

CIRCULARS
OF THE
ELECTROTECHNICAL LABORATORY

No. 228

January, 1999

UDC 512.5:514.14:514.742:517.53:519.2:519.6

**Current Status of Research on Neural Networks
with High-dimensional Parameters**

by

Tohru NITTA, Masaru TANAKA

SYNOPSIS

Complex systems have been attracting much interest in recent years all over the world. There exists a computational tool called *Complex Adaptive Systems* for dealing with complex systems, which is based on a comprehensive concept including the features of evolvable or learnable systems such as living creatures. Artificial neural networks can be considered to be a type of complex adaptive system. So far, it has been confirmed that artificial neural networks have the ability to learn, generalize and associate, and they have been applied to various fields such as image processing and speech recognition. Although the usual neural network is a sort of complex adaptive system, it is too simple to adequately handle complex systems such as economic phenomena, global environmental phenomena and the like.

In the meantime, there have been several attempts to extend the parameters (weights and thresholds) of the usual neural networks to higher dimensions (complex numbers, three-dimensional vectors, quaternions, etc.). In this paper, we call such neural networks with high-dimensional parameters *high-dimensional neural networks*. High-dimensional neural networks will extend the applicable domains of artificial neural networks, and will facilitate the application of artificial neural networks. We believe that the high-dimensional neural networks can also treat complex systems very well.

This paper will survey the current status of research on high-dimensional neural networks.

In Chapter 2, the current status of research on *high-dimensional autoregressive models*, which are the fundamental theoretical basis of high-dimensional neural networks, is described. In general, there is a close relationship between autoregressive models and neural networks in the sense that the autoregressive model is a linear approximation

KEYWORDS: neural network, autoregressive model, invariant, complex number, 3-dimensional vector, vector product, quaternion

to the artificial neural network with non-linearity. The progress of research on high-dimensional autoregressive models facilitates research on high-dimensional neural networks.

Chapter 3 surveys the current status of research on *multi-layered* high-dimensional neural networks. Some kinds of complex-valued, . . . , quaternion-valued neural network models, their characteristics and their applications will be described.

Chapter 4 outlines the current status of research on *fully connected recurrent* high-dimensional neural networks. Only complex-valued ones have been proposed so far.

Chapter 5 discusses future research topics.

目 次

第 1 章はじめに	1
第 2 章高次元自己回帰モデル	3
2.1 複素 (2 次元) 自己回帰モデル	3
2.1.1 複素自己回帰モデルによる形の識別	3
2.1.1.1 従来法の問題点	4
2.1.1.2 複素自己回帰モデルと複素 PARCOR 係数	4
2.1.1.3 複素自己回帰係数と複素 PARCOR 係数の高速計算法	6
2.1.1.4 複素自己回帰モデルを用いた形の認識	6
2.2 3 次元実自己回帰モデル	7
2.2.1 2 次元行列型実自己回帰モデル	7
2.2.2 3 次元実自己回帰モデル	9
2.3 4 次元実自己回帰モデル	11
2.4 K 次元実自己回帰モデル	12
2.5 本章のまとめ	13
第 3 章多層型高次元ニューラルネットワーク	14
3.1 多層型複素ニューラル ネットワーク	14
3.1.1 モデル	14
3.1.1.1 Widrow モデル	14
3.1.1.2 Kim-Leung モデル	14
3.1.1.3 新田・Benvenuto モデル	15
3.1.1.4 Georgiou モデル	18
3.1.1.5 複素一般化ヘブ学習	19
3.1.2 新田・Benvenuto モデルの性質	19
3.1.2.1 決定表面の構造	19
3.1.2.2 学習特性	24
3.1.2.3 2 次元アフィン変換学習能力	29
3.1.2.4 2 次元アフィン変換学習能力の数学的解析	35
3.1.3 写像近似能力	36
3.1.3.1 正則写像	36
3.1.3.2 複素ニューラルネットワーク	36
3.1.3.3 複素ニューラルネットワーク設計問題	37
3.1.4 複素 BP 学習の高速化	38
3.1.5 コンピュータビジョンへの応用	39
3.2 多層型 3 次元ニューラルネットワーク	41
3.2.1 3 次元ベクトルニューラルネットワーク	41

3.2.2	3次元外積ニューラルネットワーク	44
3.2.3	コンピュータビジョンへの応用	46
3.3	多層型4元数ニューラルネットワーク	48
3.3.1	新田モデル	48
3.3.2	関数近似能力と時系列データ予測への応用	49
	第4章相互結合型複素ニューラルネットワーク	51
	第5章おわりに	53
	謝辞	53
	参考文献	54

高次元ニューラルネットワーク関連研究の現状

通商産業技官 新田 徹
通商産業技官 田中 勝

第 1 章 はじめに

情報技術は、計算基盤技術(ソフトウェア、ハードウェア、ネットワークなど)、複雑系技術、知的ヒューマンインターフェース技術、知能ロボット技術に大きく分けることができ、インターネットの普及にも支えられ、近年研究が活発化している(図 1.1)。

ここ数年複雑系が話題になっているが[92]、複雑系を扱うためのツールとして複雑適応系がある。複雑適応系とは生物のような進化・学習するシステムや免疫システムなどを指す包括的な概念であり、ニューラルネットワークもそれに該当しており、ニューラルネットワークは複雑系技術に分類することができる[22]。ニューラルネットワークは、現在までに学習能力、汎化能力、連想能力などを持っていることが確認されていて、様々な分野に応用されている[18]。たとえば、知能ロボット(運動制御など)、知的ヒューマンインターフェース(音声認識など)にも使われていて、もはやひとつの広義の計算基盤技術になりつつあると言えるかもしれない。

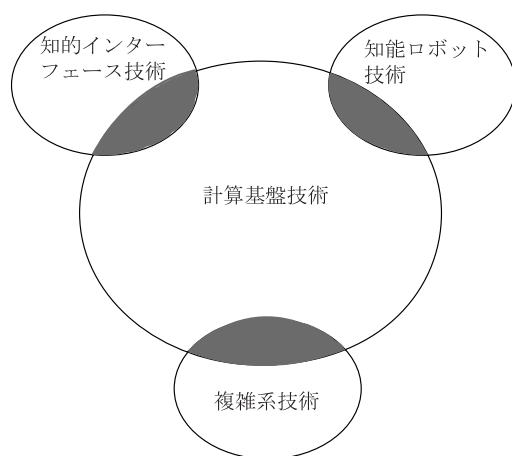


図 1.1 情報技術の全体像

このような状況のなかで、一般のニューラルネットワークは複雑適応系的一种ではあるものの、それだけでは単純なため、経済現象、社会変動、地球環境などの複雑系を十分に扱い切れているわけではないのが現状である。カオスニューラルネットワーク[4]は、ニューラルネットワークを複雑系に対処することができるようにする試みの一つと言ってもいいかもしれない。

ところで、一般のニューラルネットワークの重みや閾値といったパラメータを、複素数(2次元)や3次元に高次元化する試みがある。本稿では、高次元化されたパラメータを持つニューラルネットワークを高次元ニューラルネットワークと呼ぶことにする。特に複素領域への拡張は、通信方式等の複素数を扱う分野での応用が特に期待されているものである。従来は、実数の場合に使われていた手法を実部と虚部に分けて適応する必要があったものが、複素領域に拡張された手法を使うことにより直接的にデータの処理を行うことができる。また、複素数の持つ回転に対する性質の良い振る舞いを自動的に取り込むことができるという利点もある。このように、高次元ニューラルネットワークは陽に複雑系に対処することができるようにとの目的で提案されたわけではなく、その目的は高次元情報(例えば、複素数、3次元情報など)を自然に表現することができ、素直に処理することのできるニューラルネットワークを提供することにあつた。ところが、その性質を調べてゆく中で、通常のニューラルネットワークには見られない2次元アフィン変換学習能力などの特異な性質があることがわかってきた。このような性質は複雑系に対処することができる可能性があると考えられる。すなわち、高次元ニューラルネットワークは、知的インターフェースや知能ロボットなどへの応用を広げ、促進させることは言うまでもなく、複雑系をうまく処理することができる可能性があるものとして我々

は見ている。

本調査報告書では、以上のようなことを踏まえて、高次元ニューラルネットワーク関連研究の現状のサーベイを行なう。

本論文の構成は次のようになっている。

まず第 2 章では、高次元ニューラルネットワークの理論的基盤となる高次元自己回帰モデルの研究の現状について述べる。一般に、自己回帰モデルは、非線形性を持つニューラルネットの線形近似でもあり、高次元ニューラルネットと高次元自己回帰モデルとは互いに密接な関係がある。高次元自己回帰モデルの研究の進展は、高次元ニューラルネットの研究に良い刺激を与え、研究を促進させる。高次元自己回帰モデルに備わっている性質は、高次元ニューラルネットにも備わっている可能性が高い。逆も同じである。たとえば、新田らが 1991 年に複素ニューラルネットワーク [43] を提案した際には、複素自己回帰モデルを参考にしている。また、高次元自己回帰モデルには、高次元図形の不変量を学習することができるという通常の自己回帰モデルには備わっていない固有の性質があるが、それに対応するものが高次元ニューラルネットワークにも備わっているものと予想される。

第 3 章では、信号の流れが一方向である多層型の高次元ニューラルネットワーク研究の現状について述べる。複素数から 4 元数にまで高次元化されたニューラルネットワークの各種のモデル、性質、応用について述べる。

第 4 章では、信号がネットワークの中を何度も流れることができる相互結合型の高次元ニューラルネットワーク研究の現状について述べる。現在のところ、相互結合型については、複素数タイプのニューラルネットワークしか提案されていないので、複素ニューラルネットワークについてのみ述べることになる。

最後に第 5 章では、将来に対する展望を述べる。

第 2 章 高次元自己回帰モデル

2.1 複素 (2 次元) 自己回帰モデル

複素数への自己回帰モデルの拡張は、元々パターン認識における最も基本的な課題の一つである平面図形の認識や分類を行うために提案されたものである。特に、図形の平面内での回転変換に対して不変な特徴量を得ることは、対象図形の回転不変な認識を行うために重要であり、この性質を持つものとして複素自己回帰モデルにより得られる複素自己回帰係数もしくは複素偏自己回帰係数 (複素 PARCOR 係数) が提案されている。以下では、栗田による研究報告 [34] に基づいて、複素自己回帰モデルについて紹介する。

平面図形 (2 値画像) の認識は、パターン認識において最も基本的であり、それゆえに認識に必要なさまざまな問題を抽出し、明確にしていくのに有用である。特に、形の本質的な情報を担う外形 (輪郭線) の記述は、認識や分類のために重要であり、これに関して、これまでも平面図形を表現するための多くの形の記述子 (shape descriptors) が提案されている [71]。たとえば、輪郭線の記述子としては、Fourier 記述子や平滑化曲率関数が有名である。ここでは、統計的手法を用いて形の相似変換によらない形の認識や分類のために好ましい特徴を構成するための統計的手法として複素自己回帰モデルについて考えていく。

Dubois ら [14] は、実自己回帰モデルを用いて外形を記述し、形を識別することを試みた。彼らは、平面図形の重心を原点として輪郭を等角度で標本化し、原点から輪郭までの距離をデータ点列として実自己回帰モデルを当てはめ、その係数を形の識別に用いている。また、Kashap ら [25] は、平面曲線の符号化の立場から実自己回帰モデルの理論的な解析を行なった。実自己回帰モデルは、音声波形等の時系列データの解析・記述のために有効な手法である。音声認識の分野では、それは線形予測法あるいは PARCOR 法として知られており、音声波形を分析するための基本的な手法となっている。しかし、彼らの定式化では、本来 2 次元の点列として表現される輪郭点列に対して 1 次元の実自己回帰モデルを当てはめたため

にいくつかの問題が生じている。例えば、輪郭を極座標表現し、原点から輪郭点までの距離をデータ点とする方法では、角度に関して輪郭点が多価となるような図形が存在し、任意の輪郭を一意に表現できなくなるという欠点がある。

そこで、栗田ら [32,37,75,77,78] は、輪郭上の各輪郭点を複素数で表現した輪郭点列に対して複素自己回帰モデルを当てはめ、このときの係数を用いて形を識別する手法を提案した。それによると、複素自己回帰モデルを用いることにより、輪郭に実自己回帰モデルを当てはめたときに生じていた問題が解消され、輪郭を少数のパラメータでモデル化することが可能となる。この係数 (複素自己回帰係数) は輪郭の回転や輪郭点列を追跡する際の始点の選び方に依存しない量であり、さらに、輪郭点の量子化法を工夫すると、平行移動や大きさに対しても不変となる [32,37,75,77,78]。また、この複素自己回帰モデルに基づき複素 PARCOR 係数と呼ばれる量を定義することができる [32,75,77,78]。これも形の相似変換に関して不変な特徴となる。また、彼らは、これらの特徴を高速に計算するためのアルゴリズムも示し [32,77,78]、この複素自己回帰モデルに基づいて形の相似変換に不変な 2 つの輪郭形状間の距離を定義し [33,76]、形の自動分類や類似した形状の検索などの問題に適用している。

2.1.1 複素自己回帰モデルによる形の識別

ここでは、形状認識の問題を通して、複素自己回帰モデルについて考察する。そのためにまず、形の記述子として実自己回帰モデルを用いる従来法を概観し、その問題点を指摘する。それから、複素自己回帰モデルに基づいた複素自己回帰係数および複素偏自己回帰係数 (複素 PARCOR 係数) を定義し、その性質について形の認識の観点から考察し、複素自己回帰係数および複素 PARCOR 係数の高速計算法についても紹介する。この高速計算法は、音声波形等の実数値列に対する実自己回帰係数および実 PARCOR 係数をもとめる Levinson-Durbin のアルゴリズム [13] を、複素数値列に対する複素自己回帰係数および複素 PARCOR 係数を求めるために自然に拡張した計算法になっている [32,33]。また、これらの係数を用いた形の相似変換に不変な識別方法も合わせて紹介する [75]。

2.1.1.1 従来法の問題点

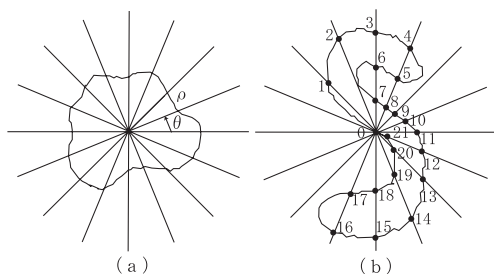


図 2.1 極座標表

Dubois ら [14] は、実自己回帰モデルを以下のようにして、図形の輪郭の認識に適用している。まず、輪郭線を、図 2.1(a) に示すように、輪郭線の重心を原点とする極座標を用い偏角を N 等分した放射状の線で標準化することで、 (ρ_j, θ_j) , $(\theta_j = 2\pi j/N)$ のように表現している。こうして得られた輪郭点列 $\{\rho_j\}_{j=0}^{N-1}$ に対して、 m 次の実自己回帰モデル

$$\hat{\rho}_j = \sum_{k=1}^m c_k \rho_{j-k} \quad (2.1)$$

の係数 $\{c_k\}_{k=1}^m$ を最小二乗法により求め、それを形の認識に利用している。このとき、モデルの係数は図形の相似変換（平行移動、大小伸縮、回転）に関して不変な表現となっている。しかし、図 2.1(b) に示すように、極座標による等角度標準化を用いたのでは、放射状に伸びる輪郭線（図 2.1(b) の 20, 21, 0, 1 の付近の輪郭）に対して輪郭点位置が不安定になる。また、その輪郭点間隔が不均等となってしまう。さらに、図形によっては θ に関して ρ が多価となることもあり得る。こうした多価性を解消するために、Dubois ら [14] は、輪郭点の追跡順（図 2.1(b) の数字の順）に実自己回帰モデルを当てはめる方法を提案している。しかし、これによって得られる $\{\rho_j\}$ からでは θ_j が一意に定まらないので、元の輪郭を一意的に表現できなくなってしまう。

輪郭線に実自己回帰モデルを当てはめる他の手法としては、輪郭線を曲率関数で表現する方法も考えることができる。しかし、曲率関数は図形の回転に不変な量であり、形の記述には適しているものの、輪郭線の微分に関係する量であり、輪郭線の量子化等のノイズの影響を受

けやすいという欠点を持っている。そのため、得られる実自己回帰係数はノイズに敏感なものになってしまう [25]。

以上のような問題は、本来 2 次元の平面曲線を 1 次元で表現しようとすることに起因している。そこで、2 次元平面上の輪郭点列に対して、2 次元の実自己回帰モデルを適用することが考えられる。すなわち、輪郭点列 $\{(x_j, y_j)\}$ に 2 次元の実自己回帰モデル

$$\begin{bmatrix} \hat{x}_j \\ \hat{y}_j \end{bmatrix} = \sum_{k=1}^m A_k \begin{bmatrix} x_{j-k} \\ y_{j-k} \end{bmatrix} \quad (2.2)$$

を当てはめることが考えられる。しかし、この場合には、得られる実自己回帰係数 $\{A_k\}_{k=1}^m$ は、残念ながら図形の回転に伴う輪郭点列の回転に対して不変とはならない。さらに、モデルの係数は 2 行 2 列の行列となり、他の方法に比べてモデルのパラメータ数が多くなってしま（しかし、後で見るように行列の性質をうまく使えば、モデルの係数が行列であることは、それほど問題ではない）

2.1.1.2 複素自己回帰モデルと複素 PARCOR 係数

ここでは、実数値列に対する従来の実自己回帰モデルを複素数値列に対する複素自己回帰モデルへ拡張する。まず、複素自己回帰モデルを定義し、複素自己回帰係数の性質について考察する。次に、複素自己回帰モデルに基づく複素偏自己相関係数（複素 PARCOR 係数）を定義し、これらの係数の高速計算法を与える。

平面図形の輪郭線（平面曲線）を追跡して得られる点列を $\{(x_j, y_j)\}_{j=0}^{N-1}$ とし、その複素表現を $z_j = x_j + iy_j$ とする。

このとき、 m 次の複素自己回帰モデルは、

$$\hat{z}_j = \sum_{k=1}^m a_k z_{j-k} \quad (2.3)$$

のように輪郭点を m 個前までの輪郭点の線形結合で近似するモデルとして定義される。ここで、モデルの係数 $\{a_k\}_{k=1}^m$ は平均二乗予測誤差 $\varepsilon^2(m)$

$$\begin{aligned} \varepsilon^2(m) &= E_j |\hat{z}_j - z_j|^2 \\ &= \bar{a}^T R a - \bar{r}^T a - \bar{a}^T r + r_0 \end{aligned} \quad (2.4)$$

が最小となるように決められる。ただし,

$$R = \begin{bmatrix} r_0 & \bar{r}_1 & \cdots & \bar{r}_{m-1} \\ r_1 & r_0 & \cdots & \bar{r}_{m-2} \\ r_2 & r_1 & \cdots & \bar{r}_{m-3} \\ \cdots & \cdots & \cdots & \cdots \\ r_{m-1} & r_{m-2} & \cdots & r_0 \end{bmatrix}, \quad (2.5)$$

$$\mathbf{a} = [a_1, a_2, a_3, \dots, a_m]^T, \quad (2.6)$$

$$\mathbf{r} = [r_1, r_2, r_3, \dots, r_m]^T, \quad (2.7)$$

$$r_k = E_j(z_j \bar{z}_{j-k}) \quad (2.8)$$

である。また, E_j は次のような輪郭点列に関する平均操作 $(1/N) \sum_{j=0}^{N-1}$ を表わし, $\bar{\cdot}$ は複素共役, T は行列の転置を表わしている。 r_k は複素相関係数であり, R は Hermite 行列, すなわち $R^* = R$ となる ($*$ は共役転置をとることを意味する)。

平均二乗予測誤差 (式 (2.4)) を最小にするという基準において, 最適であるような複素自己回帰係数 $\{a_k\}_{k=1}^m$ は,

$$\mathbf{a} = R^{-1} \mathbf{r} \quad (2.9)$$

で与えられる。

また, このとき達成される最小平均二乗予測誤差 $\hat{\epsilon}^2(m)$ は

$$\hat{\epsilon}^2(m) = r_0 - \mathbf{r}^T \bar{\mathbf{a}} \quad (2.10)$$

となる。

こうして得られた複素自己回帰係数の性質について考えてみる。与えられた輪郭点列が複素表現されている場合, 原点まわりの角度 θ だけの回転は各輪郭点列を $e^{i\theta}$ 倍することに対応している。回転された輪郭点列 $\{e^{i\theta} z_j\}_{j=0}^{N-1}$ に対する r_k は, 式 (2.8) から求められるが, そこでは $e^{i\theta}$ 同士は互いに打ち消しあい, r_k は回転によって不変であることがわかる。したがって, 式 (2.9) も不変となり, 結局, 複素自己回帰係数も回転不変な量となる。また, 式 (2.8) から r_k は輪郭を追跡する際の開始点の選び方に依存しない量であることがわかる。そのため, それに基づいて計算される複素自己回帰係数も輪郭点列の開始点の選び方には依存しない量となる。これらの性質は, 形の記述, 表現を考える上で好ましいものである。また, さらに興味深い性質として輪郭線の追跡方向 (時計回りか反時計回り) を変えると, 得られる複素自己回

帰係数は, 追跡方向を変更する前の複素自己回帰係数の複素共役をとることにより得ることができる。このことは輪郭点列を逆方向に追跡した場合には, 式 (2.3) の右辺は z の添字 $-k$ を $+k$ と置き換えたものとなり, r_k 自身が複素共役となることから明らかである。

次に複素 PARCOR 係数について考える。実 PARCOR 係数は実自己回帰係数と数学的に等価なものであるが, 実際の信号から実自己回帰モデルを構成していく上で実自己回帰係数にはない優れた利点があることが知られている。また, 実 PARCOR 係数は, いろいろな問題に対して直接適用することも可能である。そのため, 音声信号処理においては, 実 PARCOR 係数がしばしば用いられている。ここでは, 輪郭点列の記述のために, 実数値列に対する実 PARCOR 係数を複素数値列の場合に拡張することによって, 複素 PARCOR 係数を定義する。

実数値列に対する m 次の実 PARCOR 係数は, 順方向および逆方向の $(m-1)$ 次実自己回帰モデルによる予測誤差の相互相関係数として定義されている。そこで, m 次の複素 PARCOR 係数 p_m を, 順方向および逆方向の $(m-1)$ 次の複素自己回帰モデルによる予測誤差 $\epsilon_j^f(m-1)$ と $\epsilon_j^b(m-1)$ の複素相互相関係数として,

$$p_m = \frac{E_j(\epsilon_j^f(m-1) \bar{\epsilon}_{j-m}^b(m-1))}{\left\{ E_j(\epsilon_j^f(m-1) \bar{\epsilon}_j^f(m-1)) \times E_j(\epsilon_{j-m}^b(m-1) \bar{\epsilon}_{j-m}^b(m-1)) \right\}^{1/2}} \quad (2.11)$$

のように定義する。ただし,

$$\epsilon_j^f(m-1) = z_j - \sum_{k=1}^{m-1} a_k z_{j-k}, \quad (2.12)$$

$$\epsilon_j^b(m-1) = z_j - \sum_{k=1}^{m-1} b_k z_{j+k} \quad (2.13)$$

である。ここで, $\{b_k\}_{k=1}^{m-1}$ は逆方向の $(m-1)$ 次複素自己回帰係数である。

実はここで定義した m 次の複素 PARCOR 係数 p_m は, m 次の複素自己回帰係数 $\{a_k\}_{k=1}^m$ の a_m に等しいものとなっている。また, m 次の複素自己回帰モデルに対して一つの複素 PARCOR 係数 p_m のみが定義されるため, モデルの次数が増加しても, その次数より低次の複素 PARCOR 係数は影響を受けないという利点がある。

2.1.1.3 複素自己回帰係数と複素 PARCOR 係数の高速計算法

複素自己回帰係数あるいは複素 PARCOR 係数を用いて形の認識を高速に行なうためには、これらの係数を高速に求めることが必要である。 m 次の複素自己回帰係数を式(2.9)から Gauss の消去法などで求めると、 $O(m^3)$ の演算回数が必要となる。さらに 1 次から m 次までの複素 PARCOR 係数を全て求めるためには $O(m^4)$ の演算回数が必要になる。栗田ら [34] は、1 次から m 次までの複素自己回帰係数と複素 PARCOR 係数を $O(m^2)$ で求める手法を与えている。ここではその手法を紹介する。

式(2.9)の次数に関する再帰式を考えよう。ここで次数を明示するために、 m 次複素自己回帰モデルにおける式(2.5)(2.6)(2.7)の R, a, r を、それぞれ、 $R(m), a(m), r(m)$ と書くことにする。

1 次の複素自己回帰モデルの複素自己回帰係数 a_1 は、明らかに

$$a(1) = r_1/r_0 \quad (2.14)$$

である。2 次以上の複素自己回帰モデルに対しては、 $R(m)$ が

$$R(m) = \begin{bmatrix} R(m-1), & \bar{r}(m-1)^\dagger \\ \bar{r}(m-1)^*, & r_0 \end{bmatrix} \quad (2.15)$$

と表わせるので、ブロック行列の逆行列の公式を用いて整理すると、 m 次のモデルの複素自己回帰係数列ベクトル $a(m)$ は、 $(m-1)$ 次のモデルの複素自己回帰係数 $a(m-1)$ を用いて、

$$a(m) = \begin{bmatrix} a(m-1) - \bar{a}(m-1)^\dagger p_m \\ p_m \end{bmatrix}, \quad (2.16)$$

$$p_m = (r_m - r(m-1)^* a(m-1)) / (r_0 - r(m-1)^T \bar{a}(m-1)) \quad (2.17)$$

と書ける。但し、 $a(m-1)^\dagger$ は $a(m-1)$ の要素の順番を逆にしたベクトル $[a_{m-1}, a_{m-2}, \dots, a_1]^T$ を表し、 $a(m-1)^*$ は $a(m-1)^\dagger$ の転置を意味する。

従って、1 次のモデル(式(2.14))から始めて、式(2.16)(2.17)を繰り返し適用すれば、高次のモデルの係数を次々と順番に求めていくことができる。また、式(2.17)で与えられる p_m は実は m 次の複素 PARCOR

係数と等しい。つまり、この方法で複素自己回帰係数と複素 PARCOR 係数が同時に求められることになる。この方法は、結果的に実数値列に対する実自己回帰係数を計算する手法として知られている Levinson-Durbin によるアルゴリズム [13] の複素数値列に対する複素自己回帰係数を計算する場合への自然な拡張になっている。

