卒業論文

ターミナルアトラクタを組み込んだ
複製・競合メカニズムによる効率的な機械学習

Building Efficient Machine Learning
with Reproduction and Competition Mechanisms Incorporating
Terminal Attractor

富山県立大学 工学部 情報システム工学科
2120014 小澤 翔太
指導教員 奥原 浩之 教授
提出年月: 令和7年 (2025年)2月

目 次

図一覧

表一覧

記号一覧

以下に本論文において用いられる用語と記号の対応表を示す.

用語	記号
基底関数の番号	j
基底関数の数	M
入力ベクトルの番号	i
入力ベクトルの数	N
第 j 基底関数	$\xi_j(\mathbf{x})$
第 j 基底関数の内的自然増加率	α_j
第 j 基底関数と第 h 基底関数による競合の効果	γ_{jh}
第 j 基底関数の重み	w_j
第 j 基底関数の中心位置	$\mathbf{m}_j$
第 j 基底関数の広がり	$\mathbf{\Sigma}_j$
第 j ニューロンが入手できる NGF の量	g_j
第 j ニューロンの微小領域における環境の揺らぎ	f_j
微小領域への NGF の供給速度	G
教師信号	$\eta(\mathbf{x})$
正の定数	θ_j, θ'_j
累積二乗誤差関数	$E(\mathbf{w})$
線形和 (近似結果)	$s(\mathbf{x})$
逆温度パラメータ	β
学習率	Δ

第1章

はじめに

§ 1.1　本研究の背景

近年，人工知能や機械学習の発展に伴い，複雑なデータの解析や予測を可能にするさまざまな手法が提案されている．その中でも関数近似問題やパターン識別に適したニューラルネットワークの一つとして注目されているのが，動径基底関数ネットワーク（Radial Basis Function Network: RBFN）[1] である．RBFN は，最初に有限個の入出力データを補完する方法として提案されたもので，3 層構造を持つニューラルネットワークである．その設計において，RBFN は多層パーセプトロンと同様に任意の非線形関数の近似が可能であるという特長を持つ．

RBFN の最大の特徴の一つは，基底関数としてガウス関数を採用している点である．これにより，階層型ニューラルネットワークと比較してニューロンごとの局所的な学習が可能となり，収束の速さやモデルの柔軟性において優れた特性を示す．RBFN は様々な応用領域においてその有用性が示されている．例えば，非線形関数の線形和を用いてがけ崩れ発生限界雨量線を設定する研究 [4] や，複数の波源から発生した信号の到来方向を予測するためのツール [5] として活用されている．これらの応用例は，RBFN が非線形関数の近似器として非常に強力なツールであることを示している．

しかしながら，RBFN にはいくつかの課題が存在する．その一つが，未知の非線形関数を近似する際に必要となるニューロン数が事前には分からないという問題である．特に多くの冗長なニューロンを含む場合，学習プロセスが遅延し，さらに過学習のリスクが増大することが知られている．

これらの問題を解決するために，適者生存型学習則に基づいたシナプス可塑性方程式を適用した競合動径基底関数ネットワーク（Competitive RBFN: CRBFN）[2] が提案されている．CRBFN は，ニューロン間に競合を発生させることで学習に必要なニューロンのみが選択され，不要な冗長ニューロンを自然に排除することが可能である．この特性により，モデルの簡潔化と学習効率の向上が実現される．一方で，CRBFN は基底関数を新たに追加する機能を備えていないため，ニューロンの数が不足している場合には十分な関数近似が困難になるという欠点がある．

これを受けて，CRBFN に基底関数を複製する機能を付加した，複製・競合動径基底関数ネットワーク（Reproductive CRBFN: RC-RBFN）[3] が提案されている．RC-RBFN は新しいデータや状況に対応するために必要な基底関数を動的に複製し，さらに競合を活用して不要な基底関数を排除することで，モデルの柔軟性と効率を向上させている．この機構により，基底関数の過剰複製を抑制しつつ関数近似の精度向上が期待されている．

§ 1.2　本研究の目的

機械学習の主な手法である教師あり学習や教師なし学習，強化学習，そして逆強化学習にいたるまでガウス関数を動径基底関数としたRBFNを用いた学習手法は存在するものの，多くの研究事例でガウス関数の数が過多であったり過少であったりすることが課題となっている．この問題は過学習を発生させるリスクを生じさせたり，近似精度への悪影響を及ぼしたりする可能性がある．

その大きな理由として任意の非線形関数を近似するにあたって，事前に近似に必要なガウス関数の数を把握することが非常に困難であることが挙げられる．この問題に対処するために，RC-RBFNが提案され，基底関数の動的な複製と競合による冗長な基底関数の削除を組み合わせることで，モデルの柔軟性や効率性を大幅に向上させる仕組みが実現された．RC-RBFNは基底関数の数をデータや状況に応じて動的に調整できる点で，従来の固定された基底関数の数を前提とした手法に比べて高い適応性を持ち，特に多様なデータ構造や非線形関数の近似において優れた性能を発揮することが期待されている．

しかし，RC-RBFNの理論は提唱されているものの実際にシステムとして実装されている事例はない．また，RC-RBFNやその基盤であるCRBFNは目的関数の次元数が増加するに伴い，計算量が指数関数的に増大する，いわゆる「次元の呪い」の影響を受けやすいという課題を抱えている．この問題は特に高次元データを扱う場合に顕著であり，計算負荷が非常に大きくなることで，現実的な時間内で学習を終了させることが困難になる可能性がある．

このような背景を踏まえ，従来のRC-RBFNを基盤としながら，次元の呪いに起因する計算負荷の増大を軽減するための新たな手法の提案を目的とする．本研究では特に，RC-RBFNの計算負荷の主要因となっている高次元データにおける多重積分処理部分に着目し，このアルゴリズムを改良することで効率的かつ精度の高い学習を実現する方法を検討する．さらに，学習時間の上限を決定することのできるターミナルアトラクタについても説明し，RC-RBFNの重みの学習に適用することで学習の効率化を図る．

また，プログラムの開発にPythonを用いることで豊富な標準ライブラリと，データ分析や科学計算に適したサードパーティライブラリを扱えるようにした．特に，NumPy，Numba，SciPyなどのライブラリは演算処理において有用なライブラリであり，並列化処理や実行時(Just In Time: JIT)コンパイルによる高速化，高度な数値演算を実現している．そして数値実験では，粒子群最適化や進化的アルゴリズムのベンチマークに用いられるベンチマーク関数を目的関数として設定し，関数近似の結果を示す．

さらに，既存の教師あり学習や教師なし学習におけるガウス関数を動径基底関数としたRBFNの事例を紹介したのち，今後の展望としてRC-RBFNをそのシステムに組み込むことを提案する．また，強化学習における価値関数や逆強化学習における報酬関数の近似への適用も提案する．これにより，RC-RBFNが従来の固定構造モデルの限界を克服し，学習効率や汎化性能を向上させる可能性を示す．最後に得られた実験結果についての考察を行い，今後の課題についても言及する．

§ 1.3　本論文の概要

本論文は次のように構成される．

第 1 章 本研究の背景と目的について説明した．背景では関数近似手法として用いられる RBFN やそれを発展させた RC-RBFN について述べた．目的では RC-RBFN の数値計算上の課題について述べ，本研究の意義について述べた．

第 2 章 関数近似に不要な動径基底関数を効率的に削除する手法として，適者生存型学習則を適用した CRBFN について述べる．また，新たな動径基底関数を追加する機能を備えた CRBFN である RC-RBFN についてまとめる．

第 3 章 現在の機械学習手法の概要を示し，その中でも RBFN を活用した手法について詳しく紹介する．また，RC-RBFN のを既存の学習システムに組み込むことの有用性について紹介する．

第 4 章 複製機能の数値計算上の課題に対するアプローチ方法を述べ，学習の流れとプログラムの高速化に用いた Python ライブラリについて述べる．

第 5 章 第 4 章で述べた提案手法を用いてベンチマーク関数の近似を行い，関数近似器の評価を行う．

第 6 章 本論文における前章までの内容をまとめつつ，本研究で実現できたことと今後の展望について述べる．

第2章

複製・競合を考慮した動径基底関数ネットワーク

§ 2.1　競合を考慮した動径基底関数ネットワーク

ニューラルネットワークは大きく分けて，素子であるニューロン，それらを結合するシナプス，そして動作規則により構成される．なかでも，記憶にもっとも関係した情報処理は，シナプスにおいて行われているとされる．記憶には種々のものが考えられるが，本研究では短期記憶と長期記憶に着目し，短期記憶はニューロンの発火頻度，長期記憶は細胞膜の特性の変化により生じるものとする．シナプスの可塑性を記述する方程式は，これらの要因を含んだものとなっていなければならない．さらに，成長や活動に必要な神経成長因子（Nerve Growth Factor: NGF）はシナプス間隔の微小な領域において競合が発生することによりシナプスに摂取される．第 j ニューロンの微小領域におけるシナプス前終末発火頻度を ξ_j とし，これが作用するニューロンの細胞膜におけるシナプス後発火頻度を η とする．また，第 j ニューロンへの入力数を $i\,(i = 1, ..., N)$ とする．

そこで，発火頻度や膜の特性変化を生じる物質の時間変化と，微小な領域での競合を考慮したシナプス結合荷重の大きさの時間変化は以下の方程式に従うものとする．

$$\frac{dw_j}{dt} = \alpha_j w_j + g_j w_j + f_j \tag{2.1}$$

ここで，$w_j \geq 0$ である．また，g_j は微小領域に供給される NGF のうち，その領域に付着している第 j ニューロンのシナプスが入手できる量であり，f_j は NGF と環境因子に依存するゆらぎである．α_j は内的自然増加率である．内的自然増加率は脳の可塑性を表す Hebb 則 [7] を意味する．内的自然増加率 α_j は

$$\alpha_j = \sum_{i=1}^{N} \eta\left(\mathbf{x}_i\right) \xi_j\left(\mathbf{x}_i\right) \tag{2.2}$$

で定義される．また，NGF の量 g_j は次の方程式に従う．

$$\frac{dg_j}{dt} = \theta_j\left(G - g_j\right) - \left(\theta'_j w_j + \sum_{h \neq j} \theta'_h w_h\right) = \theta_j\left(G - g_j\right) - \sum_{h} \theta'_h w_h \tag{2.3}$$

G は微小領域への NGF の供給速度であり，膜の特性により決定される変数である．θ_j，θ'_h は正の定数である．NGF の量の時間変化もシナプスの興奮性，抑制性によらず，そのシナプス間感度の大きさに依存する．また，領域へ付着するシナプスが入手し得る NGF の

量の時間変化に対し，NGF の供給速度の時間変化が無視できるとして G を定数とみなす．シナプス間感度の時間変化は発火頻度に依存するため，シナプス間感度の時間変化に対し，NGF の量の時間変化は無視できるものとする．そこで，第 j ニューロンと第 h ニューロンが同時に NGF を消費することによる競合の効果 γ_{jh} を導入すると以下のシナプス可塑性方程式が導かれる．

$$\frac{dw_j}{dt} = \left(G + \alpha_j - \frac{1}{\theta_j} \sum_h \theta'_h w_h \right) w_j + f_j = \left(G + \alpha_j - \sum_h \gamma_{jh} w_h \right) w_j + f_j \tag{2.4}$$

ここでは，競合係数 γ_{jh} を

$$\gamma_{jh} = \frac{\theta'_h}{\theta_j} = \sum_{i=1}^{N} \xi_j (\mathbf{x}_i) \, \xi_h (\mathbf{x}_i) \tag{2.5}$$

で定義する．簡単のため，NGF の摂取量が一定 $(G = 0)$ で揺らぎのない場合 $(f_j = 0)$ を考える．このとき，シナプス可塑性方程式は

$$\frac{dw_j}{dt} = \left(\alpha_j - \sum_h \gamma_{jh} w_h \right) w_j \tag{2.6}$$

と表される．

適者生存型学習則を適用した RBFN

ここで，シナプス後発火頻度 $\eta (\mathbf{x})$ を入力 $\mathbf{x}$ に対する望ましい出力，シナプス前発火頻度 $\xi_j (\mathbf{x})$ を動径基底関数であるとみなすことで，シナプス結合荷重 w_j は競合を行いながら RBFN と同様に望ましい出力と動径基底関数の二乗誤差関数を減少させることが示される．本論文では，式 (2.6) のシナプス可塑性方程式を適者生存型学習則とする．このようなモデル式は一般 Lotka-Volttera 競争系と呼ばれ，競争関係にある生物の個体数の変動を表す数理モデルの一種である．

次に，この適者生存型学習則を RBFN に適用する．RBFN では未知の非線形関数を近似するためにあらかじめ必要なニューロン数が不明であり，冗長なニューロンを必要とする．そこで，適者生存型学習則によって近似に不要な冗長なニューロンを削除する機能を備えた RBFN として CRBFN がある．CRBFN ではシナプス結合荷重間に競合を生じさせ，学習に必要なニューロンのみが自然に生き残り，学習の効率化を図ることができる．RBFN における動径基底関数として一般的にガウス関数が用いられることが多く，本研究でもガウス関数を用いる．このとき，動径基底関数 $\xi_j (\mathbf{x})$ は

$$\xi_j (\mathbf{x}) = exp \left\{ -\frac{1}{2} (\mathbf{x} - \mathbf{m}_j)^{\mathrm{T}} \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \mathbf{m}_j) \right\} \tag{2.7}$$

で定義される．T は行列の転置を表す．パラメータ $\mathbf{m}_j$，$\boldsymbol{\Sigma}_j^{-1}$ はそれぞれ j 番目の動径基底関数の中心位置と分散共分散行列であり，$\mathbf{m} \equiv [\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_M]$，$\boldsymbol{\Sigma} \equiv [\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, ..., \boldsymbol{\Sigma}_M]$ である．

