自律分散系の適応アルゴリズムによる強化学習のための関数近似†

小林 祐一*・湯浅 秀男*,**・新井 民夫*

Function Approximation for Reinforcement Learning Using Autonomous-Decentralized Algorithm†

Yuichi KOBAYASHI*, Hideo YUASA*,** and Tamio ARAI*

The adaptability of resolution to the complexity of approximated function has a great influence on the performance of learning in the function approximation for reinforcement learning. We propose applying the reaction-diffusion equation on a graph to function approximation for reinforcement learning. The function approximator expressed by nodes can change its resolution adaptively by distributing them densely in the complex region of the state space with the proposed algorithm. A function is expressed in a plane. The successive least square method is adopted to approximate the function from the data. Each plane corresponds to a node, which is an element of the graph. Each node moves to diffuse the complexity of the approximated function in the neighborhood based on the reaction-diffusion equation. The complexity of the function is defined by the change of gradient. The simulation shows the two points: 1) The proposed algorithm provides the adaptability for function approximation. 2) The function approximation improves the efficiency of the reinforcement learning.

Key Words: reinforcement learning, reaction-diffusion equation, graph, function approximation

1. はじめに

強化学習では一般に、報酬をもとに構成される状態価値関数・行動価値関数などを、離散化あるいは関数近似で表現し、それをもとに行動学習を行なう。この関数表現においては、状態空間内の学習にとって重要な領域において、適切な分解能を獲得することが学習性能に大きな影響を与える。つまり、状態空間の複雑な領域に高い分解能を持ち、そうでない領域は分解能を低く抑えて計算量の増大を抑えるように適応できることが重要な意味を持つ。

強化学習に適用されている関数近似の手法としては、階層型ニューラルネットワーク、RBF(Radial Basis Function) ネットワーク、CMACなどを挙げることができる¹⁾。CMAC²⁾はメモリ効率の良いグリッド状状態空間分割を行なう方法であるが、格子の細かさや形状をあらかじめ設計しなくてはならない。また、階層型ニューラルネットワークでは関数近似の要素(たとえばシグモイド関数)の動的な追加・削除が困難である。RBFネットワークは動径基底関数という分散性の高い要素を用いた関数近似方法であり、動的な基底の追加に

対応したアルゴリズムが提案されている^{6),7)}。しかし、RBFのアルゴリズムは一般に、計算が複雑になることと設計者によるパラメータ調整を要することが多いことが指摘されている¹⁾。たとえばMorimoto *et al.*⁷⁾は、関数近似の精度を判断基準とした基底関数の動的な追加アルゴリズムを提案しているが、近似精度不足の原因がガウス関数の形状にあるのか、基底関数の数が不足しているのかの判断は困難であるために、変形、追加に関するパラメータ設計が複雑になってしまう。また、階層型ニューラルネットワークのような非線形近似手法は直接強化学習に適用した場合の収束性が証明されておらず、LWR(Locally Weighted Regression)⁴⁾のような観測データを直接記憶する近似手法と同様に、状態価値関数近似が発散する例が報告されている⁵⁾。

また、状態を離散化する手法として、浅田らのクラスタリング手法⁸⁾、Ishiguro *et al.*の統計的分割法⁹⁾、高橋らの漸次的分割手法¹⁰⁾などを挙げることができる。これらの逐次分割手法は、関数近似との類比では、関数近似の要素を動的に追加することに対応している。これらの方法では状態分割の指針として、同じ要素行動に対する遷移可能性、多数の試行における報酬のばらつき、局所線形モデルへの当てはめをそれぞれ用いている。これらの方法には、状態空間の「複雑さ」を定義し、それに基づいて状態を離散化していくという共通点がある。しかし、適切な分解能を獲得するためには、逐次的に分割してだけでなく、すでにある要素を動的に再配置することも重要である。そのような観点での研究例として、分割した状態同士の統合を扱う小林ら¹²⁾の方法が挙

† 第13回自律分散システム・シンポジウムで発表(2001・1)

* 東京大学工学系研究科 東京都文教区本郷7-3-1

** 理化学研究所バイオ・ミメティックコントロール研究センター

* Graduate School of Engineering, The University of Tokyo, Bunkyo-ku, Tokyo

** Bio-mimetic Control Research Center, RIKEN, Anagahora, Nagoya

(Received June 11, 2001)

(Revised October 12, 2001)

げられるが、これは2値の即時評価を前提としている点が本研究の問題設定と異なる。

そこで本研究では、状態価値関数の形状の複雑さに即した解像度を適応的に得る関数近似を行なうことを目的とする。具体的には、

1. 位置・勾配情報を持ったノードにより超平面を構成して関数近似を行なう
2. グラフ上の反応拡散方程式¹⁴⁾に基づいてノードを関数形状の複雑度に応じて移動させる

という方法をとる。グラフ上の反応拡散方程式は自律分散系の適応アルゴリズムと見なすことができ、個々の要素の挙動を全体の秩序を形成するように設計することができる。関数形状の複雑度として勾配変化に着目し、勾配変化の大きいところに密に、小さいところに疎に分布するような設計指針を取る。