また、上述の逐次式を用いて複素 PARCOR 係数と平均二乗誤差との間には

$$\varepsilon^2(m)/\varepsilon^2(m-1) = 1 - p_m \bar{p}_m, \quad \varepsilon^2(0) = r_0 \quad (2.18)$$

のような関係式が成り立つことも示すことができる。この関係式は、モデルの次数を情報量基準等で決める際の目安として利用できる。

2.1.1.4 複素自己回帰モデルを用いた形の認識

ここでは複素自己回帰モデルを形の認識に用いるために、形の相似変換に不変な複素自己回帰係数および複素 PARCOR 係数を求める方法を示し、これらの係数を特徴量として形を識別する方法を示す。

「形(shape)」は、本来、パターンの相似変換に不変な概念である。ここでは、複素自己回帰モデルに基づき、形の認識のために必要な相似不変な特徴を得るための方法を示す。式(2.16)からわかるように、複素 PARCOR 係数 p_m は m 次の複素自己回帰モデルの m 次の係数 a_m と等しいので、複素自己回帰係数について成立する性質は同時に複素 PARCOR 係数でも成り立つ。

パターンの相似変換は、平行移動 $z_j + d$ と重心まわりの回転 $e^{i\theta} z_j$ 、および大小伸縮 ρz_j (ρ : 正実数)とで表わすことができる。前述のように複素自己回帰係数および複素 PARCOR 係数は原点まわりの回転と輪郭線(閉曲線)を追跡する際の開始点の選び方に関して不変である。そこで、これら以外の変換に関して不変となるように複素自己回帰モデルを輪郭点列に当てはめる方法について考える。

平行移動に不変とするためには、重心を原点にとり輪郭点列を表現する方法や、隣り合う輪郭点の差分に対して複素自己回帰モデルを当てはめる方法などが考えられる。輪郭の量子化誤差等のノイズの影響をより少なくするためには、前者の手法が好ましいと思われる。また、



図 2.2 輪郭の量子化

前者の手法を採用することにより、重心まわりの回転に対して係数が不変となる。

大小伸縮に不変とするためには、輪郭の全周長を N 等分する区間に分割し、各区間内のデータ点を平均値で代表させ、輪郭点列 $\{z_j\}_{j=1}^{N-1}$ とすればよい(図 2.2)。大きさの違う図形の輪郭点列は $\{\rho z_j\}$ のように表現されるので、 r_k は ρ^2 倍になる。したがって、 R や r も ρ^2 倍になる。しかし、これらは、式(2.9)右辺においてすべて打ち消しあうので、結局、複素自己回帰係数 a は大小伸縮に対して不変となる。また、この方法は、輪郭の量子化誤差を軽減する効果もある。

実際の形の認識においては、必ずしも大きさに不変にする必要はなく、むしろ大きさの違いも区別したい場合がある。そういう場合には、輪郭をある一定の周長で標準化して複素自己回帰モデルを当てはめれば良いであろう。

次に、複素自己回帰係数や複素 PARCOR 係数を特徴ベクトルとして形を識別する方法について考える。理想的には、複素自己回帰係数や複素 PARCOR 係数は形の相似変換に不変にできるので、同じ図形に対しては同じ値をとることになる。したがって、例えば、特徴ベクトル間のユークリッド距離を使うことにより、同一の形と他の形とを識別できるはずである。しかし実際には、輪郭点列を抽出する際の量子化誤差などの影響により同じ形に対する係数も微妙に異なってくる。また、木の葉のように同じ種類の木から取ったものでも形そのものが微妙に変形している場合もある。そこで、Bayes の識別方式を採用することにする。Bayes の識別方式は、各クラス内の特徴ベクトルの真の確率密度分布がわかっている場

合には、誤識別率を最小にすることが知られている。しかし、実際には真の分布を完全に知ることは出来ないのので、正規分布を仮定することが多い。クラス C_k の平均特徴ベクトルを μ_k 、共分散行列を Σ_k 、先験確率を $P(C_k)$ とすると、正規分布を仮定した Bayes の識別では、特徴ベクトル v は、事後確率

$$p(C_k|v) = -\frac{1}{2}(v - \mu_k)^T \Sigma_k^{-1} (v - \mu_k) - \frac{1}{2} \ln |\Sigma_k| + \ln P(C_k) \quad (2.19)$$

が最大となるようなクラスに識別される。ただし、特徴ベクトルのクラス内変動は標準化誤差に起因するものであり、その値は非常に小さく、クラス内の共分散行列 Σ_k の行列式の値は次元を大きくすると指数的に小さくなる。このため、次元を大きくすると正則でなくなることがある点に注意が必要である。

一方、判別分析を用いて、クラス内の変動を吸収しクラス間の分離を最大にする空間へ特徴ベクトル v を線形写像して w を求め、 w の空間(判別空間)上で識別する方法も考えることができる。

2.2 3次元実自己回帰モデル

前述の複素自己回帰モデルを 2 次元の行列型自己回帰モデルで表現し直すことにより、より高次元でのデータ点列の回転に対して、性質のよい振る舞いを示す自己回帰係数を持つ自己回帰モデルを構成することができる[82]。まず、この 2 次元行列型自己回帰モデルを定義し、そこでの知見を元に 3 次元実自己回帰モデルを定義していく。

2.2.1 2次元行列型実自己回帰モデル

ここでは、得られた輪郭点列を複素表現することなく、実数の組として扱いながら前述の複素自己回帰モデルと同様に、その自己回帰係数もしくは PARCOR 係数が輪郭点列の回転に対して不変であるような 2 次元の行列型自己回帰モデルについて考える。

この行列型自己回帰モデルと複素自己回帰モデルは次のような関係を通して繋がっており、独立なものというわけではなく、むしろ数学的には等価なものである。

$$\Leftrightarrow \begin{pmatrix} x & -y \\ y & x \end{pmatrix} = x \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + y \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \quad (2.20)$$

ここで,

$$\mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad (2.21)$$

$$\mathbf{J} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad (2.22)$$

とおけば, $\mathbf{J}^2 = -\mathbf{I}$ を満たす。また, 複素共役に対応する演算は, $\mathbf{J}^T = -\mathbf{J}$ のように行列の転置をとることで実現できる。つまり, 虚数単位 i と同様の性質をこの行列 \mathbf{J} が持っていることがわかる。そこで, 複素数に対応して次のような行列から成る集合 M^2 を定義する。すべての $(x, y)^T \in \mathbb{R}^2$ に対して,

$$M^2 = \left\{ \mathbf{Z} \mid \mathbf{Z} = \begin{pmatrix} x & -y \\ y & x \end{pmatrix} \right\}. \quad (2.23)$$

このとき, 2次元行列型自己回帰モデルは次のように定義することができる。 N 個の輪郭点列

$\{(x_j, y_j)\}_{j=0}^{N-1}$ が与えられたとき,

$$\mathbf{Z}_j = \begin{pmatrix} x_j & -y_j \\ y_j & x_j \end{pmatrix}, \text{ for } j = 0, \dots, N-1 \quad (2.24)$$

として, m 次元行列型自己回帰モデルは, 次のように定義される。

$$\hat{\mathbf{Z}}_j = \sum_{l=1}^m \mathbf{Z}_{j-l} \mathbf{A}_l, \quad (2.25)$$

ただし, $\mathbf{A}_l \in M^2$ であり, 自己回帰係数行列 \mathbf{A}_l は, このときの平均二乗予測誤差

$$\begin{aligned} \varepsilon^2 &= \text{tr} E_j (\mathbf{Z}_j - \hat{\mathbf{Z}}_j)^T (\mathbf{Z}_j - \hat{\mathbf{Z}}_j) \\ &= \text{tr} \sum_{k,l=1}^m \mathbf{A}_k^T R_{kl} \mathbf{A}_l - \text{tr} \sum_{k=1}^m \mathbf{A}_k^T \mathbf{R}_k \\ &\quad - \text{tr} \sum_{k=1}^m \mathbf{R}_k^T \mathbf{A}_k + \text{tr} \mathbf{R}_0 \end{aligned} \quad (2.26)$$

を最小にするものとして与えられる。ただし,

$$\begin{aligned} R_{kl} &= E_j \mathbf{Z}_{j-k}^T \mathbf{Z}_{j-l} \\ &= E_j (x_{j-k} x_{j-l} + y_{j-k} y_{j-l}) \mathbf{I} \\ &\quad + E_j (-x_{j-k} y_{j-l} + y_{j-k} x_{j-l}) \mathbf{J}, \end{aligned} \quad (2.27)$$

$$\begin{aligned} \mathbf{R}_k &= E_j \mathbf{Z}_{j-k}^T \mathbf{Z}_j \\ &= E_j (x_j x_{j-k} + y_j y_{j-k}) \mathbf{I} \\ &\quad + E_j (-x_j y_{j-k} + y_j x_{j-k}) \mathbf{J}, \end{aligned} \quad (2.28)$$

$$\begin{aligned} \mathbf{R}_0 &= E_j (x_j x_j + y_j y_j) \mathbf{I} \\ &\quad + E_j (-x_j y_j + y_j x_j) \mathbf{J}, \end{aligned} \quad (2.29)$$

$$\mathbf{A}_k = A_k^x \mathbf{I} + A_k^y \mathbf{J} \quad (2.30)$$

である。このとき \mathbf{A}_k は次の連立方程式を解くことにより求めることができる。

$$\sum_{l=1}^m R_{kl} \mathbf{A}_l = \mathbf{R}_k. \quad (2.31)$$

まず, 次の諸量を定義する。

$$t_p = E_j (x_j x_{j-p} + y_j y_{j-p}), \quad (2.32)$$

$$s_p = E_j (-x_j y_{j-p} + y_j x_{j-p}). \quad (2.33)$$

このとき, $\alpha = 2(k-1) + i$, $\beta = 2(l-1) + j$ とするとき, $\alpha\beta$ 成分が $(R_{kl})^{ij}$ であるような $2m \times 2m$ の行列 Ξ_m と, α 成分が $(t_k, s_k)^T$ の第 i 成分であるような $2m$ 次元のベクトル Ψ_m および β 成分が $(A_l^x, A_l^y)^T$ の第 j 成分であるような $2m$ 次元のベクトル Λ_m を定義すると, 式 (2.31) は次のように書きなおすことができる。

$$\Xi_m \Lambda_m = \Psi_m, \quad (2.34)$$

ただし,

$$\Xi_m = \begin{pmatrix} t_0 & 0 & \cdots & t_{m-1} & -s_{m-1} \\ 0 & t_0 & \cdots & s_{m-1} & t_{m-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ t_{m-1} & -s_{m-1} & \cdots & t_0 & 0 \\ s_{m-1} & t_{m-1} & \cdots & 0 & t_0 \end{pmatrix}, \quad (2.35)$$

であり,

$$\Lambda_m = \begin{pmatrix} A_1^x \\ A_1^y \\ \vdots \\ A_m^x \\ A_m^y \end{pmatrix}, \quad \Psi_m = \begin{pmatrix} t_1 \\ s_1 \\ \vdots \\ t_m \\ s_m \end{pmatrix} \quad (2.36)$$

である。これを解くことにより自己回帰係数 A_k^a を求めることができる。すなわち,

$$\Lambda_m = \Xi_m^{-1} \Psi_m. \quad (2.37)$$

さてここで, この A_k が M^2 内の回転すなわち輪郭点列の回転変換に対して不変であることを確認しておこう。輪郭点列に対して回転変換が施されたとき,

$$\begin{pmatrix} x_{j-k} \\ y_{j-k} \end{pmatrix} \rightarrow \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_{j-k} \\ y_{j-k} \end{pmatrix}, \quad (2.38)$$

のように変換される。このとき, 各 \mathbf{Z}_{j-k} ($k = 0, 1, \dots, m$) は,

$$\mathbf{Z}_{j-k} \rightarrow \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \mathbf{Z}_{j-k} \quad (k = 0, 1, \dots, m) \quad (2.39)$$

のように変換されるので,

$$\begin{aligned} \mathbf{Z}_{j-k}^T \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}^T \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \mathbf{Z}_{j-l} \\ = \mathbf{Z}_{j-k}^T \mathbf{Z}_{j-l} \end{aligned} \quad (2.40)$$

となる。したがって, 式 (2.27), (2.28), (2.31) から M^2 における回転に対して自己回帰係数行列 \mathbf{A}_k は不変であることがわかる。以上で, 複素自己回帰モデルの行列型自己回帰モデルによる再構成ができたことになる。

2.2.2 3次元実自己回帰モデル

ここでは, 前述の2次元行列型自己回帰モデルのときの経験を元に, 輪郭点列に回転変換が施されたときに, 性質のよい振る舞いを示す自己回帰係数を持つような3次元行列型自己回帰モデルについて考える [82][83] (これらの参考文献中の自己回帰係数に対する変換性についての結果は正しいが, その導出には一部誤りがあるので注意されたい)。

このとき, ポイントとなるのは複素自己回帰モデルのときには $\{1, i\}$ が, 2次元行列型自己回帰モデルのときには $\{\mathbf{I}, \mathbf{J}\}$ が, それぞれの自己回帰モデルを構成する際の基底となっていたことである。この基底の持つ回転変換に対する性質が, 輪郭点列の回転に伴う自己回帰係数の変換性を特徴付けていたということができる。つまり, これに対応するような3次元空間での基底を見つけることであれば回転変換に対して性質の良い変換性を持った自己回帰モデルを構成できるものと期待できる。

さて, 複素数の自然な拡張が, いわゆるハミルトンの4元数

$$\begin{aligned} \sigma^0 &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \sigma^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \\ \sigma^2 &= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma^3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \end{aligned} \quad (2.41)$$

であることはよく知られている。したがって, 自己回帰モデルを構成する空間の基底が $\{1, i\}$ の自然な拡張になっているべきであるという立場に立てば, 3次元空間において, 複素自己回帰モデルの自然な拡張としてのモデルを構成することはできないことになる。

そこで, ここでは, ダミーの4番目の変数を3次元のデータに加えることにより, 4次元化して3次元自己回帰モデルを構成することにする。

まず, N 個の輪郭点列 $\{\mathbf{Z}_j = (x_j, y_j, z_j)^T\}_{j=0}^{N-1}$ が与えられたとき,

$$\tilde{\mathbf{Z}}_j = \begin{pmatrix} t_j \\ \mathbf{Z}_j \end{pmatrix}, \quad \begin{cases} j = 0, \dots, N-1 \\ t_j = 0 \end{cases} \quad (2.42)$$

として,

$$\begin{aligned} \sum_{a=0}^3 \sigma^a \hat{\mathbf{Z}}_j^a &= -i \sum_{k=1}^m \left(\sum_{a=0}^3 \sigma^a \tilde{\mathbf{Z}}_{j-k}^a \right) \left(\sum_{b=0}^3 \sigma^b \mathbf{A}_k^b \right) \\ &= -i \sigma^0 \left(\sum_{k=1}^m \mathbf{Z}_{j-k} \cdot \mathbf{A}_k \right) \\ &\quad + \sum_{a=1}^3 \sigma^a \left(\sum_{k=1}^m \mathbf{Z}_{j-k} \times \mathbf{A}_k \right)^a, \end{aligned} \quad (2.43)$$

ただし, $\mathbf{A}_k^0 = 0$ とする。このとき,

$$\hat{t}_j = -i \sum_{k=1}^m \mathbf{Z}_{j-k} \cdot \mathbf{A}_k, \quad (2.44)$$

$$\hat{\mathbf{Z}}_j = \sum_{k=1}^m \mathbf{Z}_{j-k} \times \mathbf{A}_k, \quad (2.45)$$

のように, 3次元自己回帰モデルを定義することができる。

次に, 自己回帰係数を求めなければならない。それには平均二乗予測誤差 ε^2 を最小にするように自己回帰係数を選ばばよい。そこで,

$$\mathbf{R}_k = E_j \mathbf{Z}_j \times \mathbf{Z}_{j-k}, \quad (2.46)$$

$$R_{kl} = E_j \{ (\mathbf{Z}_{j-k} \cdot \mathbf{Z}_{j-l}) \mathbf{I} + \mathbf{Z}_{j-k} \mathbf{Z}_{j-l}^T - \mathbf{Z}_{j-l} \mathbf{Z}_{j-k}^T \} \quad (2.47)$$

とすれば, 平均二乗予測誤差は,

$$\begin{aligned} \varepsilon^2 &= E_j \left(\sum_{k=1}^m \mathbf{Z}_{j-k} \cdot \mathbf{A}_k \right) \left(\sum_{k=1}^m \mathbf{Z}_{j-k} \cdot \mathbf{A}_k \right) \\ &\quad + E_j \left(\mathbf{Z}_j - \sum_{k=1}^m \mathbf{Z}_{j-k} \times \mathbf{A}_k \right) \cdot \\ &\quad \left(\mathbf{Z}_j - \sum_{l=1}^m \mathbf{Z}_{j-l} \times \mathbf{A}_l \right) \\ &= E_j \left[\mathbf{Z}_j \cdot \mathbf{Z}_j - 2 \sum_{k=1}^m \mathbf{A}_k^T (\mathbf{Z}_j \times \mathbf{Z}_{j-k}) \right. \\ &\quad \left. + \sum_{k,l=1}^m \mathbf{A}_k^T ((\mathbf{Z}_{j-k} \cdot \mathbf{Z}_{j-l}) \mathbf{I} + \mathbf{Z}_{j-k} \mathbf{Z}_{j-l}^T - \mathbf{Z}_{j-l} \mathbf{Z}_{j-k}^T) \mathbf{A}_l \right] \\ &= E_j \mathbf{Z}_j \cdot \mathbf{Z}_j - 2 \sum_{k=1}^m \mathbf{A}_k^T \mathbf{R}_k + \sum_{k,l=1}^m \mathbf{A}_k^T R_{kl} \mathbf{A}_l \end{aligned} \quad (2.48)$$

と表すことができる。これを最小にするという条件の下で次の式を得ることができる。

$$\sum_{l=1}^m R_{kl} \mathbf{A}_l = \mathbf{R}_k. \quad (2.49)$$

そこでまず, 次の諸量を定義する。

$$t_p = E_j \mathbf{Z}_j \cdot \mathbf{Z}_{j-p}, \quad (2.50)$$

$$s_p^1 = E_j (Z_j^2 Z_{j-p}^3 - Z_j^3 Z_{j-p}^2), \quad (2.51)$$

$$s_p^2 = E_j (Z_j^3 Z_{j-p}^1 - Z_j^1 Z_{j-p}^3), \quad (2.52)$$

$$s_p^3 = E_j (Z_j^1 Z_{j-p}^2 - Z_j^2 Z_{j-p}^1). \quad (2.53)$$

これらを用いて, \mathbf{R}_k と R_{kl} は次のように書き直すこと

ができる。

$$\mathbf{R}_k = \begin{pmatrix} s_k^1 \\ s_k^2 \\ s_k^3 \end{pmatrix}, \quad (2.54)$$

$$R_{kl} = \begin{pmatrix} t_{l-k} & s_{l-k}^3 & -s_{l-k}^2 \\ -s_{l-k}^3 & t_{l-k} & s_{l-k}^1 \\ s_{l-k}^2 & -s_{l-k}^1 & t_{l-k} \end{pmatrix}. \quad (2.55)$$

ここで,

$$R_{kl}^T = R_{lk} \quad (2.56)$$

である。

さらに, $\alpha = 3(k-1) + i$, $\beta = 3(l-1) + j$ とするとき, $\alpha\beta$ 成分が $(R_{kl})^{ij}$ であるような $3m \times 3m$ の行列 Ξ_m と, α 成分が $(\mathbf{R}_k)^i$ であるような $3m$ 次元のベクトル Ψ_m および β 成分が $(\mathbf{A}_l)^j$ であるような $3m$ 次元のベクトル Λ_m を定義すると, 式 (2.49) は次のように書きなおすことができる。

$$\Xi_m \Lambda_m = \Psi_m, \quad (2.57)$$

ただし,

$$\Xi_m = \begin{pmatrix} t_0 & 0 & 0 & \cdots & t_{m-1} & s_{m-1}^3 - s_{m-1}^2 \\ 0 & t_0 & 0 & \cdots & -s_{m-1}^3 & t_{m-1} & s_{m-1}^1 \\ 0 & 0 & t_0 & \cdots & s_{m-1}^2 & -s_{m-1}^1 & t_{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ t_{m-1} & -s_{m-1}^3 & s_{m-1}^2 & \cdots & t_0 & 0 & 0 \\ s_{m-1}^3 & t_{m-1} & -s_{m-1}^1 & \cdots & 0 & t_0 & 0 \\ -s_{m-1}^2 & s_{m-1}^1 & t_{m-1} & \cdots & 0 & 0 & t_0 \end{pmatrix}, \quad (2.58)$$

であり,

$$\Lambda_m = \begin{pmatrix} A_1^1 \\ A_1^2 \\ A_1^3 \\ \vdots \\ A_m^1 \\ A_m^2 \\ A_m^3 \end{pmatrix}, \quad \Psi_m = \begin{pmatrix} s_1^1 \\ s_1^2 \\ s_1^3 \\ \vdots \\ s_m^1 \\ s_m^2 \\ s_m^3 \end{pmatrix} \quad (2.59)$$

である。これを解くことにより自己回帰係数 A_k^a を求めることができる。すなわち,

$$\Lambda_m = \Xi_m^{-1} \Psi_m \quad (2.60)$$

である。また、このときの平均二乗予測誤差は次のようになる。

$$\varepsilon(m, \mathbf{A})^2 = t_0 - \Psi_m^T \Lambda_m. \quad (2.61)$$

次に、こうして得られた自己回帰係数が、回転変換に対してベクトルのように振る舞うことを示そう。 \mathbf{Z} が、回転変換により \mathbf{MZ} (\mathbf{M} は 3 次元の回転行列を表している) に写されるとき、式 (2.47) より R_{kl} は、 $\mathbf{MR}_{kl}\mathbf{M}^T$ に写される。また、 \mathbf{R}_k は、軸性ベクトルなので回転変換の下では \mathbf{MR}_k のように変換する。このとき、式 (2.49) より

$$\sum_{l=1}^m \mathbf{MR}_{kl}\mathbf{M}^T \mathbf{A}_l = \sum_{l=1}^m \mathbf{MR}_l \quad (2.62)$$

が成立するので、回転変換の下で \mathbf{A}_k はベクトルのように変換されなければならない、つまり \mathbf{MA}_k のようにベクトル的に変換されなければならないことがわかる。

2.3 4次元実自己回帰モデル

ここではハミルトンの 4 元数を基底に持つ空間上に自己回帰モデルを構築することで、4 次元行列型自己回帰モデルについて考察する [83](この参考文献中の自己回帰係数に対する変換性についての結果は正しいが、その導出には一部誤りがあるので注意されたい)。ハミルトンの 4 元数は $SU(2)$ 、すなわち 2 次元複素空間での回転群の基底になっているので、この空間での回転は $SU(2)$ の元による随伴変換で表すことができる。

ハミルトンの 4 元数 $\{\sigma^\alpha\}$ は、積について閉じているので(定数倍は無視する)、これを基底として 4 次元行列型自己回帰モデルを構成することにする。まず、次のような線形空間 \tilde{M}^2 を定義する。すべての $(x, y, z, w)^T \in \mathbb{R}^4$ に対して、

$$\tilde{M}^2 = \left\{ \mathbf{Z} \mid \mathbf{Z} = \begin{pmatrix} x - iw & -iy - z \\ -iy + z & x + iw \end{pmatrix} \right\} \quad (2.63)$$

このとき、4 次元行列型自己回帰モデルは次のようなものとなる。 N 個の輪郭点列 $\{(x_j, y_j, z_j, w_j)\}_{j=0}^{N-1}$ が与え

られたとき、

$$\mathbf{Z}_j = \begin{pmatrix} x_j - iw_j & -iy_j - z_j \\ -iy_j + z_j & x_j + iw_j \end{pmatrix}, \text{ for } j = 0, \dots, N-1 \quad (2.64)$$

として、

$$\hat{\mathbf{Z}}_j = \sum_{k=1}^m \mathbf{Z}_{j-k} \mathbf{A}_k. \quad (2.65)$$

ただし、 $\mathbf{A}_k \in \tilde{M}^2$ であり、このときの自己回帰係数 $\{\mathbf{A}_k\}$ は、次の平均二乗予測誤差 ε^2 を最小にするものとして求めることができる。

$$\begin{aligned} \varepsilon^2 &= \text{tr} E_j (\mathbf{Z}_j - \hat{\mathbf{Z}}_j)^\dagger (\mathbf{Z}_j - \hat{\mathbf{Z}}_j) \\ &= \text{tr} \sum_{k,l=1}^m \mathbf{A}_k^\dagger R_{kl} \mathbf{A}_l - \text{tr} \sum_{k=1}^m \mathbf{A}_k^\dagger \mathbf{R}_k \\ &\quad - \text{tr} \sum_{k=1}^m \mathbf{R}_k^\dagger \mathbf{A}_k + \text{tr} \mathbf{R}_0, \end{aligned} \quad (2.66)$$

ただし、

$$R_{kl} = E_j \mathbf{Z}_{j-k}^\dagger \mathbf{Z}_{j-l}, \quad (2.67)$$

$$\mathbf{R}_k = E_j \mathbf{Z}_{j-k}^\dagger \mathbf{Z}_j, \quad (2.68)$$

$$\mathbf{R}_0 = E_j \mathbf{Z}_j^\dagger \mathbf{Z}_j, \quad (2.69)$$

$$\mathbf{A}_k = \begin{pmatrix} A_k^x - iA_k^w & -iA_k^y - A_k^z \\ -iA_k^y + A_k^z & A_k^x + iA_k^w \end{pmatrix}. \quad (2.70)$$

