k,t}(\mathbf{x} - \mathbf{c}_k)}$$

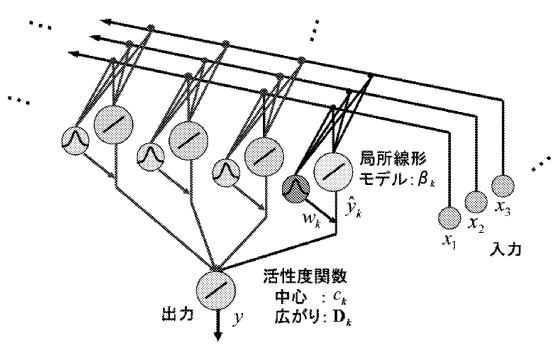

Fig. 2　LWRのネットワーク的表現

この更新は，ユニット毎の精度の比較に基づいている．

以上のLWR手法の概念をネットワーク的に表現したものをFig. 2に示す．

2.2　シミュレーション

2.2.1　Achimら(1999)の検証

まず，Achimら(1999)[6]の検証を行った．２変数１出力の次の関数の近似を行う．

$$y = \max\left(e^{-10{x_1}^2}, e^{-50{x_2}^2}, 1.25e^{-5({x_1}^2+{x_2}^2)}\right) + N(0, 0.01) \qquad (8)$$

この関数はFig. 3に示すように，２つの尾根とひとつのピークをもつ．

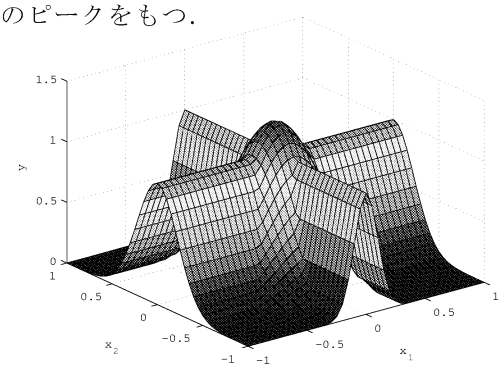

Fig. 3　目標関数

本研究では強化学習への適用を前提としていることから，Achimらとは若干異なる条件でシミュレーションを行った．１点目として，Achimらはトレーニングデータとして特定の500点を，20回繰り返し用いているが，本研究ではx_1, x_2ともに

モデルの配置	ランダム	格子状
Achimら	0.02	——
本研究	0.0281	0.0429

Table 1　学習後のnMSEの比較

[−1 1]の範囲内で一様乱数により10,000点を与えるものとする．２点目として，線形モデルをAchimらは48個用意したのに対し，本研究では49個を用いている．これは線形モデルをランダムに並べた場合と，格子状に並べた場合の比較を行うためである．その他，全てのパラメータや条件はAchimらに従った．また，近似精度の指標としては，ノイズを除いた目標値との誤差を用い，出力の分散により正規化を行った平均２乗誤差(nMSE)を用いた．Achimらによると，モデルをランダムに並べ，500点×20回の学習後にnMSE=0.02が得られたと報告されている．

本シミュレーションの結果をTable 1に示す．本シミュレーションの結果は10回の平均値を採用している．

モデルをランダムに配置した場合でも，Achimらの結果よりも特性が悪かった．この差は，シミュレーション条件の差であると考えられる．一方，モデルの配置の仕方による違いでは，ランダムに配置した方が，より特性が良好であった．これはランダムに配置した方が，より適切な活性度の広がりの変化，中心点の移動が行われたためであると考えられる．学習後の活性度の分布をFig. 4に示す．この図は，活性度$w_k = 0.5$の等高線を図示したものである．

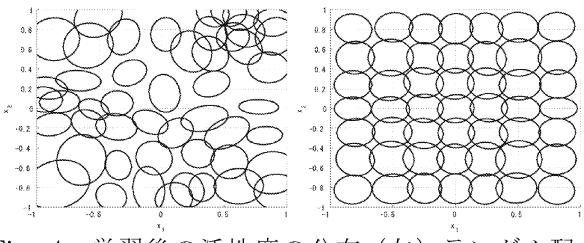

Fig. 4　学習後の活性度の分布（左）ランダム配置,（右）格子状配置

しかしながら，ランダムに配置した場合，収束値のばらつきが大きい．分散を求めてみたところ，格子状配置で2.66×10^{-6}，ランダム配置で1.51×10^{-5}であり，５倍以上の差が認められた．中心$\mathbf{c}$の更新により適切な位置に移動する能力には限界があり，モデルの初期配置に依存している部分があることが明らかになった．