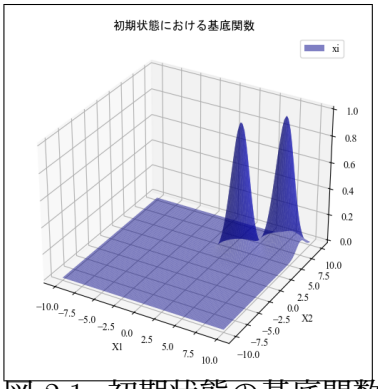

図 2.1: 初期状態の基底関数

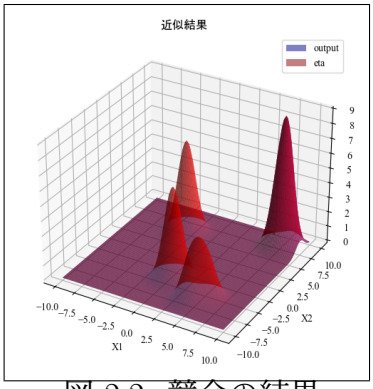

図 2.2: 競合の結果

RBFN による関数近似は二乗誤差関数 $E(\mathbf{w})$

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{\eta(\mathbf{x}_i) - s(\mathbf{x}_i)\}^2 \tag{2.8}$$

を減少させることにより実現される．ここで，$\mathbf{w} \equiv [w_1, w_2, ..., w_M]$ であり，

$$s(\mathbf{x}) = \sum_{j=1}^{M} w_j \xi_j(\mathbf{x}) \tag{2.9}$$

である．式 (2.9) の右辺は微小領域に付着しているすべてのシナプス前終末のシナプス前発火頻度 $\xi_j(\mathbf{x}_i)$ と，シナプス結合荷重 w_j との積の総和であるので，$s(\mathbf{x})$ は神経伝達物質放出量となる．

つまり，RBFN が学習により獲得しなければならないのは，第 j ニューロンのシナプス結合荷重 w_j，パラメータ $\mathbf{m}_j$ ならびにパラメータ $\mathbf{\Sigma}_j$ である．学習アルゴリズムに最急降下法を適用することで

$$\frac{dw_j}{dt} = -\Delta\frac{\partial E(\mathbf{w})}{\partial w_j} = \Delta\sum_{i=1}^{N}\{\eta(\mathbf{x}_i) - s(\mathbf{x}_i)\} w_j \xi_j(\mathbf{x}_i) \tag{2.10}$$

$$\frac{d\mathbf{m}_j}{dt} = -\Delta\frac{\partial E(\mathbf{w})}{\partial \mathbf{m}_j} = \Delta\sum_{i=1}^{N}\{\eta(\mathbf{x}_i) - s(\mathbf{x}_i)\} w_j \xi_j(\mathbf{x}_i)\frac{(\mathbf{x}_i - \mathbf{m}_j)}{{\mathbf{\Sigma}_j}^2} \tag{2.11}$$

$$\frac{d\mathbf{\Sigma}_j}{dt} = -\Delta\frac{\partial E(\mathbf{w})}{\partial \mathbf{\Sigma}_j} = \Delta\sum_{i=1}^{N}\{\eta(\mathbf{x}_i) - s(\mathbf{x}_i)\} w_j \xi_j(\mathbf{x}_i)\frac{(\mathbf{x}_i - \mathbf{m}_j)^2}{{\mathbf{\Sigma}_j}^3} \tag{2.12}$$

が得られる．Δ は学習率であり，適当な正の定数である．このような適者生存型学習則に従う RBFN を CRBFN と呼ぶ．図 2.1，図 2.2 にシナプス結合荷重の競合の様子を示す．初期状態では二つあった基底関数が CRBFN によって学習が行われることで競合が生じ，学習終了時には一つに減少していることがわかる．

§ 2.2　望ましい時刻で収束できるターミナルアトラクタ

ある動的なシステム系が特別な条件を満たす場合に収束時間を指定することが可能になるターミナルアトラクタについて述べる．まず，微分方程式における初期値問題に対する解の一意性を保証する条件であるリプシッツ条件について説明する．

リプシッツ条件

$f(t,x)$ を，閉区間 $\Omega = (t,x) \,|\, |t-t_0| \leqq a, |x-x_0| \leqq b$ で定義された関数とする．ある定数 L があり，Ω 内の 2 点 $(t,x),(t,y)$ において，

$$|f(t,x) - f(t,y)| \leqq L|x-y| \tag{2.13}$$

となるとき，f はリプシッツ条件を満たす．

このリプシッツ条件が満たされると，それぞれの初期値問題に対して一意な解が存在し，その解の軌道は漸近的に平衡点に近づく．すなわち，軌道は平衡点に近づくだけで，有限時間内で平衡点に到達できない．これは CRBFN に置き換えると，競合に負けたシナプス結合荷重は有限時間で 0 に到達することができないということである．

そこで，このリプシッツ条件を破るという考えに基づいて解の一意性を破ることにより，有限時間内でニューラルネットワークが平衡点に収束することを示した [9] [10]．このような安定な平衡点をターミナルアトラクタと呼ぶ．このターミナルアトラクタの概念を CRBFN に適用することで，収束時間の上限値を指定できるようにした．ここでは，時刻 t^* で平衡解へ収束できるように修正されたシナプス可塑性方程式を導出する．

いま，正定関数 $V(\mathbf{w})$ として

$$V(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{\eta(\mathbf{x}_i) - s(\mathbf{x}_i)\}^2 \tag{2.14}$$

を定義する．正定関数 $V(\mathbf{w})$ はシナプス後発火頻度 $\eta(\mathbf{x}_i)$ と神経伝達物質放出量 $s(\mathbf{x}_i)$ の差を表す指標である．シナプス後発火頻度 $\eta(\mathbf{x}_i)$ が時間に依存しないと仮定すると，その時間変化が

$$\begin{aligned}
\frac{dV(\mathbf{w})}{dt} &= \sum_{j=1}^{M}\frac{\partial V(\mathbf{w})}{\partial w_j}\frac{dw_j}{dt} \\
&= -\sum_{j=1}^{M}\left[\sum_{i=1}^{N}\eta(\mathbf{x}_i)\,\xi_j(\mathbf{x}_i) - \sum_{h=1}^{M}\sum_{i=1}^{N}\xi_j(\mathbf{x}_i)\,\xi_h(\mathbf{x}_i)\,w_h\right]\frac{dw_j}{dt} \\
&= -\sum_{j=1}^{M} w_j\left(\alpha_j - \sum_{h=1}^{M}\gamma_{jh}w_h\right)^2 \leq 0
\end{aligned} \tag{2.15}$$

となるため，正定関数 $V(\mathbf{w})$ が Lyapunov 関数となることがわかる．さらに，$V(\mathbf{w}) > 0$ であり $\frac{dV(\mathbf{w})}{dt} \leq 0$ であることから，このシステムは漸近安定であるということもわかる．

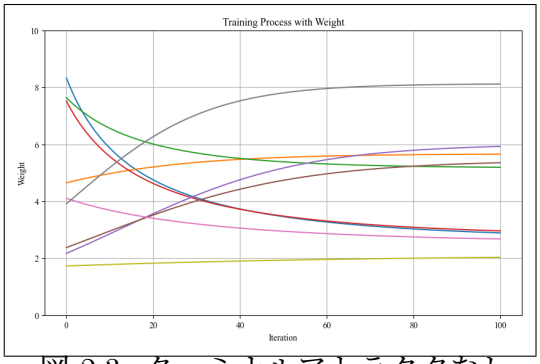

図 2.3: ターミナルアトラクタなし

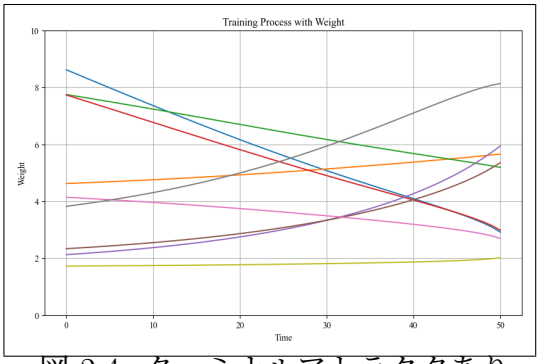

図 2.4: ターミナルアトラクタあり

ここで，重み w_j の時間微分を

$$\frac{dw_j}{dt} = -\Delta \frac{V(\mathbf{w})^c}{\sum_{j=1}^{M} \left(\frac{\partial V(\mathbf{w})}{\partial w_j} \right)^2} \frac{\partial V(\mathbf{w})}{\partial w_j} \tag{2.16}$$

とおくと，$V(\mathbf{w})$ の時間微分は

$$\frac{dV(\mathbf{w})}{dt} = \sum_{j=1}^{M} \left\{ \frac{\partial V(\mathbf{w})}{\partial w_j} \frac{dw_j}{dt} \right\} = -\Delta V(\mathbf{w})^c \leq 0 \tag{2.17}$$

となる．ただし，c は $0 < c < 1$ を満たす．$V(\mathbf{w})$ は時間とともに単調減少し，平衡点は漸近安定となることがわかる．このときの収束時間 t^* は，

$$\begin{aligned} t^* &= \int_0^{t^*} dt = \int_{V(\mathbf{w}_0)}^{V(\mathbf{w}_{t^*})} dV(\mathbf{w}) \frac{dt}{dV(\mathbf{w})} \\ &= \frac{V(\mathbf{w}_0)^{1-c} - V(\mathbf{w}_{t^*})^{1-c}}{\Delta(1-c)} \leq \frac{V(\mathbf{w}_0)^{1-c}}{\Delta(1-c)} \end{aligned} \tag{2.18}$$

で与えられ，有限時間内で収束することがわかる．$V(\mathbf{w}_0)$ は重みの初期値を用いて計算した Lyapunov 関数 $V(\mathbf{w})$ の初期値で，$V(\mathbf{w}_{t^*})$ は平衡点での $V(\mathbf{w})$ の値である．$V(\mathbf{w}_{t^*}) = 0$ の場合，式 (2.18) の等号が成立する．そこで学習率 Δ を，

$$\Delta = \frac{t^*(1-c)}{V(\mathbf{w}_0)^{1-c}} \tag{2.19}$$

とすると収束時間を指定できる．このターミナルアトラクタの概念を CRBFN における重みの学習に適用することで，初期状態において基底関数の数が多い場合でも従来の更新則よりも速く学習を終了させることが可能となる．

また，[9] [10] で提案されているターミナルアトラクタを基に，違う形で提案されたターミナルアトラクタ [2] も存在する．本研究では [2] で提案されているターミナルアトラクタを用いて基底関数の重みの学習を行う．

まず，望ましい時刻 t^* で収束するシナプス結合荷重の時間変化を Lyapunov 関数を用いて規定する．そこで，Lyapunov 関数の時間変化を

表 2.1: ターミナルアトラクタ適用前後の数値比較

重み	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
適用前	2.88	5.65	5.19	2.95	5.92	5.35	2.67	8.11	2.03
適用後	2.91	5.65	5.18	2.98	5.93	5.35	2.69	8.12	2.01

$$\frac{dV(\mathbf{w})}{dt} = -\frac{V(\mathbf{w}_0)^R V(\mathbf{w})^{\frac{1}{r}}}{Rt^*} \tag{2.20}$$

で定義する．ここで，r は 1 ではない任意の奇数であり，$R = \frac{(r-1)}{r}$ である．このような定義が可能になったのは，Lyapunov 関数が導出され，望ましい出力が動径基底関数を定数倍して足し合わせることで実現できる特別の場合を考えているからである．シナプス可塑性方程式は

$$\frac{dw_j}{dt} = \Delta \left(\alpha_j - \sum_{h=1}^{M} \gamma_{jh} w_h \right) w_j \tag{2.21}$$

とすることができる．このとき，Lyapunov 関数の時間変化は

$$\frac{dV(\mathbf{w})}{dt} = -\Delta \sum_{j=1}^{M} w_j \left(\alpha_j - \sum_{h=1}^{M} \gamma_{jh} w_h \right)^2 \tag{2.22}$$

となる．ここで，Δ は式 (2.20) と式 (2.22) から決定することが可能で，

$$\Delta = \frac{1}{\sum_{j=1}^{M} w_j \left(\alpha_j - \sum_{h=1}^{M} \gamma_{jh} w_h \right)^2} \frac{V(\mathbf{w}_0)^R V(\mathbf{w})^{\frac{1}{r}}}{Rt^*} \tag{2.23}$$

と導出される．以上より，望ましい時刻 t^* で平衡解へ収束するシナプス可塑性方程式を

$$\frac{dw_j}{dt} = \frac{\left(\alpha_j - \sum_{h=1}^{M} \gamma_{jh} w_h \right) w_j}{\sum_{j=1}^{M} w_j \left(\alpha_j - \sum_{h=1}^{M} \gamma_{jh} w_h \right)^2} \frac{V(\mathbf{w}_0)^R V(\mathbf{w})^{\frac{1}{r}}}{Rt^*} \tag{2.24}$$

で定義することができる．

図 2.3，図 2.4 にターミナルアトラクタ適用前後の重み w の学習過程を，表 2.1 に実際の数値による比較を示す．ターミナルアトラクタ適用前では重み w_1〜w_9 の数値を得るのに 100 回の学習を必要としていたが，適用後の数値を見ると指定した 50 回の学習の時点で適用前と同様の結果を得られていることがわかる．

§ 2.3　基底関数の複製を考慮した関数近似

2.1 節で冗長なニューロンを削除する手法を提示してきた．これに対して，ニューラルネットワークでの関数近似に必要な数のニューロンが存在しない場合は，関数近似をすること自体が不可能となる．そこで，新たに必要なニューロンを追加する手法が提案されている [11]．ところが，このような手法では基準となるしきい値の決定自体が困難であることが予想できる．また，教師信号が動的に変化する環境では削除する手法と追加する手法を組み合わせて学習を行わなければならない．当然それぞれにとって良い手法を，ただそのまま組み合わせただけでは，動径基底関数の数が振動するなどして望ましい結果が得られるとは限らない．

そこで新しい動径基底関数を追加する手法がある．この手法は必要な動径基底関数を効率的に追加することができる．そして，先に提案された CRBFN にこの手法を組み合わせたニューラルネットワークが RC-RBFN である．この RC-RBFN は，環境の変化に適応する能力を備えたものとなっている．

まず，パラメータが従う確率密度関数の導出を行う．ここでは，CRBFN のシナプス結合荷重と平均ベクトル，標準偏差が学習終了時にとる同時確率密度 $p(\mathbf{w}, \mathbf{m}, \mathbf{\Sigma})$ を導出する．シナプス結合荷重 w_j を

$$y_j^2 = w_j \tag{2.25}$$

と変数変換する．y_j の定義域は任意の実数である．このとき，式 (2.6) は

$$\frac{dy_j}{dt} = \left(\frac{\alpha_j}{2} - \sum_{h=1}^{M} \frac{\gamma_{jh}}{2} y_h^2 \right) y_j \tag{2.26}$$