である。このとき、自己回帰係数行列は、

$$\mathbf{R}_k = \sum_{l=1}^m R_{kl} \mathbf{A}_l, \quad (2.71)$$

を、これまでと同様にして解くことにより得ることが出来る。また、ここで次のような諸量を定義すれば、

$$t_p = E_j (x_{j-p} x_j + y_{j-p} y_j + z_{j-p} z_j + w_{j-p} w_j), \quad (2.72)$$

$$s_p^1 = E_j (x_{j-k} w_j - w_{j-k} x_j), \quad (2.73)$$

$$s_p^2 = E_j (y_{j-k} z_j - z_{j-k} y_j), \quad (2.74)$$

$$s_p^3 = E_j (x_{j-k} z_j - z_{j-k} x_j), \quad (2.75)$$

$$s_p^4 = E_j (y_{j-k} w_j - w_{j-k} y_j), \quad (2.76)$$

$$s_p^5 = E_j (x_{j-k} y_j - y_{j-k} x_j), \quad (2.77)$$

$$s_p^6 = E_j (z_{j-k} w_j - w_{j-k} z_j), \quad (2.78)$$

$R_{kl}, \mathbf{R}_k, \mathbf{R}_0$ は、以下のように書き直すことができる。

$$R_{kl} = \begin{pmatrix} t_{k-l} - i(s_{k-l}^1 - s_{k-l}^2) \\ s_{k-l}^3 + s_{k-l}^4 - i(s_{k-l}^5 - s_{k-l}^6) \\ -(s_{k-l}^3 + s_{k-l}^4) - i(s_{k-l}^5 - s_{k-l}^6) \\ t_k + i(s_{k-l}^1 - s_{k-l}^2) \end{pmatrix}, \quad (2.79)$$

$$\mathbf{R}_k = \begin{pmatrix} t_k - i(s_k^1 - s_k^2) & -(s_k^3 + s_k^4) - i(s_k^5 - s_k^6) \\ s_k^3 + s_k^4 - i(s_k^5 - s_k^6) & t_k + i(s_k^1 - s_k^2) \end{pmatrix}, \quad (2.80)$$

$$\mathbf{R}_0 = \begin{pmatrix} t_0 & 0 \\ 0 & t_0 \end{pmatrix}. \quad (2.81)$$

ところで、この自己回帰係数は輪郭点列の回転に際して一般には不変ではない。なぜなら、この空間上での回転は $SU(2)$ の元 G を使って、

$$G\mathbf{Z}_j G^\dagger, \quad (2.82)$$

と書けるが、たとえば G として σ^2 により生成される回転

$$G_2 = \exp\left\{\frac{\theta}{2}\sigma^2\right\} = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad (2.83)$$

を考えてみると、この回転変換の下で $E_j \mathbf{Z}_{j-k} \mathbf{Z}_j$ は、

$$\begin{aligned} & G_2 E_j \mathbf{Z}_{j-k} \mathbf{Z}_j G_2^\dagger \\ &= \begin{pmatrix} t_k - i(s_k^1 - s_k^2) \cos\theta + i(s_k^5 - s_k^6) \sin\theta \\ s_k^3 + s_k^4 - i(s_k^5 - s_k^6) \cos\theta - i(s_k^1 - s_k^2) \sin\theta \\ -(s_k^3 + s_k^4) - i(s_k^5 - s_k^6) \cos\theta - i(s_k^1 - s_k^2) \sin\theta \\ t_k + i(s_k^1 - s_k^2) \cos\theta - i(s_k^5 - s_k^6) \sin\theta \end{pmatrix}, \end{aligned} \quad (2.84)$$

のように変換される。したがって式 (2.67), (2.68), (2.69), (2.71) より、自己回帰係数行列 $\{\mathbf{A}_k\}$ が回転に対して不変になっていないことがわかる。

しかし、 \tilde{M}^2 での回転が $SU(2)$ の元 G を使って $G\mathbf{Z}G^\dagger$ のように表せるために、輪郭点列の回転を \tilde{M}^2 上での回転として見直すことにより、自己回帰係数行列はテンソルのように振舞うことがわかる。すなわち、

$$G\mathbf{R}_k G^\dagger = G \sum_{l=1}^m R_{kl} \mathbf{A}_l G^\dagger,$$

$$= \sum_{l=1}^m G R_{kl} G^\dagger \mathbf{A}_l G^\dagger, \quad (2.85)$$

$$\mathbf{A}_l \rightarrow G \mathbf{A}_l G^\dagger. \quad (2.86)$$

2.4 K 次元実自己回帰モデル

ここでは、より一般の高次元自己回帰モデルについて考察する。ただし、ここで考察される自己回帰モデルは、すべてこれまで同様に輪郭点列の回転に対して、その自己回帰係数が特徴的な振る舞いを示す、すなわち性質の良い変換性を示すものである。ただ単に高次元へ自己回帰モデルを拡張しただけでは、輪郭点列の回転に対する自己回帰係数の変換性は性質の良いものになるとは限らない。

さて、 K 次元の輪郭点列が与えられたとき $SO(K)$ (K 次元回転群) の基底もしくは $SO(K)$ の被覆群の基底がわかっているならば、 K 次元自己回帰モデルは、3 次元自己回帰モデルと同様な手続きで構成することができる。

しかし、ここでは少し違ったやり方で K 次元自己回帰モデルを構成してみる。それは、 $SU(K)$ (特殊ユニタリ群、すなわち K 次元複素空間での回転群) の Lie 代数の満たす交換関係を利用したものである (ちなみに、 $SU(2)$ の Lie 代数としてパウリ行列が登場する)。そこで、この方法で構成される自己回帰モデルを $SU(K)$ 自己回帰モデルと呼ぶことにする。

まず、 $SU(K)$ の Lie 代数 $\{L^a\}$ の張る線形空間 M^{K^2-1} を定義する。すべての \mathbb{R}^{K^2-1} の元 $\mathbf{V} = (V^1, \dots, V^{K^2-1})^T$ に対して、

$$M^{K^2-1} = \left\{ \mathbf{X} \mid \mathbf{X} = \sum_{a=1}^{K^2-1} V^a L^a \right\}. \quad (2.87)$$

$SU(K)$ の Lie 代数 $\{L^a\}_{a=1}^{K^2-1}$ は次の交換関係を満たす。

$$[L^a, L^b] = i \sum_{c=1}^{K^2-1} f^{abc} L^c, \quad (2.88)$$

ただし、 $[A, B] = AB - BA$ であり、 f^{abc} は $SU(K)$ の構造定数と呼ばれるものである。この構造定数 f^{abc} は、 $SU(K)$ が半単純 Lie 群であるために、添え字 a, b, c の入れ換えに対して完全反対称である。また、この構造定数を与えることにより $SU(K)$ の構造を完全に決定

できることが知られている。 $(K^2 - 1)$ 次元の輪郭点列 $\left\{ \mathbf{Z}_j \mid \mathbf{Z}_j = \left(Z_j^1, \dots, Z_j^{K^2-1} \right)^T \right\}_{j=0}^{N-1}$ が与えられたとき, m 次の $SU(K)$ 自己回帰モデルは線形空間 M^{K^2-1} 上に次のように構成することができる。

$$\begin{aligned} \sum_{a=1}^{K^2-1} L^a \hat{Z}_j^a &= -i \sum_{k=1}^m \left[\sum_{a=1}^{K^2-1} L^a Z_{j-k}^a, \sum_{b=1}^{K^2-1} L^b A_k^b \right] \\ &= \sum_{k=1}^m \sum_{a,b,c=1}^{K^2-1} f^{abc} L^c Z_{j-k}^a A_k^b, \end{aligned} \quad (2.89)$$

すなわち,

$$\hat{Z}_j^a = \sum_{k=1}^m \sum_{b,c=1}^{K^2-1} f^{abc} Z_{j-k}^b A_k^c. \quad (2.90)$$

このとき自己回帰係数 \mathbf{A}_k は, 次の平均二乗予測誤差を最小にするように定められる。

$$\begin{aligned} \varepsilon(m)^2 &= E_j \sum_{a=1}^{K^2-1} \left(Z_j^a - \sum_{k=1}^m \sum_{b,c=1}^{K^2-1} f^{abc} Z_{j-k}^b A_k^c \right)^2 \\ &= E_j \mathbf{Z}_j \cdot \mathbf{Z}_j - 2 \sum_{k=1}^m \mathbf{A}_k \cdot \mathbf{R}_k \\ &\quad + \sum_{k,l=1}^m \mathbf{A}_k^T R_{kl} \mathbf{A}_l, \end{aligned} \quad (2.91)$$

ただし,

$$(\mathbf{R}_k)^a = E_j \sum_{b,c=1}^{K^2-1} f^{abc} Z_j^b Z_{j-k}^c, \quad (2.92)$$

$$(R_{kl})^{ab} = E_j \sum_{c,d,e=1}^{K^2-1} f^{acd} f^{bce} Z_{j-k}^d Z_{j-l}^e, \quad (2.93)$$

$$(R_{kl})^{ab} = (R_{lk})^{ba}. \quad (2.94)$$

このとき, 自己回帰係数は,

$$\sum_{l=1}^m R_{kl} \mathbf{A}_l = \mathbf{R}_k, \quad (2.95)$$

を解くことにより得ることができる。

この自己回帰係数の $SU(K)$ 回転に対する性質は, \mathbf{R}_k の変換性に依存して決定される。つまり, もし \mathbf{R}_k が $SU(K)$ 回転の下でベクトルとして振舞えば, 得られる自己回帰係数もベクトルとして振舞う。

2.5 本章のまとめ

本章では, 図形の形(輪郭)などの平面上の曲線の認識のための平面曲線の自然な記述(不変特徴抽出)法として, 複素自己回帰モデルに基づく手法を紹介した。さらに, この複素自己回帰モデルを行列形式で書き直すことで, 自己回帰係数が輪郭点列の回転変換に対して性質の良い変換性を示すような高次元の自己回帰モデルを構成することができることを示した。特に, 3次元自己回帰モデルは, 3次元空間での輪郭点列の回転変換に対してその自己回帰係数がベクトルのように振舞うことを確かめた。この性質を利用して, 物体の3次元空間での回転の前後での輪郭点列を与えることにより, その物体の3次元空間での回転軸および回転角を推定するために利用することもできるだろう。

さまざまなデータから, 認識に必要な特徴を, ある特定の変換の下でのその特徴量の変換性に注目して抽出することはパターン認識の基本であり, どういう変換に注目するかが問題を容易にもしたり難しくしたりもする。安直に低次元空間のデータ解析手法を高次元に当てはめることは複素自己回帰モデルの必要性を説明したところでみたようにあまり良いことばかりとは言えないだろう。そして, このことが複素数へのモデルの拡張およびより高次元空間へのモデル拡張を行うことの重要性を示しているとも言える。

第 3 章 多層型高次元ニューラルネットワーク

本章では、多層型の高次元ニューラルネットワークの研究の現状について述べる。

3.1 多層型複素ニューラルネットワーク

まず、多層型の複素ニューラルネットワークについて述べる。

3.1.1 モデル

今までに提案された多層型複素ニューラルネットワークには、いくつかの種類がある。本項では、それらを概観する。

3.1.1.1 Widrow モデル

多層型、相互結合型にかかわらず、最初に、ニューラルネットワークを複素数に拡張したのは、Widrow ら [90] である。彼らが対象としたのは、1つの複素ニューロンである(図 3.1 参照)。

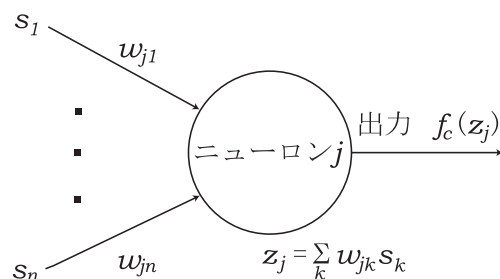


図 3.1 複素ニューロン

$s_1, \dots, s_n, w_{j1}, \dots, w_{jn}, f_C(z_j)$ は、すべて複素数。

すなわち、入力信号、重みおよび出力信号はすべて複素数である。ニューロンモデルでは、通常、閾値が想定されて

いるが、本モデルでは使われていない(つまり、閾値はゼロ)。複素ニューロン j の内部ポテンシャル z_j は、

$$z_j = \sum_k w_{jk} s_k \quad (3.1)$$

と定義される。ここで、 w_{jk} は複素ニューロン j と複素ニューロン k との間の重みを表す複素数、 s_k は複素ニューロン k から出力された複素ニューロン j への入力信号を表す複素数である。出力信号 $f_C(z_j)$ は次のように定義される：

$$f_C(z_j) = z_j, \quad z_j = x_j + iy_j, \quad i \stackrel{\text{def}}{=} \sqrt{-1}. \quad (3.2)$$

この関数は正則であるが、有界ではない。また、非線形性を含んでいない。

Widrow らは、この複素ニューロンに対する学習アルゴリズムとして、複素 LMS アルゴリズムというものを導いた。これは Widrow-Hoff LMS (Least-Mean-Square) アルゴリズム [31,88,89] の複素数版である。LMS アルゴリズムは、実ニューロン(通常の実数タイプのニューロン)に対して適用されるものであり、基本的に最急降下法に基づいている。つまり、重みを逐次的に修正し、学習誤差を徐々に小さくしてゆくことによって所望の出力値を得るものである。重みの変化規則は次のとおりである：

$$w_{jk}(n+1) = w_{jk}(n) + 2\mu \varepsilon s_k. \quad (3.3)$$

ここで、 n は時刻、 μ は学習率、 $\varepsilon = d - y$ は学習誤差を表す。 d は教師信号、 y は実際の出力値である。 $w_{jk}(n+1)$ 、 $w_{jk}(n)$ 、 μ 、 ε 、 s_k 、 d 、 y はすべて実数である。

これに対して、複素 LMS アルゴリズムも、基本的に最急降下法に基づいて導かれる。その重みの変化規則は次のとおりである：

$$w_{jk}(n+1) = w_{jk}(n) + 2\mu \varepsilon \bar{s}_k. \quad (3.4)$$

ここで、 μ は学習率を表す実数、 $w_{jk}(n+1)$ および $w_{jk}(n)$ は重みを表す複素数、 ε は学習誤差を表す複素数、 s_k は入力信号を表す複素数、 \bar{s}_k は複素数 s_k の共役複素数である。

3.1.1.2 Kim-Leung モデル

ここで述べるモデルは、Kim ら (1990 年) と Leung ら (1991 年) が独立に提案したものである。

まず, Kim モデルについて述べる [26]. 彼らが対象とした複素ニューロンを図 3.2 に示す。

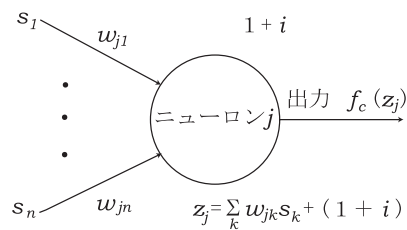


図 3.2 複素ニューロン

$s_1, \dots, s_n, w_{j1}, \dots, w_{jn}, f_C(z_j)$ は, すべて複素数。

入力信号, 重み, 閾値および出力信号はすべて複素数である。ただし, 閾値は $1+i$ に固定されている。ニューロンの出力関数 $f_C: C \rightarrow C$ は次のとおりである (C は複素数全体の集合):

$$f_C(z) = \frac{1}{1 + \exp(-z)}, \quad z = x + iy. \quad (3.5)$$

この関数は正則であるが, 有界ではない。また, 非線形性を含んでいる。

彼らは, このような複素ニューロンから構成される 3 層のニューラルネットワーク (図 3.3 参照) を対象として, 最急降下法を使って学習アルゴリズムを導いた。ただし, 具体的な学習アルゴリズムは記載されていない。

また, 学習の対象は重みだけであり, 閾値は $1+i$ に固定されていて, 学習の対象にはなっていない。学習誤差は $\sum_{k=1}^N |d_k - y_k|^2$, ただし, d_k は出力ニューロン k の教師信号, y_k は出力ニューロン k の実際の出力値である。

そして, 提案した複素ニューラルネットワークの有効性を確かめるために, 排他的論理和 (XOR) を使って計算機実験を行なった結果, 学習誤差は 10% にまでなつたと報告している。実験には, 入力ニューロン 2 個, 中間ニューロン 2 個, 出力ニューロン 1 個の 2-2-1 ネットワークが用いられた。

次に, Leung モデルについて述べる [36]。Kim モデルとの相違点のみ述べる。異なる点は次のとおりである:

1. 閾値も学習の対象にしている,
2. 一般の n 層のネットワークに対する学習アルゴリズムを導いている,

3. 出力関数 (式 (3.5)) に特異点があるため収束しないということにも触れていて, その回避策として, 特異点を避けるために入力パターンをスケールリングすることによって, ある領域に押し込めているという点である (具体的なスケールリング方法については明示されていない)。

そして, 提案した複素ニューラルネットワークの有効性を確かめるために, 簡単な学習パターンを使って計算機実験を行なった結果, 確かに収束することを確認している。

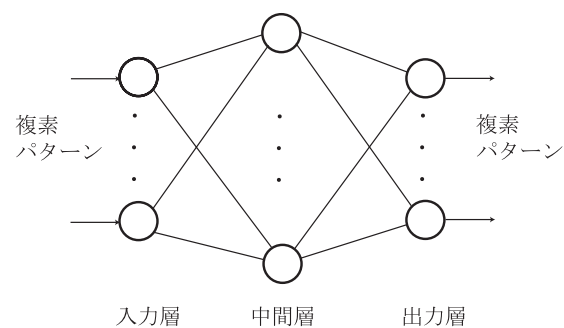


図 3.3 3層複素ニューラルネットワーク

3.1.1.3 新田・Benvenuto モデル

ここで述べるモデルは, 新田ら (1991 年) と Benvenuto ら (1992 年)[11] が独立に提案したものである。

まず, 新田らが提案したモデルについて, 文献 [43,48,50,52,66,67,69] に従って述べる。

新田らは, 従来のバックプロパゲーション学習アルゴリズム (実 B P)[73] の複素数版である複素バックプロパゲーション学習アルゴリズム (複素 B P) を提案した。複素 B P は, ニューロン間の結合の重みおよび各ニューロンが持つ閾値がすべて複素数である階層型ニューラルネットワークに適用される。

以下では, まず, 実 B P の理論的な基礎となっている学習識別理論 [5] を複素数に拡張することにより複素学習識別理論を用意し, 複素 B P の理論的基礎とする。複素 B P はこの複素学習識別理論から導出され, 学習収束性についても, これにより保証される。

(a) 複素学習識別モデル

実 B P の理論的基礎となっている学習識別理論 [5] におけるモデルに複素数要素を加味したモデル (複素学習識別モデル) を考える。まず、2つの複素パターン情報源を考え、情報源 1 から複素パターン $x \in C^n$ 、情報源 2 から複素パターン $y \in C^m$ が未知なる同時確率 $P(x, y)$ で発生するものとする ($x \in C^n, y \in C^m$ であることに注意)。ここで、 $\{(x, y)\}$ は階層型ニューラルネットワークにおける学習パターンに対応するものであり、有限個とする。学習の目的は、情報源 1 からの複素パターン x が与えられたときに、情報源 2 からの複素パターン y を推定できるようになることである。推定を与える関数を $z(w, x) : R^p \times C^n \rightarrow C^m$ とする (R は実数の全体をあらわす)。 $w \in R^p$ は階層型ニューラルネットワークにおける結合の重みと各ニューロンの閾値のそれぞれの実部と虚部をすべてまとめたものに対応し、推定関数 z の出力値は階層型ニューラルネットワークが出力する出力パターンに対応する。また、 $r(y', y) : C^m \times C^m \rightarrow R^+$ (R^+ は 0 以上の実数の全体をあらわす) を、複素パターン y を複素パターン y' で推定したときの損失関数とし、 $R(w) : R^p \rightarrow R^1$ をパラメータ w を用いたときの平均損失とする。すなわち、

$$R(w) \stackrel{\text{def}}{=} \sum_x \sum_y r(z(w, x), y) P(x, y) \quad (3.6)$$

と定義する。 $R(w)$ は階層型ニューラルネットワークが出力する出力パターンと出力学習パターン (教師パターン) との誤差に対応するものであり、この値が小さいほど良い推定であることになる。

(b) 学習収束定理

複素学習識別モデルにおける学習アルゴリズムを示し、その収束性を確認する。それは確率的降下法 [5] の複素数版である。

離散時間パラメータ n を導入し、 (x_n, y_n) を時刻 n において発生する複素パターンとし、パラメータ w を

$$w_{n+1} = w_n + \Delta w_n \quad (3.7)$$

に従って修正するものとする。 $R_n(w_n)$ を時刻 n における平均損失とする。また、 $R(w)$ を最小または極小ならしめるパラメータ w を最適なパラメータと定義する。

このとき、次の定理が成り立つ。証明については [43, 66, 69] を参照されたい。

定理 1. 十分小さな正数 ε と正定値行列 A に対して、パラメータ w を修正量

$$\Delta w_n = -\varepsilon A \nabla r(z(w_n, x_n), y_n) \quad n = 0, 1, \dots \quad (3.8)$$

に従って修正を行なうことにより、平均損失 R に関して最適なパラメータ w がいくらかでも良い精度で得られる (∇ は w に関する gradient)。

(c) 複素 B P ネットワーク

次に、対象とする複素 B P ネットワークについて述べる。

複素ニューロンは図 3.4 に示すとおりである。入力信号、重み、閾値および出力信号はすべて複素数とする。複素ニューロン j の内部ポテンシャル z_j は、

$$z_j = \sum_k w_{jk} s_k + t_j \quad (3.9)$$

と定義する。ここで、 w_{jk} は複素ニューロン j と複素ニューロン k との間の重みを表す複素数、 s_k は複素ニューロン k から出力された複素ニューロン j への入力信号を表す複素数、 t_j は複素ニューロン j の閾値を表す複素数である。出力信号 $f_C(z_j)$ は次のように定義される：

$$f_C(z_j) = f_R(x_j) + i f_R(y_j), \quad z_j = x_j + i y_j, \quad (3.10)$$

ただし、 $f_R(u) = 1/(1 + \exp(-u))$ 、 $i = \sqrt{-1}$ である。すなわち、複素ニューロンの出力値の実部および虚部は、それぞれ、複素ニューロンへの入力の総和 z_j の実部、虚部に関するシグモイド関数であり、明らかに、 $0 \leq \text{Re}[f_C], \text{Im}[f_C] \leq 1$ 、 $|f_C(z_j)| \leq \sqrt{2}$ を満たす ($\text{Re}[z]$ 、 $\text{Im}[z]$ はそれぞれ複素数 z の実部、虚部)。

次に、上で定式化した複素ニューロンを使って、複素 B P ネットワークを定式化する。

簡単のために、3層のネットワークを例にとる (図 3.3)。 w_{ji} を入力複素ニューロン i と中間複素ニューロン j との間の結合の重み、 v_{kj} を中間複素ニューロン j と出力複素ニューロン k との間の結合の重み、 θ_j を中間複素ニューロン j が持つ閾値、 γ_k を出力複素ニューロン k が持つ閾値とする。また、 I_i 、 H_j 、 O_k を、それぞれ、入力複素ニューロン i 、中間複素ニューロン j 、出力複素ニューロン k の

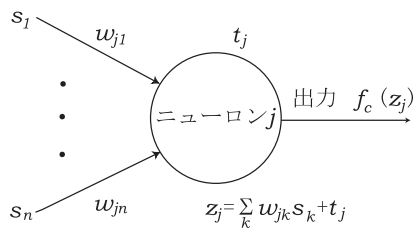


図 3.4 複素ニューロン

$s_1, \dots, s_n, w_{j1}, \dots, w_{jn}, t_j, f_C(z_j)$ は、すべて複素数.