2.2.2　目標値の変更

先ほど用いた関数をx_1方向へ+1.0，x_2方向へ+1.5シフトさせたものを目標値とする.この関数をFig. 5に示す．その他の条件は全て同一である．

$$y = \max\left(e^{-10(x_1-0.1)^2}, e^{-50(x_2-0.15)^2}, 1.25e^{-5((x_1-0.1)^2+(x_2-0.15)^2)}\right) + N(0, 0.01) \qquad (9)$$

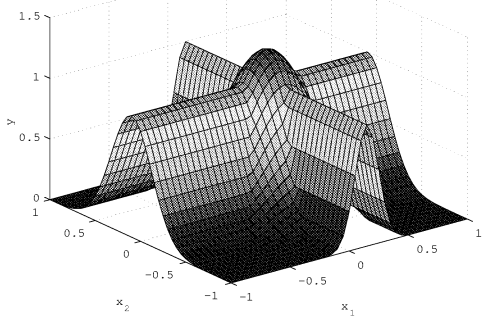

Fig. 5　変更後の目標関数

本シミュレーションの結果をTable 2に示す．結果は全て10回の平均値を採用している．

モデルの配置	ランダム	格子状
目標値変更前	0.0281	0.0429
目標値変更後	0.0282	0.0336

Table 2　学習後のnMSEの比較

シミュレーションの結果得られた値は，モデルをランダムに並べた場合はほぼ変化なく，格子状に並べた場合は特性が改善された．ランダムな場合は，予想通りの結果であると言える．一方，格子状に並べた場合に特性が向上した理由としては，

今回の目標値が現在のモデルの並べ方に適していたためであるといえる．以前の目標値では尾根のピークに沿って中心点が並んでいたため，精度の向上に支障をきたしていたが，今回の目標値はそれがシフトすることにより解消されたためである．モデルを格子状に並べた場合は，活性度の広がりや中心点の変化量が少なく，目標値との相性があることが明らかになった．

2.2.3 更新則の簡単化

次に，この手法を強化学習へ利用するため，デルタルール的更新則へ簡単化を行い，特性の変化を確認した．具体的にはベクトルの推定量$\mathbf{P}$を，スカラーのステップサイズパラメータαで置き換えることとする．(5)式を次式で置き換える．

$$\beta_{k,t+1} = \beta_{k,t} + \alpha w_k \tilde{\mathbf{x}} \mathbf{e}_k^T \qquad (10)$$

10,000点までのシミュレーション結果の一例をFig. 6に示す．

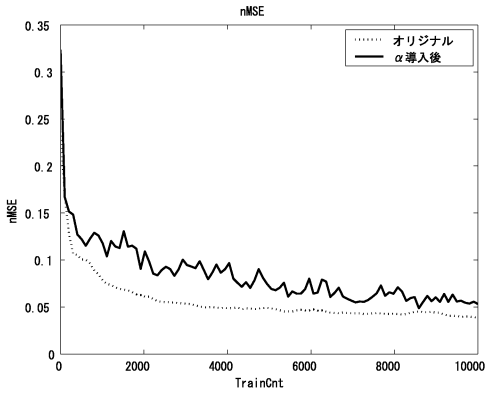

Fig. 6　nMSE遷移グラフ

これより，振動の増加と，収束速度の悪化が確認された．そこで，トレーニングデータをさらに増やし，20,000点までシミュレーションを行った．最終的なnMSEの比較をTable 3に示す．

トレーニングデータを２倍にした結果，更新則を簡単化する以前と同等の結果を得ることができた．以上より，この簡単化を行うことで振動の増加，収束速度の悪化が起こるが，トレーニングデータを２倍程度にすることにより同等の近似精度が得られることが確認された．次節以降の学習への適用の際には，この簡単化を行った更新則を用いることとする．