提案手法の利点として、自律分散システムのアルゴリズムであることから

- すでにある関数近似の要素(ノード)全体を動的に再配置することができる
- 関数近似の要素を動的に追加・削除することが容易である

を挙げることができる。勾配に着目した適応手法である点では高橋らの線形モデルによる分割と共通する部分があるが、逐次分割によらず同数の要素のままで適応的に再配置できる点異なる。これにより要素数の不要な増加を防ぐことができる。また、RBFネットワークの更新アルゴリズムに対して、局所近傍における複雑度が定義され、それと関数近似全体との関係が明確に記述されているために設計者のパラメータ調整の複雑さを低減できるという利点がある。

以下、2章では各ノードにおける逐次最小二乗法、3章でグラフ上の反応拡散方程式について説明し、4章でその適用方法について述べる。5章で提案手法の強化学習への適用方法について述べ、6章でのシミュレーションにより、関数近似問題および強化学習における標準問題によって提案手法の有効性を検証する。特に本稿では、ノードを動的に再配置することの効果を経験価値関数の近似において検証する。

2. 関数近似のための超平面の構成

本章では、位置・勾配を持ったノードにより構成される超平面による出力決定方法、およびデータをもとに各ノードの勾配を逐次最小二乗法で更新する方法を示す。 V を、 n 次元入力1次元出力の位置情報をもつノード(頂点)の集合とする。各ノード $i \in V$ は位置ベクトル $\mathbf{w}_i = [\mathbf{w}_i^T \ w_i^y]^T \in R^{n+1}$ (T は転置を表す) と勾配ベクトル $\mathbf{a}_i \in R^n$ を持っている(Fig.1)。入力ベクトル $\mathbf{x} \in R^n$ が与えられたとき、最整合(best matching)ノード b は

$$b = \arg \min_i \|\mathbf{x} - \mathbf{w}_i^x\| \quad (2.1)$$

によって決定される。 $\tilde{\mathbf{x}} \equiv [(\mathbf{x} - \mathbf{w}_b^x)^T \ 1]^T$ とすると、出力 y

はノード b によって

$$y = \mathbf{a}_b^T (\mathbf{x} - \mathbf{w}_b^x) + w_b^y = [\mathbf{a}_b^T \ w_b^y] \tilde{\mathbf{x}} \quad (2.2)$$

で与えられる。これらのノードで表現される超平面の組み合わせにより関数近似を行なう。

Fig. 1 Plane construction Fig. 2 Least square method

最整合ノード b が N 個のデータを得た(注1)としたときのデータ行列 A_N とベクトル $\mathbf{y}_N$ をそれぞれ

$$A_N = \begin{bmatrix} \mathbf{x}_1 - \mathbf{w}_b^x & \cdots & \mathbf{x}_N - \mathbf{w}_b^x \\ 1 & & 1 \end{bmatrix} \quad (2.3)$$

$$\equiv [\tilde{\mathbf{x}}_1 \tilde{\mathbf{x}}_2 \cdots \tilde{\mathbf{x}}_N]$$

$$\mathbf{y}_N = [y_1 \ y_2 \ \cdots \ y_N]^T \quad (2.4)$$

と表わす。これらの点がノード b で作られる平面上にあるとすると、

$$\mathbf{y}_N^T = [\mathbf{a}_b^T \ w_b^y] A_N \quad (2.5)$$

が成り立つ。ここで、二乗誤差 $\|\mathbf{y}_N^T - [\mathbf{a}_b^T \ w_b^y] A_N\|^2$ を最小にする w_b^y と $\mathbf{a}_b$ を求める(Fig.2)。rank $A_N = n + 1$ のとき、 $\mathbf{a}_b$ は擬逆行列を用いて以下のように計算できる。

$$[\mathbf{a}_b^T \ w_b^y] = \mathbf{y}_N^T A_N^T (A_N A_N^T)^{-1} \quad (2.6)$$

データ行列 A_N とデータベクトル $\mathbf{y}_N$ を新しい観測の情報によって逐次的に更新するために、 B_N と $\mathbf{c}_N$ を以下のように定義する。

$$B_N = A_N A_N^T, \quad \mathbf{c}_N^T = \mathbf{y}_N^T A_N^T \quad (2.7)$$

この両式は、

$$B_N = B_{N-1} + \tilde{\mathbf{x}}_N \tilde{\mathbf{x}}_N^T, \quad \mathbf{c}_N = \mathbf{c}_{N-1} + \tilde{\mathbf{x}}_N y_N \quad (2.8)$$

と逐次表現することができる。各ステップごとに $B_{N-1}, \mathbf{c}_{N-1}$ の情報を保存することで B_N および $\mathbf{c}_N$ を更新することができる。このようにして、勾配および位置 $[\mathbf{a}_b^T \ w_b^y]$ は

$$[\mathbf{a}_b^T \ w_b^y] = \mathbf{c}_N^T B_N^{-1} \quad (2.9)$$

のように更新することができる。

3. グラフ上の反応拡散方程式

位置・勾配を持つノードによる関数表現方法が前章で定義された。このノードの密度を適応的に変えるために、

1. 各ノード u の近傍での近似関数形状の複雑度を定義する

(注1) 実際のノードは逐次最小二乗パラメータを保持する。

2. 各ノードの複雑度を周囲のノードと均一にするようにノード u を移動させる

という方法で各ノードを移動させる。ここで扱うノードによる関数表現システムは、空間内に同一の性質をもつ動的要素が不均一・不連続に分布しているシステムといえる。このようなシステムにおいて全体として秩序を形成するように個々の要素の挙動を設計する方法としてグラフ上の反応拡散方程式¹⁴⁾が提案されている。これを用いて、ノード上に表現された複雑度を全体として均一にするためのノードの挙動を設計する。本章では、グラフ上の反応拡散方程式について簡単に述べる。