出力値とし、 U_j, S_k をそれぞれ中間複素ニューロン j , 出力複素ニューロン k の内部ポテンシャルとする。すなわち、 $U_j = \sum_i w_{ji} I_i + \theta_j$, $S_k = \sum_j v_{kj} H_j + \gamma_k$, $H_j = f_C(U_j)$, $O_k = f_C(S_k)$ と定義する。また、出力複素ニューロン k に対する教師パターン (出力学習パターン) T_k と出力複素ニューロン k の出力値 O_k との誤差を δ^k とする。つまり、 $\delta^k = T_k - O_k$ と定義する。最後に、複素パターン p に対する 2 乗誤差を $E_p = (1/2) \sum_k |T_k - O_k|^2$ と定義する。

(d) 複素 BP 学習アルゴリズムの導出

複素学習識別理論を、先に定義した複素 BP ネットワークに適用することにより、複素 BP 学習アルゴリズムを導出する。

学習率 $\varepsilon > 0$ を十分小さくとり、正定値行列 A を単位行列として、定理 1 を適用すると、結合の重みおよび閾値の修正量は、

$$\Delta v_{kj} = -\varepsilon \frac{\partial E_p}{\partial \text{Re}[v_{kj}]} - i\varepsilon \frac{\partial E_p}{\partial \text{Im}[v_{kj}]}, \quad (3.11)$$

$$\Delta \gamma_k = -\varepsilon \frac{\partial E_p}{\partial \text{Re}[\gamma_k]} - i\varepsilon \frac{\partial E_p}{\partial \text{Im}[\gamma_k]}, \quad (3.12)$$

$$\Delta w_{ji} = -\varepsilon \frac{\partial E_p}{\partial \text{Re}[w_{ji}]} - i\varepsilon \frac{\partial E_p}{\partial \text{Im}[w_{ji}]}, \quad (3.13)$$

$$\Delta \theta_j = -\varepsilon \frac{\partial E_p}{\partial \text{Re}[\theta_j]} - i\varepsilon \frac{\partial E_p}{\partial \text{Im}[\theta_j]} \quad (3.14)$$

とすれば良いことになり、結局、

$$\Delta v_{kj} = \overline{H_j} \Delta \gamma_k, \quad (3.15)$$

$$\Delta \gamma_k = \varepsilon \left\{ \text{Re}[\delta^k] (1 - \text{Re}[O_k]) \text{Re}[O_k] + i \text{Im}[\delta^k] (1 - \text{Im}[O_k]) \text{Im}[O_k] \right\}, \quad (3.16)$$

$$\Delta w_{ji} = \overline{I_i} \Delta \theta_j, \quad (3.17)$$

$$\Delta \theta_j = \varepsilon \left[(1 - \text{Re}[H_j]) \text{Re}[H_j] \sum_k \left\{ \text{Re}[\delta^k] (1 - \text{Re}[O_k]) \text{Re}[O_k] \text{Re}[v_{kj}] + \text{Im}[\delta^k] (1 - \text{Im}[O_k]) \text{Im}[O_k] \text{Im}[v_{kj}] \right\} - i (1 - \text{Im}[H_j]) \text{Im}[H_j] \sum_k \left\{ \text{Re}[\delta^k] (1 - \text{Re}[O_k]) \text{Re}[O_k] \text{Im}[v_{kj}] - \text{Im}[\delta^k] (1 - \text{Im}[O_k]) \text{Im}[O_k] \text{Re}[v_{kj}] \right\} \right] \quad (3.18)$$

が得られる。ここで、 ε は複素数 z の共役複素数である。

参考のために、実 BP におけるパラメータの修正量を次に記しておく。式 (3.19) - (3.22) と式 (3.15) - (3.18) は互いに似通った形をしていることがわかる。ただし、式 (3.19) - (3.22) においては、 $\delta^k, I_i, H_j, O_k, v_{kj}, \gamma_k, w_{ji}, \theta_j$ はすべて実数である。

$$\Delta v_{kj} = H_j \Delta \gamma_k, \quad (3.19)$$

$$\Delta \gamma_k = \varepsilon \delta^k (1 - O_k) O_k, \quad (3.20)$$

$$\Delta w_{ji} = I_i \Delta \theta_j, \quad (3.21)$$

$$\Delta \theta_j = \varepsilon (1 - H_j) H_j \sum_k \{ \delta^k (1 - O_k) O_k v_{kj} \}. \quad (3.22)$$

(e) 出力関数について

新田は、ニューロンの出力関数に関して、文献 [66] において次のように述べている:

『我々は、当初、Kim ら [26] とは独立に、出力関数に彼らと同じ $f_C(z) = 1/(1 + \exp(-z))$, ただし、 $z = x + iy$ を使った定式化を行なった。この定式化が最も自然であると思われたし、この関数は正則関数であることから、興味深い結果が出てくるであろうと予想したからである。しかし、実験の結果、我々の行なった実験の範囲内では収束しなかった (発散した)。そこで、出力関数が有界でなかったことに着目し、正則ではないが、有界な関数 (式 (3.10)) を用いたという経緯があった。正則かつ有界な出力関数を用いることを考えると、リュービルの定理 (たとえば、[24]) により、全複素平面 C で正則かつ有界な

関数は定数だけとなる。よって、出力関数を有界にしようとする、どうしても正則性を破らざるを得ない。そのひとつの結果が、式 (3.10) で与えた正則ではないが有界な関数である。ちなみに、有界でない出力関数を持つ Widrow ら [90] による定式化に従って実験を行なったが、我々の行なった範囲では学習はすべて発散した。』

すなわち、新田らの提案した複素 BP は、Widrow モデルおよび Kim-Leung モデルの定式化が持っていた収束性に関する問題点を解消した形のものとなっているのである。

次に、Benvenuto らが提案したモデル [11] について述べるが、それは新田らが提案したモデルと同じものである。ただし、モデルの考察に関しては次の視点が欠けていた：

1. 新田らが複素学習識別理論の枠組で行なった学習収束性に関する議論は行なわれていない。
2. 学習アルゴリズムが導かれているが、計算機実験は行なわれていない。

ちなみに、Benvenuto らも、先に『(e) 出力関数について』において述べたことにも触れている。

3.1.1.4 Georgiou モデル

1992 年に Georgiou らが提案したモデルについて述べる [17]。

彼らは、Kim-Leung モデルの不備を指摘した上で、それを改善した複素ニューラルネットワークモデルを提案した。すなわち、出力関数だけが Kim-Leung モデルとは異なったモデルを提案した。Kim-Leung モデルの不備とは出力関数の定式化であり、彼らは、Kim-Leung モデルにおける出力関数は特異点を持っており、収束性に問題があることを指摘した（このことについては、3.1.1.3 項においても述べたように、新田も指摘していることである）。具体的には、Kim-Leung モデルにおける出力関数は、領域 $\{0 \pm i(2n+1)\pi \mid n \in \mathbb{Z}\}$ の任意の値に近づくと、その絶対値が無限大に発散してしまうのである（ \mathbb{Z} は

整数全体からなる集合）。Leung ら [36] は、この領域を避けるために、入力値のスケーリングを行なったが、最急降下法を使っている限りは、重みの値を制御することは不可能であるから、この方法は不適切である。

以下、次のような出力関数を考える。

$$f(z) = u(x, y) + iv(x, y), \quad z = x + iy. \quad (3.23)$$

まず、彼らは適切でない出力関数の条件を調べた。その結果を以下に示す。証明については、原論文を参照されたい。

命題 1. 全複素平面上で正則な関数は、出力関数としては不適切である。 □

リュービルの定理（たとえば、[24]）により、全複素平面上で正則な関数は定数しか存在しないから、この命題は明らかである。

命題 2. もし、

$$\frac{\partial u}{\partial x} \frac{\partial v}{\partial y} \equiv \frac{\partial v}{\partial x} \frac{\partial u}{\partial y} \quad (3.24)$$

ならば、 f は出力関数として不適切である。 □

命題 3. もし、

$$u \equiv v + k \quad (3.25)$$

ならば、 f は出力関数として不適切である。ただし、 $k \in \mathbb{C}$ は定数である。 □

命題 3 から、不適切な出力関数のクラスがいかに広いものであるかがわかる。

次に、適切な出力関数であるための条件を調べた。その結果を以下に示す。

命題 4. 出力関数 f が適切であるためには、次の条件を満たす必要がある。

1. 偏導関数 $\partial u/\partial x, \partial u/\partial y, \partial v/\partial x, \partial v/\partial y$ が存在し、かつ、それらが有界である。
2. $f(z)$ が x と y に関して非線形である（線形な f では、たとえば、非線形分離問題に対応できない）。
3. $f(z)$ は有界である。このためには、 u と v が有界であることが必要である。

4. $f(z)$ は全複素平面上で正則ではない。
 5. $\frac{\partial u}{\partial x} \frac{\partial v}{\partial y} \equiv \frac{\partial v}{\partial x} \frac{\partial u}{\partial y}$ が成り立たない。 □

以上のように、不適切な出力関数と適切な出力関数に関する考察を行なった上で、命題 4 の条件を満たす次の出力関数を提案した：

$$f(z) = \frac{z}{c + \frac{1}{r}|z|}, \quad (3.26)$$

ただし、 c と r は正の実数である。 c は $|f(z)|$ を変化させるものである。また、 z と $f(z)$ の位相は等しい。 u, v の偏導関数は次のように定義した。

$$\frac{\partial u}{\partial x} = \begin{cases} \frac{r(y^2 + cr|z|)}{|z|(cr + |z|)^2} & (\text{if } |z| \neq 0) \\ \frac{1}{c} & (\text{if } |z| = 0) \end{cases}, \quad (3.27)$$

$$\frac{\partial u}{\partial y} = \begin{cases} -\frac{rxy}{|z|(cr + |z|)^2} & (\text{if } |z| \neq 0) \\ 0 & (\text{if } |z| = 0) \end{cases}, \quad (3.28)$$

$$\frac{\partial v}{\partial x} = \begin{cases} -\frac{rxy}{|z|(cr + |z|)^2} & (\text{if } |z| \neq 0) \\ 0 & (\text{if } |z| = 0) \end{cases}, \quad (3.29)$$

$$\frac{\partial v}{\partial y} = \begin{cases} \frac{r(x^2 + cr|z|)}{|z|(cr + |z|)^2} & (\text{if } |z| \neq 0) \\ \frac{1}{c} & (\text{if } |z| = 0) \end{cases}. \quad (3.30)$$

最後にいくつかの学習パターンに対して計算機実験を行なったと述べた後、そのうちの 1 つを学習が収束した例として示している。

ちなみに、新田・Benvenuto モデルにおける出力関数 (式 (3.10)) が、命題 4 の条件を満たしていることは容易に確かめることができる。

3.1.1.5 複素一般化ヘブ学習

ここでは、一般化ヘブ学習の複素領域への拡張 [93] について紹介する。まず、一般化ヘブ学習は、Sanger[74] によって提案された入力の相関行列の主成分ベクトルを抽出する学習則である。これは、単一層のニューラルネットワークを用いて、入力の相関行列の主成分ベクトルを逐次的に推定していくものであり、すなわち、主成分分析と等価である。しかし、ニューラルネットワークへの入力としては通常の主成分分析を行う場合と違い、あらかじめ入力の相関行列を計算しておく必要がないという利点がある。また、各ニューロンに対して局所的な演算

のみが要求され、ニューロンの学習が同時に起こるので並列化することが可能である。画像圧縮やテキストチャープ分割などへの応用 [74] が試みられている。

ニューラルネットワークへの入力として、 N 次元の平均が 0 であるような線形空間の元

$$X = (x_1, x_2, \dots, x_N)^T \quad (3.31)$$

をとるとき、ニューラルネットワークの出力として M ($M < N$) 次元の線形空間の元

$$Y = (y_1, y_2, \dots, y_M)^T \quad (3.32)$$

が得られるものとする。このとき Y は $N \times M$ の行列 W 、すなわち、ニューロンの結合荷重行列を用いて次のように関係している。

$$Y = W^* X, \quad (3.33)$$

ただし、 $*$ は共役転置をとることを表している。この結合荷重行列 W を如何に求めるかが重要である。

複素一般化ヘブ学習の場合は、通常的一般化ヘブ学習の複素領域への拡張として、次のように学習則が定義される。結合荷重 W の j 番目の列 W_j は、

$$W_j(n+1) = W_j(n) + \mu(n) \overline{y_j(n)} \cdot [X(n) - y_j(n)W_j(n) - \sum_{i < j} y_i(n)W_i(n)] \quad (3.34)$$

に従って逐次的に更新される。ただし、 $\overline{\cdot}$ は複素共役をとることを意味し、 $y_j(n) = W_j^*(n)X(n)$ であり、 $\mu(n)$ は学習係数である。この学習則を用いた場合に、荷重 W_j に任意の初期値を与えたときにニューラルネットワークの出力が入力の共分散行列 $R_{XX} = E[XX^*]$ の j 番目の規格化された固有ベクトルに収束していくことが示されている [93]。これは、入力データが実数の場合に適應される一般化ヘブ学習則の単純な一般化になっている。

3.1.2 新田・Benvenuto モデルの性質

本項では、3.1.1.3 項において述べた新田・Benvenuto モデルの性質を概観する。

3.1.2.1 決定表面の構造

まず、決定表面の構造について述べる [44, 59, 62, 66]。

新田らは、ネットワークアーキテクチャの観点から、複素BPの基本特性を解析的、実験的に調べ、実BPとの差異を明確にした。主要な結果は次のとおりである。(1) 複素BPネットワークにおける重み係数は、実BPネットワークにおけるそれとは異なり、2次元アフィン変換に関係した制約を持っており、学習は基本的にその制約を保ちつつ行われる。(2) 複素ニューロンの実部の決定表面と虚部の決定表面とは互いに直交しており、決定領域を均等に4つに分割する。3層の複素BPネットワークにおける決定表面は基本的にこの構造を内包しており、中間ニューロンへの総入力十分大きいときに直交する。

以下、順を追ってこれらのことについて述べる。

(a) 実ニューロンの重みパラメータ

まず、実ニューロンの入力の結合重みの基本的な構造について確認しておく。重み $w_k \in \mathbf{R}^1$ ($1 \leq k \leq n$) および閾値 $\theta \in \mathbf{R}^1$ を持つ n 入力実ニューロンを考える。出力関数 $f_R: \mathbf{R}^1 \rightarrow \mathbf{R}^1$ は $f_R(u) = 1/(1 + \exp(-u))$ とする。このとき、入力 $x_k \in \mathbf{R}^1$ ($1 \leq k \leq n$) が与えられると、実ニューロンは $f_R(\sum_{k=1}^n w_k x_k + \theta)$ を出力する。つまり、実ニューロンは実数直線(1次元)上の n 個の点 x_1, \dots, x_n を原点との距離に関して、それぞれ、 w_1, \dots, w_n 倍し、それらを1次元実ベクトルとみなして加えた結果得られるベクトルの示す点を更に θ だけ移動させたと解釈することができる(図3.5)。その後、非線形変換 f_R を施したものを出力値としている。

このように、実ニューロンは基本的に1次元上の点列の移動を司っており、重みパラメータ w_1, \dots, w_n は互いに独立であって、何らの影響も与え合わない。ここで、実ニューロンの出力値は非線形変換 f_R が施されることによって得られるため、実ニューロンの出力値に関して上記の幾何学的解釈が厳密に成り立つのは非線形変換 f_R が線形変換とみなせるとき、すなわち、実ニューロンへの入力の総和の絶対値 $|\sum_{k=1}^n w_k x_k + \theta|$ が小さな値であるときに限ることに注意しなければならない。

(b) 複素ニューロンの重みパラメータ

次に、複素ニューロンの入力の結合重みの基本的な構造について調べる。重み $w_k = w_k^r + iw_k^i \in \mathbf{C}^1$ ($1 \leq k \leq n$) および閾値 $\theta = \theta^r + i\theta^i \in \mathbf{C}^1$ を持つ n 入力複素ニューロンを考える。入力 $x_k + iy_k \in \mathbf{C}^1$ ($1 \leq k \leq n$) が与え

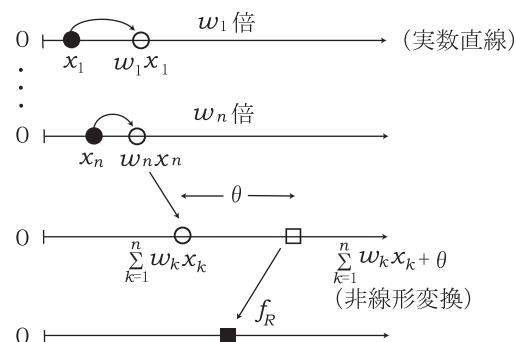


図 3.5 実ニューロンにおける処理のイメージ

られると、この複素ニューロンの出力 $X + iY$ は、

$$\begin{aligned} & X + iY \\ &= f_C \left(\sum_{k=1}^n (w_k^r + iw_k^i)(x_k + iy_k) + (\theta^r + i\theta^i) \right) \\ &= f_R \left(\sum_{k=1}^n (w_k^r x_k - w_k^i y_k) + \theta^r \right) \\ & \quad + i f_R \left(\sum_{k=1}^n (w_k^i x_k + w_k^r y_k) + \theta^i \right) \end{aligned} \quad (3.35)$$

となる。故に、 n 入力複素ニューロンは図3.6に示す2つの n 入力実ニューロンと等価である。複素ニューロンの出力の実部 X に相当する実ニューロンを“実部”ニューロン、虚部 Y に相当する実ニューロンを“虚部”ニューロンと呼ぶことにする。ここで、

$$\begin{aligned} & \begin{bmatrix} X \\ Y \end{bmatrix} \\ &= F_C \left(\begin{bmatrix} w_1^r & -w_1^i & \cdots & w_n^r & -w_n^i \\ w_1^i & w_1^r & \cdots & w_n^i & w_n^r \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix} + \begin{bmatrix} \theta^r \\ \theta^i \end{bmatrix} \right) \\ &= F_C \left(|w_1| \begin{bmatrix} \cos \alpha_1 & -\sin \alpha_1 \\ \sin \alpha_1 & \cos \alpha_1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \cdots \right. \\ & \quad \left. + |w_n| \begin{bmatrix} \cos \alpha_n & -\sin \alpha_n \\ \sin \alpha_n & \cos \alpha_n \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \begin{bmatrix} \theta^r \\ \theta^i \end{bmatrix} \right), \end{aligned} \quad (3.36)$$

ただし, $F_C \left(\begin{bmatrix} x \\ y \end{bmatrix} \right) = \begin{bmatrix} f_R(x) \\ f_R(y) \end{bmatrix}$, $\alpha_k = \arctan(w_k^i/w_k^r)$ ($1 \leq k \leq n$) である。

式 (3.36) において, $|w_k|$ は 2 次元平面における点 (x_k, y_k) の原点との距離に関する縮小・拡大, $\begin{bmatrix} \cos \alpha_k & -\sin \alpha_k \\ \sin \alpha_k & \cos \alpha_k \end{bmatrix}$ は原点を中心にした正の方向 (反時計回り) への α_k 度の回転, $t[\theta^r \ \theta^i]$ は平行移動を意味する。このように, n 入力複素ニューロンは, 入力された 2 次元情報 (複素数) それぞれに対して “アフィン変換” を施しているのである (図 3.7)。

先に見たように, 実ニューロンは基本的に実数直線 (1 次元) 上の点列の移動を司っており, 実ニューロン間の重みパラメータは互いに独立であって, 何らの影響も与え合わない。それに対して, 複素ニューロンは基本的に 2 次元平面上の “アフィン変換” を司っており, 学習においてはそのアフィン変換の調整を行うものと解釈することができる。それに伴って, 複素ニューロンには,

“実部” ニューロンへの入力値の実部 x_k に対する重み

=

“虚部” ニューロンへの入力値の虚部 y_k に対する重み,

“実部” ニューロンへの入力値の虚部 y_k に対する重み

=

- “虚部” ニューロンへの入力値の実部 x_k に対する重み

という制約が重みに課されており (図 3.6 参照), 学習はその制約を保ちつつ行われる。見方を変えると “実部” ニューロンと “虚部” ニューロンはそれらの重みパラメータを通じて互いに影響を与え合っていると言える。このように, 基本的に数直線 (1 次元) 上の移動を司っていた実 BP は, 複素数に拡張されることにより, 平面 (2 次元) 上のアフィン変換に関する構造を有するものへと変質していることがわかった。次に述べるように, 重みパラメータのこの構造は決定表面の直交性として現れる。

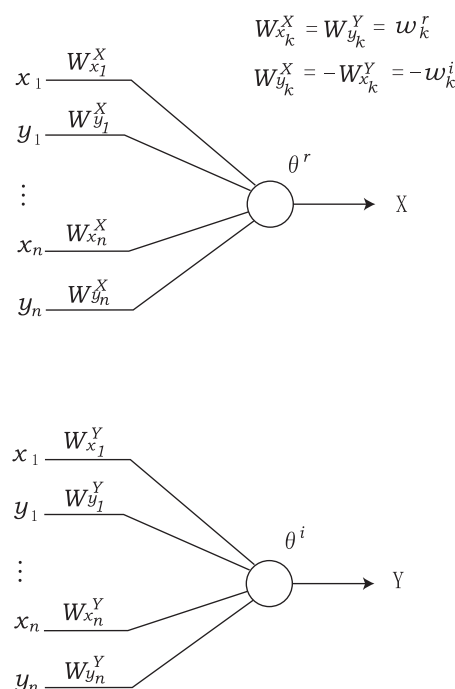


図 3.6 複素ニューロンと等価な実ニューロン

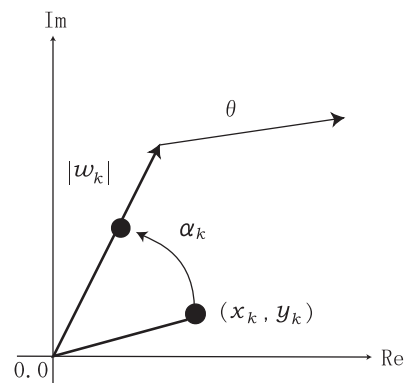


図 3.7 複素信号に対する 2 次元アフィン変換のイメージ

(c) 単一複素ニューロンの決定表面の直交性

決定表面 (decision boundary) とは, 学習識別系 (たとえば, 実 BP) がパターンを切り分ける境界線のことであり, 一般に超曲面から成るものである。

まず, 単一の複素ニューロン (中間層 0 個) の決定表面について調べる。

重みを $w = {}^t[w_1 \cdots w_n] = w^r + iw^i$, $w^r = {}^t[w_1^r \cdots w_n^r]$, $w^i = {}^t[w_1^i \cdots w_n^i]$, 閾値を $\theta = \theta^r + i\theta^i$ とする。出力関数 $f_1: C^1 \rightarrow C^1$ は、簡単のために、

$$f_1(z) = 1(x) + i1(y), \quad z = x + iy, \quad (3.37)$$

ただし、 $1(u) = \begin{cases} 1 & (\text{if } u \geq 0) \\ 0 & (\text{if } u < 0) \end{cases}$ とする。このとき、複素パターン $z = {}^t[z_1 \cdots z_n] = x + iy$, $x = {}^t[x_1 \cdots x_n]$, $y = {}^t[y_1 \cdots y_n]$ が入力されると複素ニューロンは、

$$1\left({}^t[w^r \quad -{}^t[w^i] \begin{bmatrix} x \\ y \end{bmatrix} + \theta^r\right) + i1\left({}^t[w^i \quad {}^t[w^r] \begin{bmatrix} x \\ y \end{bmatrix} + \theta^i\right) \quad (3.38)$$

を出力する。故に、複素ニューロンの出力値の実部は $[{}^t[w^r \quad -{}^t[w^i]]$ を法線ベクトルとする超平面 (決定表面) $[{}^t[w^r \quad -{}^t[w^i]] \begin{bmatrix} x \\ y \end{bmatrix} + \theta^r = 0$ を境にして定まる。つまり、超平面 ≥ 0 ならば 1, そうでなければ 0 となる。同様に、複素ニューロンの出力値の虚部は $[{}^t[w^i \quad {}^t[w^r]]$ を法線ベクトルとする超平面 (決定表面) $[{}^t[w^i \quad {}^t[w^r]] \begin{bmatrix} x \\ y \end{bmatrix} + \theta^i = 0$ を境にして定まる。ここで、これら 2 つの決定表面の法線ベクトルの内積をとると $[{}^t[w^r \quad -{}^t[w^i]] \begin{bmatrix} w^i \\ w^r \end{bmatrix} = 0$ となり、2 つの決定表面は直交しているということがわかる。

次に、式 (3.10) で示される出力関数を持つ場合について調べる。複素パターン $z = {}^t[z_1 \cdots z_n] = x + iy$, $x = {}^t[x_1 \cdots x_n]$, $y = {}^t[y_1 \cdots y_n]$ が入力されると複素ニューロンは、

$$X + iY = f_R\left({}^t[w^r \quad -{}^t[w^i] \begin{bmatrix} x \\ y \end{bmatrix} + \theta^r\right) + if_R\left({}^t[w^i \quad {}^t[w^r] \begin{bmatrix} x \\ y \end{bmatrix} + \theta^i\right) \quad (3.39)$$

を出力する。ここで、常数 $C^R, C^I \in (0, 1)$ を任意に選び、

$$X(x, y) = f_R\left({}^t[w^r \quad -{}^t[w^i] \begin{bmatrix} x \\ y \end{bmatrix} + \theta^r\right) = C^R, \quad (3.40)$$

$$Y(x, y) = f_R\left({}^t[w^i \quad {}^t[w^r] \begin{bmatrix} x \\ y \end{bmatrix} + \theta^i\right) = C^I \quad (3.41)$$

とおく。式 (3.40) は n 入力複素ニューロンの出力値の実部に関する決定表面である、すなわち、入力信号 $(x, y) \in R^{2n}$ は、超曲面 (式 (3.40)) を境界線にして、2 つの決定領域 (decision region) $\{(x, y) \in R^{2n} | X(x, y) \geq C^R\}$, $\{(x, y) \in R^{2n} | X(x, y) < C^R\}$ に分類される。同様にして、式 (3.41) で示される超曲面は n 入力複素ニューロンの虚部に関する決定表面である。そこで、式 (3.40) および式 (3.41) のそれぞれの法線ベクトル $H^R(x, y)$, $H^I(x, y)$ を求めると、

$$H^R(x, y) = \left(\frac{\partial X}{\partial x_1} \cdots \frac{\partial X}{\partial x_n} \frac{\partial X}{\partial y_1} \cdots \frac{\partial X}{\partial y_n} \right) = f'_R\left({}^t[w^r \quad -{}^t[w^i] \begin{bmatrix} x \\ y \end{bmatrix} + \theta^r\right) \cdot [{}^t[w^r \quad -{}^t[w^i]], \quad (3.42)$$

$$H^I(x, y) = \left(\frac{\partial Y}{\partial x_1} \cdots \frac{\partial Y}{\partial x_n} \frac{\partial Y}{\partial y_1} \cdots \frac{\partial Y}{\partial y_n} \right) = f'_R\left({}^t[w^i \quad {}^t[w^r] \begin{bmatrix} x \\ y \end{bmatrix} + \theta^i\right) \cdot [{}^t[w^i \quad {}^t[w^r]] \quad (3.43)$$

となり、それらの内積は 0 となるから、実部に関する決定表面と虚部に関する決定表面は互いに直交していることがわかる。

実ニューロンは入力された実パターンを 2 つの (0 および 1 という値で代表される) クラスに分類する。一方、複素ニューロンは入力された複素パターンを 4 つの (0, 1, i , $1+i$ という値で代表される) クラスに分類する。先の議論により、複素ニューロンの決定表面は 2 つの直交した超曲面 (超平面) から構成されており、領域を 4 つに均等に分割しようとする。つまり、一般には複素ニューロンは複素パターンに対する自然な決定表面を有していると言える。

(d) 3 層ネットワークの決定表面の直交性

次に、3 層ネットワーク (中間層 1 個) の決定表面について調べる。入力ニューロン L 個、中間ニューロン M 個、出力ニューロン N 個の 3 層ネットワークを考える。 $w_{ji} = w_{ji}^r + iw_{ji}^i$ を入力ニューロン i と中間ニューロン j との間の結合の重み、 $v_{kj} = v_{kj}^r + iv_{kj}^i$ を中間ニューロン j と出力ニューロン k との間の結合の重み、 $\theta_j = \theta_j^r + i\theta_j^i$ を中間ニューロン j が持つ閾値、 $\gamma_k = \gamma_k^r + i\gamma_k^i$ を出力ニュー

ロン k が持つ閾値とする。出力関数は、式 (3.10) で示されるものとする。このとき、複素パターン $z = {}^t[z_1 \cdots z_L] = \mathbf{x} + i\mathbf{y}$, $\mathbf{x} = {}^t[x_1 \cdots x_L]$, $\mathbf{y} = {}^t[y_1 \cdots y_L]$ が入力されると、中間ニューロン j への入力の総和 U_j は、

$$\begin{aligned} U_j &= U_j^r + iU_j^i \\ &= \left[\sum_{i=1}^L (w_{ji}^r x_i - w_{ji}^i y_i) + \theta_j^r \right] \\ &\quad + i \left[\sum_{i=1}^L (w_{ji}^i x_i + w_{ji}^r y_i) + \theta_j^i \right] \end{aligned} \quad (3.44)$$

となる。よって、その出力 H_j は、

$$H_j = H_j^r + iH_j^i = f_R(U_j^r) + if_R(U_j^i) \quad (3.45)$$

である。さらに、出力ニューロン k への総入力 S_k は、

$$\begin{aligned} S_k &= S_k^r + iS_k^i \\ &= \left[\sum_{j=1}^M (v_{kj}^r H_j^r - v_{kj}^i H_j^i) + \gamma_k^r \right] \\ &\quad + i \left[\sum_{j=1}^M (v_{kj}^i H_j^r + v_{kj}^r H_j^i) + \gamma_k^i \right] \end{aligned} \quad (3.46)$$