となる．式 (2.26) は積分条件

$$\frac{\partial}{\partial y_h} \frac{dy_j}{dt} = \frac{\partial}{\partial y_j} \frac{dy_h}{dt} \tag{2.27}$$

を満たすため，ポテンシャル

$$U'(\mathbf{y}, \phi) = -\sum_{j=1}^{M} \int_a^{y_j} \left(\frac{\alpha_j}{2} - \sum_{h=1}^{M} \frac{\gamma_{jh}}{2} \right) y_j' dy_j' \tag{2.28}$$

を考えることができる．ただし $\mathbf{y} \equiv [y_1, y_2, ..., y_M]$ であり，パラメータ ϕ_j は平均ベクトルと共分散行列の集合 $\{\mathbf{m}_j, \mathbf{\Sigma}_j\}$ である．そして ϕ で集合 $\{\phi_1, \phi_2, ..., \phi_M\}$ を表す．変数 y_j の時間変化はポテンシャル $U'(\mathbf{y}, \phi_j)$ から，

$$\frac{dy_j}{dt} = -\frac{\partial U'(\mathbf{y}, \phi)}{\partial y_j} \tag{2.29}$$

で導くことができ，式 (2.26) からポテンシャル $U'(\mathbf{y}, \phi)$ は

$$U(\mathbf{w}, \phi) = -\sum_{j=1}^{M} \left\{ \frac{\alpha_j}{4} - \sum_{h \neq j}^{M} \frac{\gamma_{jh}}{4} w_j w_h - \frac{\gamma_{jj}}{8} w_j^2 \right\} \tag{2.30}$$

と書き直すことができる．この結果，

$$E(\mathbf{w},\phi) = 4U(\mathbf{w},\phi) + \frac{1}{2}\sum_{i=1}^{N}\eta^2(\mathbf{x}_i) \tag{2.31}$$

であることが示される．ただし，$\mathbf{x}_i \equiv [x_1, x_2, ..., x_N]$ である．よって，累積二乗誤差関数 $E(\mathbf{w},\phi)$ の最小化はポテンシャル $U(\mathbf{w},\phi)$ の最小化と等価であることがわかる．

いま，式 (2.26) に従う y_j はポテンシャル $U'(\mathbf{w},\phi)$ の最急降下方向に更新される．その結果，ひとたび極小解に収束すると，そこから逃れることができなくなる．そこで，極小解から脱出させるための手法として，y_j の更新則を

$$y_j(t+\Delta t) = y_j(t) - \frac{\partial U(\mathbf{w},\phi)}{\partial y_j}\delta t + \sqrt{Q\Delta t}n_j(t) \tag{2.32}$$

のようにノイズを考慮し離散近似した見本過程で与えることが考えられる．ただし，$n_j(t)$ は独立な確率変数であり，平均 0，分散 1 の正規分布 $N(0,1)$ に従う．Q は任意の正の定数である．このとき，学習終了時に CRBFN のシナプス結合荷重と平均ベクトル，標準偏差が満たす同時確率密度 $p_\beta(\mathbf{w},\phi)$ は

$$p_\beta(\mathbf{w},\phi) = Z_\beta^{-1} exp\{-\beta U(\mathbf{w},\phi)\} \tag{2.33}$$

で得ることができる．ここで β は逆温度である．Z_β は分配関数であり

$$Z_\beta = \int_{\mathbf{w}}\int_{\phi} exp\{-\beta V(\mathbf{w},\phi)\}\, d\mathbf{w}d\phi \tag{2.34}$$

で定義される．また，式 (2.33) はポテンシャル $V(\mathbf{w})$ と累積二乗誤差関数 $E(\mathbf{w})$ の関係式 (2.31) より

$$p_{\beta'}(\mathbf{w}) = Z_{\beta'}^{-1} exp\{-\beta' E(\mathbf{w})\} \tag{2.35}$$

と書き直すことができる．ここで，$\beta' = \frac{\beta}{4}$ である．また，$Z_{\beta'}$ は分配関数である．

以上のようにして，パラメータが従う確率密度関数が導出できたことにより，与えられた条件のもとで累積二乗誤差関数 $E(\mathbf{w})$ を最小とするパラメータの値が検出できることを示す．ここでは，教師信号 $\eta(\mathbf{x})$ を

$$\eta(\mathbf{x}) = \sum_k w_k N(\mathbf{m}_k, \mathbf{\Sigma}_k) \tag{2.36}$$

で与えることとする．$N(\mathbf{m},\mathbf{\Sigma})$ は中心位置 $\mathbf{m}$，分散共分散行列 $\mathbf{\Sigma}$ の正規密度関数を表す．この教師信号を 1 つの動径基底関数だけを用いて近似することを考える．この場合，近似しようとしている非線形関数 $\eta(\mathbf{x})$ の複雑さに対し，必要とされる動径基底関数が十分に存在していないため，累積二乗誤差関数を 0 にすること自体が不可能である．しかし，この動径基底関数のパラメータ m が従う条件付き確率密度関数

$$p_{\beta'}(m|w,\Sigma) = \frac{p_{\beta'}(w)}{\int_m p_{\beta'}(w)\, dm} \tag{2.37}$$

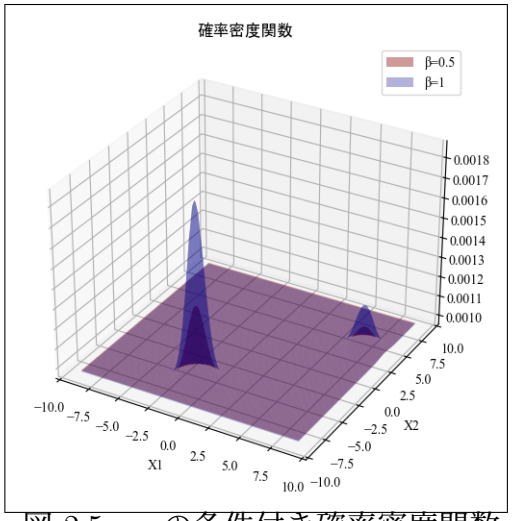

図 2.5: $\mathbf{m}$ の条件付き確率密度関数

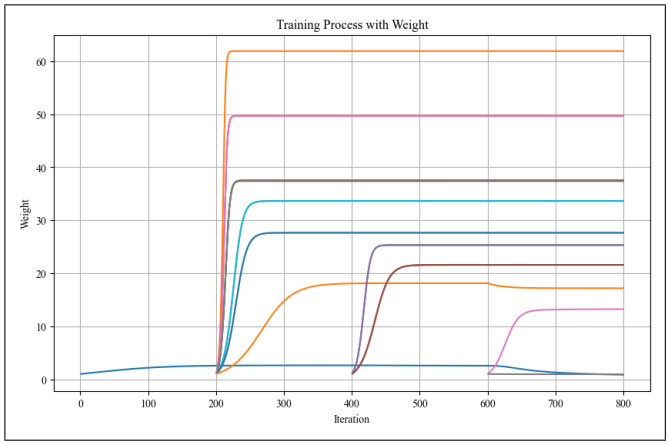

図 2.6: 複製の様子

は導出することができ，図 2.5 のようになる．

この結果から，あるパラメータをもつ動径基底関数が与えられた条件のもとで累積 2 乗誤差関数を最小とするためには，パラメータ $\mathbf{m}$ を条件付き確率 $p_{\beta'}(\mathbf{m}|w, \mathbf{\Sigma})$ を最大とする値に定めればよいことがわかる．また，もし同じ形質をもつ動径基底関数を一つ追加することができるなら，条件付き確率 $p_{\beta'}(\mathbf{m}|w, \mathbf{\Sigma})$ を極大とするパラメータ $\mathbf{m}$ へ配置することが最も累積二乗誤差関数を小さくできることもわかる．

一般に，式 (2.11) に従いパラメータを更新し続けると極小解にとらわれ，累積二乗誤差関数の値を 0 にすることができないことがある．または，近似しようとしている非線形関数 $\eta(\mathbf{x})$ の複雑さに対し，必要とされる動径基底関数が十分に存在していないときには，二乗誤差関数の値を 0 にすること自体が不可能である．

ところで，確率的な要素や未知の教師信号などが存在しないものとするなら，すべての入力ベクトル $\mathbf{x}_i$ ごとに動径基底関数を作成し，シナプス結合荷重が $w_i = \eta(\mathbf{x}_i)$ かつパラメータ $\mathbf{\Sigma}_i \to 0$ であるときに，パラメータ $\mathbf{m}_i$ が $\mathbf{x}_i$ となることで近似的に 0 とできる場合がある．もちろん，多くの問題ではすべての入力ベクトルについて動径基底関数を用意しなくても，このようなことが可能であるものと思われる．

そこで，累積二乗誤差関数の値がある正数 $\epsilon > 0$ より大きな値に収束して学習が収束したと判断されるときに，新たに必要な動径基底関数を追加する．この手法では，先ほど導出した確率密度関数を利用しているため，学習が収束した時点で得られている動径基底関数の一部の形質（シナプス結合荷重 w_j，パラメータ $\mathbf{\Sigma}_j$）が新たに追加される動径基底関数がもつパラメータに引き継がれている．そのため，効率的に最も累積二乗誤差関数を小さくするパラメータ $\mathbf{m}$ に動径底関数を追加していくことができる．なおかつ，最悪の場合にはすべての入力ベクトル $\mathbf{x}_i$ をパラメータ $\mathbf{m}_i$ とする動径基底関数を作成することができる．図 2.6 に学習の過程で不足している動径基底関数が複製されている様子を示す．

第3章

機械学習手法と動径基底関数ネットワーク

§ 3.1　代表的な機械学習の手法について

現在，機械学習は学術界だけでなく産業界においても幅広く用いられ，人工知能技術のコア技術として重要な役割を果たしている．機械学習はもともと人と同様の知能的機能を実現させるために研究開発が進められてきた分野であるが,「データから学ぶ」という過程とデータ解析・統計学の方法論がうまくマッチし，現在では狭い意味での人工知能としての使い方にとどまらず幅広いデータ科学の方法論として発展している．

その中でも RBFN は機械学習の分野で注目される手法の一つであり，特に関数近似や分類，クラスタリングなどの問題で優れた性能を示す．RBFN は入力空間内のデータポイントに基づいて局所的な基底関数を活用し，非線形な関係を効果的にモデル化できる点で特徴的である．さらに，RBFN は教師あり学習と教師なし学習の両方で活用可能であり，前者では分類や回帰の問題に，後者ではクラスタリングや特徴抽出に応用される．機械学習の概要として，RBFN を含む機械学習の代表的な問題設定および RBFN を活用した機械学習のツール (教師あり学習，教師なし学習，半教師あり学習) について述べる．

教師あり学習

教師あり学習は機械学習の中でも特に基本的な問題である．教師あり学習はあらかじめ定めた分類に学習データを振り分けるための「分類」と，連続するデータにおいて入力データ (特徴量) とそれに対応する連続値の出力との関係をモデル化する「回帰」に分けることができる．

分類問題における目標点はある入力 (画像やテキストなど) に対して，そのラベル (画像に写っている物体や人などの離散的なカテゴリ) を予測し，分類することである．与えられた学習データが明らかに線形分離可能であれば，最小二乗法などの回帰分析手法によって，境界線をある程度正確に引くことが可能であると考えられる．しかし，必ずしも直線的な境界線を引くことが可能であるわけではなく，非線形の境界線により学習データを分類する必要性も生じる．

そこで，非線形の境界線を引く方法として，サポートベクターマシンやニューラルネットワークの適用が考えられる．実際にニューラルネットワークの一つである RBFN を用いて非線形境界線を引く試みを行った研究がある [19]．また，運転中の視線情報から各特徴をガウスカーネルにより分類し，学習したサポートベクターマシンによって運転行動の予測を行う研究もある [20]．

次に回帰問題においては，ある入力（画像，時系列データ，センサー値など）に対して，連続値の出力（気温，株価，距離などの連続的なカテゴリ）を予測することを目標として学習を行う．良い回帰モデルを構成する要素としては近似精度や非線形性に強い，中間層のニューロン数を少なく設定できるなど多様な要素がある．この中でも現実の応答は非線形性を持つものが多く，非線形関数を近似できるということは大きなメリットになると考えられる．

実際に動径基底関数としてガウス関数を採用したRBFNにより，コミュニティサイクル利用動態の短期予測を行った研究事例が存在する [21]．また，RBFNを用いることにより，シグモイド関数型のニューラルネットよりも中間層の数を少なく設定可能であり，構造の簡素化により学習効率の面でも優れているという報告もある [22]．しかし，ガウス関数を採用したRBFNの弱点として，複雑な応答の近似には適するが単純な応答に対しては不適切であるという指摘もある [23]．応答が強い線形性を持つ場合は特にそれが顕著になることを考慮してニューラルネットワークを構築する必要がある．

教師なし学習

教師なし学習では過去のデータを何かしらの観点に基づいて似ているデータ同士を分類する．教師がない (正解ラベルがない) 状態での分類では，教師なし学習ではこのグルーピングそのものが学習となり，主なタスクとなる．教師なし学習の特徴は，分類を行うことはできるがそれぞれのグループが持つ意味を明らかにすることはできない点である．そのため，機械が分類した結果に対して人が解釈をつける必要がある．人の目では判断できない大量のデータやグループに対しても機械は分類を行うことができ，潜在的なクラスターの発見や異常なデータの発見に役立てることができる．

教師なし学習の具体的な手法として，クラスター分析や主成分分析，自己組織化マップなどが挙げられる．この中でもクラスター分析の場合，観測データからその裏にある真の分布を推定することで実現されることが多い．例えば，混合ガウス分布の密度関数

$$p\left(\mathbf{x}|\left\{a_k,\mathbf{m}_k,\mathbf{\Sigma}_k\right\}_{k=1}^{K}\right)=\sum_{k=1}^{K}a_k\frac{1}{\sqrt{(2\pi)^d\left|\mathbf{\Sigma}\right|}}exp\left(-\frac{1}{2}\left(\mathbf{x}-\mathbf{m}_k\right)^{\mathrm{T}}{\mathbf{\Sigma}_k}^{-1}\left(\mathbf{x}-\mathbf{m}_k\right)\right)\quad(3.1)$$