モデルの配置	ランダム	格子状
P(10,000点)	0.0281	0.0429
α(10,000点)	0.0470	0.0544
α(20,000点)	0.0361	0.0421

Table 3　学習後のnMSEの比較

3. 強化学習への適用

3.1 強化学習とLWR

まず，強化学習に関数近似を用いる場合，テーブルルックアップ方式における教師付き学習の，単純な代用ではないことに注意する必要がある．全ての教師付き学習は，実際のデータに対して学習を行う際に予測誤差の影響を受ける．しかしながら，関数近似による逐次学習を行う場合，これらの小さな誤差が蓄積され，その近似が使い物にならなくなる．例として，オンラインでの一般的な強化学習手法であるSarsa(0)アルゴリズムを取り上げる．状態sで行動aをとり，報酬rを受け取って，遷移先の状態s'で行動a'をとったとき，テーブルに基づく価値関数近似の更新式は次式による．

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \quad (11)$$

状態・行動対(s,a)における推定値は，学習率α，割引率γ，遷移先の状態における現在の推定値により更新される．ここで重要なことは，新しい推定値$Q(s,a)$は，現在の推定値$Q(s,a)$と遷移先の状態における推定値$Q(s',a')$に基づいている点である．これは，予測に用いている推定値に含まれる誤差が，更新時に取り込まれることを意味する．このような場合，近似アルゴリズムによって学習され

た新しい値は，多少の不正確さを含んでいる．この値が他の状態･行動対の更新に用いられると，誤差がさらに増幅される．この誤差は急速に増大し，近似を使い物にならなくする．

また，一般に関数近似による予測値は，全ての点（状態・行動対）において妥当であるわけではない．それは現在用いている関数の形式について大きな仮定を設けない限り，トレーニングデータのカバーしている入力領域においてのみしか信頼性がない．言い換えれば，関数近似は外挿法的な使い方ではなく，内挿法的に使うことしかできないのである．

教師付き学習においては，学習している関数を入力空間の異なる領域に対して交差検定を行うことにより，正確さを検証すればよいと考えるかもしれない．それにより，経験的にどの点においては妥当であり，どの点においては妥当でないか判断可能になるはずである．しかしながら，逐次的な推定を行う場合に関しては，これらの検証は適さない．推定した関数が，トレーニングデータそのものを近似できているかどうかは判断できても，学習により導き出されたトレーニングデータそのものが正確かどうかは判断できないためである．

この問題に対するひとつのアプローチとしては，トレーニング点を中心に中高の外形をした（例えばガウス関数のようなもの）妥当性を表す活性度関数を設けることである．そうすることで問題は，この活性度関数をどのように配置し，変化させるかという問題に置き換えることができる．

この点と，収束速度が速いなどの点が，強化学習への適用にLWR手法が適していると考えられる根拠である．

3.2 アルゴリズム

本研究においては，状態は連続であり，行動は離散であるという仮定に基づいて議論を行う．また，基本となるアルゴリズムにSarsa(0)アルゴリズムを採用した．

まず，価値推定についてであるが，各行動ごとに価値空間を用意し，独立に推定を行う．すなわち，回帰モデルで扱った空間を行動の数だけ用意することになる．また今回，モデルを格子状に初期配置することとする．

行動選択法については，各状態・行動ごとの推定値を用いて，ϵ-greedy政策をとる．これは，微小な確率ϵでランダムな行動を，残りの確率で最適な行動を選択する政策である．

問題となるのは，βの更新についてである．回帰モデルのように目標出力は観測できないため，単純に更新則を適用することができない．ここでは，TD学習による予測値で置き換えることにする．したがって，(10)式を次のように変更する．今，状態$\mathbf{s}$で行動aをとり，遷移先の状態$\mathbf{s}'$で行動a'をとったとき，次のように更新を行う[7]．

$$\beta_{k,t+1} = \beta_{k,t} + \alpha w_k \tilde{\mathbf{s}} d_k \tag{12}$$

ここで，

$$\begin{aligned} d_k &= r(\mathbf{s},a) + \gamma \hat{Q}(\mathbf{s}',a') - \hat{Q}(\mathbf{s},a) \\ &= r(\mathbf{s},a) + \gamma \hat{Q}(\mathbf{s}',a') - \beta_k^T(\mathbf{s},a)\tilde{\mathbf{s}} \end{aligned}$$