である。故に、その出力 O_k は、

$$O_k = O_k^r + iO_k^i = f_R(S_k^r) + if_R(S_k^i) \quad (3.47)$$

となる。ここで、常数 $C^R, C^I \in (0, 1)$ を任意に選び、

$$O_k^r(\mathbf{x}, \mathbf{y}) = C_R, \quad (3.48)$$

$$O_k^i(\mathbf{x}, \mathbf{y}) = C_I \quad (3.49)$$

とおく。式 (3.48) および式 (3.49) は、それぞれ、3層の複素BPネットワークにおける出力ニューロン k の実部および虚部に関する決定表面である。これら2つの超曲面の法線ベクトル $H^R(\mathbf{x}, \mathbf{y}), H^I(\mathbf{x}, \mathbf{y})$ は、それぞれ、

$$H^R(\mathbf{x}, \mathbf{y}) = \left(\frac{\partial O_k^r}{\partial x_1} \cdots \frac{\partial O_k^r}{\partial x_L} \frac{\partial O_k^r}{\partial y_1} \cdots \frac{\partial O_k^r}{\partial y_L} \right), \quad (3.50)$$

$$H^I(\mathbf{x}, \mathbf{y}) = \left(\frac{\partial O_k^i}{\partial x_1} \cdots \frac{\partial O_k^i}{\partial x_L} \frac{\partial O_k^i}{\partial y_1} \cdots \frac{\partial O_k^i}{\partial y_L} \right) \quad (3.51)$$

であり、これらの内積は、

$$\begin{aligned} &H^R(\mathbf{x}, \mathbf{y}) \cdot {}^t H^I(\mathbf{x}, \mathbf{y}) \\ &= \frac{\partial O_k^r}{\partial x_1} \cdot \frac{\partial O_k^i}{\partial x_1} + \cdots + \frac{\partial O_k^r}{\partial x_L} \cdot \frac{\partial O_k^i}{\partial x_L} \\ &+ \frac{\partial O_k^r}{\partial y_1} \cdot \frac{\partial O_k^i}{\partial y_1} + \cdots + \frac{\partial O_k^r}{\partial y_L} \cdot \frac{\partial O_k^i}{\partial y_L} \end{aligned} \quad (3.52)$$

で与えられる。ここで、任意の $1 \leq i \leq L$ に対して、

$$\begin{aligned} &\frac{\partial O_k^r}{\partial x_i} \cdot \frac{\partial O_k^i}{\partial x_i} + \frac{\partial O_k^r}{\partial y_i} \cdot \frac{\partial O_k^i}{\partial y_i} \\ &= \frac{\partial f_R(S_k^r)}{\partial S_k^r} \cdot \frac{\partial f_R(S_k^i)}{\partial S_k^i} \\ &\quad \cdot \left[\sum_{j=1}^M \left(v_{kj}^r w_{ji}^r \cdot \frac{\partial f_R(U_j^r)}{\partial U_j^r} - v_{kj}^i w_{ji}^i \cdot \frac{\partial f_R(U_j^i)}{\partial U_j^i} \right) \right] \\ &\quad \cdot \left[\sum_{j=1}^M \left(v_{kj}^i w_{ji}^r \cdot \frac{\partial f_R(U_j^r)}{\partial U_j^r} + v_{kj}^r w_{ji}^i \cdot \frac{\partial f_R(U_j^i)}{\partial U_j^i} \right) \right] \\ &\quad - \frac{\partial f_R(S_k^r)}{\partial S_k^r} \cdot \frac{\partial f_R(S_k^i)}{\partial S_k^i} \\ &\quad \cdot \left[\sum_{j=1}^M \left(v_{kj}^r w_{ji}^r \cdot \frac{\partial f_R(U_j^r)}{\partial U_j^r} - v_{kj}^i w_{ji}^i \cdot \frac{\partial f_R(U_j^i)}{\partial U_j^i} \right) \right] \\ &\quad \cdot \left[\sum_{j=1}^M \left(v_{kj}^i w_{ji}^r \cdot \frac{\partial f_R(U_j^r)}{\partial U_j^r} + v_{kj}^r w_{ji}^i \cdot \frac{\partial f_R(U_j^i)}{\partial U_j^i} \right) \right]. \end{aligned} \quad (3.53)$$

よって、法線ベクトルの内積は必ずしも0になるわけではなく、実部と虚部のそれぞれの決定表面は互いに単純に直交するとは言えない。しかしながら、式 (3.53) を注意深くみると、任意の $1 \leq j \leq M$ に対して、

$$\frac{\partial f_R(U_j^r)}{\partial U_j^r} = \frac{\partial f_R(U_j^i)}{\partial U_j^i} \quad (3.54)$$

すなわち、

$$\begin{aligned} &f_R' \left[\sum_{i=1}^L (w_{ji}^r x_i - w_{ji}^i y_i) + \theta_j^r \right] \\ &= f_R' \left[\sum_{i=1}^L (w_{ji}^i x_i + w_{ji}^r y_i) + \theta_j^i \right] \end{aligned} \quad (3.55)$$

が成り立つならば、内積 = 0 となることがわかる。一般に、 $f_R'(u_1)$ と $f_R'(u_2)$ は $|u_1|$ および $|u_2|$ がある大きな値以上であれば、ほぼ同じ値をとるものとみなせる。故に、任意の $1 \leq j \leq M$ に対して、十分大きな正数 K_1, K_2 が存在して、

$$\begin{aligned} |U_j^r| &= \left| \sum_{i=1}^L (w_{ji}^r x_i - w_{ji}^i y_i) + \theta_j^r \right| \\ &= \left| \sum_{i=1}^L |w_{ji}| |z_i| \cos(\alpha_{ji} + \beta_i) + \theta_j^r \right| \\ &> K_1, \end{aligned} \quad (3.56)$$

$$|U_j^i| = \left| \sum_{i=1}^L (w_{ji}^i x_i + w_{ji}^r y_i) + \theta_j^i \right|$$

$$= \left| \sum_{i=1}^L |w_{ji}^i| |z_i| \sin(\alpha_{ji} + \beta_i) + \theta_j^i \right| > K_2, \quad (3.57)$$

ただし, $\tan(\alpha_{ji}) = w_{ji}^i/w_{ji}^r$, $\tan(\beta_i) = y_i/x_i$ が成り立つとき, 2つの決定表面は互いに直交する。つまり, 任意の $1 \leq j \leq M$ に対して, 中間ニューロン j への総入力(複素数)の実部および虚部の絶対値が共に十分大きいとき, 2つの決定表面は互いに直交する。あるいは, さらに, $|z| = \sqrt{|z_1|^2 + \dots + |z_L|^2}$ が十分大きい値をとる (x, y) において, 2つの決定表面は互いに直交する可能性が高いと言える。

このことを定理および系の形でまとめておく。

定理 2. すべての中間ニューロンへの総入力(複素数)の実部および虚部の絶対値が共に十分大きいとき, 実部と虚部のそれぞれの決定表面は互いに直交する。 □

系 1. $|z|$ が十分大きい値をとる (x, y) において, 実部と虚部のそれぞれの決定表面は互いに直交する可能性が高い。 □

さらに, 新田らは, 3層の複素BPネットワークにおける決定表面の特性について計算機実験を通じて調べ, 実BPネットワークのものとの比較を行った。その結果, 実部の決定表面と虚部の決定表面とが直交する傾向が見られ, 複素BPネットワークの決定表面の直交性の性質が反映されているものとしている。詳細については, 文献[44,66]を参照されたい。

3.1.2.2 学習特性

次に, 学習特性について述べる [46,55,58,61,66,69]。

新田らは, 学習アルゴリズムの観点から複素BPの基本特性を調べた。主な結果は次のとおりである。(1) 複素BPにおける誤差逆伝播は2次元アフィン変換に基づいた構造を有している。(2) 複素BPはネットワークを流れる複素信号を1つの処理単位として学習を進める。(3) 複素BPでは, 学習則における実部要因と虚部要因

の補完構造により, 学習停滞状態の発生が押えられる。その結果, 実BPに比べて, 複素パターンに対する学習が数倍速くなっている。しかも, そのときに必要となる領域計算量(重みおよび閾値の総数)は約半分で済む。これらの意味で, 複素BPは複素パターンの学習に適したアルゴリズムであると言える。

以下では, まず, 複素BP学習アルゴリズムの解析を行い, 学習の高速性の要因などの基本特性について調べる。次に, 2つの例題を用いた計算機実験の結果を示す。

(a) 学習パラメータの修正量の構造

まず, 複素BPにおける学習パラメータの修正量の構造を3.1.1.3項で定義した3層ニューラルネットワークを例にとって調べる。学習パラメータ x の修正量を Δx , その実部を Δx^R , 虚部を Δx^I で表すことにすると, 3.1.1.3節で導出した複素BPの学習則(式(3.15) - (3.18))は次のように表現できる。

$$\begin{aligned} \begin{bmatrix} \Delta v_{kj}^R \\ \Delta v_{kj}^I \end{bmatrix} &= \begin{bmatrix} \operatorname{Re}[H_j] & \operatorname{Im}[H_j] \\ -\operatorname{Im}[H_j] & \operatorname{Re}[H_j] \end{bmatrix} \begin{bmatrix} \Delta \gamma_k^R \\ \Delta \gamma_k^I \end{bmatrix} \\ &= |H_j| \begin{bmatrix} \cos \beta_j & \sin \beta_j \\ -\sin \beta_j & \cos \beta_j \end{bmatrix} \begin{bmatrix} \Delta \gamma_k^R \\ \Delta \gamma_k^I \end{bmatrix}, \end{aligned} \quad (3.58)$$

$$\begin{bmatrix} \Delta \gamma_k^R \\ \Delta \gamma_k^I \end{bmatrix} = \varepsilon \begin{bmatrix} A_k & 0 \\ 0 & B_k \end{bmatrix} \begin{bmatrix} \operatorname{Re}[\delta^k] \\ \operatorname{Im}[\delta^k] \end{bmatrix}, \quad (3.59)$$

$$\begin{aligned} \begin{bmatrix} \Delta w_{ji}^R \\ \Delta w_{ji}^I \end{bmatrix} &= \begin{bmatrix} \operatorname{Re}[I_i] & \operatorname{Im}[I_i] \\ -\operatorname{Im}[I_i] & \operatorname{Re}[I_i] \end{bmatrix} \begin{bmatrix} \Delta \theta_j^R \\ \Delta \theta_j^I \end{bmatrix} \\ &= |I_i| \begin{bmatrix} \cos \phi_i & \sin \phi_i \\ -\sin \phi_i & \cos \phi_i \end{bmatrix} \begin{bmatrix} \Delta \theta_j^R \\ \Delta \theta_j^I \end{bmatrix}, \end{aligned} \quad (3.60)$$

$$\begin{bmatrix} \Delta \theta_j^R \\ \Delta \theta_j^I \end{bmatrix} = \begin{bmatrix} C_j & 0 \\ 0 & D_j \end{bmatrix} \sum_k \begin{bmatrix} \operatorname{Re}[v_{kj}] & \operatorname{Im}[v_{kj}] \\ -\operatorname{Im}[v_{kj}] & \operatorname{Re}[v_{kj}] \end{bmatrix} \begin{bmatrix} \Delta \gamma_k^R \\ \Delta \gamma_k^I \end{bmatrix}$$

$$= \begin{bmatrix} C_j & 0 \\ 0 & D_j \end{bmatrix} \sum_k |v_{kj}| \begin{bmatrix} \cos \varphi_{kj} & \sin \varphi_{kj} \\ -\sin \varphi_{kj} & \cos \varphi_{kj} \end{bmatrix} \begin{bmatrix} \Delta \gamma_k^R \\ \Delta \gamma_k^I \end{bmatrix}, \quad (3.61)$$

ただし,

$$\begin{aligned} A_k &= (1 - \operatorname{Re}[O_k])\operatorname{Re}[O_k], \\ B_k &= (1 - \operatorname{Im}[O_k])\operatorname{Im}[O_k], \\ C_j &= (1 - \operatorname{Re}[H_j])\operatorname{Re}[H_j], \\ D_j &= (1 - \operatorname{Im}[H_j])\operatorname{Im}[H_j], \\ \beta_j &= \arctan(\operatorname{Im}[H_j]/\operatorname{Re}[H_j]), \\ \phi_i &= \arctan(\operatorname{Im}[I_i]/\operatorname{Re}[I_i]), \\ \varphi_{kj} &= \arctan(\operatorname{Im}[v_{kj}]/\operatorname{Re}[v_{kj}]) \end{aligned} \quad (3.62)$$

である。

まず, Δv_{kj} について調べる。式 (3.58) において, $|H_j|$ は 2次元平面上の点の原点との距離に関する縮小・拡大, $\begin{bmatrix} \cos \beta_j & \sin \beta_j \\ -\sin \beta_j & \cos \beta_j \end{bmatrix}$ は原点を中心とした負の方向(時計回り)への β_j 度の回転を意味する。つまり, 式 (3.58) では 2次元アフィン変換が行われていると言える。よって, 学習パラメータの修正量(複素数)を 2次元平面上の点として表現した場合, 中間 - 出力層の重みの修正量 $(\Delta v_{kj}^R, \Delta v_{kj}^I)$ は, 出力層の閾値の修正量 $(\Delta \gamma_k^R, \Delta \gamma_k^I)$ に上記のアフィン変換を施すことによって得られるというわけである(図 3.8)。同様にして, 中間層の閾値の修正量 $(\Delta \theta_j^R, \Delta \theta_j^I)$ は, 出力層の各閾値の修正量 $(\Delta \gamma_k^R, \Delta \gamma_k^I)$ に中間 - 出力層の重み v_{kj} に関する 2次元アフィン変換を施すことによって得られる値に基づいて定まる(式 (3.61))。また, 入力 - 中間層の重みの修正量 $(\Delta w_{ji}^R, \Delta w_{ji}^I)$ は, 中間層の閾値の修正量 $(\Delta \theta_j^R, \Delta \theta_j^I)$ に入力層の出力値 I_i に関する 2次元アフィン変換を施すことによって得られる(式 (3.60))。要するに, 複素BPにおける誤差逆伝播は 2次元アフィン変換に基づいた構造を有していると言える。

ここで, 重みパラメータ自身には, 3.1.2.1 項で見たように, 2次元アフィン変換に関する制約が課せられており, 学習はその制約を保ちながら進められる。よって, 複素BPでは, 学習パラメータ自身およびその修正量には

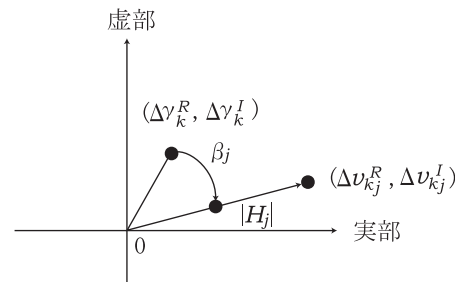


図 3.8 複素BPにおける誤差逆伝播のイメージ

2次元アフィン変換に関係した構造が備わっていると言える。ただし, 学習パラメータ自身に課せられている制約は正の方向への回転変換

$$P = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix} \quad (3.63)$$

に関係しており, 学習パラメータの修正量に課せられている制約は負の方向への回転変換

$$Q = \begin{bmatrix} \cos \beta & \sin \beta \\ -\sin \beta & \cos \beta \end{bmatrix} \quad (3.64)$$

に関係している。ここで, P および Q は, 共に, 直交行列であり,

$$Q^{-1} = {}^t Q = \begin{bmatrix} \cos \beta & -\sin \beta \\ \sin \beta & \cos \beta \end{bmatrix} \quad (3.65)$$

を満たし, Q の逆行列 Q^{-1} および転置行列 ${}^t Q$ は P と同様な形をしている。

ここで, 一方の実BPにおける学習パラメータの修正量の構造について確認しておく。実BPにおける学習パラメータの修正量は,

$$\Delta v_{kj} = H_j \Delta \gamma_k, \quad (3.66)$$

$$\Delta \gamma_k = \varepsilon(1 - O_k)O_k \delta^k, \quad (3.67)$$

$$\Delta w_{ji} = I_i \Delta \theta_j, \quad (3.68)$$

$$\Delta \theta_j = (1 - H_j)H_j \sum_k v_{kj} \Delta \gamma_k \quad (3.69)$$

で与えられる(ただし, 各パラメータはすべて実数であり, その定義は複素BPのパラメータに対応する)。学習パラメータの修正量(実数)を 1次元直線上の点として表

現した場合、中間 - 出力層の重みの修正量 Δv_{kj} は出力層の閾値の修正量 $\Delta \gamma_k$ を原点との距離に関して H_j 倍することによって得られる (式 (3.66), 図 3.9)。同様に、中間層の閾値 $\Delta \theta_j$ は、出力層の各閾値の修正量 $\Delta \gamma_k$ をそれぞれ原点との距離に関して v_{kj} 倍した値に基づいて定まる (式 (3.69))。また、入力 - 中間層の重みの修正量 Δw_{ji} は、中間層の閾値の修正量 $\Delta \theta_j$ を原点との距離に関して I_i 倍することによって得られる (式 (3.68))。つまり、実 BP における誤差逆伝播は 1 次元直線上の点の移動に基づいた構造をしていると言える。このように、実 BP が複素数へ拡張されることによって、誤差逆伝播の構造が 1 次元的なものから 2 次元的なものへと変質していることがわかった。

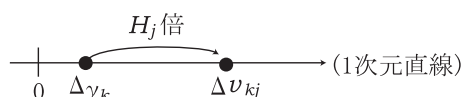


図 3.9 実 BP における誤差逆伝播のイメージ

また、見方を変えると、誤差逆伝播の 2 次元的な構造に伴い、複素 BP はネットワークを流れる複素信号を 1 つの処理単位として学習を進めているということがわかる。たとえば、 Δv_{kj}^R および Δv_{kj}^I は、ネットワークを流れる信号の実部 ($Re[H_j]$, $Re[O_k]$) および虚部 ($Im[H_j]$, $Im[O_k]$) を共に含んでおり、これらに基づいて値が定まる (式 (3.58))。つまり、 Δv_{kj}^R は複素信号の実部 ($Re[H_j]$, $Re[O_k]$) だけから、 Δv_{kj}^I は虚部 ($Im[H_j]$, $Im[O_k]$) だけから、という具合にお互いに独立して決まるのではなく、ネットワークを流れる信号の実部および虚部の両方の値を通じて依存し合いながら定まるものであると言える。入力 - 中間層の結合の重み w_{ji} 、中間層の閾値 θ_j 、出力層の閾値 γ_k についても同様のことが言える。

このように、複素 BP における学習パラメータの実部および虚部の値の修正は、それぞれ、ネットワークを流れる信号の実部および虚部の両方の値に基づいて、そして、それらを通じて相互に依存し合いながら行われる (図 3.10)。要するに、複素 BP は、ネットワークを流れる信号 (複素数) を 1 つの処理単位として学習を進めるものであると言える。

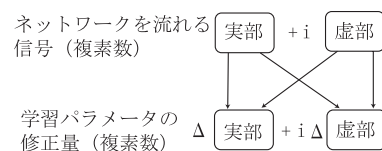


図 3.10 学習パラメータの修正量の決定要因。

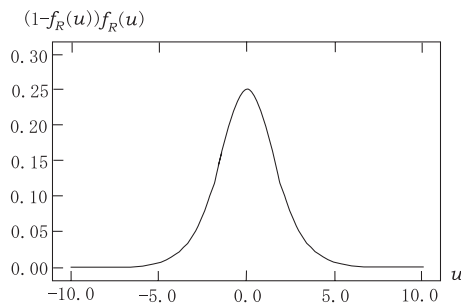
矢印の始点は、矢印の終点の決定要因である。

(b) 学習の高速性の要因

このように、複素 BP における誤差逆伝播は 2 次元アフィン変換に基づいた構造を有しており、その結果、複素 BP 学習はネットワークを流れる複素信号を 1 つの処理の単位としている。ここでは、更に、このことがその構造において、学習の高速化を促すものであることを示す。

実 BP の学習パラメータの修正量 (式 (3.66) - (3.69)) において、 $(1 - H_j)H_j \in \mathbf{R}$ および $(1 - O_k)O_k \in \mathbf{R}$ は、各ニューロンの出力関数であるシグモイド関数 $f_R(u) = 1/(1 + \exp(-u))$ の導関数 $(1 - f_R(u))f_R(u)$ であり、図 3.11 のような曲線を描くものである。つまり、ニューロンへの入力の総和 u の絶対値が大きくなるにつれて 0 に近づいてゆく関数である。よって、ニューロンの出力値を 0 もしくは 1 に少しでも近づけるために、ニューロンへの入力の総和 u の絶対値が大きくなるとシグモイド導関数 $(1 - f_R(u))f_R(u)$ は微小な値を取ることになり、その結果、学習は進みにくくなる。その際、特に、誤差が大きな値であり、そのような状態が長く続く場合には、“ローカルミニマムに陥った”と称されることになる。よく知られているように、これが実 BP における学習停滞のメカニズムである。

一方、複素 BP においては、学習パラメータの修正量 (式 (3.58) - (3.61)) には 2 種類のシグモイド導関数が現れる。1 つは出力関数の実部の導関数 $(1 - Re[O_k])Re[O_k]$, $(1 - Re[H_j])Re[H_j]$, もう一つは虚部の導関数 $(1 - Im[O_k])Im[O_k]$, $(1 - Im[H_j])Im[H_j]$ であり、複素 BP の学習則は基本的にこれらの 2 つの線

図 3.11 シグモイド関数 $f_R(u)$ の導関数

形結合

$$\alpha_1(1-\text{Re}[O_k])\text{Re}[O_k] + \beta_1(1-\text{Im}[O_k])\text{Im}[O_k], \quad (3.70)$$

$$\alpha_2(1-\text{Re}[H_j])\text{Re}[H_j] + \beta_2(1-\text{Im}[H_j])\text{Im}[H_j] \quad (3.71)$$

から構成されている ($\alpha_k, \beta_k \in \mathbf{R} (k=1,2)$)。ここで、式 (3.70) が微小な値になるのは、 $(1 - \text{Re}[O_k])\text{Re}[O_k]$ と $(1 - \text{Im}[O_k])\text{Im}[O_k]$ の両方の値が微小になるときである (式 (3.71) についても同様)。よって、たとえば、 $(1 - \text{Re}[O_k])\text{Re}[O_k]$ が微小な値をとったとしても、 $(1 - \text{Im}[O_k])\text{Im}[O_k]$ もそうであるとは限らないので、式 (3.70) が微小値をとらない可能性があり、この点で、 $(1 - O_k)O_k \in \mathbf{R}$ が微小値をとれば必ず学習パラメータの修正量が微小になってしまう実 B P の学習則とは異なっている。言い換えると、実部要因 $\left((1 - \text{Re}[O_k])\text{Re}[O_k], (1 - \text{Re}[H_j])\text{Re}[H_j] \right)$ と虚部要因 $\left((1 - \text{Im}[O_k])\text{Im}[O_k], (1 - \text{Im}[H_j])\text{Im}[H_j] \right)$ は、学習パラメータの修正量が異常に小さくならないようにお互いに補い合っているとも言える。このように、複素 B P の学習則は、実 B P に比べて、学習の停滞状態が発生しにくい構造を有していると言える (注意: 特に、 $(1 - \text{Re}[O_k])\text{Re}[O_k]$ および $(1 - \text{Im}[O_k])\text{Im}[O_k]$ が共に微小な値になることが稀であるときに顕著であろう)。この結果、複素 B P は、実 B P に比べて、学習が高速化されているものと思われる。

(c) シミュレーション

次に、複素パターンを使った計算機実験を通じて、(b)

において述べた複素 B P の学習特性の確認を行う。

学習速度の評価を計算量の観点からトータルに行なうために、ここでは、アルゴリズムの中で行なわれる (実数の) 四則演算の総回数を時間計算量、学習パラメータ数を領域計算量とする。ただし、時間計算量において、(1回の加算の計算量) と (1回の減算の計算量) は等しいものとする。また、一般に、(1回の乗算の計算量) \geq (1回の加算の計算量) であるが、ここでは念の為、(1回の加算の計算量) = (1回の乗算の計算量) および (1回の加算の計算量) = 0 という極端な 2 つのケースを考える。なお、除算は実 B P および複素 B P では使用されないのが常に 0 回である。

以下では、1回の学習の時間計算量が実 B P と複素 B P とで等しくなるようにネットワークの構造を調節したうえで、それぞれの収束するまでに必要な平均学習回数を基準にして学習速度の評価を行なう。加えて、そのときの領域計算量も評価する。

実験において、学習パラメータの初期値は、 -0.3 から $+0.3$ の間の乱数により設定する。学習は誤差

$$\sqrt{2 \sum_p E_p} \quad (3.72)$$

が 0.10 となったときに収束 (終了) したものとみなし、収束の規範とする (E_p は 3.1.1.3 節で定義したパターン p に対する 2 乗誤差である)。また、学習パターンは、通常、数種類のパターンから構成されているが、その数種類のパターンをひと通り提示することで 1 回の学習と数えることとする。

例題 1

例題 1 では、複素パターンから構成される単純な学習パターン (表 3.1) を使った実験を行なった。複素 B P には 1 - 3 - 1 ネットワーク、実 B P には 2 - 6 - 2 ネットワークおよび 2 - 7 - 2 ネットワークを使った。複素 B P および実 B P をこれらのネットワークに適用するときのそれぞれの計算量を表 3.2 に示す。学習 1 回あたりの時間計算量に関して、複素 B P 1 - 3 - 1 ネットワークは、(1回の加算の計算量) = 0 のとき、実 B P 2 - 6 - 2 ネットワークに等しく、(1回の加算の計算量) = (1回の乗算の計算量) のとき、実 B P 2 - 7 - 2 ネットワークにほぼ等しい。

表 3. 1 学習パターン [例題 1]

パターン番号	入力パターン	出力パターン
1	0	0
2	i	1
3	1	$1 + i$
4	$1 + i$	i

表 3. 2 複素BPおよび実BPの計算量 [例題 1]

ネットワーク	時間計算量			領域計算量		
	乗除算	加減算	計	重み	閾値	計
複素BP 1-3-1	78	52	130	12	8	20
実BP 2-6-2	78	40	118	24	8	32
実BP 2-7-2	90	46	136	28	9	37

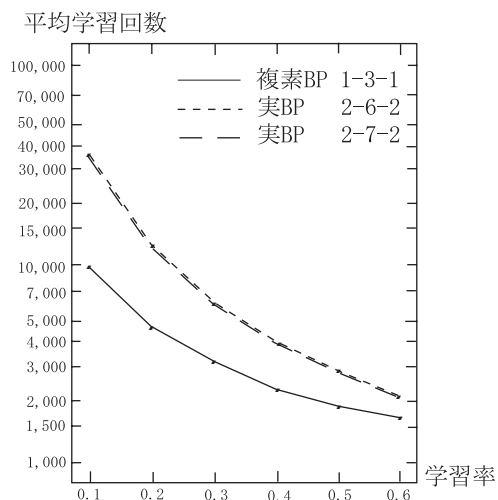


図 3.12 平均学習速度 (複素BPと実BPの比較)[例題 1]