をデータにあてはめることでソフトクラスタリングが実現可能な混合ガウスモデル (Gaussian Mixture Model: GMM) がある．ここで，$\{a_k\}_{k=1}^{K}$ は混合を表した変数で $\sum_{k=1}^{K}a_k=1$ を満たす．$\{\mathbf{m}_k\}_{k=1}^{K}$ は各クラスタの平均を表し，$\{\mathbf{\Sigma}_k\}_{k=1}^{K}$ は各クラスタの分散共分散行列を表す．データ $(x_i)_{i=1}^{n}$ へのあてはめは基本的に負の対数尤度の最小化

$$\underset{(a_k,\mathbf{m}_k,\mathbf{\Sigma}_k)_{k=1}^{K}}{min}\sum_{i=1}^{n}-\log\left(p\left(x_i|\left\{a_k,\mathbf{m}_k,\mathbf{\Sigma}_k\right\}_{k=1}^{K}\right)\right)\quad(3.2)$$

で実現可能となる．実際にオープンソースのデータセットであるIrisデータセットを用いてソフトクラスタリングを行った．Irisデータセットはアヤメの形状データを花びら，がく片のそれぞれの長さと幅 (cm) という4つの特徴量と，setosa，versicolor，virginicaという3つのアヤメの種類のラベルで構成されるデータセットである．Irisデータセットの正解

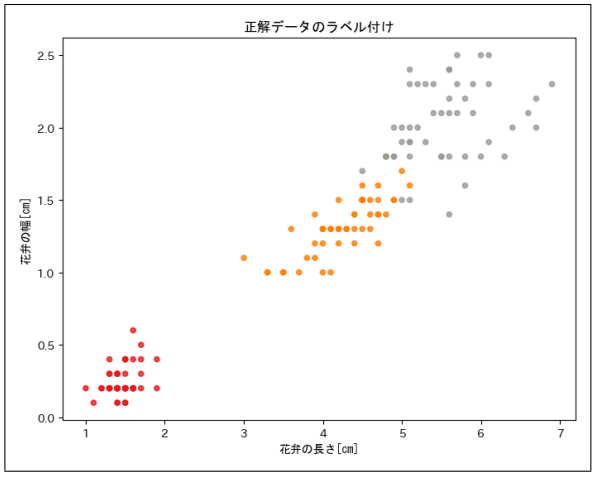

図 3.1: 正解データの分布

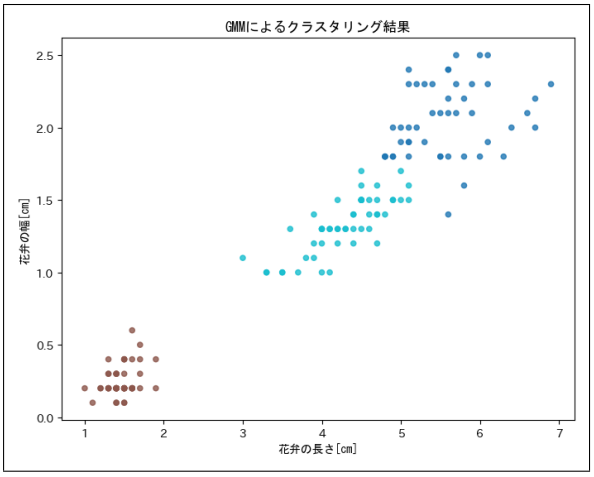

図 3.2: GMM によるクラスタリング結果

データの分布を図 3.1 に，GMM を用いてソフトクラスタリングを行った結果を図 3.2 に示す．図 3.1 を見ると，アヤメの種類ごとに平均が異なる多変量ガウス分布に従って，各データが生成されていると解釈できる．図 3.2 ではほぼ正解データ通りにクラスタリングできていることがわかる．

教師なし学習の手法は，教師あり学習のように学習手法自体に RBFN を適用できるわけではないが，RBFN の基底関数としてガウス関数を選択した場合にガウス関数の中心位置を決定する一つの手法として用いられる [24].

半教師あり学習

半教師あり学習は，教師あり学習と教師なし学習の中間に位置する問題として広く研究されている分野である．この学習手法は，データセットの一部に教師ラベル（正解ラベル）が与えられ，残りのデータには教師ラベルが得られていないという状況を前提としている．このような状況で，ラベル付きデータとラベルなしデータの両方を活用して判別機や予測モデルを構成することを目的とする．

半教師あり学習の特徴は，ラベルなしデータを有効活用することでデータ全体の構造や分布を理解し，それを学習プロセスに活かす点にある．ラベル付きデータのみを用いた場合，データのサンプル数が少ない場合や分布が偏っている場合にモデルの性能が限定されることがある．しかし，ラベルなしデータを活用することで，データの全体像を推定しやすくなり，結果としてモデルの性能を向上させることが可能となる．これは特に，ラベル付け作業が時間的・費用的に困難な場合に有効であり，現実の多くの応用分野で利用されている．

例えば，画像認識や自然言語処理の分野では，大量のデータが容易に得られる一方で，それに対して正確なラベルを付与することは非常に手間がかかり，無駄が生じてしまう．このような場合，半教師あり学習は少数のラベル付きデータから効率的に情報を抽出し，大量のラベルなしデータを活用してモデルを補強する手法として注目されている．

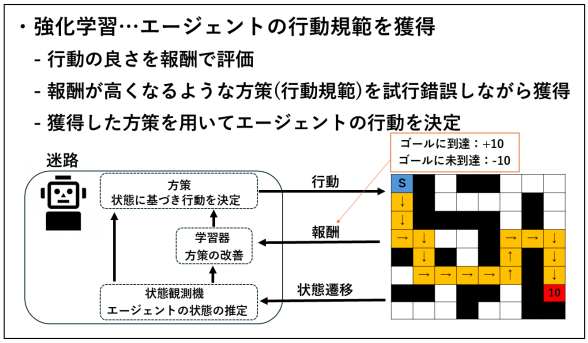

図 3.3: 強化学習の仕組み

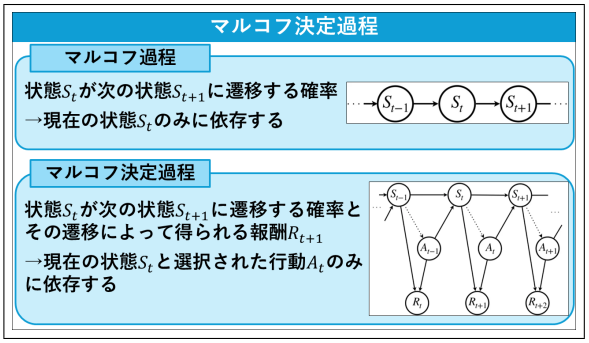

図 3.4: マルコフ過程とマルコフ決定過程

§ 3.2　マルコフ決定過程と強化学習

強化学習とはある環境内におけるエージェントが現在の状態を観測し，得られる報酬を最大化するために，どのような行動を取るべきかを決定する機械学習の一分野である．図 3.3 に強化学習の仕組みを示す．実用例としてはゲームを解く AI やロボットの動作学習などに用いられている．その中でも AlphaGo は強化学習を要素技術として用いており，その実力は数多くのプロに勝利するほどの実力を誇っている [6].

強化学習が教師あり学習や教師なし学習と異なる点として，ラベル付きの入力と出力の組を提示する必要がなく，最適でない行動を明示的に修正する必要もない点が挙げられる．その応用範囲の広さから，強化学習に深層学習を組み込んだ深層強化学習による応用研究が盛んに行われている [12] [13].

マルコフ決定過程

一般的に強化学習に用いられる環境として定式化されているのがマルコフ決定過程である．まず，マルコフ過程は未来の状態は過去には依存せず，現在の状態と遷移確率に依存するというアルゴリズムであり，マルコフ決定過程はこれに行動の概念を加えたものである．図 3.4 にマルコフ決定過程の概略図を示す.

マルコフ決定過程は次の 4 つの要素で構成される．ゲームの局面などを表す状態 $s_t \in S$ とその局面に対して行う行動 $a \in \mathbb{A}$，状態 s_t において行動 a を取ったときに次の状態 s_{t+1} になる遷移確率 $P(s_{t+1}|s_t, a)$，状態 s_t において行動 a を取って状態 s_{t+1} に遷移した際の報酬 $R(s_{t+1}, a)$ である.

Bellman 方程式

強化学習アルゴリズムの理論的基盤として重要な役割を果たす原理として Bellman の最適性原理がある．この原理はある一連の決定事項について，決定の全系列にわたる最適化を行うためには，その初期の状態と初期の決定がいかなるものであっても，以後のすべての決定は初期の決定によってもたらされる状態に対して最適の決定でなければならないという原理である．この Bellman の最適性原理を定式化したものが Bellman 方程式である.

ここで，ある方策 (各状態で次に起こす行動を決める関数)π に従って行動するときの報酬の総和の期待値 $U^{\pi}(s_t)$ を

$$U^{\pi}(s_t) = E[G|s_0 = s_t] = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1}, a) | s_0 = s_t\right] \tag{3.3}$$

と定義し，状態価値関数と呼ぶ．これは，状態 s_t から出発して未来に得られる報酬の総和であり，$0 < \gamma < 1$ は時間割引を表す係数である．また，G は割引収益を表し，報酬に割引率を乗じた将来の割引報酬の和として定義される．

$$G = \sum_{t=0}^{\infty} \gamma^t R(s_{t+1}, a) \tag{3.4}$$

強化学習が目標とするのは報酬 $U^{\pi}(s_t)$ を最大にする方策 π を求めることにある．遷移確率による状態の枝分かれを考慮すると状態価値関数 $U^{\pi}(s_t)$ は次のように表される．

$$\begin{aligned} U^{\pi}(s_t) &= E\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1}, a) | s_0 = s_t\right] \\ &= E[R(s_1, a) | s_0 = s_t] + \gamma E\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+2}, a) | s_0 = s_t\right] \\ &= E[R(s_1, a) | s_0 = s_t] + \gamma E[U^{\pi}(s') | s_{t+1} = s'] \\ &= \sum_{a \in A} \pi(a|s_t) \sum_{s' \in S} P(s'|s_t, a)[R(s_t, a) + \gamma U^{\pi}(s')] \end{aligned} \tag{3.5}$$

このような漸化式を状態価値関数 $U^{\pi}(s_t)$ に対する Bellman 方程式と呼ぶ．また，Bellman 方程式は状態価値だけでなく行動価値でも定義することができ，最適性を定義する上で有用となる．状態 s_t，行動 a，方策 π が与えられたとき，π の下で状態-行動ペアの (s_t, a) の行動価値関数 $Q^{\pi}(s_t, a)$ は次のように定義される．

$$Q^{\pi}(s_t, a) = E[G|s_t, a, \pi] = R(s, a) + \gamma \sum_{s' \in S} P(s'|s_t, a) \sum_{a' \in A} \pi(a'|s') Q^{\pi}(s', a') \tag{3.6}$$

マルコフ決定過程の理論では，π^* が最適な方策であれば Q^{π^*} は最適な行動価値関数となり，Q^{π^*} を知ることができれば最適な行動方法がわかる．Q^{π^*} を計算するための手法は 2 つあり，価値反復法と方策反復法である．どちらのアルゴリズムも，Q^{π^*} に収束する一連の関数 $Q_k(k = 0, 1, 2, ...)$ を計算する．これらの関数を計算するには，状態空間全体に対する期待行動価値を計算する必要があるが，これは最小のマルコフ決定過程を除いては非現実的である．

そこで強化学習法では，大きな状態行動空間上の行動価値関数を表現する必要性に対処するために，サンプルの平均化や関数近似の手法を使用して期待値を近似する．

時間差分学習

関数近似法を適用した強化学習の手法の一つに時間差分 (Temporal Difference: TD) 学習がある．TD 学習は，モンテカルロ法と動的計画法の特徴を組み合わせた価値関数の推定手

法である．モンテカルロ法は，環境から多数のサンプルを収集してエピソード全体の報酬を基に価値関数を推定する手法である．一方で動的計画法は，環境の遷移モデルが既知である場合に現在の推定値に基づいて逐次的に価値関数を更新する方法である．

TD 学習では，状態価値関数 $U(s_t)$ がある方策 π に従った際の期待報酬の総和を正確に近似するように調整される．具体的には，方策 π に従う場合の真の状態価値関数 $U^{\pi}(s_t)$ に近づけることが目標となる．まず，現在の推定値と新たに得られた情報に基づく推定値との差 (TD 誤差) を計算する．

$$\delta_t = R(s_{t+1}) + \gamma U(s_{t+1}) - U(s_t) \tag{3.7}$$

次に，求められた TD 誤差を用いて状態価値関数を逐次更新する．

$$U(s_t) \leftarrow U(s_t) + \Delta\delta_t \tag{3.8}$$

この更新を繰り返すことで状態価値 $U(s_t)$ は真の状態価値関数 $U^{\pi}(s_t)$ に近づいていく．TD 学習は，モンテカルロ法のように環境から得られる経験を活用しつつ，動的計画法のように現在の推定値を使って次の推定を改善する．このため，環境モデルが未知でも逐次学習が可能であり，現実の問題に適用しやすい点が特徴である．

Q 学習

TD 学習では状態価値関数を使用していたが，Q 学習では行動価値関数 $Q(s_t, a)$ を使用する．Q 学習では実行するルールに対し，そのルールの有効性を示す行動価値の Q 値という値を持たせ，エージェントが行動するたびにその値を更新する．ここでいうルールとはある状態とその状態下においてエージェントが可能な行動を対にしたものである．例えばエージェントの現在の状態を s_t とし，この状態で可能な行動が A, B, C の 3 通りであるとする．このとき，エージェントは 3 つの Q 値，$Q(s_t, A)$，$Q(s_t, B)$，$Q(s_t, C)$ をもとに行う行動を決定する．

行動を決定した場合，次にその状態と行動の行動価値関数を更新する．状態 s_t のエージェントが行動 a を選び，報酬 $R(s_{t+1}, a)$ を得て，次の状態 s_{t+1} に遷移したとする．行動価値関数の更新には以下の式が用いられる．