有限グラフ $G = (V, E)$ の頂点 (ノード) V が内部点の集合 $\bar{V}$ と境界点の集合 ∂V からなり、辺の集合 E が内部辺の集合 $\bar{E}$ と境界辺の集合 ∂E からなるとき (Fig.3), これを境界付きグラフと呼ぶ¹⁴⁾。

Fig. 3 Graph with boundaries

各頂点 u における実数値関数を $f(u)$ と表わし、 $C(V)$ を V 上の、 $C(E)$ を E 上のそれぞれ実数値関数全体の空間とする。

$$df(e) = f(t(e)) - f(o(e)) \quad (3.1)$$

を対応づける作用素 $d: C(V) \rightarrow C(E)$ を余微分といい、 df を f の勾配という (Fig.4)。ここで、 $o(e)$ は辺 e の始点となる頂点、 $t(e)$ は辺 e の終点となる頂点である。

Fig. 4 Function $f(u)$ and $df(e)$

この f を関数とする汎関数 $W(f)$ を、系全体の状態を表わす関数として定義する。本研究では、各ノードについて関数値 f を大きくし、近傍との差を減らして均一にするという設計指針に基づき、汎関数を以下のように定める。

$$W(f) = W_0(f) + W_1(df) \quad (3.2)$$

$$W_0(f) = \alpha_R \sum_{u \in \bar{V}} f(u) \quad (3.3)$$

$$W_1(df) = \beta_D \sum_{u \in \bar{V}} \|df(u)\|^2 \quad (3.4)$$

ただし、 $E(u)$ を u を頂点とする辺の集合とし、 $df(u) = [df(e)]_{e \in E(u)}$ 、また $\alpha_R < 0, \beta_D > 0$ である。これをポテンシャル汎関数とする関数空間での勾配系を構成する。ポテンシャル汎関数 $W(f)$ を極小値に向かわせるための関数 f の変化を以下のように与えることができる。

$$\frac{\partial f}{\partial t} = -\frac{\delta W(f)}{\delta f} \quad (3.5)$$

Fig.5に関数空間における最急降下法概念を表わす。ここで $W(f)$ の最小値はシステム全体の望ましい状態を表わしており、そこに向かうように関数値 f を変更することでシステムを望ましい状態に近づけることができる。

Fig. 5 Gradient descent in function space

4. 反応拡散方程式に基づいたノードの挙動設計

2章で述べたノードに3章の反応拡散方程式を適用し、前述の複雑度に即した分布を実現するための各ノードの挙動 (移動) 設計について述べる。ただしノードを移動したときには、ノードの移動モデルにしたがって近似関数の形状を保つように位置・勾配を更新する。以下に、

- 境界付きグラフへの適用
- 複雑度を表わした各ノード上の関数 $f(u)$ の定義
- 近似関数曲面に沿ったノードの移動モデル
- 関数値の位置微分によるノードの挙動設計

の順で述べる。

4.1 境界付きグラフへの適用

各ノードは辺によって接続されたノード (以下、近傍ノードと呼ぶ) と情報を交換することができる。近傍ノードおよび境界ノードは以下のように定義され (Fig.6), ノード移動ごとに更新される。

- 境界に近いノードは境界ノード (境界点) を認識し、境界辺によって結ばれる。
- 各辺はノード移動の各ステップごとに入力空間での位置 w_u^x を基づいて Topology Representing Networks アルゴリズム¹⁶⁾により更新され、新規生成および消去を行なう。
- 境界ノードは境界辺で結ばれるノード u と同じ関数値 $f(u)$ をもち、 w_u^x を境界上に射影した位置、勾配は $a = 0$ であると仮定する (ノイマン条件)。

4.2 グラフ上の関数 $f(u)$ の設計

ノード u の関数値 $f(u)$ は、その近傍における複雑度としてノードの密度を反映した勾配変化を表現するために、

$$f(u) = \frac{1}{m(u)} \sum_{h \in N(u)} \|a_u - a_h\|^2 \|w_u^x - w_h^x\|^2 \quad (4.1)$$

Fig. 6 Node definition

Fig. 7 Gradient change

のように定義する。ここで $m(u)$ はノード u の次数 (近傍ノードの数), $N(u)$ はノード u の近傍ノードの集合である。ここで, 記述の簡略化のために $b_{hu} \equiv a_h - a_u$, $c_{hu} \equiv w_h^x - w_u^x$ を定義する。

4.3 近似関数曲面に沿ったノードの移動モデル

ノード u が入力空間内を $r \equiv \Delta w_u^x$ 移動したときの勾配を $a_u(r)$ と表わし, その変化を表わすモデルについて述べる。

$$a_u(r) = \sum_{h \in N(u)} q_h^u a_h^u(r) \quad (4.2)$$

のように, 各近傍ノード h との関係により決定される勾配 $a_h^u(r)$ の重み付き和として表わす。ここで重み係数 q_h^u は距離の近い近傍の影響を強くするために距離の二乗に反比例する重みを正規化したものとして,

$$q_h^u = \frac{1/\|c_{hu}\|^2}{\sum_{i \in N(u)} 1/\|c_{iu}\|^2} \quad (4.3)$$

により与える。以下では, 特定の近傍ノード $h \in N(u)$ との関係による勾配変化 $a_h^u(r)$ について述べる。

位置・勾配の変化は $t = \frac{r \cdot c_{hu}}{\|c_{hu}\|^2}$ の関数として表わし (Fig.7),

1. c_{hu} 方向成分は, 近傍ノード h の位置 w_h^y ・勾配 a_h との間を滑らかにつなぐために Ferguson 曲線²⁶⁾で表現する (Fig.8)。