実BPに複素パターンを学習させる方法の1つは、入出力層において実部、虚部を担当させるニューロンをあらかじめ決めておくことである。ここでは、入出力される複素数の実部を入出力ニューロン1、虚部を入出力ニューロン2に担当させた。

学習率をパラメータとして0.1から0.6まで0.1刻みで変化させた。各学習率について、それぞれ50回の試行を行い、それらの収束に要した学習回数の平均を評価の基準とした。学習は5万回で打ち切りとしたが、すべての試行は収束した。図3.12がその実験結果である。

例題 2

次に、表3.3に示す学習パターンを使った実験を行った。この学習パターンは複素パターンから構成されており、次の2つの規則に従っている。

(1) 入力される複素数1と複素数2が等しいならば、出力される複素数3の実部は1、そうでなければ0。

(2) 入力される複素数2が1または i ならば、出力される複素数3の虚部は1、そうでなければ0。

例題1と異なるのは、複素BPには2-4-1ネットワーク、実BPには4-8-2ネットワークおよび4-9-2ネットワークを使った点である。これらのネット

ワークが使われる際のそれぞれの計算量を表3.4に示す。学習1回あたりの時間計算量に関して、複素BP 2-4-1ネットワークは、(1回の加算の計算量) = 0のとき、実BP 4-8-2ネットワークに等しく、(1回の加算の計算量) = (1回の乗算の計算量)のとき、実BP 4-9-2ネットワークに等しい。

表 3. 3 学習パターン [例題 2]

パターン番号	入力パターン		出力パターン
	複素数 1	複素数 2	複素数 3
1	0	0	1
2	0	i	i
3	i	i	$1 + i$
4	i	1	i
5	1	1	$1 + i$
6	i	0	0
7	$1 + i$	$1 + i$	1
8	$1 + i$	i	i

表 3.4 複素BPおよび実BPの計算量 [例題 2]

ネットワーク	時間計算量			領域計算量		
	乗除算	加減算	計	重み	閾値	計
複素BP 2-4-1	134	92	226	24	10	34
実BP 4-8-2	134	68	202	48	10	58
実BP 4-9-2	150	76	226	54	11	65

実BPに上述の学習パターンを与える際には、例題1と同様にして、入出力層において実部、虚部を担当させるニューロンをあらかじめ決めておいた。すなわち、入力される複素数1の実部、虚部、複素数2の実部、虚部を、この順番で、入力ニューロン1, 2, 3そして4に担当させた。また、出力される複素数3の実部は出力ニューロン1, 虚部は出力ニューロン2に担当させた。

学習は10万回で打ち切りとした。図3.13に実験結果を示す。参考のために、各ネットワークでの収束率(50回の試行のうち収束した試行の割合)を表3.5に示しておく。

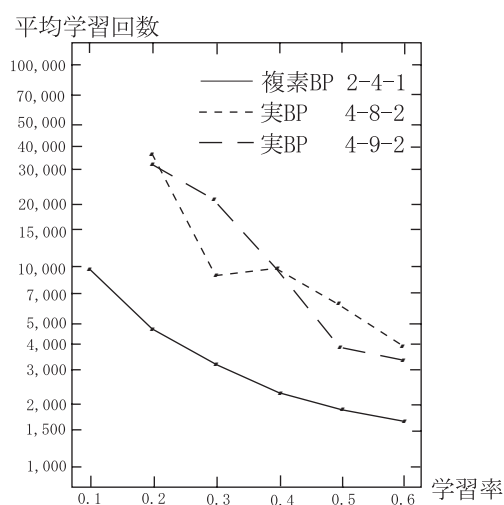


図 3.13 平均学習速度 (複素BPと実BPの比較)[例題 2]

表 3.5 収束率 [例題 2]

ネットワーク	学習率					
	0.1	0.2	0.3	0.4	0.5	0.6
複素BP 2-4-1	100	96	88	92	90	98
実BP 4-8-2	0	48	70	84	94	94
実BP 4-9-2	0	22	64	78	90	100

以上の実験結果により、複素BPには複素パターンの学習に関する以下の特徴が備わっていることがわかる。

複素BPは、実BPに比べて、学習が数倍速くなっている(図3.12, 図3.13)。しかも、そのときの領域計算量は約半分で済む(表3.2, 表3.4)。

これらは先に述べた出力関数の導関数の実部と虚部の(学習則における)線形結合性による学習停滞状態の解消構造が、学習特性に現れたものであると考えられる。

3.1.2.3 2次元アフィン変換学習能力

次に、2次元アフィン変換学習能力について述べる[43,48,50,52,66,67,69]。

新田らは、複素BPが、通常のニューラルネットワークには見られない2次元アフィン変換学習能力(図形変換能力)を備えていることを計算機実験によりいくつかの例を通して確認した。2次元アフィン変換を学習した複素BPネットワークのふるまいと複素解析における一致の定理との関係についても言及している。一致の定理は複素数世界においてのみ生ずる特異な性質を明らかにしたものである。

以下では、まず、計算機実験の条件について述べる。次に、単純な変換を通して、2次元アフィン変換学習能力が備わっていることを示す。最後に、一致の定理との類推について触れる。

(a) 実験の諸条件

複素BPが2次元アフィン変換学習能力を持っていることを確認するために以下で行なう実験の諸条件について述べる。

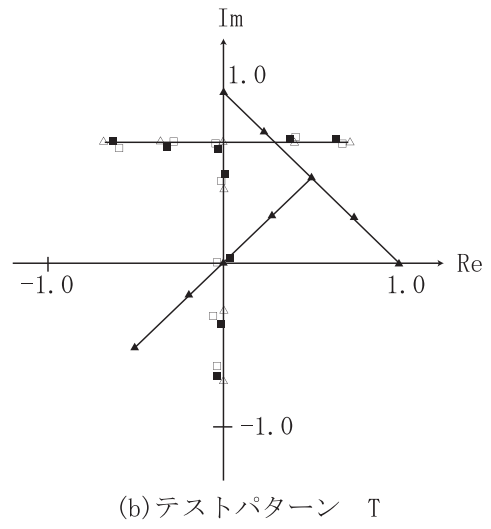
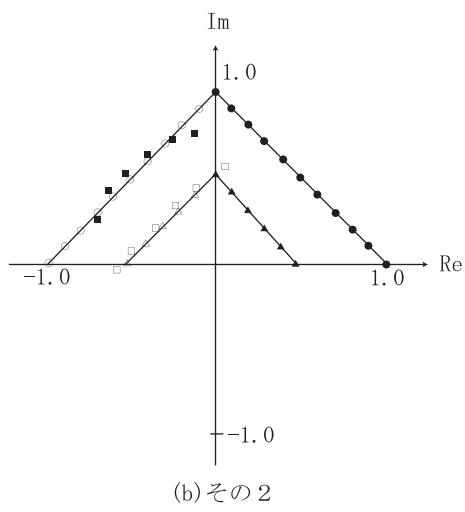
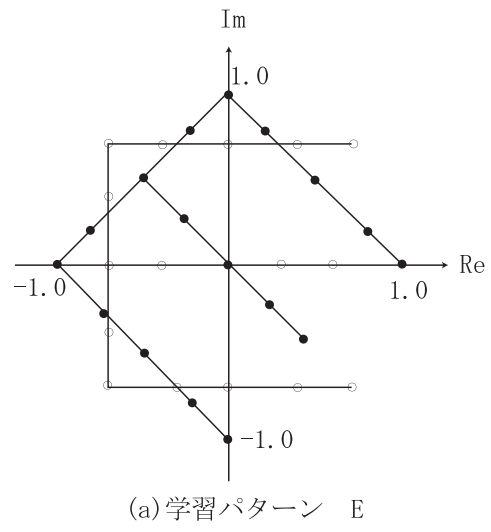
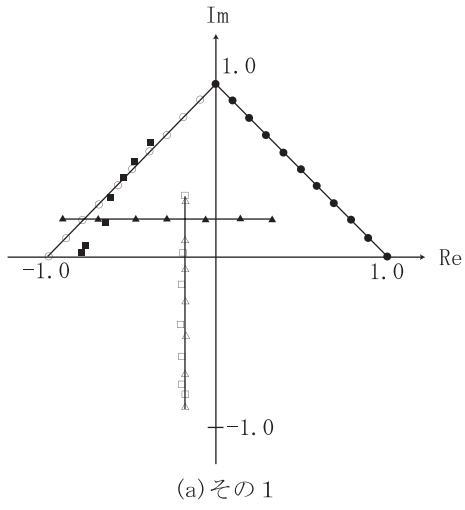
実験に用いる複素BPネットワークは、入力層1ニューロン、中間層6ニューロン、出力層1ニューロンの3層構造である。つまり、1つの複素数 z_1 を入力し、1つの複素数 z_2 を出力するものである。一般に、1つの複素数 $z = x + iy$ は2次元平面上の1点 (x, y) に対応する。よって、当該ネットワークは2次元平面上のある1点 (x, y) を他の1点 (x', y') に変換するものであると考えることができる。ここでは、2次元平面を $-1 \leq x, y \leq 1$ に制限しておく。ニューラルネットワークが出力する値は、 $0 \leq \text{Re}[z], \text{Im}[z] \leq 1$ であるが、便宜上、以下に示す図では、それらを $-1 \leq \text{Re}[z], \text{Im}[z] \leq 1$ に変換した値で示す。比較のために、実BPによる実験も行ったが、そのネットワークは入力層2ニューロン、中間層12ニューロン、出力層2ニューロンとし、複素数 z の実部 $\text{Re}[z]$ を入出力ニューロン1に、虚部 $\text{Im}[z]$ を入出力ニューロン2に対応させた。実験はいずれも、学習率 $\varepsilon = 0.5$ 、学習回数1,000回であり、結合の重みおよび閾値は0と1の間の乱数により設定した。

(b) 回転

まず、直線の回転を行った(図3.14)。学習パターンは11個であり、直線 $y = -x + 1$ ($0 \leq x \leq 1$) 上の11個の点を入力パターン(印)とし、それらの点を原点を中心にして正の方向(反時計回り)に90度回転させた点を出力パターン(印)とした。出力パターンは直線 $y = x + 1$ ($-1 \leq x \leq 0$) 上の点になっている。学習の後、テストパターン(印)として直線 $y = 0.2$ ($-0.9 \leq x \leq 0.3$) 上の7つの点(図3.14(a))および直線 $y = -x + 0.5$ ($0 \leq x \leq 0.5$) 上の6個の点(図3.14(b))を与えたところ、ニューラルネットワークはそれらをほぼ正の方向に90度回転させた点を出力した(印)。同様にして、実BPに従って実験したところ、当然ではあるが、ニューラルネットワークは教師パターン(出力学習パターン)に位置的に近い点(印)を出力した(図3.14)。

次に、斜文字の回転を行った(図3.15, 図3.16)。3文字からなる単語ETLが負の方向に45度回転させられているとする。このとき、学習パターンとして、最初の1文字E上の点17個を入力パターン(印)とし、それらの点を正の方向に45度回転したものを出力パターン(

印)とした(図3.15(a))。学習の後、テストパターン(印)として残る2文字TおよびL上の点を与えたところ、複素BPおよび実BPともにニューラルネットワークはテストパターンをほぼ正の方向に45度回転させた点を出力した(図3.15(b), (c))。同様にして、単語ISOについて行ったところ、テストパターンSおよびOは複素BPでは正の方向に45度回転したが(印)、実BPでは、出力学習パターンに位置的に近い点を出力したため(印)、SおよびOの形状はくずれた(図3.16)。実BPが単語ETLをうまく回転させることができたのは、テストパターンT, L上の点列が学習パターンE上の点列に含まれていたり、位置的に近かったからであると思われる。



- 入学習パターン
- 出学習パターン
- ▲ 入力テストパターン
- △ 期待される出力テストパターン
- 実BP学習による出力テストパターン
- 複素BP学習による出力テストパターン

図 3.14 直線の回転

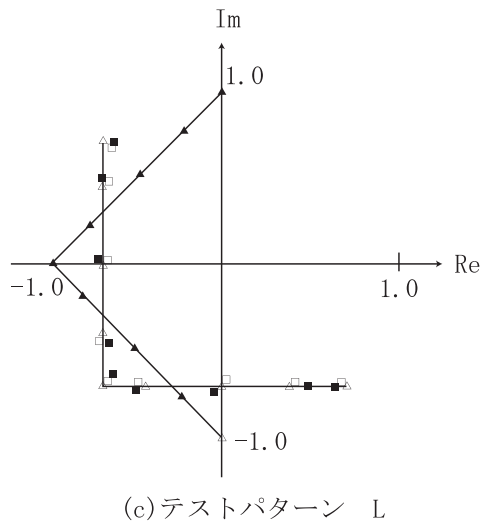
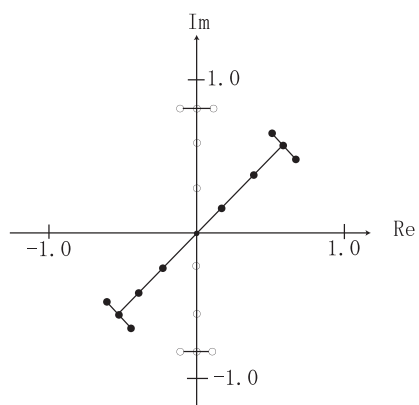
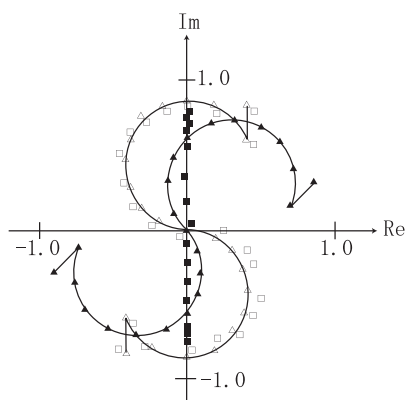


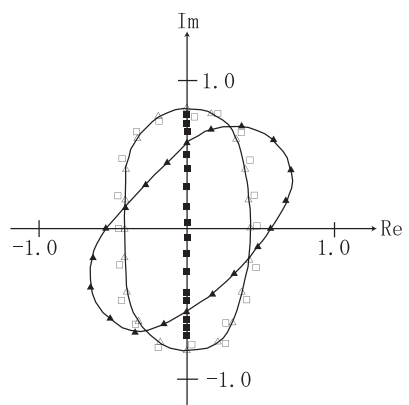
図 3.15 単語 ETL の回転



(a) 学習パターン I



(b) テストパターン S



(c) テストパターン 0

図 3.16 単語 ISO の回転

(c) 相似変換

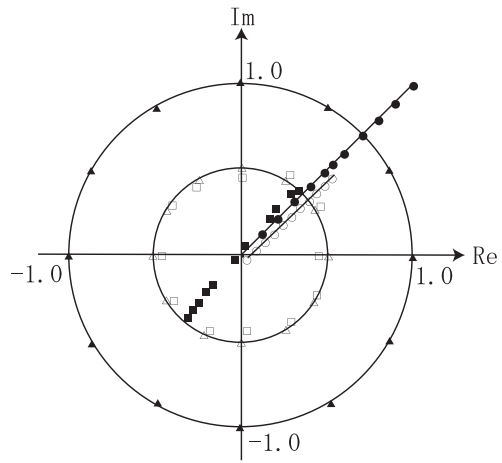
次に、縮小の例として、円 $x^2 + y^2 = 1$ から円 $x^2 + y^2 = (1/2)^2$ への相似変換を行った (図 3.17(a))。学習パターンは 11 個であり、直線 $y = x$ ($0 \leq x \leq 1$) 上の 11 個の点を入力パターン (印) とし、それらの点と原点との距離を $1/2$ にした点 (印) を出力パターンとした。出力パターンは直線 $y = x$ ($0 \leq x \leq 0.5$) 上の点になっている。学習の後、テストパターンとして、円 $x^2 + y^2 = 1$ 上の 12 個の点 (印) を与えたところ、ニューラルネットワークは円 $x^2 + y^2 = 1$ を $1/2$ に縮小した円 $x^2 + y^2 = (1/2)^2$ 上の点 (印) を出力した。因みに、実 B P では直線 $y = x$ 上の点 (印) を出力し、そのような縮小を行うことはできなかった。さらに、テストパターンとして任意の曲線上の点を与えた結果が図 3.17(b) である。円の場合と同様に曲線がその形状を保ったまま $1/2$ に縮小されていることがわかる。

次に、学習パターンとして直線 $y = x$ ($0 \leq x \leq 0.3$) 上の 11 個の点を入力パターン (印) とし、それらの点と原点との距離を 3.33 倍した点を出力パターン (印) として与えることにより、1 辺 0.3 の正方形の拡大を行なった。図 3.17(c) がその結果である。実 B P ネットワークは直線 $y = x$ 上の点列 (印) を出力したが、複素 B P ネットワークはほぼ 1 辺 1.0 の正方形を出力する傾向がうかがえる (印)。

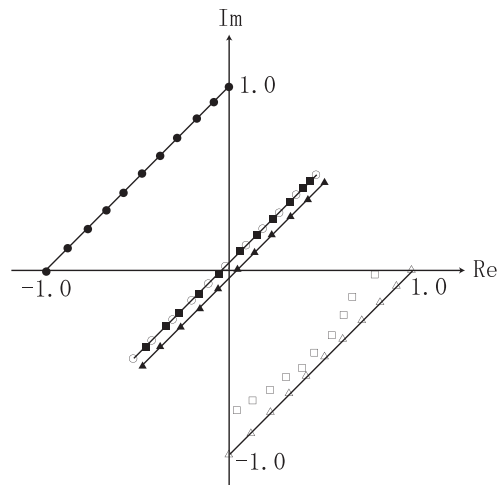
(d) 平行移動

直線の平行移動を行った (図 3.18(a))。学習パターンは 11 個であり、直線 $y = x + 1$ ($-1 \leq x \leq 0$) 上の 11 個の点を入力パターン (印) とし、それらの点を右下 45 度の方向に $1/\sqrt{2}$ だけ平行移動させた点を出力パターン (印) とした。出力パターンは直線 $y = x$ ($-0.5 \leq x \leq 0.5$) に乗った点となっている。学習の後、直線 $y = x$ ($-0.5 \leq x \leq 0.5$) 上の点 (印) を与えたところ、それらの点をほぼ右下 45 度の方向に $1/\sqrt{2}$ だけ平行移動させた点 (印) が得られたが、実 B P による実験においては平行移動は行われなかった (印)。

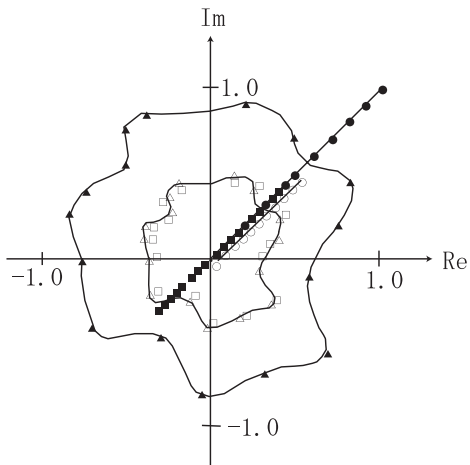
次に、任意の図形について実験したところ、図 3.18(b) に示すとおり、複素 B P ではうまく平行移動させることができた。



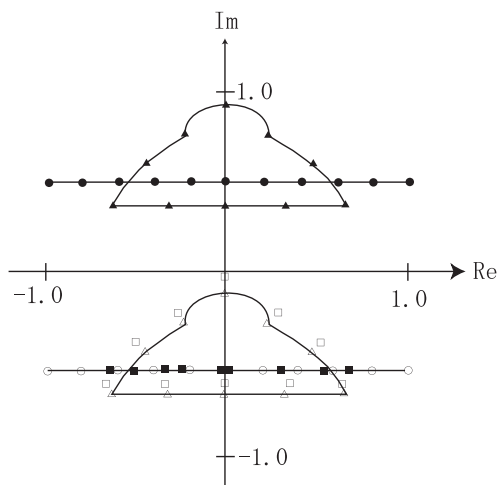
(a) 円の縮小



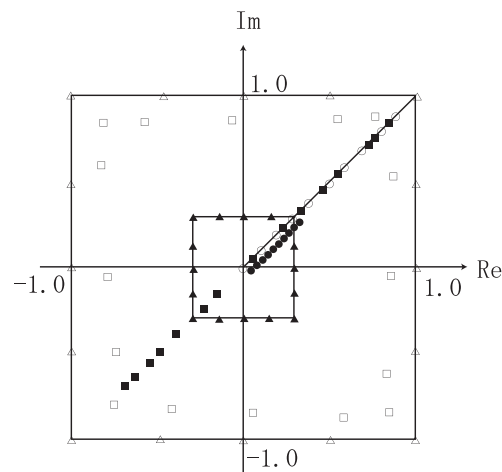
(a) 直線



(b) 曲線の縮小



(b) 曲線



(c) 正方形の拡大

図 3.18 平行移動

図 3.17 相似変換

(e) 考察

複素 BP によって学習させたニューラルネットワークには、従来の多層ニューラルネットワークには見られない 2 次元アフィン変換学習能力があることがわかった。

実験結果において、正方形が完全に拡大されなかったり (図 3.17(c)), 平行移動により直線がいくらかねじ曲がっているが (図 3.18(a)), これらの現象はシグモイド関数の出力値が完全に 1 にならないために起こるものと考えられるので、若干の工夫を施すことにより改善できるものと思われる。

一般に、2 次元平面上の点 (x, y) が原点を中心として正の方向に θ 度だけ回転することは、複素数 $z_1 = x + iy$ に、動径が 1 で偏角が θ 度の複素数 $z_2 = e^{i\theta}$ を掛け合わせることに対応する。つまり、 $z_1 z_2$ は、点 (x, y) を原点を中心として正の方向に θ 度だけ回転させた点を意味する。たとえば、回転の実験では、点をあらかず複素数 $z_1 = x + iy$ に $z_2 = e^{i90}$ または $z_2 = e^{i45}$ が掛け合わされていたことになる。また、2 次元平面上の点 (x, y) の相似変換 (縮小, 拡大) および平行移動は、それぞれ、複素数 $z_1 = x + iy$ と実数 α との乗算、複素数 $z_1 = x + iy$ と複素数 w との加算に対応する。図 3.17(a) に示す縮小は $\alpha = 0.5$ の場合、図 3.17(c) の拡大は $\alpha = 3.33$ の場合であり、図 3.18(a) に示す平行移動は $w = 0.5 - 0.5i$ の場合であった。要するに、ニューラルネットワークは複素関数 $g(z) = ze^{i\theta}$, $g(z) = \alpha z$ あるいは $g(z) = z + w$ を学習したと考えることができる。ここで、注意すべきことは、複素関数 g の定義域は $[-1, 1] \times [-1, 1]$ であるにもかかわらず、ニューラルネットワークが学習したのはその領域のある直線上の複数個の点にすぎないということである。定義域の一部の点列を学習したニューラルネットワークは、その定義域のすべての点に対して、学習した複素関数に従った出力を行っている。ニューラルネットワークのこのふるまいは複素解析における一致の定理 (たとえば、文献 [24]) との関係を持っていることが予想される。一致の定理は、実解析には見られない複素解析の特徴的な性質を示したものである。すなわち、

定理 3 (一致の定理). F, G は領域 D 上の正則関数とする。このとき、 D 内のある曲線上で $F(z) = G(z)$ ならば、 D で恒等的に $F(z) = G(z)$ である。

□

平たく言うと、一致の定理は次のようなことを主張している。平野に、高さが複素数値で表されている 2 つの山 A, B があって、それらの形状が同一であるか否かを知りたいとする。ただし、山 A と B の底面は重なり合わせることができるとする。山 A と B が、それぞれ、正則な複素数値関数 f, g (定義域は同じ) で表現できるとき、私達は山の底面内のある曲線上の山の高さが等しいか否かをチェックしさえすれば、山 A と B の形状は全く同じであるか否かを判定することができる。一方、山 A と B が正則な複素数値関数で表現できない場合、一致の定理は適用できないので、私達は山の底面内のすべての点における山の高さが等しいか否かを逐一チェックする必要がある。

そこで、この一致の定理を使って前述のニューラルネットワークのふるまいを解釈すると次のようになる。学習データは、複素関数 $G : [-1, 1] \times [-1, 1] \rightarrow [-1, 1] \times [-1, 1]$ の定義域内のある曲線上の点から得られたものとする。ニューラルネットワークは、この与えられた学習データをもとに真の複素関数 G を推定しようとし、その結果、推定関数 F を求める (ニューラルネットワークは少なくとも学習データについては $F(z) = G(z)$ としようとするはずである)。このとき、ニューラルネットワークは一致の定理を満たすかのように定義域 $[-1, 1] \times [-1, 1]$ のすべての点 z に対して真の複素関数 $G(z)$ に近い値を出力する。つまり、ニューラルネットワークが推定した複素関数 F は定義域 $[-1, 1] \times [-1, 1]$ 上で恒等的に $F(z) = G(z)$ となっていることが予想される。

この他に、新田らは、次のような結果を得た。その詳細については、文献 [66] を参照されたい。

1. 2 つの変換率 (たとえば、10 分の 1 の縮小と 2 分の 1 の縮小) を複素 BP により学習させた場合のネットワークのふるまいについて調べ、アフィン変換 (図形変換) に関する汎化能力があることを明らかにした。
2. 2 次元アフィン変換学習能力の誤差の性質が明らかになった。つまり、2 次元アフィン変換学習能力の誤差はテストパターンが入力学習パターンから離れるにつれて増加する。そして、最も距離が大きくなるあたりで誤差が最大となる。さらに、テストパターンが入力学習パターンに近づくにつれて減少する。

3. 「学習させるアフィン変換」と「学習パラメータの値」との明確な関係が実験的に示唆された。

3.1.2.4 2次元アフィン変換学習能力の数学的解析

新田は、2次元アフィン変換学習能力の数学的解析を行なった [47,60,64,66,69]。得られた主要な結果は次のとおりである。

1. 1種類の変換率(たとえば、1種類の回転角度)を学習した複素BPネットワークの定性的性質を明らかにした。変換率に関する汎化能力の(距離に関する)誤差は、未学習パターンと学習パターンの偏角の差に関する正弦関数によって表現される。
2. 2種類の変換率を学習した複素BPネットワークのふるまいを回転角度を例にとって解析した結果、次の結果を得た。
 - (a) 2種類の回転角度を学習した複素BPネットワークの回転角度に関する汎化能力の(距離に関する)誤差の評価式を与えた。
 - (b) 2種類の回転角度を学習した複素BPネットワークの回転角度に関する汎化能力の『(角度に関する)誤差が無視できるほど小さな値であるための十分条件』を求めた。
3. これらの数値例を示し、解析結果の妥当性を確認した。