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \Delta\left[R(s_{t+1}, a) + \gamma \underset{a}{max} Q(s_{t+1}, a)\right] \tag{3.9}$$

上記の更新式は現在の状態から次の状態に移った際，その Q 値を（報酬＋割引率×次の状態）によって最も高い Q 値に近づけることを意味している．このことにより，ある状態で高い報酬を得た場合はその状態に到達することが可能な状態にもその報酬が更新ごとに伝播することになる．これにより，最適な状態遷移の学習が行われる．

§ 3.3　逆強化学習の仕組み

逆強化学習は，強化学習の根本的な課題である報酬関数の設計を回避するためのアプローチとして注目されている．強化学習においてエージェントが望ましい行動を学習するためには，目標に基づいた適切な報酬関数を設計する必要がある．しかし，現実世界の問題は複雑で報酬関数を設計する際に考慮すべき要因は膨大であり，時には定量化が難しい場合もある．そのため，報酬関数の設計が不適切であるとエージェントは意図しない振る舞いを学習する可能性が高まり，全体的なパフォーマンスを著しく損なう結果となることがある．

例えば，ロボット工学や自動運転分野では，エージェントの行動を正確に誘導するための報酬関数を設計することは非常に困難とされている．その理由として，専門家やエンジニアが試行錯誤の過程で設計した報酬関数が必ずしも現実的な条件下で最適な行動を生むとは限らず，設計の不備が安全性や効率性の欠如に繋がることもあるからである．このような課題を解決するために，逆強化学習ではエキスパートのデモンストレーションデータを利用して報酬関数そのものを推定し，エージェントがその報酬関数に基づいて適切な行動を学習できるようにすることを目標とする．

逆強化学習の有用性は，特定のドメインに依存せず幅広い分野で適用可能であることにある．例えば，医療分野においては熟練医の治療手順や手術プロセスを基に報酬関数の推定を行った研究事例もある [14]．また，製造業では熟練工の作業を模倣するロボットを設計する際に，逆強化学習を活用して人間の効率的な作業方法を学習させることができる [15]．

さらに自動運転車の開発においては，熟練したドライバーの運転データを解析し，理想的な運転行動を推定することが可能である．このようなアプローチは，特に安全性が重要なシステムにおいて有効であり，人間の直感的な意思決定プロセスをシステムに反映させるための手段となる．

このように逆強化学習は強化学習の適用範囲を広げ，実世界の問題における報酬設計の課題を克服するための強力なアプローチとして注目されている．その発展と応用は今後さらに多様な分野でのイノベーションを生むことが期待される．

線形計画法

線形計画法を用いた逆強化学習は主に Ng ら [16] による手法が用いられている．ここで逆強化学習では各状態 s_t における最適な行動 a^* が既知であるとする．このとき，各状態における報酬 R を求める．これは以下のような最適化を解く問題として解釈できる．

$$maximize \quad \sum_{s_t \in S} \min_{a \in A \setminus a^*} \left\{ \left(P_{a^*} - P_a\right) \left(I - \gamma P_{a^*}\right)^{-1} R\left(s_t, a\right) \right\} - \lambda \left\| R\left(s_t, a\right) \right\|_1 \tag{3.10}$$

$$subject\ to \quad \left(P_{a^*} - P_a\right) \left(I - \gamma P_{a^*}\right)^{-1} R\left(s_t, a\right) \geq 0, \quad \forall a \in A \setminus a^* \tag{3.11}$$

$$\left\| R\left(s_t, a\right) \right\| \leq R_{\max} \tag{3.12}$$

上式における $\left(P_{a^*} - P_a\right) \left(I - \gamma P_{a^*}\right)^{-1} R\left(s_t, a\right)$ の部分は最適な行動 a^* とそれ以外の行動の期待報酬の差を求めており，この目的関数は a^* との期待報酬の差が最も小さい行動 (2 番目に良い行動) との期待報酬の差を最も大きくするような報酬を求めていることになる．つまり，常に 2 番目の行動よりも良い行動を選べる方策を探索できるようになっている．上

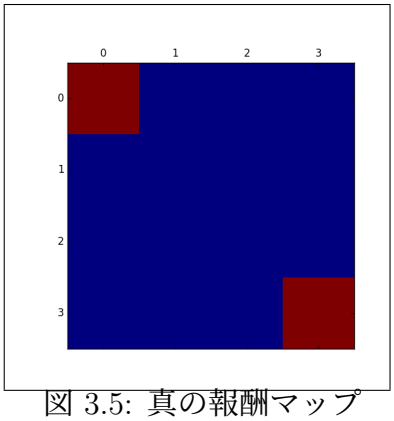

図 3.5: 真の報酬マップ

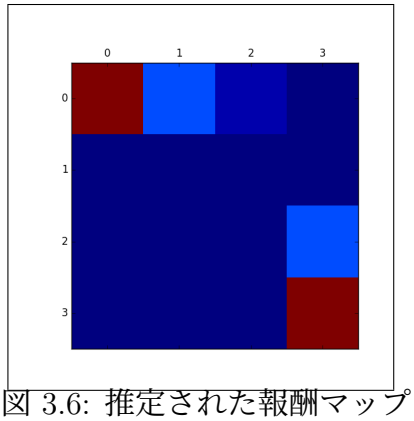

図 3.6: 推定された報酬マップ

記の式はこの状態では線形計画問題として扱うことが出来ないため，新たな変数として u_t, v_t という変数を用意し，以下のように変形を行う．

$$max \qquad \sum_{s_t \in S} \{u_t - \lambda v_t\} \tag{3.13}$$

$$subject\ to \qquad (P_{a^*} - P_a)\left(I - \gamma P_{a^*}\right)^{-1} R\left(s_t, a\right) \geq u_t \tag{3.14}$$

$$(P_{a^*} - P_a)\left(I - \gamma P_{a^*}\right)^{-1} R\left(s_t, a\right) \geq 0 \tag{3.15}$$

$$-v_t \leq R\left(s_t, a\right) \leq v_t \qquad \left|R\left(s_t, a\right)\right| \leq R_{max} \tag{3.16}$$

実際に OpenAI が提供する Grid World と呼ばれる，エージェントがマス目状の 2 次元グリッド上を移動しながら目的を達成する方法を学ぶ環境を用いて報酬関数の推定を行った．今回は報酬が高いグリッドが赤で表示され，報酬が低いグリッドが青で表示される．真の報酬マップを図 3.5 に，線形計画法によって推定された報酬マップを図 3.6 に示す．

最大エントロピー逆強化学習

通常の逆強化学習では「観測された行動が報酬を最大化する」という仮定を用いるが，これにエントロピー最大化の原理を組み合わせたのが最大エントロピー逆強化学習である [18]．

エントロピー最大化の原理とは，確率変数 X について X が条件 I を満たすことだけがわかっており，それ以外に X に関して何一つ知らなかったとき，X が従う分布は条件 I の下で X の不確かさが最大になるような分布を選ぶのが適切であるという原理である．この原理を組み合わせることにより，多様で確率的な人間の行動に関する方策を学習することが可能となり，逆強化学習の結果が現実の行動と近いものになる．

最大エントロピー逆強化学習では報酬関数 $R\left(s_t, a\right)$ をパラメータ θ と，one-hot 表現であらわされるエキスパートの行動軌跡の特徴量 $\mathbf{f}_\zeta$ を用いて以下のような線形関数で近似することを考える．このときエキスパートは経路 $\zeta = \left[\left(s_0, a_0\right), \left(s_1, a_1\right), \left(s_2, a_2\right), ..., \left(s_T, a_T\right)\right]$ を選択していると仮定する．

$$R\left(\zeta|\theta\right) = \theta \mathbf{f}_\zeta \tag{3.17}$$

ここで，パラメータ θ が $\hat{\theta}$ と推定できたとすると，状態 s の特徴量 $\mathbf{f}_s$ から状態 s の報酬 $R\left(s\right)$ が定まる．

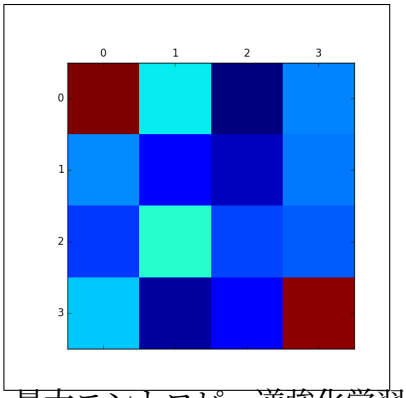

図 3.7: 最大エントロピー逆強化学習によって推定された報酬マップ

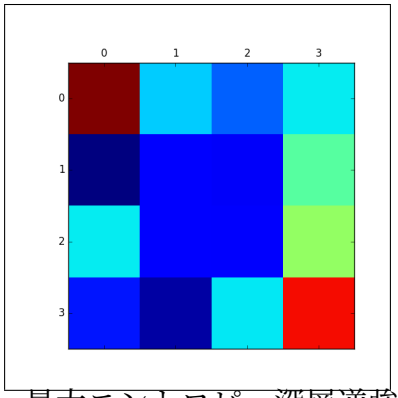

図 3.8: 最大エントロピー深層逆強化学習によって推定された報酬マップ

$$R(s) = \hat{\theta}\mathbf{f}_s \tag{3.18}$$

この $R(s)$ が最大エントロピー逆強化学習において求めたい報酬関数となる．また，行動軌跡の発生確率 $P(\zeta|\theta)$ についてはエキスパートの行動履歴を制約条件にして，エキスパートの行動履歴において想定していない事象に対しては確率 $P(\zeta|\theta)$ を等しく割り当てることを考える．$P(\zeta|\theta)$ は以下のように与えられる．

$$P(\zeta|\theta) = \frac{exp(\theta\mathbf{f}_\zeta)}{\sum_\zeta exp(\theta\mathbf{f}_\zeta)} = \frac{exp(R(\zeta|\theta))}{\sum_\zeta exp(R(\zeta|\theta))} \tag{3.19}$$

これは報酬 $R(\zeta|\theta)$ が大きくなるほど，行動軌跡の発生確率 $P(\zeta|\theta)$ が大きくなることを表している．最適な N 個の行動列 $D = [\zeta_0, \zeta_1, \zeta_2, ..., \zeta_N]$ が与えられた際に，以下の対数尤度関数 $L(\theta)$ を最大にする θ を求めることで報酬関数の推定を行う．

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} \log P(\zeta_i|\theta) \tag{3.20}$$

次に，勾配法を用いて θ の更新を行うことで尤度最大化を図る．

$$\theta_{k+1} \leftarrow \theta_k + \Delta\frac{\partial L(\theta)}{\partial\theta} \tag{3.21}$$

そして，エキスパートの行動軌跡の特徴量の平均値と θ に基づいて行動した場合の行動軌跡の特徴量の平均値が一致するように，パラメータ θ を更新することにより最適化を行う．

$$\frac{\partial L(\theta)}{\partial\theta} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{f}_{\zeta_i} - \sum_{i=1}^{N} P(\zeta_i|\theta)\,\mathbf{f}_{\zeta_i} \tag{3.22}$$

以上の手順により最大エントロピー逆強化学習ではエキスパートの行動軌跡に基づいて，報酬関数の推定が確率的かつ多様性を考慮した形で行われる．このアプローチは現実世界の不確実性や複雑な意思決定を反映するモデル構築を可能にし，多くの応用分野で現実的かつ柔軟な解を提供する点で有用である．さらに発展として深層学習と組み合わせた手法も提案されている [17].

第4章

提案手法

§ 4.1　計算量を考慮した複製方法の提案

2.3 節で述べた累積二乗誤差関数 $E(\mathbf{w}, \phi)$ の最小化は各入力ベクトル $\mathbf{x}_i$ ごとに二乗誤差関数 $E(\mathbf{x}_i, \mathbf{w}, \phi)$ を最小化することに等価である．そこで，各入力ベクトル $\mathbf{x}_i$ に依存した平均ベクトル $\mathbf{m}_{j[i]}$ を考える．そして，学習収束の時点で得られている j 番目の動径基底関数に着目すると，入力ベクトル $\mathbf{x}_i$ の条件付き確率密度関数は

$$p_{\beta'}\left(\mathbf{x}_i|\mathbf{m}_{j[i]}, \phi'_j, \phi''_j\right) = Z^{-1}_{\beta'}\left(\mathbf{m}_j, \phi'_j, \phi''_j\right) exp\left\{-\beta' E\left(\mathbf{x}_i, \mathbf{m}_{j[i]}, \phi'_j, \phi''_j\right)\right\} \tag{4.1}$$

と導出できる．ここで，パラメータ ϕ'_j は着目した第 j 番目の動径基底関数のシナプス結合荷重 w_j と共分散行列 Σ_j の集合であり，パラメータ ϕ''_j は着目した第 j 番目の動径基底関数以外のシナプス結合荷重，共分散行列並びに平均ベクトルの集合である．以後は記法の簡便のため，パラメータ ϕ'_j とパラメータ ϕ''_j は省略する．また，分配関数は

$$Z_{\beta'}(\mathbf{m}_j) = \sum_{i=1}^{N} exp\left\{-\beta' E\left(\mathbf{x}_i, \mathbf{m}_{j[i]}\right)\right\} \tag{4.2}$$

で定義される．条件付き確率密度関数 $p_{\beta'}\left(\mathbf{x}_i|\mathbf{m}_{j[i]}\right)$ は，確率の正規化と二乗誤差関数 $E\left(\mathbf{x}_i, \mathbf{m}_{j[i]}\right)$ の条件付き期待値

$$\langle E(\mathbf{m}_j)\rangle_{\beta'} = \sum_{i=1}^{N} p_{\beta'}\left(\mathbf{x}_i|\mathbf{m}_{j[i]}\right) E\left(\mathbf{x}_i, \mathbf{m}_{j[i]}\right) \tag{4.3}$$

が一定となるという二つの制約のもとで，エントロピー

$$\langle E(\mathbf{m}_j)\rangle_{\beta'} = \sum_{i=1}^{N} p_{\beta'}\left(\mathbf{x}_i|\mathbf{m}_{j[i]}\right) E\left(\mathbf{x}_i, \mathbf{m}_{j[i]}\right) \tag{4.4}$$