2. c_{hu} に直交する成分は, t に応じた線形補間によって勾配を a_h に近づける (Fig.9)。

Ferguson 曲線 $P(s) = [P_{x1}(s) \cdots P_{xn}(s) P_y(s)]^T$ は, 位置, 方向ベクトルを持った2点を s を媒介変数とする3次多項式により滑らかに結んだ曲線である。本研究での適用の詳細については付録Aで述べる。

Fig. 8 Ferguson model

Fig. 9 Interpolation model

勾配の方向成分と直交成分を分離するために, ノード u から h へ向かう単位ベクトル $n_c = \frac{c_{hu}}{\|c_{hu}\|}$ に対するヌル空間の直交基底を $n_1 \cdots n_{n-1}$ とし, 行列 N を

$$N \equiv [n_1 \cdots n_{n-1}] \quad (4.4)$$

と定義する。Ferguson 曲線により決定される勾配を $a_1(s)$, 線形補間による勾配を $a_2(t)$ とし,

$$a_u(r) = (a_1(s) \cdot n_c) n_c + N N^T a_2(t)^T \quad (4.5)$$

として勾配変化を記述する。各成分の変化は,

$$a_1(s) = \left[\frac{\dot{P}_y}{\dot{P}_{x1}}, \dots, \frac{\dot{P}_y}{\dot{P}_{xn}} \right], \quad \dot{p} = \frac{dP}{ds} \quad (4.6)$$

$$a_2(t) = (1-t)a_u + t a_h \quad (4.7)$$

のように表わされる。各項の r による微分は

$$\begin{aligned} \frac{\partial a_u(r)}{\partial r} &= \frac{\partial t}{\partial r} \frac{\partial s}{\partial t} \left(\frac{da_1(s)}{ds} \cdot n_c \right) n_c \\ &+ \frac{\partial t}{\partial r} N \frac{da_2(t)}{dt} N^T \end{aligned} \quad (4.8)$$

と表わされる。

4.4 関数の位置微分によるノードの挙動設計

$f(u)$ を変化させるためのノード u の移動方向は, $f_u(r)$ を r で微分し, $r = 0$ を代入することで得られる。ここで, $g_{hu}(r)$ を

$$g_{hu}(r) \equiv \|a_u(r) - a_h\|^2 \|r - c_h^u\|^2 \quad (4.9)$$

と定義すると,

$$f_u(r) = \frac{1}{m(u)} \sum_{h \in N(u)} g_{hu}(r) \quad (4.10)$$

と表わせる。 $g_{hu}(0)$ を得ることで, $\frac{\partial f_u(r)}{\partial r} \Big|_{r=0}$ は

$$\frac{\partial f_u(r)}{\partial r} \Big|_{r=0} = \frac{1}{m(u)} \sum_{h \in N(u)} \left(\frac{\partial g_{hu}(r)}{\partial r} \Big|_{r=0} \right) \quad (4.11)$$

と求められる。この方向にノードを移動させることで関数値 $f(u)$ を任意に変化させることができる。なお, (4.8) 式を用いた $\frac{\partial g_{hu}(r)}{\partial r}$ の詳細は付録Bに譲る。

(3.5) 式における関数値の勾配法による設計によりポテンシャル汎関数を極小にする解に導くことができるが, これにはノードの関数値を他のノードの関数値とは独立に制御できるという仮定が含まれている。本研究での関数値の定義では, ノード u を移動させて関数値 $f(u)$ を更新すると, (4.1) 式にしたがってその近傍の関数値も変化してしまう。このような近傍への影響を考慮するために, 近傍も含めたノード u の反応項 $D_u^R(r)$ および拡散項 $D_u^D(r)$ を

$$D_u^R(r) = f_u(r) + \sum_{d \in N(u)} f_d^u(r) \quad (4.12)$$

$$D_u^D(r) = \sum_{d \in N(u)} (f_u(r) - f_d^u(r))^2 \quad (4.13)$$

と定義する。ここで $f_d^u(r)$ は, 近傍ノード u が r 移動したときのノード d の関数値を表わす。ポテンシャル汎関数はこの D_u^R, D_u^D を用いて,

$$W(f) = \frac{\alpha_R}{2} \sum_{u \in \bar{V}} D_u^R + \frac{\beta_D}{2} \sum_{u \in \bar{V}} D_u^D \quad (4.14)$$

と改める. これらを直接 $\mathbf{r}$ で微分し,

$$\frac{\partial D_u^R(\mathbf{r})}{\partial \mathbf{r}} = \sum_{h \in N(u)} \left(\frac{1}{m(u)} + \frac{1}{m(h)} \right) \frac{\partial g_{hu}(\mathbf{r})}{\partial \mathbf{r}} \quad (4.15)$$

$$\frac{\partial D_u^D(\mathbf{r})}{\partial \mathbf{r}} = \sum_{d \in N(u)} \left\{ (f(u) - f(d)) \sum_{h \in N(u,d)} m_{hu}^D \frac{\partial g_{hu}(\mathbf{r})}{\partial \mathbf{r}} \right\} \quad (4.16)$$

を得る. ただし $N(u, d)$ は $N(u)$ からノード d を除いた集合, $m_{hu}^D \equiv \frac{1}{m(u)} - \frac{1}{m(h)}$ である. ポテンシャル汎関数を減少させるための各ノード u の移動方向 $\mathbf{v}_u$ は,

$$\mathbf{v}_u = \alpha_R \frac{\partial D_u^R(\mathbf{r})}{\partial \mathbf{r}} \Big|_{\mathbf{r}=\mathbf{0}} + \beta_D \frac{\partial D_u^D(\mathbf{r})}{\partial \mathbf{r}} \Big|_{\mathbf{r}=\mathbf{0}} \quad (4.17)$$

と定めることができる.