式中で，目標値は$r(\mathbf{s},a)+\gamma\hat{Q}(\mathbf{s}',a')$の部分である．十分に学習が進めば，行動価値

$$Q(s,a) = \beta^T(\mathbf{s},a)\tilde{\mathbf{s}}$$

は割引報酬

$$R = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \cdots = \textstyle\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

に収束する．

3.3 問題設定

本研究では，Mountain-Car問題[8]と呼ばれる制御問題に適用を行った．Fig. 7に示すように斜面に沿って車を走らせる．目的は，車が右側の斜面を登

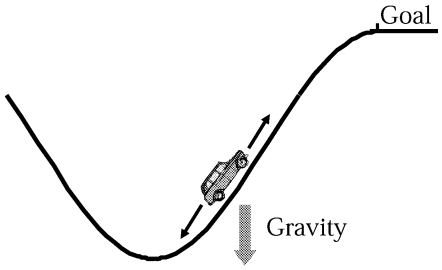

Fig. 7　Mountain-Car問題

りきることである．しかしながら，重力はエンジンの力よりも強く，最も急な斜面ではフルスロットルでも加速することができない．この問題を解決する唯一の方法は，いったんゴールとは逆方向に加速して，その後でゴール方向に加速し続けることである．これにより，右側の斜面で減速しても登りきれるだけの十分な速度が得られる．このように，ゴールに到達するためには，一度ゴールから遠ざかる必要がある．これは，良い方向を目指すために，いったん悪い方向へ向かう必要のある問題の単純な例である．多くの制御手法は，設計者の明示的な手助けなしに，この種の問題を取り扱うことは非常に困難である．

この問題における報酬は，ゴールするまでの各ステップごとに-1を与える．右側の斜面を登りきりゴールに到達するか，最大ステップ数に達したときはそのトライアルを終了する．エージェント（学習者）はゴールするまでの報酬の総和を最小化するため，最小時間でゴールしようとする．各時間ステップにおいて，エージェントは前方フルスロットル，加減速なし，後方フルスロットルの３つの行動からひとつを選択する．この行動と重力の影響により，次の時間ステップでの位置と速度が決定する．

3.4　シミュレーション

実際にシミュレーションを行うに当たり，状態を離散化した場合を比較対象とした．アルゴリズムは同じSarsa(0)を採用し，行動選択もϵ-greedy政策

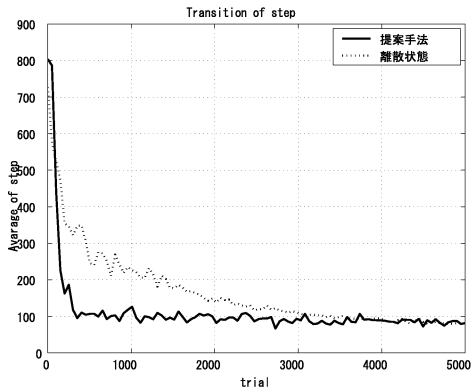

Fig. 8　離散化手法と提案手法のstep数遷移の比較

を取る．パラメータは状態数30^2，学習率$\alpha = 0.1$とした．一方，LWRを用いた提案手法については，モデルの初期配置は格子状，モデル数7^2，学習率$\alpha = 0.05$，$\mathbf{c}$の移動のステップサイズパラメータ$\eta = 0.001$，$\mathbf{D}$の更新に関するパラメータ$\rho = 0.001$，$\mathbf{D}$の初期値は$diag[9000\ 70]$とした．このときの２つの手法のStep数の遷移グラフをFig. 8に示す．なお，使用したデータはどちらも10回のシミュレーションの平均値を用いている．

グラフより，状態を連続で扱っているにもかかわらず，提案手法の方が優れた収束特性を示していることが分かる．また，学習により獲得された軌跡を確認したところ，離散化をした場合と提案手法との間に大きな違いはなく，滑らかな準最適と思われる軌跡が確認された．