を最大にする確率密度関数として導出できる．ここで記号

$\langle\cdots\rangle_{\beta'}$ は，$p_{\beta'}\left(\mathbf{x}_i|\mathbf{m}_{j[i]}\right)$ を掛けて $\mathbf{x}_i$ に関する和をとる演算を表すものとする．このとき，自由エネルギーを

$$F_{\beta'}(\mathbf{m}_j) = -\frac{1}{\beta'} \log Z_{\beta'}(\mathbf{m}_j) \tag{4.5}$$

で定義すれば，

$$S_{\beta'}\left(\mathbf{m}_j\right) = -F_{\beta'}\left(\mathbf{m}_j\right) + \beta' \langle E\left(\mathbf{m}_j\right) \rangle_{\beta'} \tag{4.6}$$

と表すことができる．この式はエントロピー $S_{\beta'}\left(\mathbf{m}_j\right)$ を最大化する条件付き確率密度関数 $p_{\beta'}\left(\mathbf{x}_i | \mathbf{m}_{j[i]}\right)$ は，自由エネルギー $F_{\beta'}\left(\mathbf{m}_j\right)$ を最小化するものであることを示している．

このような自由エネルギーは，データのクラスタリングのための手法であるメルティング [8] においても同様に定義されている．メルティングとは，$m_{j[i]} = \mathbf{x}_i$ かつ β' が ∞ である初期状態から，徐々に β' を 0 へ近づけていきながら，パラメータ $m_{j[i]}$ を自由エネルギー $F_{\beta'}\left(\mathbf{m}_j\right)$ の最急降下方向に更新していくものである．その結果，パラメータ $m_{j[i]}$ は徐々に同じ値をとりはじめ，最終的に一つの値 $m_{j[i]} = m_j\,(\forall i)$ に収束する．

そこで，RC-RBFN ではパラメータ m_j の更新則を式 (2.11) の代わりに

$$\begin{aligned}\Delta \mathbf{m}_{\beta'} &= -\epsilon \sum_{i=1}^{N} \frac{\partial F_{\beta'}\left(\mathbf{m}_j\right)}{\partial \mathbf{m}_{j[i]}} = -\epsilon \sum_{i=1}^{N} \frac{\partial F_{\beta'}\left(\mathbf{m}_j\right)}{\partial Z_{\beta'}\left(\mathbf{m}_j\right)} \frac{\partial Z_{\beta'}\left(\mathbf{m}_j\right)}{\partial \mathbf{m}_{j[i]}} \\ &= \sum_{i=1}^{N} p_{\beta'}\left(\mathbf{x}_i | \mathbf{m}_{j[i]}\right) \Delta \mathbf{m}_{j[i]} = \langle \Delta \mathbf{m}_j \rangle_{\beta'}\end{aligned} \tag{4.7}$$

で与えることとする．ここで，

$$\Delta m_{j[i]} = -\epsilon \frac{\partial E\left(\mathbf{x}_i, \mathbf{m}_{j[i]}\right)}{\partial \mathbf{m}_{j[i]}} \tag{4.8}$$

である．特に $\beta' = 0$ であり，初期の状態が $\mathbf{m}_{j[i]} = \mathbf{m}_j\,(\forall i)$ である場合は

$$\Delta_0 \mathbf{m}_j = \Delta \mathbf{m}_j \tag{4.9}$$

であることが示される．この場合は，RC-RBFN のパラメータ $\mathbf{m}_j$ の更新則が従来の RBFN のパラメータ $\mathbf{m}_j$ の更新則そのものとなっていることがわかる．このとき，$\beta' = 0$ で固定したままパラメータを $\mathbf{\Sigma}_j \to 0$ にすると，$\Delta_0 \mathbf{m}_j = 0$ とするパラメータ $\mathbf{m}_{j[i]}$ は

$$\sum_{i=1}^{N} \xi\left(\mathbf{x}_i, \mathbf{m}_{j[i]}\right)\left(\mathbf{x}_i - \mathbf{m}_{j[i]}\right)\left\{\eta\left(\mathbf{x}_i\right) - s\left(\mathbf{x}_i, \mathbf{m}_{j[i]}\right)\right\} = 0 \tag{4.10}$$

を満たし，$\mathbf{m}_{j[i]} = \mathbf{x}_i\,(\forall i)$ であることがわかる．つまり，教師入力信号がパラメータ $\mathbf{m}_j$ の収束点として検出されることとなる．逆に，パラメータ $\mathbf{\Sigma}_j$ を固定したまま $\beta' \to \infty$ にすると，$\Delta_\infty \mathbf{m}_j = 0$ とするパラメータ $\mathbf{m}_{j[i]}$ は

$$\sum_{i=1}^{N} exp\left\{-\beta' E\left(\mathbf{x}_i, \mathbf{m}_{j[i]}\right)\right\} \xi\left(\mathbf{x}_i, \mathbf{m}_{j[i]}\right)\left(\mathbf{x}_i - \mathbf{m}_{j[i]}\right)\left\{\eta\left(\mathbf{x}_i\right) - s\left(\mathbf{x}_i, \mathbf{m}_{j[i]}\right)\right\} = 0 \tag{4.11}$$

を満たし，$\mathbf{x}_i\,(\forall i)$ を含む任意の値となることがわかる．これらの結果から，提案する RC-RBFN のパラメータ $\mathbf{m}_j$ の更新則 $\Delta_{\beta'} \mathbf{m}_j$ では，$\Delta_0 \mathbf{m}_j$ で従来の RBFN のパラメータ $\mathbf{m}_j$ の更新則を実現する．さらに，$\Delta_\infty \mathbf{m}_j$ とすることで，すべての入力ベクトル $\mathbf{x}_i$ の任意の値を安定な収束点とすることができる．

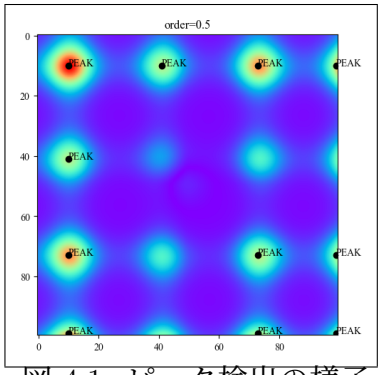

図 4.1: ピーク検出の様子

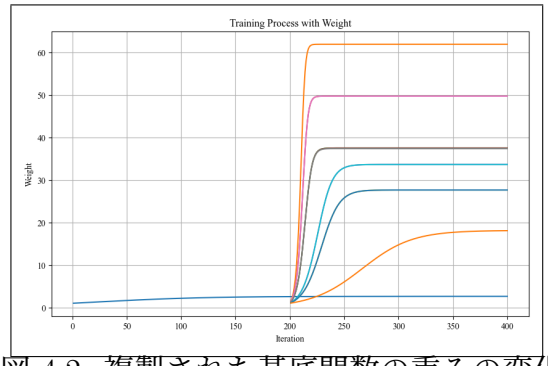

図 4.2: 複製された基底関数の重みの変化

ここで問題となるのが式 (4.11) において β' を徐々に大きくしながら計算する部分である．1 変数や 2 変数の場合は計算量はそれほど多くならず，計算時間も少なくて済む．しかし，変数の数が多くなるにつれて各 β' について式 (4.11) を計算するのは計算負荷が大きくなり，学習に多大な時間がかかる．

新たな複製方法の提案

そこで，本研究では教師信号 $\eta(\mathbf{x})$ と重みと動径基底関数の線形和 (近似結果)$s(\mathbf{x})$ の差 $o(\mathbf{x})$ を計算し，$o(\mathbf{x})$ の極大値 (ピーク) を検出することにより複製すべき点を導出する．$o(\mathbf{x})$ は以下のように計算する．

$$o(\mathbf{x}) = |\eta(\mathbf{x}) - s(\mathbf{x})| \tag{4.12}$$

この手法により，疑似的に式 (4.11) を満足するようなパラメータ $\mathbf{m}$ を求め，重みが 1，中心位置 $\mathbf{m}$，共分散行列 $\mathbf{\Sigma}$ をもつようなガウス関数を新たな動径基底関数として複製する．ここで，$\mathbf{\Sigma}$ は N 次元の単位行列とする．また，式 (4.11) の方法では一つ一つ複製を行っていくため，細かい部分の精度は良くなるが学習終了までの時間がその分増えてしまう．しかし，提案手法では $o(\mathbf{x})$ の形状によっては一度に複数の基底関数を複製することが可能であり，効率よく近似を行うことができると考えられる．

ピーク検出には SciPy ライブラリの ndimage モジュールを用いた．ndimage モジュールは，多次元画像処理をサポートするためのツールが提供されており，画像処理に関連する多くの操作を簡単に実行するための機能が含まれている．その中でも今回は，maximum_filter を用いた．maximum_filter は N 次元の画像 (N 次元の NumPy 配列) におけるピーク検出を行うためのフィルターであり，実際にピーク検出を行っている様子を図 4.1 に示す．図 4.1 内の order とはフィルタリングによって得られた最大のピーク時の値を order 倍し，その値以下のピークを排除するための数値であり，動径基底関数が複製されすぎないように抑制する効果を持つ．order の値は最初 0.5 として設定し，複製を行うごとに 0.1 増加させ，重みの学習へ移る．この作業を繰り返し，order=1.0 に到達した時点で複製終了とし，その時点での近似結果を評価する．図 4.2 は図 4.1 で検出された位置に複製を行い，各基底関数の重みを更新している様子である．図 2.6 と比較しても同様に複製が機能していることがわかる．

§ 4.2　機械学習への組み込みと高速化手法

4.1 節で述べた新しい複製方法を適用した RC-RBFN の組み込む先として，強化学習における価値関数の近似と逆強化学習における報酬関数の近似を提案する．また，プログラム開発に用いた Python の豊富なライブラリをいくつか紹介し，実際に有効であることを確かめる．

強化学習への適用

価値関数の近似に RBFN を用いている先行研究として，RBFN の一つである正規化ガウス関数ネットワーク (Normalized Gaussian network: NGnet) を用いた研究がある [25] [26]．[25] では学習の安定化のために中心位置とガウス関数の広がりは固定で関数近似を行っており，初期状態のガウス関数の中心位置によっては近似精度が落ちる場合がある．[26] では NGnet を用いて近似を行う点では [25] と同じであるが，基底関数分割アルゴリズムと呼ばれる価値観数の予測誤差である TD 誤差の観測空間における分布をもとに分割先を決定するという点で異なる．しかし，分割する方向を求めるために基底関数の分散共分散行列の固有値，固有ベクトルを求める計算が必要になるという課題がある．

そこで，本研究の手法では初期状態の基底関数自体が誤差を小さくする方向に移動することで，複製する前にある程度誤差を減らすことが可能となると考えられる．

逆強化学習への適用

報酬関数の近似手法として 2.3 節で述べたような線形計画法やエントロピー最大化の原理を用いた最大エントロピー逆強化学習，報酬関数を推定する過程で最適な方策を獲得する見習い学習の手法 [27] が存在する．エキスパートの報酬関数の線形近似が困難な問題に対しては，報酬関数に確率分布を仮定し，ベイズの定理を用いた報酬関数の推定法が提案されている [28]．この中でも本研究では最大エントロピー逆強化学習に着目した．

ところで，報酬の設計と報酬による最適化については先行研究がある [29]．この研究では報酬駆動型システムとしての遺伝的アルゴリズムについて述べられており，N 人の進化ゲームにおけるプレイヤー数や要素数が動的に変化する環境においても適用できるように，RC-RBFN の複製機能についても述べられている．さらに報酬による最適化を行う際のコスト関数 l を次のように定義している．

$$l = \beta E\left[\log\left(\frac{f(\mathbf{x})}{f(\mathbf{x}')}\right)\right] - S \tag{4.13}$$

ここで，$\mathbf{x}$，$\mathbf{x}'$ はそれぞれ遺伝子 $\mathbf{x}$，$\mathbf{x}'$ を表し，$f(\mathbf{x})$ は適合度関数，β は逆温度である．また，S はエントロピーを表す．このコスト関数 l の最小化はエントロピーの最大化によって実現可能であり，コスト関数 l の最小化を行うことは報酬の最大化とみなすことも可能である．つまり，4.1 節で述べた自由エネルギーの最小化とエントロピーの最大化による複製機能を用いることで，動的な環境においても報酬の最大化を図ることができると考えられる．

また，エントロピーを用いた報酬の最大化については様々な論文で議論が交わされており，[30] では式 (3.4) の割引収益を次のように再定義している．

$$G' = \sum_{t=0}^{\infty} \gamma^t \left(R\left(s_{t+1}, a\right) - \beta^{-1} \log P\left(s_{t+1} | s_t, a\right) \right) \tag{4.14}$$

従来の強化学習は $\beta = \infty$ の場合に対応する．このような割引収益が与えられた際，この割引収益の期待値は

$$\tau \equiv E\left[\sum_{t=0}^{\infty} \gamma^t R\left(s_{t+1}, a\right) \right] + \beta^{-1} H \tag{4.15}$$

となる．ただし，H はエントロピーである．ここで，式 (4.15) の両辺に β をかけることで式 (4.13) と等価とみなせる．第二項の符号については式 (4.15) と式 (4.13) の意味を考えることで等価であると解釈できる．

したがって，自由エネルギーの最小化とエントロピーの最大化による複製機能は，逆機械学習におけるエントロピーを用いた報酬の最大化に寄与できると考える．

Numba

本研究のプログラムの開発に用いた Python は，高い可読性と豊富なライブラリを持つ汎用プログラミング言語であり，学術研究やプロトタイピングに適した選択肢である．しかし，Python はインタプリタ型の言語であり，コードを実行時に逐次的に解釈するためにコンパイル型の言語である C や C++と比較すると，処理速度が遅くなる場合がある．特に，大規模な数値計算や複雑なアルゴリズムを多用する場合にはこの性能差が顕著となる．

この欠点を補うため，近年では Python の実行速度を向上させるための高速化ライブラリが数多く開発されている．例えば，数値計算を効率化するための NumPy や，Python コードを JIT コンパイルする Numba が挙げられる．さらに，Python コードを C 言語に変換してコンパイルする Cython や，並列処理を簡単に実現する multiprocessing ライブラリも広く利用されている．これらの高速化ライブラリの 1 つの Numba を使用することで計算コストを低減し，効率的なプログラム開発を実現した．これにより，Python の持つ可読性や開発の柔軟性を維持しつつ，処理速度の課題を克服が期待できる．