5. 連続値強化学習への適用

本研究では, 提案手法の強化学習における有効性の検討方法として状態価値関数の近似に的を絞るために Value Gradient¹⁷⁾を用いる. Value Gradient は, 状態価値関数とシステムダイナミクスをもとに行動を決定する手法であり, 行動出力 (制御入力) と状態遷移の関係が既知あるいは容易に獲得できるような問題で有効である.

状態変数を $\mathbf{x} \in \mathbf{R}^n$, 行動出力を $\mathbf{u} \in \mathbf{R}^m$ としたときに, システムのダイナミクスが $\dot{\mathbf{x}} = \mathbf{h}(\mathbf{x}, \mathbf{u})$ で表わされるとする. 行動出力 $\mathbf{u}$ の状態変化に対する寄与 $\frac{\partial \mathbf{h}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}$ が既知である場合^(注2)には, 状態価値関数 $V(\mathbf{x})$ をもとに (5.1) 式のように直接行動を決定することができる. また, 状態価値関数の更新は TD(0) アルゴリズムによる.

1. 初期化:
 $V(\mathbf{x})$ (ノードの位置情報 $\mathbf{w}$) の初期化
2. 各試行の初期化
 各試行における初期状態 $\mathbf{x}_0$ を乱数により決定
3. 行動決定 (ϵ -greedy 戦略):
 ϵ の確率で乱数により行動決定
 それ以外の場合: $j = 1, \dots, m$ に対して

$$u_j = u_j^{\max} \left(\frac{\partial \mathbf{h}(\mathbf{x}, \mathbf{u})}{\partial u_j} \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \right) \quad (5.1)$$

4. $\mathbf{x}'$ に状態遷移し, 報酬 r を受け取る

$$\mathbf{x}' = \mathbf{x} + \mathbf{h}(\mathbf{x}, \mathbf{u}) \Delta t \quad (5.2)$$

5. 状態価値関数 $V(\mathbf{x})$ の更新

$$\delta = r + \gamma V(\mathbf{x}') - V(\mathbf{x}) \quad (5.3)$$

$$V(\mathbf{x}) := V(\mathbf{x}) + \alpha \delta \quad (5.4)$$

6. 試行終了判定:
 反応拡散方程式によるノード移動を行ない, 2. に戻る
7. 繰り返し: 3. に戻る

(注2) 後述する本研究での例題では, 1 自由度振子振り上げ問題ではトルク T と角速度 ω の比例関係, 水たまり問題では各行動によりエージェントの位置が上下左右に移動する関係がこれに相当する.

(5.1) 式の s は値域 $[0, 1]$ の単調増加関数であり, 本研究の例ではステップ関数を用いている. (5.4) 式の更新則に基づき, 最整合ノードの位置を $\mathbf{w}_b^y = V(\mathbf{x})$ として更新する. 逆に $V(\mathbf{x})$ 参照時には $V(\mathbf{x}) = \mathbf{w}_b^y$ のように位置情報のみを用いる. ノードの移動は 4 章で述べた設計に基づき, 移動の際に用いるノードの勾配情報は 4.1 節の近傍ノードの位置 $\{\mathbf{w}_h | h \in N(b)\}$ をデータとして用いて (2.6) 式により得る.

Tsitsiklis *et al.* は, TD 学習の収束証明において, 関数近似に $V_t(\mathbf{x}) = \sum_{i=1}^n \theta_i(i) \phi_i(\mathbf{x})$ と表現できるような線形アーキテクチャを仮定している²¹⁾. 上記のように関数近似方法を限定することで, 本近似手法はノード固定時にこの線形アーキテクチャと等価になる. ノード移動時には, ノード密度が充分にあればノード移動による関数近似器の出力は大きく変わらないが, 基底関数が増えるため収束証明の仮定を逸脱する. 本適用では, ノード移動により学習の収束性に影響が出る場合は, ノード移動のダイナミクスは強化学習のそれより十分小さくする必要がある.

6. シミュレーション

提案手法の有効性を, 定常関数近似問題, 連続値強化学習問題において検証する. ノードの位置を格子状に固定した場合と, 同様の初期状態から提案手法により移動させた場合との比較を行なう. 定常関数近似問題では 2 章で述べた近似法にしたがい, 強化学習問題では 5 章で述べた近似法を用いる.

6.1 定常関数近似問題

1 次元および 2 次元のガウス関数

$$y = p \exp(-q \|\mathbf{x} - \mathbf{a}\|^2) \quad (6.1)$$

により生成されるデータをもとに関数近似を行なう. ノード移動のパラメータはいずれの場合も $\beta_D = 0.01$ である. 格子状の初期配置では, (4.14) 式の反応項の働きによりノードが空間上に拡散するように移動する必要がなく, 拡散項の働きのみで分布が達成されるため, 反応項は用いない ($\alpha_R = 0$).