さらに，ひとつの行動における学習後の活性度の分布をFig. 9と，価値関数をFig. 10に示す．

この図より，平面に近い価値関数の部分では，活性度が大きく広がっているユニットがあり，活性度の分布の更新が適切に行われている様子が確認できる．しかしながら，大きく広がったモデルの付近には，小さな活性度の分布をもった多数のモデルが存在している．これらのユニットは存在しなかったとしても，精度に大きな影響を与えるものではないと考えられる．これらの不必要なユニットの削除の機構に関しては今後の課題である．

次に，提案手法において，活性度の更新がどの

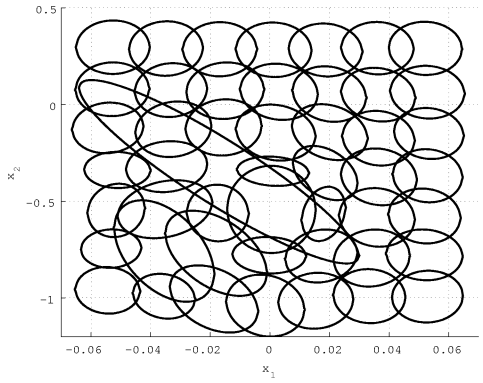

Fig. 9　提案手法のパラメータの更新の違いによるstep数遷移の比較

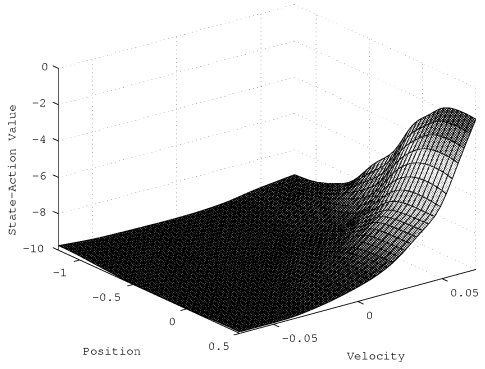

Fig. 10　提案手法のパラメータの更新の違いによるstep数遷移の比較

程度，近似精度に貢献しているかを検証するため，活性度の中心$\mathbf{c}$，広がり$\mathbf{D}$の更新を行わない場合について検討を行った．２つの条件におけるStep数の遷移グラフをFig. 11に示す．

グラフより，活性度の中心や広がりも更新を行う場合と，これらの更新を行わず，局所的な線形近似のみに頼った場合とで特性の差が確認されなかった．これは，この問題のために必要な精度が局所線形近似のみの能力で十分な程度であったためである可能性がある．この点については，他の環境に適用を行った上での十分な検討が必要である．

しかし一方で，現在の活性度の更新法が必ずしも有効に働いていない可能性も残されている．

現在の活性度の中心$\mathbf{c}$の更新法では，目標値が一定の場合にはある程度有効であることが示されているが，オンラインで目標値が変化しつづける状況

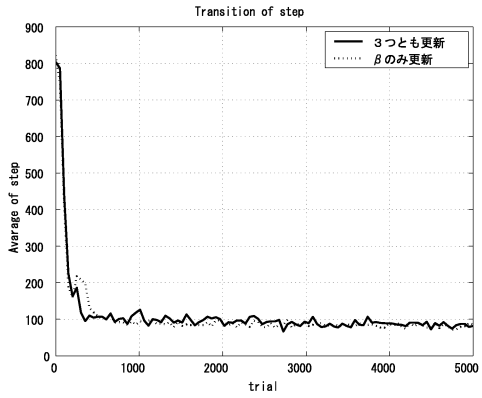

Fig. 11　提案手法のパラメータの更新の違いによるstep数遷移の比較

では．特性を向上させることができなかった．これに対しては，中心を移動するのではなく，必要な位置へは新たなモデルの配置を行い，不必要なユニットは削除する方法をとることで，解決できる可能性がある．

また，活性度の広がり$\mathbf{D}$については，今の更新則では他のモデルとの精度を比較して更新を行っている．これも目標値一定の場合では非常に有効ではあるが，目標値が変化する場合には効果がない又は特性を悪化させることが示された．これに対する解決策としては，他のモデルとの比較に基づく更新ではなく，各モデルごと独立に，誤差に基づいて更新する方法が有効ではないかと考えている．