Numba は Python コードの高速化を助けるためのモジュールであり，Python コードを JIT コンパイルすることで CPU や GPU 上でネイティブマシンコードとしての実行が可能になる．Numba がどのように機能しているかというと，次の 3 つの手順に分けることができる．

Step 1: 最初の呼び出しで関数を JIT コンパイルし，型推論を行う

関数が最初に呼び出される際，Numba はその関数の JIT コンパイルを実行する．まず関数のソースコードを解析し，どのような型の引数が与えられているかを推論する．その後，Python のコードを低レベルな中間表現に変換し，さらにそれをマシンコードへと変換する．このプロセスによって，関数は Python のインタープリターを経由せずに直接 CPU 上で実行できる形式になる．

Step 2: 引数の型を推論した後，コンパイルされたマシンコードをキャッシュする

型推論が完了してマシンコードが生成された後，Numba はこのコンパイル済みコー

表 4.1: Numba 適用前後の数値比較

呼出回数	10^5	10^6	10^7	10^8	10^9
適用前 [s]	3.64×10^{-3}	3.96×10^{-2}	3.83×10^{-1}	3.78	37.4
適用後 [s]	2.40×10^{-6}	7.29×10^{-6}	9.80×10^{-6}	7.80×10^{-6}	3.18×10^{-3}

ドをキャッシュする．キャッシュは関数ごとだけでなく，異なる引数の型ごとにも保存される．つまり，同じ関数が異なる型の引数で呼び出された場合，それぞれの型に対して別々のマシンコードが生成されて管理されることになる．この仕組みにより，同じ型の引数で再度関数を呼び出した際には再コンパイルの必要がなくなり，処理時間を短縮できる．

Step 3: 以降の関数呼び出しでは，キャッシュされたバージョンを使用する
一度コンパイルされた関数が再び呼び出される際には，Numba はキャッシュに保存されたマシンコードを直接実行する．これにより，Python のインタープリターを経由せずに純粋なネイティブコードとして関数を実行することが可能になる．その結果としてインタープリターのオーバーヘッドが排除され，関数の実行速度の向上が期待される．特に，ループや数値計算が多く含まれる関数ではこの最適化の効果が顕著に現れる．

JIT デコレーターには「nopython=True」オプションを指定することもでき，このオプションを設定することで完全なネイティブコードが生成され，さらなる性能向上が期待できる．Numba を使用することで，Python コードのパフォーマンスを C や Fortran と同等の水準に引き上げることができるため，科学計算や機械学習，シミュレーション，データ分析など計算負荷の高いアプリケーションにおいて非常に有用である．従来の手法ではボトルネックとなっていた処理の高速化を容易に実現できる点が，Numba の大きな特徴である．

Numba を使用した場合と未使用の場合の比較結果を表 4.1 示す．この数値比較では，$s = s + 1$ を行う関数を 10^5，10^6，10^7，10^8，10^9 回ずつ呼び出し，Numba 適用前と適用後で処理にかかった時間をそれぞれ計測した．適用前は対数グラフで線形的に処理時間が増加しており，指数的に処理時間が増加していることがわかる．適用後は 100 倍から 1000 倍近く高速になっており，処理高速化の効果がみられる．また，Numba には並列化オプションも実装されており，「parallel=True」オプションを指定することにより CPU での並列分散処理も実現している．

§ 4.3　複製・競合を考慮した学習の流れ

2 章と 4.1 節で述べた競合と複製の機能を用いて任意の非線形関数の近似を行うための学習の流れの説明を行う．

Step 1: 初期状態における基底関数の配置

RBFN を関数近似器として用いる際，初期状態における基底関数の個数の決定は重要な課題の 1 つである．この問題に対して，過去の研究ではさまざまな手法が提案されており，例えば多目的遺伝的アルゴリズムを用いて適切な基底関数の数を決定するアプローチが報告されている [31]．このような手法では，関数の近似精度とモデルの複雑さのバランスを考慮しながら，最適な基底関数の個数を自動的に探索することが可能となる．

本研究では複製機能を活用することによって一度に複数の基底関数を効率的に複製できる点に着目し，初期状態における基底関数の数を 1 つに設定する．この理由として，近似対象の関数（目的関数）が単純な場合に 1 つの基底関数で十分に表現可能なケースがあることも考慮している．初期状態における基底関数の数を最小限に抑えることで計算コストを削減しつつ，必要に応じて基底関数を動的に増加させる柔軟な構造を実現する．

Step 2: 重みの学習と基底関数の複製

RBFN における 3 つのパラメータのうち，中心位置の学習から始める方が効率的であると考えられるかもしれない．なぜなら，適切な中心位置を早期に決定することで，関数近似の精度向上が期待できるからである．しかし，中心位置の学習に関しては重みの学習と同様に適者生存型学習則が適用されるため，初期状態において教師信号と基底関数の分布に大きなズレがある場合，そのまま中心位置の学習を行っても有効な結果を得ることが難しい．特に，初期状態の基底関数が教師信号の分布と重なっていない場合，中心位置の更新を何度繰り返しても誤差の減少が見込めない可能性が高い．

そこで，まず重みの学習を適者生存型学習則に基づいて実施し，基底関数が近似に必要かどうかを決定する．この手法では初期状態に存在する基底関数のうち，十分な貢献ができないもの（誤差削減に寄与しないもの）は削除し，有用な基底関数のみを生存させる．この過程を経ることで中心位置の学習を開始する前に，必要最小限の基底関数が適切に選別される．その結果として学習プロセス全体の効率が向上し，無駄なパラメータ調整を削減することができる．

重みの学習を終えると，次に基底関数の複製を行う．複製方法については 4.1 節で述べた通りである．order=1.0 に到達した時点で Step 3 へ進む．

Step 3: 広がりの学習

4.1 節で提案した基底関数の複製方法では，まず関数近似の誤差が最も大きい箇所を見つけ出し，その位置に基底関数を追加することで誤差を効率的に削減することを目指している．複製された基底関数は教師信号と近い位置に配置されるため，自然と誤差を減少させる役割を果たす．これにより，誤差が大きい領域に基底関数が追加され，より精度の高い近似が可能となる．

このような複製方法では，重みの学習に続いて中心位置の学習を行うのではなく，まず基底関数の広がりを学習することが有効であると考える．その理由は，教師信号に近い位置に基底関数を配置することができても，基底関数の広がりが不適切であれば関数近似が十分に行われない可能性があるためである．ガウス関数による関数近似手法においては基底関数の広がりを適切に設定することが，関数の形を再現するために重要な要素となる．

Step 4: 中心位置の学習

最後に各基底関数の中心位置の学習を行う．各基底関数は複製によって誤差の大きな領域に配置されるため，複製された基底関数の中心位置は既に適切な場所に近い位置に配置されていると考えられる．そのため中心位置の学習は大幅な修正を行うものではなく，あくまで微調整の意味合いが強い．

また，中心位置の更新は既に学習された重みや基底関数の広がりとのバランスを保ちつつ行う必要がある．過度に大きく移動させると，既に適切に配置されている基底関数の役割が崩れてしまう可能性があるため学習率を適切に設定し，小さな調整を繰り返しながら最適な中心位置へと導く．

Step 5: モデルの評価

関数の近似精度を評価するために，二乗平均平方根誤差（Root Mean Squared Error: RMSE）と平均絶対誤差（Mean Absolute Error: MAE）の 2 つの指標を用いる．これらの評価指標に基づき，関数近似の精度を評価する．RMSE と MAE は，それぞれ以下の式で定義される．

$$RMSE = \sqrt{\frac{1}{q}\sum_{p=1}^{q}(y_p - \hat{y}_p)^2} \tag{4.16}$$

$$MAE = \frac{1}{q}\sum_{p=1}^{q}|y_p - \hat{y}_p| \tag{4.17}$$

ここで，q は総データ数，y_p は正解値，$\hat{y}_p$ は予測値を表す．

RMSE は誤差の二乗平均の平方根を取ることで，誤差の大きさを測定する指標である．誤差を二乗することで大きな誤差が強調されるため，大きな誤差を持つデータ点が評価に与える影響が強くなる．これにより，外れ値の影響を受けやすい性質を持つ．一方で，MAE は誤差の絶対値の平均を取る指標であり，すべての誤差を均等に扱うため外れ値の影響を受けにくいという特長がある．また，誤差が正規分布に従う場合については RMSE と MAE の比を評価することでモデルの評価が可能であることを後で示す．

RMSE と MAE の両方を用いることで，近似の精度を包括的に評価する．RMSE はモデルの全体的な性能を確認する上で有用であり，MAE は外れ値による影響を抑えつつ，誤差の分布を確認するのに適している．これら 2 つの指標を基に近似精度が満足できるかどうか判断し，Step 4 へ進むか複製を行うかを決定する．

以上の手順を踏むことで複製・競合を考慮した任意の非線形関数の近似を行うことが可能となる．

誤差が正規分布に従う場合の評価指標

Step 5 において誤差が正規分布に従う場合に RMSE と MAE の比を評価することでモデルの評価が可能であることについて述べる．

まず，良いモデルというのはデータに対してフィットしているため，モデルによって予測される値は実際のデータのパターンなどを反映する．このため，誤差は予測モデルが捉えられないランダムなノイズに起因すると考えられる．このようなランダムノイズの要因が独立しており，各要因が小さな影響を与える場合，中心極限定理よりランダムノイズの総和は正規分布に従うことが期待される．よって良いモデルの誤差は正規分布に従うと考えられる．

ここで，誤差 $e\,(=y_p-\hat{y}_p)$ が平均が 0，標準偏差 σ の正規分布に従うと仮定するとして $RMSE^2$ と MAE^2 の差を考えると，

$$RMSE^2-MAE^2=\frac{1}{q}\sum_{p=1}^{q}e_p^2-\frac{1}{q^2}\left(\sum_{p=1}^{q}e_p\right)^2 \tag{4.18}$$

であり，これは e の分散 $VAR\,(e)$ と等しい．また，e の平均 $MEAN\,(e)$ は MAE と等しいことを利用すると，MAE に対する $RMSE$ の比は

$$\frac{RMSE}{MAE}=\sqrt{1+\frac{VAR\,(e)}{MAE^2}}=\sqrt{1+\frac{VAR\,(e)}{MEAN\,(e)^2}} \tag{4.19}$$

となる．よって $\frac{RMSE}{MAE}$ の値を求めることで，そのモデルが良いモデルかどうか判定することができる．この値は $VAR\,(e)$ と $MEAN\,(e)^2$ を求めることで導くことが可能であり，

$$MEAN\,(e)=\int_0^{\infty}e\times f\,(e)\,de=\sqrt{\frac{2}{\pi}}\sigma\ \ \because(e\geq 0) \tag{4.20}$$

$$VAR\,(e)=\int_0^{\infty}\left(e-MEAN\,(e)\right)^2\times f\,(e)\,de=\left(1-\frac{2}{\pi}\right)\sigma^2\ \ \because(e\geq 0) \tag{4.21}$$

である．ここで，$f\,(e)$ は確率密度関数であり

$$f\,(e)=2\times\frac{1}{\sqrt{2\pi}\sigma}exp\left(-\frac{e^2}{2\sigma^2}\right)\quad(e\geq 0) \tag{4.22}$$

で定義される．以上より MAE に対する $RMSE$ の比は

$$\frac{RMSE}{MAE}=\sqrt{1+\frac{\left(1-\frac{2}{\pi}\right)\sigma^2}{\left(\sqrt{\frac{2}{\pi}}\sigma\right)^2}}=\sqrt{\frac{\pi}{2}}\approx 1.253 \tag{4.23}$$

となり，誤差が正規分布に従う場合にはこの数値を基にモデルの良し悪しを判別することが可能となる．

第5章

数値実験結果並びに考察

§ 5.1　数値実験の概要

数値実験では，ベンチマーク関数のデータを用いて 4 章で述べた手法で関数近似を行い，その精度を確認，および考察を行う．また，学習終了までの時間を計測して変数の数との関係を考察する．変数の数は 1，2 とし，各変数の定義域は-10 から 10 まで，これらの定義域をそれぞれ 100 分割した点をデータ点とする．これらのデータをもとに基底関数や教師信号を定義し，RC-RBFN により関数の近似を行う．最後に，近似結果と教師信号から算出される $RMSE$，MAE をもとにモデルの評価を行う．

教師信号

数値実験で使用する教師信号は，最適化手法の性能評価に用いられるベンチマーク関数である．ベンチマーク関数とは，最適化アルゴリズムや関数近似手法の性能を比較・評価するために設計された，特定の数理モデルである．これらの関数は，実験環境において目的関数として機能し，アルゴリズムや手法がどの程度正確に近似・最適化できるかを定量的に測定する指標を提供する．関数近似や最適化の分野において，ベンチマーク関数の活用は非常に重要であり，その性能を客観的に比較するための標準的な手段として広く用いられている．今回は Xin-She Yang 関数という次のように定義されるベンチマーク関数を用いる．

$$\eta\left(\mathbf{x}\right) = \left(\sum_{i=1}^{N} |x_i|\right) exp\left(-\sum_{i=1}^{N} sin\left(x_i^2\right)\right) \tag{5.1}$$

ここで，1 変数の場合と 2 変数の場合の Xin-She Yang 関数をそれぞれ図 5.1，図 5.2 に示す．この関数は多峰性関数であり，強い非線形性を持つ．その性質から非線形回帰モデルの数値実験における教師信号として有用であると考えた．

初期状態の基底関数

初期状態における基底関数の数は，4.3 節で述べたように 1 つに設定する．この設定は学習の初期段階において不要な自由度を抑え，基底関数の追加や調整を段階的に行うための基盤を確立するためである．

このときの基底関数のパラメータは，以下のように与えられる．まず，重みは 1 に設定される．これは初期状態では基底関数の影響を均等に持たせることを意図したものであり，