1 次元の例

パラメータは $a = 0.1, p = \sqrt{1/2\pi}, q = 15$ である. ノード数 10 個で等間隔のノードに充分なデータを提示した初期状態と, 全ノードを微小量移動させるのを 1 ステップとして 300 ステップ後の状態をそれぞれ Fig.10, Fig.11 に示す. 各ノード上の線分は勾配, 鉛直な線分は各ノードにおける関数値 $f(u)$ を示している. ノード移動後には勾配変化の大きい領域に密にノードが分布し, 関数値が均一化されていることがわかる. Fig.13 はそれぞれ 1, 2 次元ガウス関数での近似精度 (近似誤差の積算値) をノード移動の前後で比較した結果である. いずれのノード数に対しても, ノード移動によって分布密度が変わった結果, 関数近似の精度が向上していることがわかる. また, ノード数が少ないほど, ノードの移動による近似精度の向上が顕著である.

2 次元の例

$\mathbf{a} = [0.1, 0.1], p = \sqrt{1/2\pi}, q = 5$ で Fig.14 に示す関数の

Fig. 10 Initial state

Fig. 11 Final state

Fig. 12 Function values

Fig. 13 Comparison

近似を行なった。ノード数は49個で 7×7 の格子状に配置して開始し、十分に関数近似を達成した(1000個のデータを提示した)後、各ステップごとに近似関数上からランダムに選んだデータを100点提示しながら移動を繰り返した。Fig.15のように、右下の勾配変化の大きい領域に密に、勾配変化の小さい左上の領域に疎に分布していることがわかる。同じ

Fig. 14 Gauss function

Fig. 15 Nodes after iteration

要素(ノード)数に対して関数近似精度を向上させることができた。ただし、関数形状を超平面上で表わすために最低限必要な要素数を下回る(上記の例では4個以下)と関数近似精度は逆に低下する。本提案手法はノードの動的な追加・削除が容易に行なえるという利点を有しているので、このような状況では動的に要素を追加するアルゴリズムで対処することが求められる。また、計算時間はPentiumIII 1GHz CPUでノード移動には100ステップにノード49個で0.27[sec], 225個で1.56[sec]を要した。

6.2 1自由度振子振り上げ問題

連続値強化学習の例題である1自由度振子の振り上げ問題^{1),17)}に適用した。振子(Fig.16)は以下の運動方程式にしたがって運動する。

$$\ddot{\theta} = \omega \quad (6.2)$$

$$ml^2\ddot{\omega} = -\mu\omega + mgl\sin\theta + u \quad (6.3)$$

行動出力は回転トルク($T = u$)であり、最大トルク $u_{\max}$ を発生させても一度で頂上に到達することはできず、頂上に到達するためには、往復運動を必要とする。シミュレーション条件は、 $m = l = 1, g = 9.8, \mu = 0.01, u_{\max} = 5.0$, 報酬関数は $R(x) = \cos\theta$, また状態空間の定義域は $-2\pi < \theta < 2\pi, -3\pi < \omega < 3\pi$ である。物理モデル、強化学習ともに計算は0.02[sec]ごとに行なう。終了条件は、1回の試行時間は20[sec]経過時あるいは、定義域外に出たときである。ノード数 15×15 , 各試行1000ステップを1000試行行ない、関数近似器は各ステップでデータの更新を行なう。

Fig. 16 Pendulum swing up

Fig. 17 Puddle world

得られる報酬をより多くするためには、振り上げられるようになった後、その状態で静止させる必要がある。ノード移動なしの場合はFig.18に示すように十分な分解能が得られていないために振り上げた状態での静止を維持できていない。それに対し、500ステップ時にノード移動を行なった場合はFig.19のように、静止を維持できるように振り上げている時間が長くなっている。このことから、ノードの移動がある場合の状態価値関数(Fig.21)では、ノードの移動がない場合の状態価値関数(Fig.20)と比較して学習にとって重要な領域での分解能が大きくなっていることを意味している。しかし、この例題の状態価値関数は、勾配変化の小さい領域と大きい領域が大きく分かれられないような関数であるために、ノードの移動による効果は比較的少ない問題であると考えられる。

Fig. 18 Without motion

Fig. 19 With motion

6.3 水たまり問題(Puddle World Task)

Fig.17に示すような2次元平面上をエージェントが移動し、右上のゴール領域([0.9,0.9]から[1.0,1.0]までの正方形領域)を目指す。領域内には水たまり(puddle)が存在し、(中心溝からの距離) $\times (-400)$ の負の報酬を得る。移動は上下左

Fig. 20 Value function(1) Fig. 21 Value function(2)

右4方向で大きさ0.05, 1ステップで-1の報酬を得, 各移動ごとに標準偏差0.01のガウス分布のノイズを受ける。また, 状態価値関数の領域による勾配変化の違いがより顕著な例として, 上記と同様の問題設定で水たまりをなくし, ゴール領域を中央 $[(0.475, 0.475)]$ から $[0.525, 0.525]$ の正方形領域に設定した問題 (2D Grid world) での比較を行なった。2つの例題において得られた報酬の積算値 (10回の試行の平均値) を Table 1 に示す。ただし, 各試行におけるエージェントの初期位置は乱数により決定するが, 比較のためにノード移動のある場合とない場合とで同じ乱数の種を用いている。終了条件は, ゴール領域に到達するか 10000 ステップ行動時である。水たまり問題ではノード数 15×15 , 2D Grid world ではノード数 7×7 で試行回数はいずれも 100 である。