ここでは、項番1の回転の場合についてのみ述べる。その他の結果については、文献 [66] を参照されたい。

以下、解析の対象とする複素BPネットワークモデルを定義し、回転を学習した複素BPネットワークの回転角度に関する汎化能力の解析を行い、その定性的な性質を明らかにする。

(a) 複素BPネットワークモデル

まず、解析の対象とするモデルの説明を行なう。解析を容易にするために、入力ニューロン1個、中間ニューロン1個、出力ニューロン1個から成る3層ネットワークを考え、入力-中間ニューロン間および中間-出力ニューロン間の重みパラメータをそれぞれ $v \exp[iw]$, $c \exp[id]$ とし、中間ニューロンおよび出力ニューロンの閾値をそれぞ

れ $s \exp[it]$, $r \exp[il]$ とする。この複素BPネットワークにおいて、複素BP学習を施した後の学習パラメータを $v^0 \exp[iw^0]$, $c^0 \exp[id^0]$, $s^0 \exp[it^0]$, $r^0 \exp[il^0]$ で表すことにする。

また、次の定数を定義しておく。

$$\begin{aligned} A &= \frac{c^0 s^0}{2(kav^0 + s^0)}, \quad B = \frac{c^0}{\sqrt{2}}, \quad C = r^0, \\ H_R &= A \cos(t^0 + d^0) + B \cos\left(d^0 + \frac{\pi}{4}\right) + C \cos(l^0), \\ H_I &= A \sin(t^0 + d^0) + B \sin\left(d^0 + \frac{\pi}{4}\right) + C \sin(l^0), \\ M &= 2K\sqrt{H_R^2 + H_I^2}. \end{aligned} \quad (3.73)$$

これらは学習終了後の学習パラメータの値に依存しており、学習毎に値は異なっていることに注意されたい。

(b) 回転の学習

回転角度を複素BPによって学習させた複素ニューラルネットワークのふるまいについて調べる。

学習パターンは、複素平面において実軸と x 度の角度をなす直線上に等間隔に並んだ p 個の点列を原点を中心に正の方向(反時計回り)に α 度だけ回転させるものとする(図3.19)。つまり、任意の $1 \leq k \leq p$ に対して、入力パターンは

$$ka \exp[ix] \quad (3.74)$$

であり、それに対応する出力パターンは

$$ka \exp[i(x + \alpha)] \quad (3.75)$$

である。ただし、 $a \in \mathbf{R}^+$ は常数であり、点列の間隔を表す。また、 $k, p \in \mathbf{N}$ (\mathbf{N} は自然数全体の集合を表す); $x, \alpha \in \mathbf{R}$; $0 < pa \leq 1$ である。ここで、式(3.75)で示される出力パターン(-1から1の間の値をとる)は、ニューラルネットワークに与える際には、0から1の間の値へ変換しておく必要がある。なぜならば、複素BPネットワークの出力値が0から1の間の値であるからである。その変換の結果、任意の $1 \leq k \leq p$ に対して、ニューラルネットワークに与えられる出力パターンは

$$\frac{1}{2}ka \exp[i(x + \alpha)] + \frac{1}{\sqrt{2}} \exp\left[i\frac{\pi}{4}\right] \quad (3.76)$$

とやや複雑な形で表される。

次に示す定理は、複素BPの回転角度に関する汎化能力の定性的性質を明らかにするものである。

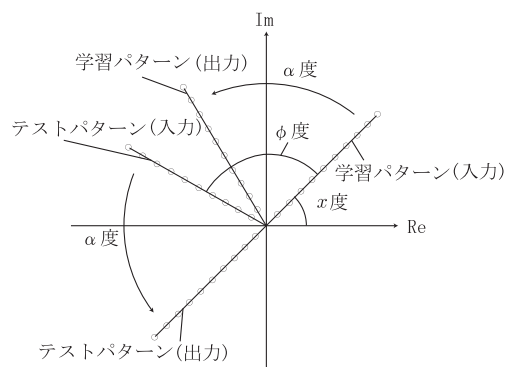


図 3.19 学習パターンとテストパターン (α 度の回転)

定理 4. $1 \leq k \leq p$ を任意に固定する。式 (3.74) および式 (3.76) で示される学習パターンを学習した複素 BP ネットワークに対して、『入力学習パターン $ka \exp[ix]$ を原点を中心にして正の方向に任意の角度 ϕ だけ回転させた点 $ka \exp[i(x+\phi)]$ 』を入力テストパターンとして与える (図 3.19)。このとき、複素 BP ネットワークが出力する値は

$$\left[\frac{1}{2}ka \exp[i(x+\phi+\alpha)] + \frac{1}{\sqrt{2}} \exp\left[i\frac{\pi}{4}\right] \right] + E^R(\phi) \in \mathbb{C} \quad (3.77)$$

で与えられる。ここで、式 (3.77) の第 1 項は未学習パターン $ka \exp[i(x+\phi)]$ を原点を中心にして正の方向に『学習した角度 α 』だけ回転させた点に相当する (図 3.19)。また、 $E^R(\phi)$ はそのときに生じるずれを表す複素数であり、その絶対値 (回転汎化誤差) は、

$$|E^R(\phi)| = M \left| \sin\left(\frac{\phi}{2}\right) \right| \quad (3.78)$$

で与えられる。□

(注意)

式 (3.78) における M は式 (3.73) で定義された常数であり、学習終了後の学習パラメータの値には依存するが、 ϕ には依存しない。

回転汎化誤差 (式 (3.78)) は、未学習パターンが学習パターンから離れる (ϕ が大きくなる) につれて増加し、最も離れた地点 ($\phi=180$ 度) において最大値 M をとり、更に学習パターンに近づくにつれて減少してゆくことを示している。数値実験を行なった結果、明らかにこの

ような傾向が見られ、この定理は複素 BP ネットワークの定性的なふるまいの大要は捉えているものと言える。

3.1.3 写像近似能力

本項では、高橋が行なった複素ニューラルネットワークの写像近似能力の解析について述べる [81]。

彼は、3 層の複素ニューラルネットワークが、与えられた正則写像を、有限個の学習データから、与えられた精度で近似できることを数学的に証明した。そして、近似することのできる最小の複素ニューラルネットワークを具体的に明示した。

3.1.3.1 正則写像

近似の対象とするのは次のような正則写像である：

$$f : \mathbb{C}^M \rightarrow \mathbb{C}^N, \quad (3.79)$$

$$f(x) = [f_1(x), \dots, f_N(x)], \quad x \in \mathbb{C}^M, \quad (3.80)$$

$$f_k : \mathbb{C}^M \rightarrow \mathbb{C}^1 \quad (1 \leq k \leq N), \quad (3.81)$$

ただし、 f_k は整関数、すなわち、 $|x| < \infty$ において正則とする。

写像される対象領域 $D \in \mathbb{C}^M$ は、単位複素超球 (単位複素超球面を含む閉集合) と 1 対 1 かつ連続的に対応する \mathbb{C}^M 内の複素領域 (領域の境界を含む閉集合) とする。たとえば、 \mathbb{C}^1 上の円板、三角形、四角形などはその例である。

\mathbb{C}^M および \mathbb{C}^N には、それぞれ、次のような距離 d^M 、 d^N を入れておく：

$$d^M(x, x') \equiv \sum_m |x_m - x'_m|, \quad x, x' \in \mathbb{C}^M, \quad (3.82)$$

$$d^N(z, z') \equiv \sum_n |z_n - z'_n|, \quad z, z' \in \mathbb{C}^N. \quad (3.83)$$

例

上記の正則写像の例を次に示す ($M = N = 1$ の場合)：

$$f(x) = xe^{i\theta}, \quad \theta \in \mathbb{R}, \quad (\text{回転}) \quad (3.84)$$

$$g(x) = rx, \quad r \in \mathbb{R}, \quad (\text{縮小・拡大}) \quad (3.85)$$

$$h(x) = x + a, \quad a \in \mathbb{R}. \quad (\text{平行移動}) \quad (3.86)$$

3.1.3.2 複素ニューラルネットワーク

次のような 3 層の複素ニューラルネットワークを考察の対象とする。入力ニューロンは、受け付けた入力をそ

のまま出力する。中間ニューロン k の出力値 y_k は、

$$y_k = \varphi_k \left(\sum_{m=1}^M \alpha_{mk} x_m - \theta_k \right) \quad (3.87)$$

とする。ここで、 $x_m \in C$ は入力ニューロン m からの入力である。中間ニューロン k の出力関数を表す $\varphi_k : C^1 \rightarrow C^1$ は正則関数とする。入力ニューロン m と中間ニューロン k との間の重み $\alpha_{mk} \in C$ は有界、つまり、ある実数 $A > 0$ が存在して、 $|\alpha_{mk}| \leq A$ とする。閾値 $\theta_k \in C$ は、有限集合 $\Theta = \{\theta^1, \dots, \theta^I\}$ 上の値をとる。また、出力ニューロン n の出力値 z_n は、

$$z_n = \sum_{k=1}^K \beta_{kn} y_k \quad (3.88)$$

とする。ここで、中間ニューロン k と出力ニューロン n との間の重み $\beta_{kn} \in C$ は有界とする。

定義 1. $Str(\Omega) \stackrel{\text{def}}{=} (K, \alpha, \theta, \beta)$ を複素ニューラルネットワーク Ω の構造という。ここで、 K は中間ニューロン数、 $\alpha = (\alpha_{mk})$ は入力ニューロンと中間ニューロンの間の重み、 $\theta = (\theta_k)$ は中間ニューロンの閾値、 $\beta = (\beta_{kn})$ は中間ニューロンと出力ニューロンの間の重みである。

また、式 (3.87) および式 (3.88) から定まる正則写像

$$\begin{aligned} \xi(x) &\equiv (\xi_1(x), \dots, \xi_N(x)), \quad x \in C^M, \\ \xi_n(x) &\equiv \sum_{k=1}^K \beta_{kn} \varphi_k \left(\sum_{m=1}^M \alpha_{mk} x_m - \theta_k \right) \\ &\quad (n = 1, \dots, N) \end{aligned} \quad (3.89)$$

を複素ニューラルネットワーク Ω が生成するニューラル写像という。□

3.1.3.3 複素ニューラルネットワーク設計問題

複素ニューラルネットワーク設計問題を次のように定式化する：

任意の正則写像 $f(x)$ と自然数 K_{\max} (使用可能な最大中間ニューロン数) および任意の非負実数 $\varepsilon \geq 0$ (許容可能な近似誤差) が与えられたものとする。このとき、 $K \leq K_{\max}$ を満たす複素ニューラルネットワーク Ω の構造 $(K, \alpha, \theta, \beta)$

の全体集合の中で、 f を D 内で ε - 近似するニューラル写像 ξ を生成し、しかも中間ニューロン数 K を最小にするという意味で最小な構造を、 f の有限個のデータ点 (学習データ) を適切に選択して構成すること。

以下、この問題を精密に定式化する。

定義 2 (ε -近似). 任意のニューラル写像 ξ が、任意の $x \in D$ に対して、

$$d^N(f(x), \xi(x)) < \varepsilon \quad (3.90)$$

を満たすとき、「 ξ は f の ε -近似である」という。□

定義 3 (ε -学習データ). 任意の複素ニューラルネットワーク Ω が生成するニューラル写像 ξ に対して、有限個の点集合

$$TD \equiv \{(a^j, b^j) \mid a^j \in D, b^j \in C^N, j = 1, \dots, J\} \quad (3.91)$$

が、条件

$$b^j = \xi(a^j), \quad (3.92)$$

$$d^N(f(a^j), \xi(a^j)) < \frac{\varepsilon}{2}. \quad (j = 1, \dots, J) \quad (3.93)$$

を満たすとき、「 TD はニューラル写像 ξ に対する f の有限な ε -学習データである」という。なお、このとき、 J を TD のデータ数と呼ぶ。□

定義 4 (設計問題). $f(x)$, K_{\max} , ε が与えられたとき、式 (3.90), (3.92), (3.93) を満たす複素ニューラルネットワーク Ω の構造 $Str(\Omega) = (K, \alpha, \theta, \beta)$ と ε -学習データ TD の組 $(Str(\Omega), TD)$ の全体集合を $S(f, K_{\max}, \varepsilon)$ と書く。このとき、複素ニューラルネットワーク設計問題は、次のように定式化できる。

$S(f, K_{\max}, \varepsilon)$ の中で、中間ニューロン数 K が最小な組 $(Str(\Omega), TD)$ を構成すること。□

次の定理は、複素ニューラルネットワーク設計問題に対する解である。

定理 5. K_{\max} が十分大きい自然数であるとき、複素ニューラルネットワーク設計問題の解 $(Str(\Omega), TD)$ は存在し、しかも、次の条件を満たすように解を構成することができる。

$$\begin{aligned} & \text{(中間ニューロン数 } K) \\ & \leq (\varepsilon\text{-学習データ } TD \text{ のデータ数 } J) + 1. \quad (3.94) \end{aligned}$$

□

以上をまとめると次のようになる：

正則写像を、有限個の ε -学習データを用いて ε -近似する複素ニューラルネットワークを理論的に構成できることを示した。さらに、その中間ニューロン数と ε -学習データ数との関係も明らかにした。

3.1.4 複素 BP 学習の高速化

これまでに提案された複素ニューラルネットワークは、実数の Back Propagation(BP) を複素数に拡張したものであり、実数の場合と同様に学習に時間がかかるという欠点をもっている。そこで高速な学習アルゴリズムが必要となる。ここでは、宮嶋等 [41] による Recursive Least-Squares(RLS) アルゴリズムを用いた複素ニューラルネットワークのための高速学習アルゴリズムを紹介する。RLS アルゴリズムは線形適応フィルタの適応アルゴリズムとして高速な収束性を持つことが知られている。評価関数に出力層のニューロンの内部ポテンシャルと望ましい内部ポテンシャル(出力関数の逆関数から求められる)の二乗誤差の和を用いることで、RLS アルゴリズムを適用することができる。このことは、通常の学習アルゴリズムでは、評価関数に出力層のニューロンの出力と望ましい出力の二乗誤差を用いることと対照的である。

L 層からなるニューラルネットワークについて考える。時刻 t での第 l 層の i 番目のニューロンの内部ポテンシャル $y_i^{(l)}$ とその出力 $x_i^{(l)}$ は次のように定義される。

$$y_i^{(l)} = \sum_{j=0}^{N_{l-1}} w_{ij}^{(l)}(t-1)x_j^{(l-1)}(t), \quad (3.95)$$

$$x_i^{(l)} = f\left(y_{iR}^{(l)}(t)\right) + if\left(y_{iI}^{(l)}(t)\right). \quad (3.96)$$

ただし、 N_l は第 l 層に含まれるニューロンの数、 $w_{ij}^{(l)}(t)$ は時刻 t での第 $l-1$ 層の j 番目のニューロンから第 l 層の i 番目のニューロンへの結合の重み係数である。出力関数は、 $f(\cdot)$ である。また、 $x_0^{(l)}(t)$ は常に 1 であり閾値を表すための定数である。

このとき評価関数は次のように定義される。

$$E(t) = \frac{1}{2} \sum_{n=0}^t \lambda^{t-n} \sum_{i=1}^{N_L} |e_i^{(L)}(n,t)|^2, \quad (3.97)$$

ただし、 $e_i^{(L)}(n,t) = d_i^{(L)}(n) - y_i^{(L)}(n,t)$ であり、 $\lambda(0 < \lambda \leq 1)$ は忘却係数である。 $y_i^{(L)}(n,t)$ と $e_i^{(L)}(n,t)$ は時刻 n での入力データに対する時刻 t での出力層の i 番目のニューロンの内部ポテンシャルと誤差をそれぞれ表しており、 $d_i^{(L)}(n)$ は時刻 n での入力データに対する出力層の i 番目のニューロンの望ましい内部ポテンシャルである。

このとき RLS アルゴリズムは次のように適用される。

[Step 1] 初期化

重み係数 $w_{ij}^{(l)}(0)$ をランダムに小さな値で初期化する。第 $l \in [2, L]$ 層の $(N_{l-1} + 1) \times (N_{l-1} + 1)$ の相関行列 $\mathbf{P}^{(l)}(0)$ を単位行列に初期化する。

[Step 2] 出力の計算

各ニューロンの内部ポテンシャル $y_i^{(l)}(t)$ および出力 $x_i^{(l)}(t)$ を計算する。

$$y_i^{(l)} = \sum_{j=0}^{N_{l-1}} w_{ij}^{(l)}(t-1)x_j^{(l-1)}(t), \quad (3.98)$$

$$x_i^{(l)} = f\left(y_{iR}^{(l)}(t)\right) + if\left(y_{iI}^{(l)}(t)\right). \quad (3.99)$$

[Step 3] カルマンゲインと相関行列の更新

第 l 層のカルマンゲイン $\mathbf{K}^{(l)}(t)$ と相関行列 $\mathbf{P}^{(l)}(t)$ を更新する。

$$\mathbf{K}^{(l)}(t) = \frac{\mathbf{P}^{(l)}(t-1)\mathbf{X}^{(l-1)*}(t)}{\lambda + \mathbf{X}^{(l-1)T}(t)\mathbf{P}^{(l)}(t-1)\mathbf{X}^{(l-1)*}(t)}, \quad (3.100)$$

$$\begin{aligned} \mathbf{P}^{(l)}(t) = & \frac{1}{\lambda} \left\{ \mathbf{P}^{(l)}(t-1) \right. \\ & \left. - \mathbf{K}^{(l)}(t)\mathbf{X}^{(l-1)T}(t)\mathbf{P}^{(l)}(t-1) \right\}. \end{aligned} \quad (3.101)$$

ただし、 $\mathbf{X}^{(l-1)}(t)$ は第 $(l-1)$ 層のニューロンの出力 $\{x_i^{(l-1)}(t)\}_{i=0}^{N_{l-1}}$ を元とするベクトルを表している。

[Step 4] 誤差信号の計算

出力層: $e_i^{(L)}(t) = d_i^{(L)}(t) - y_i^{(L)}(t)$,

中間層: $e_i^{(l)}(t) = f'(y_{iR}^{(l)}(t))\sigma_{iR}^{(l)}(t) + if'(y_{iI}^{(l)}(t))\sigma_{iI}^{(l)}(t)$

ただし、 $\sigma_i^{(l)}(t) = \sum_{j=1}^{N_{l+1}} w_{ji}^{(l+1)*}(t-1)e_j^{(l+1)}(t)$ である。

[Step 5] 重みの更新

第 $(l-1)$ 層から第 l 層の i 番目のニューロンへ結合している重みベクトルを次のように更新する。

$$\mathbf{W}_i^{(l)}(t) = \mathbf{W}_i^{(l)}(t-1) + \mathbf{K}^{(l)}(t)e_i^{(l)}(t). \quad (3.102)$$

誤差が十分に小さくなるまで [Step 2] から [Step 5] を繰り返す。

出力関数として

$$f(x) = \tanh\left(\frac{x}{2}\right), \quad (3.103)$$

を用いた場合のシミュレーション結果を図 3.20 に示す。

ただし、この RLS アルゴリズムには次のような注意すべき点がある。ニューロンへの入力数を N とするとき、BP の計算量が N に比例するのに対して、RLS アルゴリズムの計算量は N^2 に比例する。したがって、 N が大きいとき時には学習アルゴリズムの収束に要する時間は非常に長くなってしまふという欠点があることである。しかし、適応等化器などの、一定の時間間隔で学習データがシステムに与えられ、短時間で環境の変化に追従しなければならないシステムでは、学習回数が少ないことが重要になってくる。このような分野においては、ここで紹介したアルゴリズムが有効である可能性がある。

3.1.5 コンピュータビジョンへの応用

宮内らは、3.1.1.3 項で述べた新田・Benvenuto モデルをコンピュータビジョンに応用した [38-40,87]。彼らは、3.1.2.3 項において述べた 2 次元アフィン変換学習能力をうまく利用しながら、(実画像も含めた) 動きの解釈を行なわせ、その有効性を確認した。以下、そのベースになった考え方を述べる。

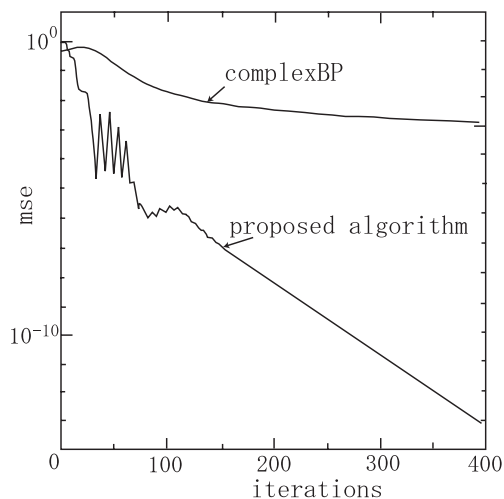


図 3.20 学習曲線

コンピュータビジョン [10,21,79] は、目で外界を見てその状況を認識、理解する人間の視覚情報処理機能を、機械を用いて実現させることを目的とした研究分野である。具体的には、カメラを通じて得られた画像から、外界の物理的対象物に対する明白で意味のある記述を計算機を通して構成することを目的としている。

コンピュータビジョンにおいて、動いている物体の動画像 (異なる時刻で得られた画像の集まり) から物体の動きパラメータを推定することは重要な課題である [2]。多くの方法は、動画像からオプティカル・フローと呼ばれる画像上の 2 次元速度ベクトル場を計算することを前処理として行ない、次にそのオプティカル・フローから画像中の動物体の動きパラメータを推定する。オプティカル・フローから動きパラメータを推定する方法の多くは、オプティカル・フロー中のいくつかのフロー・ベクトルを使って方程式を解いている [1,80,84]。しかし、この方法では時間がかかること、雑音に弱いこと、実画像に対する解が容易に得られない、といった解決すべき課題がある。

ニューラルネットワークはパターン変換にしばしば利用され、雑音にそれほど影響を受けないという特徴がある。しかも、学習後の処理に要する時間は短い。よって、これらの点で、動きパラメータの推定に応用すると有効であると考えられる。

特に、複素 BP ネットワークは入出力パターンが複素

パターン (2次元情報) であるので, 2次元ベクトルから構成されるオプティカル・フローを自然に扱うことができる。つまり, オプティカル・フロー中の2次元ベクトルを, 1つの複素数として, 複素ニューロンに素直に入出力させることができる。図 3.21 に動きの解釈を行なう複素BPネットワークの構成例を示す。入力はおプティカル・フロー, 出力は動きパラメータである。ここで, 複素BPネットワークには, 1つの平行移動や1つの回転といった基本的な動きに関するパターンを教師データとして予め学習させておく。基本的な動きを学習した複素BPネットワークは未学習の動きに対してもうまく解釈を行なうことができる。

さらに, 実画像のオプティカル・フローは必ずしも完全なものが得られるわけではなく, 1部分しか得られないことが多い(図 3.22)。このため, 正規化を行なう必要があり, この正規化に複素BPの2次元アフィン変換学習能力を利用する。つまり, 得られたオプティカル・フローの1部分を $1-n-1$ 複素BPネットワークに学習させ, オプティカル・フローの残りの部分を推定させるといった使い方をするのである。図 3.23 にオプティカル・フローの正規化を行なう複素BPネットワークの構成例を示す。フロー・ベクトルの始点を表す2次元座標を入力ニューロンに, 終点を表す2次元座標を出力ニューロンに対応させて利用する。

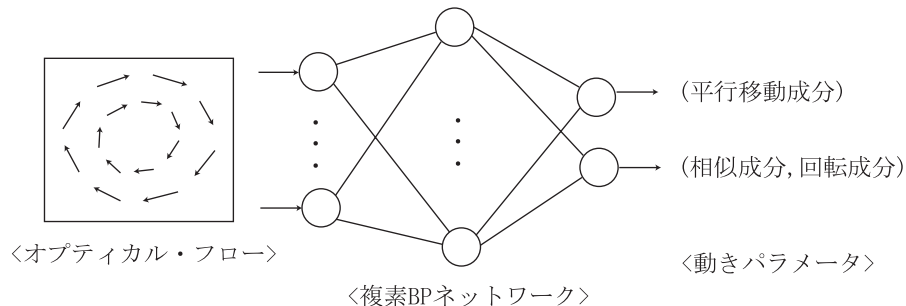


図 3.21 動きの解釈を行う複素BPネットワークの構成例

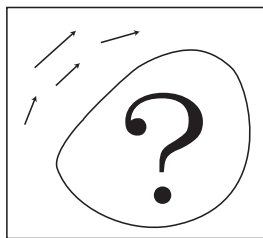


図 3.22 1部分だけが得られているオプティカル・フローのイメージ

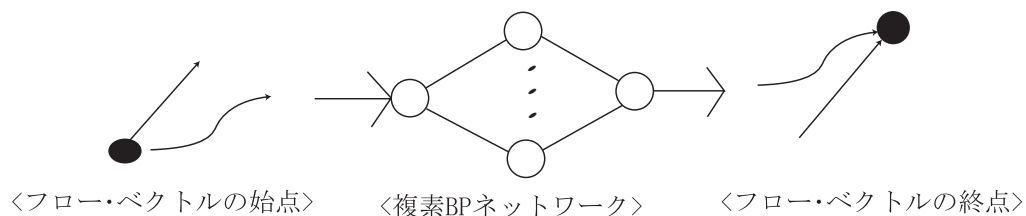


図 3.23 オプティカル・フローの正規化を行う複素BPネットワークの構成例

3.2 多層型 3 次元ニューラルネットワーク

本節では、多層型の 3 次元ニューラルネットワークについて述べる。

3.2.1 3 次元ベクトルニューラルネットワーク

新田らは、3.1.1.3 項で述べた複素 B P を、群論に基づいて自然な形で 3 次元パラメータ化した 3 次元ベクトル・バックプロパゲーション学習アルゴリズム (以下、3 次元ベクトル B P と呼ぶ) を提案した [45,49,56,66]。3 次元ベクトル B P は、閾値、入力信号および出力信号がすべて 3 次元実ベクトル、重みパラメータが 3 次直交行列である階層型ニューラルネットワークに適用される。さらに、3 次元ベクトル B P には、従来のニューラルネットワークには見られない 3 次元アフィン変換学習能力が備わっていることを確認した [56,57,63,66]。この能力は複素 B P が備えていた 2 次元アフィン変換学習能力に対応するものである。