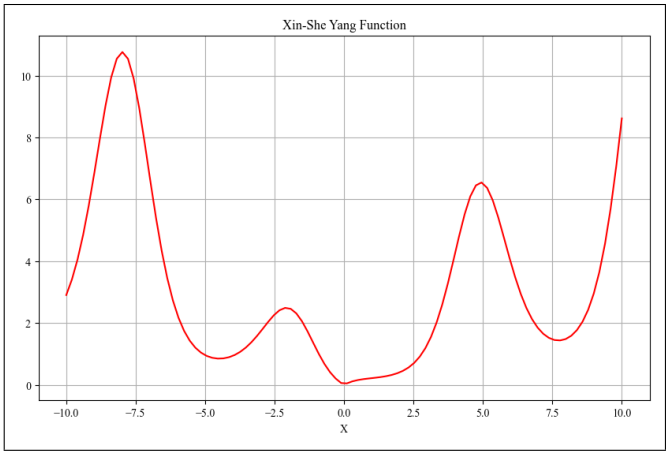

図 5.1: 1 変数 Xin-She Yang 関数

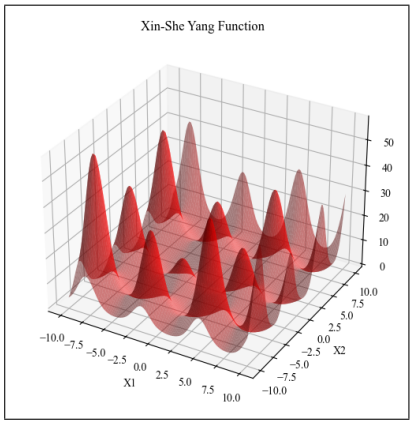

図 5.2: 2 変数 Xin-She Yang 関数

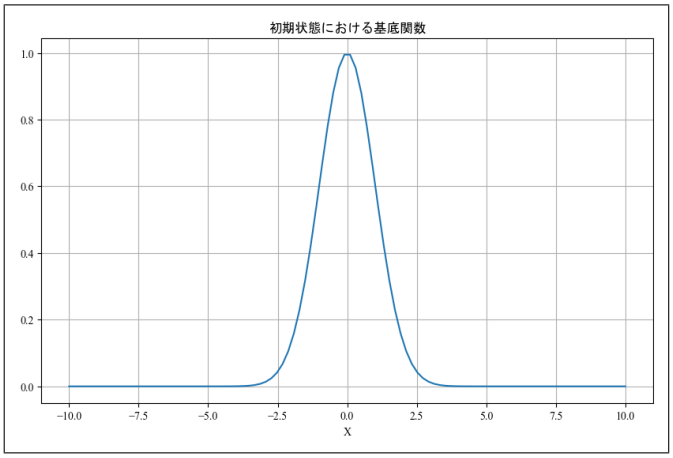

図 5.3: 1 変数の基底関数

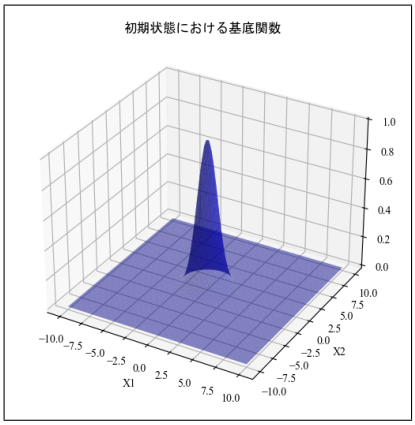

図 5.4: 2 変数の基底関数

学習の進行に伴い適応的に更新されることを前提としている．次に基底関数の中心位置は原点に固定する．最後に分散共分散行列は単位行列とする．これは初期状態では基底関数の影響範囲を等方的に設定し，後の学習によって適切な形状や大きさへと変化させることを目的としている．

初期状態における基底関数の数については，4.3 節で述べたように 1 つとして設定する．このときの基底関数のパラメータは重みが 1，中心位置が原点，分散共分散行列は単位行列として与えられる．1 変数の場合と 2 変数の場合の初期状態における基底関数をそれぞれ図 5.3，図 5.4 に示す．

このような設定の下で 4.3 節の Step 2 から Step 4 を行うことで教師信号の近似を行う．

§ 5.2　実験結果と考察

まず，1 変数の場合の関数を近似する過程において，1 回目の競合の様子と，ピーク検出による複製の様子を図 5.5，図 5.6 に示す．

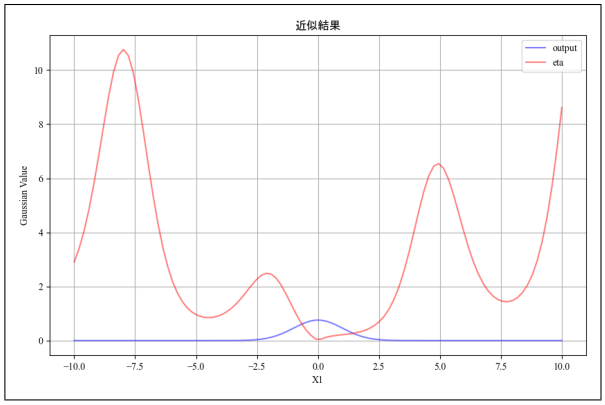

図 5.5: 競合の様子

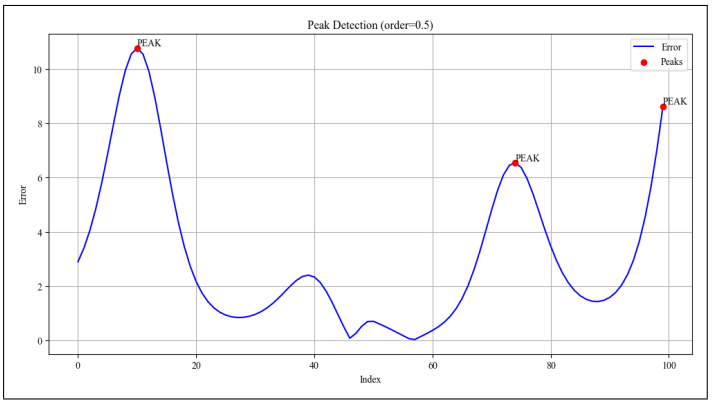

図 5.6: ピーク検出の様子

第6章

おわりに

本研究ではまず，自由エネルギーとエントロピーの観点から誤差を効率的に減らす位置に基底関数を複製する機能と，適者生存型学習則を適用することによって冗長な基底関数が存在する場合に競合を発生させ冗長な基底関数を削除する機能を持つ RC-RBFN の実装を行った．既存の RBFN では，基底関数の数が少ない場合には複雑な非線形関数を近似することができず，基底関数の数が過剰な場合には冗長な計算を行うことになっていた．また，ターミナルアトラクタを導入することによって学習回数の上限を決定することが可能になり，学習の効率化を図ることが可能となっている．

本研究で提案したシステムの特徴をまとめる．1 つ目の特徴は，複製すべき位置の導出を誤差関数から求めることにより多重積分の計算を回避している点である．従来の複製手法では多重積分を行うことで，変数が多くなるにつれて，計算量が指数関数的に増加することが課題となっていた．その課題を誤差関数のピークを複製点とすることで多重積分を回避している．このことにより，変数の増加によって発生していた計算量の課題を緩和できたと考えられる．

2 つ目の特徴は，RC-RBFN を教師あり・なし学習や強化学習，逆強化学習に組み込むことの提案である．RC-RBFN は基底関数の複製と削除を考慮できるニューラルネットワークにもかかわらず機械学習に組み込んでいる事例はない．そこで本研究では各手法における RBFN やエントロピーの観点からの関連を示し，動的な環境における機械学習の手法を提案した．

今後の課題として，実行時間の短縮があげられる．本研究では Python の高速化ライブラリである Numba によってモデルを JIT コンパイルすることで C 言語のように扱ったり，CPU による並列分散処理を用いることで高速化を図った．しかし，高次元ではまだまだ処理の時間がかかっておりさらなる高速化が可能だと考えられる．そこで，CPU ではなく GPU を用いた並列処理や，複数台のコンピュータを用いた分散処理などの手法が有効であると考えられる．さらに，学習率を動的に変化させることで学習の安定化や，適者生存型学習則による重みの制限の撤廃などもより汎化性能を高める上で重要になる点だと考えられる．以上の点を今後改善・検討することで，本手法の実用性と性能を一層向上させることができると考える．処理速度の向上こそが多変数関数の近似において不可欠な要件であるといえる．

謝辞

本研究を遂行するにあたり，多大なご指導と終始懇切丁寧なご鞭撻を賜った富山県立大学工学部電子・情報工学科情報基盤工学講座の奥原浩之教授，António Oliveira Nzinga René 講師に深甚な謝意を表します．最後になりましたが，多大な協力をしていただいた研究室の同輩諸氏に感謝致します．

2025 年 2 月

小澤 翔太

参考文献

[1] M.J.D. Powell, “Radial basis functions for multivariable interpolation: a review”, *Algorithms for approximation*, pp. 143-167, 1987.

[2] 奥原浩之, 尾崎俊治, “適者生存型学習則を適用した競合動径基底関数ネットワーク”, 電子情報通信学会論文誌, Vol. 80, pp. 3191-3199, 1997.

[3] 奥原浩之, 佐々木浩二, 尾崎俊治, “環境の変化に適応できる複製・競合動径基底関数ネットワーク”, 電子情報通信学会論文誌, Vol. 82, pp. 941-951, 1999.

[4] 倉本和正, 鉄賀博己, 東寛和, 荒川雅生, 中山弘隆, 古川浩平, “RBF ネットワークを用いた非線形がけ崩れ発生限界雨量線の設定に関する研究”, 土木学会論文集, pp. 117-132, 2001.

[5] 持田英史, 飯國洋二, “RBF ネットワークを用いた到来波方向の適応推定”, 電子情報通信学会論文誌, Vol. 87, pp. 1205-1214, 2004.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., “Mastering the game of go with deep neural networks and tree search”, *Nature*, Vol. 529, pp. 484-489, 2016.

[7] 高橋直矢, 池谷裕二, 松木則夫, Hebb 則, 脳科学辞典, `https://bsd.neuroinf.jp/wiki/`ヘブ則

[8] Y. Wong, “Clustering Data by Melting”, *Neural Computation*, Vol. 5, pp. 89-104, 1993.

[9] Zak, Michail, “Terminal attractors for addressable memory in neural networks”, *Physics Letters A*, Vol. 133, pp. 18-22, 1988.

[10] Zak, Michail, “Terminal attractors in neural networks”, *Neural Networks*, Vol. 2, pp. 259-274, 1989.

[11] Z. Wang, C. D. Massimo, M. T. Tham, and A. J. Morris, “A procedure for determining the topology for multilayer feedforward networks”, *Neural Networks*, Vol. 7, pp. 291-300, 1994.

[12] A. Charpentier, R. Elie and C. Remlinger, “Reinforcement learning in economics and finance”, *Computational Economics*, pp. 1-38, 2021.

[13] Hideki Asoh, Masanori Shiro, Shotaro Akaho, Toshihiro Kamishima, “An Application of Inverse Reinforcement Learning to Medical Records of Diabetes Treatment”, 閲覧日 2025-01-23, =https://api.semanticscholar.org/CorpusID:2159236.

[14] Z. Wang, C. D. Massimo, M. T. Tham, and A. J. Morris, “A procedure for determining the topology for multilayer feedforward networks”, *Neural Networks*, Vol. 7, pp. 291-300, 1994.

[15] 三菱電機, “「人と協調する AI」を開発”, 閲覧日 2025-1-20, https://www.mitsubishielectric.co.jp/news/2020/0603.html.

[16] A. Y. Ng, Stuart Russell, “Algorithms for Inverse Reinforcement Learning”, *Icml*, Vol. 1, 2000.

[17] W. Markus, P. Ondruska, and I. Posner, “Maximum entropy deep inverse reinforcement learning”, *arXiv preprint arXiv:1507.04888*, 2015.

[18] B. D. Ziebart, A. Maas, J. A. Bagnell, and A.K. Dey “Maximum entropy inverse reinforcement learning”, *Aaai*, Vol. 8, pp. 1433-1438, 2008.

[19] 北山雄三, 北山哲士, 山崎光悦, “RBF ネットワークによるパターン分類”, 設計工学・システム部門講演会講演論文集, Vol. 2008.18, pp. 330-332, 2008.

[20] 上坂竜規, 野田雅文, 目加田慶人, 出口大輔, 井手一郎, 村瀬洋, “ドライバの視線情報を利用した運転行動予測”, 信学技報, PRMU2011-19, 2011.

[21] 浅田拓海, 岡田和洋, 松田真宜, 有村幹治, “RBF ネットワークによるコミュニティサイクル利用動態の短期予測”, 土木学会論文集 D3 (土木計画学), Vol. 71, pp. 425-431, 2015.

[22] J. Moody, C.J. Darken, “Fast Learning in Networks of Locally-Tuned Processing Units”, *Neural computation*, Vol. 1, pp. 281-294, 1989.

[23] 白井隆晴, 荒川雅生, 中山弘隆, “回帰と RBF を利用した近似最適化”, 最適化シンポジウム講演論文集, Vol. 2004.6, pp. 279-284, 2004.

[24] 安道知寛, 井元清哉, 小西貞則, “動径基底関数ネットワークに基づく非線形回帰モデルとその推定”, 応用統計学会, Vol. 30, pp. 19-35, 2001.

[25] 森本淳, 銅谷賢治, “強化学習を用いた高次元連続状態空間における系列運動学習—起き上がり運動の獲得—”, 電子情報通信学会論文誌 D, Vol. 82, pp. 2118-2131, 1999.

[26] 鮫島和行, 大森隆司, “強化学習における適応的状態空間構成法”, 日本神経回路学会誌, Vol. 6, pp. 144-154, 1999.

[27] P. Abbeel, A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning”, *Proceedings of the twenty-first international conference on Machine learning*, 2004.

[28] D. Ramachandran, E. Amir, “Bayesian Inverse Reinforcement Learning”, *IJCAI*, Vol. 7, pp. 2586-2591, 2007.

[29] 奥原浩之, “報酬駆動型システムにおける報酬の設計と報酬による最適化”, 一般社団法人 システム制御情報学会, Vol. 58, pp. 462-467, 2014.

[30] 中口悠輝, “最大エントロピー逆強化学習の性能の理論評価”, 人工知能学会全国大会論文集 第 35 回, pp. 1G2GS2a01-1G2GS2a01, 2021.

[31] 畠中利治, 近藤伸彦, 魚崎勝司, “多目的進化計算によるパレート最適な RBF ネットワークの構成法”, 日本知能情報ファジィ学会 ファジィ システム シンポジウム 講演論文集 第 21 回ファジィ システム シンポジウム, p. 182, 2005.