Table 1 Integrated reward

	Puddle world	2-D Grid world
Without motion	-2.17×10^4	-1.97×10^4
With motion	-1.97×10^4	-1.36×10^4

いずれの場合も得られた負の報酬の積算値は小さくなっているが, 水たまり問題では水たまりを適切に回避できるかどうか, ゴール到達問題ではより少ないステップ数でゴールに到達できるかが報酬の差となっている。勾配変化の少ない領域と大きい領域の差の大きい問題ほど, 本提案手法の有効性は大きくなることがわかる。

6.4 考察

シミュレーションより, 提案アルゴリズムが, 関数形状の複雑度に応じた分布を適応的に構成できることを確認した。また, ノード数固定で関数近似および強化学習問題への適用において性能を改善できる例を示した。その一方で, 位相構造が変化する (辺が新たに生成, 消去される) 場合に関数値が不連続に変化する, ノード同士が過剰に接近しすぎるなどの現象も見られた。より多次元の空間に安定して分布させるためには, 上記の問題への対応が重要である。

また, 計算処理はノード移動と位相関係の更新からなり, 前者の計算時間はノード数, 繰り返し数にほぼ比例する。後者は TRN アルゴリズムで観測データごとに最整合と第2最整合のノード計算を行ない, ノード移動に伴って行なうが, 計算時間比は前者に比べ十分に小さい。

7. おわりに

本研究では, 強化学習に適用するための関数近似手法として, グラフにおける逐次最小二乗法およびそれをもとにしたグラフ上の拡散方程式による設計を行ない, 実装した。シミュレーションにより, 勾配変化に応じてノード分布の疎密を決定する挙動設計が, 関数近似および強化学習での状態価値関数表現に有効であることを確認した。本手法を強化学習と独立に一般的な関数近似手法とみると, 近似関数の幾何的形状を利用したメモリ圧縮方法と考えることができる。近似対象によるメモリ, 計算時間効用などの議論も重要な課題である。

本研究では状態価値関数の近似のみを扱ったが, 行動価値関数近似, 多次元評価¹³⁾などの枠組みに適用することで, より複雑な問題への適用を図ることが今後の課題である。

《付 録》

A. Ferguson 曲線による平面表現

Ferguson 曲線は勾配と位置情報をもとに曲線を構成する計算式で, Bezier 曲線, Spline 曲線の基礎である。始点および終点を w_u, w_h , 始点および終点における方向ベクトルを v_o, v_t としたとき, これをつなぐ Ferguson 曲線の軌跡 P は媒介変数 $s (0 \leq s \leq 1)$ を用いて,

$$P(s) = p_3 s^3 + p_2 s^2 + p_1 s + p_0 \quad (A.1)$$

のように表わせる。ただし係数ベクトル $(p_0, \dots, p_3)$ は以下のように表わされる。

$$\begin{aligned} p_0 &= w_u, & p_2 &= -3w_u + 3w_h - 2v_o - v_t \\ p_1 &= v_o, & p_3 &= 2w_u - 2w_h + v_o + v_t \end{aligned} \quad (A.2)$$

また, 始点, 終点の方向ベクトルは

$$v_o = c_{hu}, \quad v_t = v_o \quad (A.3)$$

で与える。また, $\dot{P} = \frac{\partial P}{\partial s}$ である。ここで, s と t の間には

$$t = \frac{(P_x(s) - w_u) \cdot c_{hu}}{\|c_{hu}\|^2} \quad (A.4)$$

という関係が成り立つ。

B. ノード上関数の移動ベクトルによる微分

$g_{hu}(r)$ の微分はつぎのように表わされる。

$$\begin{aligned} \frac{\partial g_{hu}(r)}{\partial r} &= \left(\frac{\partial}{\partial r} \|a_u(r) - a_h\|^2 \right) \|c_{hu} - r\|^2 \\ &\quad + \|a_u(r) - a_h\|^2 \left(\frac{\partial}{\partial r} \|c_{hu} - r\|^2 \right) \end{aligned} \quad (B.1)$$

r による偏微分の項は以下のように展開できる。

$$\begin{aligned} \frac{\partial}{\partial r} \|a_u(r) - a_h\|^2 &= \frac{\partial}{\partial r} \sum_{j=1}^n (a_{hj} - a_{uj}(r))^2 \\ &= -2 \sum_{j=1}^n (a_{hj} - a_{uj}(r)) \frac{\partial}{\partial r} a_{uj}(r) \end{aligned} \quad (B.2)$$

$$\frac{\partial}{\partial r} \|c_h - r\|^2 = \frac{\partial}{\partial r} \sum_{j=1}^n (c_{hj} - r_j)^2$$

$$= -2(c_h - r) \quad (\text{B.3})$$

$\frac{\partial a_{ij}(r)}{\partial r}$ は (4.8) 式により計算される.