4.　まとめと今後の課題

本研究では，連続状態をもつ環境を取り扱うための関数近似手法としてLWRに着目し，この近似手法そのものの性能評価と，実際に単純な環境における強化学習へ適用を行った．

LWRは一般化性能と収束速度など，オンラインでの学習に適していると考えられる点が多いが，線形モデルの初期配置の仕方の違いにより，特性に差がでることが確認された．それぞれの配置法にはメリットとデメリットがあり，別のより優れ

た配置法が求められる．また，強化学習との整合性を保つため，線形モデルパラメータβの更新則の簡単化を行った．収束速度の低下が起こるものの，２倍程度の学習量を確保することで，同程度の精度で近似可能であることが確認された．

また，学習への適用を行い，離散化を行った場合との比較を行った．提案手法は連続状態を扱っているにもかかわらず，離散化した場合よりも優れた収束特性が得られた．しかしながら，活性度の中心$\mathbf{c}$，広がり$\mathbf{D}$の更新を行わなかった場合と，更新を行った場合との間に特性の差は認められず，これらの更新がオンラインで有効に働いているのかどうかを確認できなかった．

今後はeligibility traceの導入によりSarsa(λ)アルゴリズムを採用することで，さらなる収束速度の向上が期待される．しかし，最も重要な課題は，よりオンラインでの学習に適した，活性度分布の更新則を提案することである．線形モデルパラメータβに関しては非常に収束が速く，オンラインでの学習でも有効であったのに対し，現在の活性度の中心$\mathbf{c}$，広がり$\mathbf{D}$の更新法はオンラインでの学習に向いていない．3.4節で検討した考え方をもとに新たな更新方法の提案が必要である．

さらに，活性度の中心の更新とも関連して，局所モデルの数の決定方法が求められる．現在の手法では最初に配置した数のまま増減は行わないため，この数の決定は重要性が高く，また試行錯誤的に決定せざるを得ない．この点からも，必要な数を必要な位置へ配置する機構が求められる．

また，今回適用を行った環境だけでなく，その他の環境，より高次元の環境への適用による十分な検証も必要である．さらに，今回は行動は離散的なものを扱ったが，連続行動を扱うことも今後の課題である．

一方，実装上の問題として，計算時間が挙げられる．本来，LWR手法は神経回路をモデルにしていることから，並列計算に向いている．しかしながら，本研究環境においては，逐次処理型の計算機に頼っており，非常に多くの計算時間を要する．これは研究を進めていく上での非常に大きな障害となっている．今後，高次元の状態や連続行動を扱う上ではさらにこの問題が深刻になることは明らかである．そのため，逐次処理型の計算機での計算時間の短縮をする工夫，またはクラスタマシンを構築して並列計算を実現するなど，計算時間を短縮するための方法も不可欠である．

参考文献

1) Sutton, S. R, Barto, G. A.: Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA. (1998)
2) Kaelbling, L. P., Littman, M. L., Moore, A. W.: Reinforcement learning: A survey,
The Journal of Artificial Intelligence Research, 4, 237/285 (1996)
3) Boyan, J. A., Moore, A. W.: Generalization in reinforcement learning: Safely approximating the value function,
Advanced in Neural Information Processing Systems: Proceedings of the 1994 Conference, Cambridge, MA: MIT Press, 369/376 (1995)
4) Atkeson, C. G., Moore, A. W., Schaal, S.:Locally weighted learning,
Artificial Intelligence Review, 11, 11/73 (1997)
5) Schaal, S., Atkeson, G. C.: Constructive incremental learning from only local information,
Neural Computation, 10, 2047/2084 (1998)
6) Achim, L., Tagscherer, M., Kindermann, L., Protzel, P.: Improving the Fit of Locally Weighted Regression Models,
The Sixth International Conference on Neural Information Processing (ICONIP'99), Perth, Australia (1999)
7) Tsitsiklis J. N., Van Roy, B.: Feature-based methods for large scale dynamic programming,
Machine Learning, 22, 59/94 (1996)
8) Moore, A. W.: Variable resolution dynamic programing: Efficiently learning action maps in multivariate real-valued state-spaces,
In Machine Learning; Proceedings of Eighth International Workshop, 333/337 (1991)