参考文献

- 1) Richard S. Sutton and Andrew G. Barto: Reinforcement Learning, MIT Press (1998)
- 2) J.S. Albus: A new approach to manipulator control: The cerebellar model articulation controller (cmac), Journal of Dynamic Systems, Measurement, and Control, Trans. ASME, **97**-3, 220/227 (1975)
- 3) 木村, 宮崎, 小林: 強化学習システムの設計指針, 計測と制御, **38**-10, 618/623 (1999)
- 4) Christopher G. Atkeson, Andrew W. Moore and Stefan Schall: Locally Weighted Learning for Control, Artificial Intelligence Review, **11**, 75/113 (1997)
- 5) Justin A. Boyan and Andrew W. Moore: Generalization in Reinforcement Learning: Safely Approximating the Value Function, Advance in Neural Information Processing Systems, **7**, MIT Press, Cambridge MA (1995)
- 6) Kazuyuki Samejima and Takashi Omori: Adaptive state space formation in reinforcement learning, Proc. of International Conference on Neural Information Processing (ICONIP)'98, 251/255 (1998)
- 7) Jun Morimoto and Kenji Doya: Reinforcement Learning of dynamic motor sequences: learning to stand up, Proc. of IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)'98, **3**, 1721/1726 (1998)
- 8) M. Asada, S. Noda and K. Hosoda: Action-based sensor space categorization for robot learning, Proc. of IEEE/RSJ Int. Conf. IROS96, 1502/1506 (1996)
- 9) H. Ishiguro, R. Sato and T. Ishida: Robot oriented state space construction, Proc. of IEEE/RSJ Int. Conf. IROS96, 1496/1501 (1996)
- 10) 高橋, 浅田: 実ロボットによる行動学習のための状態空間の漸次的構成, 日本ロボット学会誌, **17**-1, 118/124 (1999)
- 11) Y. Takahashi, M. Takeda and M. Asada: Continuous valued Q-learning for vision-guided behavior acquisition, Proc. of 1999 IEEE/SICE/RSJ Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems, 255/260 (1999)
- 12) 小林, 太田, 井上, 新井: 視覚情報を用いた状態・行動空間の自律的生成, **36**-11, 1029/1036 (2000)
- 13) Y. Kobayashi, H. Yuasa and T. Arai: Learning using multidimensional internal rewards, Proc. of IEEE/RSJ Int. Conf. on Intelligent Robot and Systems (2000)
- 14) 湯浅, 伊藤: グラフ上の反応拡散方程式と自律分散システム, 計測自動制御学会論文集, **31**-1, 1/7 (1999)
- 15) T. Kohonen: 自己組織化マップ, シュプリンガー・フェアラーク東京 (1998)
- 16) T. Martinetz and K. Shulten: Topology Representing Networks, Neural Networks, **7**-3, 507/522 (1994)
- 17) Kenji Doya: Reinforcement learning in continuous time and space, Neural Computation, No.12, 243/269 (2000)
- 18) M. E. Harmon and L. C. Baird III, Reinforcement learning applied to a differential game, Adaptive Behavior, No. 4, 3/28 (1996)
- 19) Katsunari Shibata and Yoichi Okabe: Reinforcement learning when visual sensory signals are directly given as inputs, Proc. of ICNN'97 Houston, **3**, 1716/1720 (1997)
- 20) Andrew G. Barto, Richard S. Sutton and Charles W. Anderson: Neurolike adaptive elements that can solve difficult learning control problems, IEEE Transactions on systems, man, and cybernetics, **SMC13**-5 (1983)
- 21) John N. Tsitsiklis and Benjamin Van Roy: An Analysis of Temporal-Difference Learning with Function Approxi-

mation, IEEE Transactions on Automatic Control, **42**-5, 674/690 (1997)

- 22) Richard S. Sutton: Generalization in Reinforcement Learning: Successful Example Using Sparse Coarse Coding, Advances in Neural Information Processing Systems, No.8, 1038/1044, MIT Press (1996)
- 23) Joerg Brucke, Ingo Ahrms and Gerald Sommer: An integrated architecture for learning of reactive behaviors based on dynamic cell structures, Robotics and Autonomous Systems, **22**-2, 87/101 (1997)
- 24) R. J. Williams: Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine Learning, No. 8, 229/256 (1992)
- 25) Dimitri P. Bertsekas and John N. Tsitsiklis: Neuro-Dynamic Programming, Athena Scientific (1996)
- 26) 黒瀬 能幸: 3次元図形処理工学, 共立出版 (1999)

[著者紹介]

小林 祐一

1997年東京大学工学部精密機械工学科卒業。99年同大学大学院工学系研究科精密機械工学専攻修了。同年同博士課程入学。現在に至る。修士(工学)。実ロボットを用いた行動学習の研究に従事。

湯浅 秀男 (正会員)

1986年名古屋大学大学院工学研究科博士前記課程修了(情報工学専攻)。同年名古屋大学工学部情報工学科助手。同電子機械工学科講師を経て、99年東京大学大学院工学系研究科助教授。また、93年より理化学研究所バイオ・ミメティックコントロール研究センターフロンティア研究員を併任(非常勤)。システム理論、ロボット工学の研究に従事。特に自律分散システム・自己組織化システムの研究を中心としている。博士(工学)。電気学会、システム制御情報学会などの会員。93年度、日本機学会 ROBOMECH 賞受賞。

新井 民夫 (正会員)

1977年同大学工学系大学院精密機械工学課程博士課程修了。同年同大学工学部講師。79年同助教授。工学(博士)。79年~81年英国エディンバラ大学人工知能学科客員研究員。83年~86年東京大学工学部総合試験所機械方面研究室助教授。87年より東京大学工学系研究科精密機械工学専攻教授。現在に至る。複数の柔らかいマニピュレータの協調制御、複数台の移動ロボットの動作計画、ロボットの環境設計などの研究に従事。