まず、3 次元ベクトル B P を定式化し、次に、3 次元ベクトル B P の 3 次元アフィン変換学習能力について述べる。

(a) 3 次元ベクトルニューロン

実 B P は実数を扱うという点で 1 次元的であり、複素 B P は複素数を扱うという点でいわば 2 次元的である。そのような実 B P あるいは複素 B P をさらに高次元パラメータ化する方法はいくつか考えられる。ひとつは、実数 α (1 次元) を複素数 $z = x + iy$ (2 次元)、4 元数 $q = a + ib + jc + kd$ (4 次元)、8 元数 (8 次元)、16 元数 (16 次元)、... へと拡張してゆく方法である。もうひとつは、重みおよび閾値の次元数を 1 次元あるいは 2 次元から n 次元実ベクトルを使って高次元パラメータ化する方法である。ここでは、後者の方法を使って 3 次元パラメータ化を試みる。

次のようなニューロン (3 次元ベクトルニューロンと呼ぶ) を考える。入力信号、閾値および出力信号はすべて 3 次元実ベクトルであり、重みはすべて 3 次行列である。

ニューロン j の内部ポテンシャル A_j は、実 B P において用いられるニューロンの内部ポテンシャルの類推で、

$$A_j = \sum_k W_{jk} S_k + T_j \quad (3.104)$$

と定義する。ここで、 W_{jk} はニューロン j とニューロン k との間の重みを表す 3 次行列、 S_k はニューロン k から出力されたニューロン j への入力信号を表す 3 次元実ベクトル、 T_j はニューロン j の閾値を表す 3 次元実ベクトルである。出力信号 $F_3(A_j)$ は次のように定義される：

$$F_3(A_j) = \begin{bmatrix} f_R(a_1) \\ f_R(a_2) \\ f_R(a_3) \end{bmatrix}, \quad A_j = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}. \quad (3.105)$$

$F_3(A_j)$ の各成分は、それぞれ、 A_j のそれぞれの各成分 a_m ($m = 1, 2, 3$) に関するシグモイド関数値である。

上記の定式化において、出力関数 F_3 にはいろいろなものを考えることができ、それらに応じてニューロンにはそれら固有の性質が備わることになる。また、一般に、重みを表す 3 次行列にはいろいろな制限を与えることができる。たとえば、正則性、対称性、直交性などであるが、これらはニューロンのふるまいに影響を与えることとなる。ここでは、重みは 3 次直交行列にしておく。この 3 次直交行列は、複素 B P において使われている重みの自然な拡張となっているからである。以下、このことを示す。複素 B P ネットワークにおけるニューロンの構造は次のとおりである。重み $w_k = w_k^r + iw_k^i \in C$ ($1 \leq k \leq n$) および閾値 $\theta = \theta^r + i\theta^i \in C$ を持つ n 入力ニューロンをひとつ考える。このとき、入力信号 $x_k + iy_k \in C$ ($1 \leq k \leq n$) が与えられると、このニューロンは $X + iY \in C$ を出力する。ここで、

$$\begin{aligned} & X + iY \\ &= f_C \left(\sum_{k=1}^n (w_k^r + iw_k^i)(x_k + iy_k) + (\theta^r + i\theta^i) \right) \\ &= f_R \left(\sum_{k=1}^n (w_k^r x_k - w_k^i y_k) + \theta^r \right) \\ &+ i f_R \left(\sum_{k=1}^n (w_k^i x_k + w_k^r y_k) + \theta^i \right). \end{aligned} \quad (3.106)$$

さらに、

$$\begin{bmatrix} X \\ Y \end{bmatrix}$$

$$\begin{aligned}
&= F_C \left(\begin{array}{c} \left[\begin{array}{cc|cc} w_1^r & -w_1^i & \cdots & w_n^r & -w_n^i \\ w_1^i & w_1^r & \cdots & w_n^i & w_n^r \end{array} \right] \begin{array}{c} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{array} \\ + \begin{array}{c} \theta^r \\ \theta^i \end{array} \end{array} \right) \\
&= F_C \left(|w_1| \begin{bmatrix} \cos \alpha_1 & -\sin \alpha_1 \\ \sin \alpha_1 & \cos \alpha_1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \cdots \right. \\
&\quad \left. + |w_n| \begin{bmatrix} \cos \alpha_n & -\sin \alpha_n \\ \sin \alpha_n & \cos \alpha_n \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \begin{array}{c} \theta^r \\ \theta^i \end{array} \right), \quad (3.107)
\end{aligned}$$

ここで、 $F_C \left(\begin{bmatrix} x \\ y \end{bmatrix} \right) = \begin{bmatrix} f_R(x) \\ f_R(y) \end{bmatrix}$, $\alpha_k = \arctan(w_k^i/w_k^r)$ ($1 \leq k \leq n$) である。式 (3.107) において、 $\begin{bmatrix} w_k^r & -w_k^i \\ w_k^i & w_k^r \end{bmatrix}$ は 2 次直交群 $O_2(\mathbf{R})$ の元である。このように、重みを 3 次直交群 $O_3(\mathbf{R})$ の元である 3 次直交行列にすることは自然であると言える。さらに、出力関数 F_3 の形態などについても先に与えられた 3 次元ベクトルニューロンの定式化は自然であることがわかる。ちなみに、 $\begin{bmatrix} \cos \alpha_k & -\sin \alpha_k \\ \sin \alpha_k & \cos \alpha_k \end{bmatrix}$ は 2 次回転群 $SO_2(\mathbf{R})$ の元であり、 $|w_k|$ は $\mathbf{R}^* = \{x \in \mathbf{R} \mid x > 0\}$ (乗法に関する群) の元である。したがって、 $\begin{bmatrix} w_k^r & -w_k^i \\ w_k^i & w_k^r \end{bmatrix}$ は群 $\mathbf{R}^* \times SO_2(\mathbf{R})$ の元でもあるわけである。

(b) 3次元ベクトルニューラルネットワーク

次に、3次元ベクトルBPが適用されるネットワークを、(a) で定義した 3次元ベクトルニューロンを使って構成する。一般に、3次元ベクトルニューロンから構成されるニューラルネットワークを 3次元ベクトルニューラルネットワークと呼ぶことにするが、ここで定義するネットワークは、3次元ベクトルBPが適用されるネットワークという意味で 3次元ベクトルBPネットワークと呼ぶことにする。簡単のために、3層ネットワークを例にとる。

入力ニューロン i と中間ニューロン j との間の重みを

$W_{ji} \in O_3(\mathbf{R})$ ($O_3(\mathbf{R})$ は 3 次直交群), 中間ニューロン j と出力ニューロン k との間の重みを $V_{kj} \in O_3(\mathbf{R})$, 中間ニューロン j の閾値を $\Theta_j = {}^t[\theta_j^x \ \theta_j^y \ \theta_j^z] \in \mathbf{R}^3$, 出力ニューロン k の閾値を $\Gamma_k = {}^t[\gamma_k^x \ \gamma_k^y \ \gamma_k^z] \in \mathbf{R}^3$ とする。また, 入力ニューロン i への入力信号を $I_i = {}^t[I_i^x \ I_i^y \ I_i^z] \in \mathbf{R}^3$ とし, 中間ニューロン j および出力ニューロン k の出力信号を, それぞれ, $H_j = {}^t[H_j^x \ H_j^y \ H_j^z] \in \mathbf{R}^3$, $O_k = {}^t[O_k^x \ O_k^y \ O_k^z] \in \mathbf{R}^3$ とする。さらに, $\Delta^k = {}^t[\delta_k^x \ \delta_k^y \ \delta_k^z] = T_k - O_k \in \mathbf{R}^3$ を O_k と出力ニューロン k に対する教師パターン (出力学習パターン) $T_k = {}^t[T_k^x \ T_k^y \ T_k^z] \in \mathbf{R}^3$ との間の誤差とする。パターン p に対する 2 乗誤差を $E_p = (1/2) \sum_{k=1}^N \|T_k - O_k\|^2$, と定義する。ただし, $\|x\| = \sqrt{x_1^2 + x_2^2 + x_3^2}$, $x = {}^t[x_1 \ x_2 \ x_3]$, N は出力ニューロンの総数である。重みは, ここでは, 特に次のように定義する:

$$W_{ji} = \begin{bmatrix} w_{ji}^x & -w_{ji}^y & 0 \\ w_{ji}^y & w_{ji}^x & 0 \\ 0 & 0 & w_{ji}^z \end{bmatrix} \in O_3(\mathbf{R}), \quad (3.108)$$

ただし, $w_{ji}^z = \sqrt{(w_{ji}^x)^2 + (w_{ji}^y)^2}$,

$$V_{kj} = \begin{bmatrix} v_{kj}^x & 0 & 0 \\ 0 & v_{kj}^y & -v_{kj}^z \\ 0 & v_{kj}^z & v_{kj}^y \end{bmatrix} \in O_3(\mathbf{R}), \quad (3.109)$$

ただし, $v_{kj}^x = \sqrt{(v_{kj}^y)^2 + (v_{kj}^z)^2}$.

式 (3.108) および式 (3.109) における W_{ji} , V_{kj} は群 $\mathbf{R}^* \times SO_3(\mathbf{R})$ の元でもある ($SO_3(\mathbf{R})$ は 3 次回転群) なげならば, これらは次のようにも表現できるからである:

$$\begin{aligned}
W_{ji} &= w_{ji}^z A(\alpha_{ji}) \\
&= w_{ji}^z \begin{bmatrix} \cos \alpha_{ji} & -\sin \alpha_{ji} & 0 \\ \sin \alpha_{ji} & \cos \alpha_{ji} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (3.110)
\end{aligned}$$

$$\begin{aligned}
V_{kj} &= v_{kj}^x B(\beta_{kj}) \\
&= v_{kj}^x \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \beta_{kj} & -\sin \beta_{kj} \\ 0 & \sin \beta_{kj} & \cos \beta_{kj} \end{bmatrix}, \quad (3.111)
\end{aligned}$$

ただし, $\alpha_{ji} = \arctan(w_{ji}^y/w_{ji}^x)$, $\beta_{kj} = \arctan(v_{kj}^z/v_{kj}^y)$. ここで, $B(\beta_{kj})$ は 3 次回転群 $SO_3(\mathbf{R})$ の元の標準形である, つまり, 任意の $X \in SO_3(\mathbf{R})$ に対して, ある直交行

列 $Y \in O_3(\mathbf{R})$ が存在して, $YXY^{-1} = B(\beta_{kj})$ 。また, $B(\beta_{kj})$ は X 軸のまわりの右まわりの回転, $A(\alpha_{ji})$ は Z 軸のまわりの右まわりの回転を表している。

(c) 学習アルゴリズム

次に, 3次元ベクトルBPネットワークに対する学習アルゴリズムを導出する。十分小さい学習率 $\varepsilon > 0$ に対して, 最急降下法を使うことにより,

$$\begin{aligned} & \begin{bmatrix} \Delta v_{kj}^y \\ \Delta v_{kj}^z \end{bmatrix} \\ &= \begin{cases} \begin{bmatrix} (v_{kj}^y/v_{kj}^x)H_j^x & H_j^y & H_j^z \\ (v_{kj}^z/v_{kj}^x)H_j^x & -H_j^z & H_j^y \end{bmatrix} \begin{bmatrix} \Delta \gamma_k^x \\ \Delta \gamma_k^y \\ \Delta \gamma_k^z \end{bmatrix} \\ \quad (v_{kj}^x \neq 0 \text{ のとき}) \\ \begin{bmatrix} 0 & H_j^y & H_j^z \\ 0 & -H_j^z & H_j^y \end{bmatrix} \begin{bmatrix} \Delta \gamma_k^x \\ \Delta \gamma_k^y \\ \Delta \gamma_k^z \end{bmatrix} \\ \quad (v_{kj}^x = 0 \text{ のとき}), \end{cases} \end{aligned} \quad (3.112)$$

$$\begin{aligned} & \begin{bmatrix} \Delta \gamma_k^x \\ \Delta \gamma_k^y \\ \Delta \gamma_k^z \end{bmatrix} \\ &= \varepsilon \begin{bmatrix} (1 - O_k^x)O_k^x & 0 & 0 \\ 0 & (1 - O_k^y)O_k^y & 0 \\ 0 & 0 & (1 - O_k^z)O_k^z \end{bmatrix} \begin{bmatrix} \delta_k^x \\ \delta_k^y \\ \delta_k^z \end{bmatrix}, \end{aligned} \quad (3.113)$$

$$\begin{aligned} & \begin{bmatrix} \Delta w_{ji}^x \\ \Delta w_{ji}^y \end{bmatrix} \\ &= \begin{cases} \begin{bmatrix} I_i^x & I_i^y & (w_{ji}^x/w_{ji}^z)I_i^z \\ -I_i^y & I_i^x & (w_{ji}^y/w_{ji}^z)I_i^z \end{bmatrix} \begin{bmatrix} \Delta \theta_j^x \\ \Delta \theta_j^y \\ \Delta \theta_j^z \end{bmatrix} \\ \quad (w_{ji}^z \neq 0 \text{ のとき}) \\ \begin{bmatrix} I_i^x & I_i^y & 0 \\ -I_i^y & I_i^x & 0 \end{bmatrix} \begin{bmatrix} \Delta \theta_j^x \\ \Delta \theta_j^y \\ \Delta \theta_j^z \end{bmatrix} \\ \quad (w_{ji}^z = 0 \text{ のとき}), \end{cases} \end{aligned} \quad (3.114)$$

$$\begin{aligned} & \begin{bmatrix} \Delta \theta_j^x \\ \Delta \theta_j^y \\ \Delta \theta_j^z \end{bmatrix} \\ &= \begin{bmatrix} (1 - H_j^x)H_j^x & 0 & 0 \\ 0 & (1 - H_j^y)H_j^y & 0 \\ 0 & 0 & (1 - H_j^z)H_j^z \end{bmatrix} \\ & \sum_k \begin{bmatrix} v_{kj}^x & 0 & 0 \\ 0 & v_{kj}^y & v_{kj}^z \\ 0 & -v_{kj}^z & v_{kj}^y \end{bmatrix} \begin{bmatrix} \Delta \gamma_k^x \\ \Delta \gamma_k^y \\ \Delta \gamma_k^z \end{bmatrix} \end{aligned} \quad (3.115)$$

が得られる。

(d) 3次元アフィン変換学習能力

次に, 複素BPの自然な拡張である3次元ベクトルBPに3次元アフィン変換学習能力(3次元図形変換能力)が備わっていることを計算機実験により検証する。

新田は, 回転, 縮小拡大, 平行移動といった単純な3次元アフィン変換学習能力について調べた後, やや複雑な3次元アフィン変換学習能力について調べ, さらに実BPにはこのような能力は無いことも示している。単純な場合とは, たとえば, (3次元空間における)“原点との距離に関する2分の1の縮小”などのように単独の縮小率を学習させる場合である。単純な場合の詳細については文献[56,66]を参照されたい。ここでは, やや複雑な場合について述べる。

実験に用いるネットワークは, 入力層1ニューロン, 中間層6ニューロン, 出力層1ニューロンの3層構造である。つまり, 1つの3次元ベクトル X を入力し, 1つの3次元ベクトル X' を出力するものである。一般に, 1つの3次元ベクトル $X = {}^t[x_1 \ x_2 \ x_3]$ は3次元空間内の1点 (x_1, x_2, x_3) に対応する。よって, 当該ネットワークは3次元空間内のある1点 (x_1, x_2, x_3) を他の1点 (x'_1, x'_2, x'_3) に変換するものであると考えることができる。ここでは, 対象とする空間を, x 軸, y 軸および z 軸の取る値が -1 から 1 までであるような空間に制限しておく。ニューラルネットワークが出力する値は 0 から 1 の間の値であるが, 便宜上, 以下に示す図では, それらを -1 から 1 の間の値に変換した値で示す。

学習率 ε は 0.5 とし, 学習パラメータの初期値は -0.3

と +0.3 の間の値をとる乱数により設定した。学習は $\sqrt{2 \sum_p E_p} = 0.05$ となったときに収束 (終了) したものとみなし、収束の規範とした。また、学習パターンは、通常、数種類のパターンから構成されているが、その数種類のパターンをひと通り提示することで 1 回の学習と数えることとした。

異なる 2 つの直線上の点を原点との距離に関して、それぞれ、2 分の 1、10 分の 1 に縮小させるパターンを学習させた (図 3.24)。学習パターンは 2 1 個であり、直線

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = t \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (-1.0 \leq t \leq 1.0) \quad (3.116)$$

上の 2 1 個の点 (点列の間隔は $\sqrt{3}/10$) を入力パターンとした。直線

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = t \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (0.0 \leq t \leq 1.0) \quad (3.117)$$

上の 1 1 個の点に対しては、それらの点と原点との距離を 2 分の 1 にした点を出力パターンとし、直線

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = t \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (-1.0 \leq t \leq 0.0) \quad (3.118)$$

上の 1 1 個の点に対しては、それらの点と原点との距離を 10 分の 1 にした点を出力パターンとした。ここで、前者の学習パターンを学習パターン 1、後者の学習パターンを学習パターン 2 と呼ぶことにする。出力学習パターン 1 は、直線

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = t \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (0.0 \leq t \leq 0.5) \quad (3.119)$$

上に乗っており、出力学習パターン 2 は、直線

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = t \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (-0.1 \leq t \leq 0.0) \quad (3.120)$$

上に乗っていることになる。

学習終了後、入力テストパターンとして、3 つの円 $x^2 + z^2 = 1$, $y^2 + z^2 = 1$, $x^2 + y^2 = 1$ の円周上の

60 個の点を与えたところ、ニューラルネットワークは図 3.25 に示すようなパターンを出力した。図 3.25 において、入力学習パターン 1 に近い入力テストパターンほど 2 分の 1 に縮小され、入力学習パターン 2 に近い入力テストパターンほど 10 分の 1 に縮小されるという傾向性が見られる。つまり、0.1 と 0.5 という 2 つの縮小率が反映された結果となっているようである。

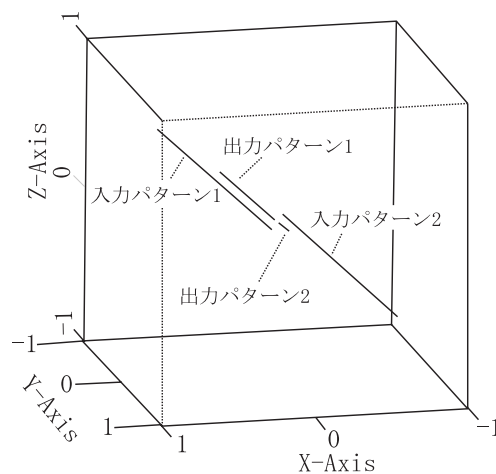


図 3.24 学習パターン

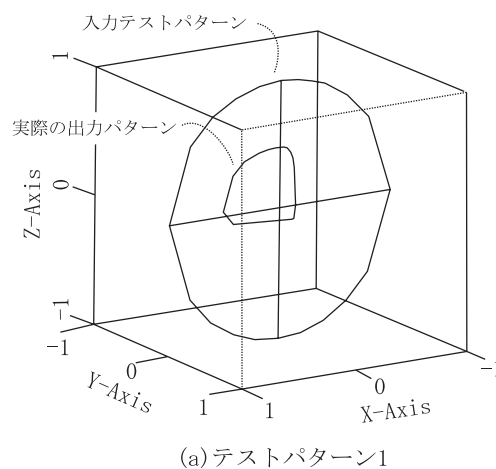


図 3.25 テストパターン

3.2.2 3次元外積ニューラルネットワーク

新田は、3.2.1 項とは異なった観点から実 B P の 3 次元パラメータ化を行なった [51,53,54,66]。すなわち、外積

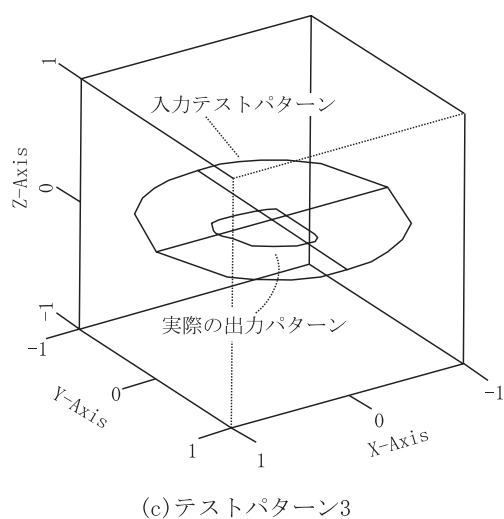
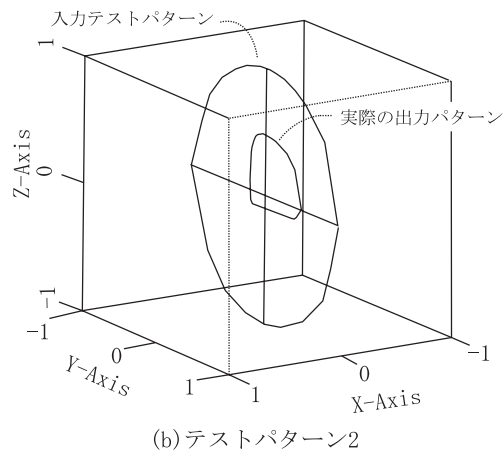


図 3.25 テストパターン

を拡張の基軸とした。3次元ベクトルニューラルネットワーク (3.2.1 項) と異なるのは、重みが3次直交行列から3次元実ベクトルになっており、その影響により、3次元ベクトルニューラルネットワークにおいて行なわれていた行列演算が外積演算になっているという点である。このようなニューラルネットワークに適用されるバックプロパゲーション学習アルゴリズム (以下、外積BPと呼ぶ) を導出する。

(a) 外積ニューロン

3.2.1 項でも述べたように、実BPあるいは複素BPをさらに高次元パラメータ化する方法にはいくつか考えられる。ひとつは、実数 α (1次元) を複素数 $z = x + iy$ (2次元, 3.1.1.3 項), 4元数 $q = a + ib + jc + kd$ (4次元),

8元数 (8次元), 16元数 (16次元), ... へと拡張してゆく方法である。もうひとつは、重みおよび閾値の次元数を1次元から n 次元実ベクトルを使って高次元パラメータ化する方法である。後者の方法にはさらに次の2つの方法が考えられる: (1) 重みを n 次行列 (3.2.1 項) とする方法, (2) 重みを n 次元実ベクトルとする方法。本章では, (2) の方法を使って3次元パラメータ化を試みる。その際, 3次元実ベクトルと3次元実ベクトルとから3次元実ベクトルを生成することが必要となる。そのような演算を自由に創ることもできるが, ここでは, 数学において標準的に使用されている外積を採用する。

次のようなニューロン (外積ニューロンと呼ぶ) を考える。入力信号, 重み, 閾値および出力信号はすべて3次元実ベクトルである。ニューロン j の内部ポテンシャル A_j は, 実BPにおいて用いられるニューロンの内部ポテンシャルの類推で,

$$A_j = \sum_k (W_{jk} \times S_k) + T_j \quad (3.121)$$

と定義する。ここで, W_{jk} はニューロン j とニューロン k との間の重みを表す3次元実ベクトル, S_k はニューロン k から出力されたニューロン j への入力信号を表す3次元実ベクトル, T_j はニューロン j の閾値を表す3次元実ベクトルである。また, $x \times y$ は $x = {}^t[x_1 \ x_2 \ x_3]$ と $y = {}^t[y_1 \ y_2 \ y_3]$ の外積を表している。すなわち, $x \times y = {}^t[x_2y_3 - x_3y_2 \ x_3y_1 - x_1y_3 \ x_1y_2 - x_2y_1]$ である。出力信号 $F_3(A_j)$ は次のように定義される:

$$F_3(A_j) = \begin{bmatrix} f_R(a_1) \\ f_R(a_2) \\ f_R(a_3) \end{bmatrix}, \quad A_j = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}. \quad (3.122)$$

$F_3(A_j)$ の各成分は, それぞれ, A_j のそれぞれの各成分 a_m ($m = 1, 2, 3$) に関するシグモイド関数値である。この出力関数 F_3 は式 (3.105) において定義したものと同一のものである。

(b) 外積ニューラルネットワーク

次に, 外積BPが適用されるネットワークを, 上で定義した外積ニューロンを使って構成する。一般に, 外積ニューロンから構成されるニューラルネットワークを外積ニューラルネットワークと呼ぶことにするが, ここで

定義するネットワークは、外積BPが適用されるネットワークという意味で外積BPネットワークと呼ぶことにする。簡単のために、3層ネットワークを例にとる。

入力ニューロン i と中間ニューロン j との間の重みを $W_{ji} = {}^t[w_{ji}^x, w_{ji}^y, w_{ji}^z] \in \mathbf{R}^3$, 中間ニューロン j と出力ニューロン k との間の重みを $V_{kj} = {}^t[v_{kj}^x, v_{kj}^y, v_{kj}^z] \in \mathbf{R}^3$, 中間ニューロン j の閾値を $\Theta_j = {}^t[\theta_j^x, \theta_j^y, \theta_j^z] \in \mathbf{R}^3$, 出力ニューロン k の閾値を $\Gamma_k = {}^t[\gamma_k^x, \gamma_k^y, \gamma_k^z] \in \mathbf{R}^3$ とする。また、入力ニューロン i への入力信号を $I_i = {}^t[I_i^x, I_i^y, I_i^z] \in \mathbf{R}^3$ とし、中間ニューロン j および出力ニューロン k の出力信号を、それぞれ、 $H_j = {}^t[H_j^x, H_j^y, H_j^z] \in \mathbf{R}^3$, $O_k = {}^t[O_k^x, O_k^y, O_k^z] \in \mathbf{R}^3$ とする。さらに、 $\Delta^k = {}^t[\delta_k^x, \delta_k^y, \delta_k^z] = T_k - O_k \in \mathbf{R}^3$ を O_k と出力ニューロン k に対する教師パターン(出力学習パターン) $T_k = {}^t[T_k^x, T_k^y, T_k^z] \in \mathbf{R}^3$ との間の誤差とする。パターン p に対する 2乗誤差を $E_p = (1/2) \sum_{k=1}^N \|T_k - O_k\|^2$, と定義する。ただし、 $\|x\|^2 = \sqrt{x_1^2 + x_2^2 + x_3^2}$, $x = {}^t[x_1, x_2, x_3]$, N は出力ニューロンの総数である。

(c) 学習アルゴリズム

次に、学習アルゴリズムを導出する。十分小さい学習率 $\varepsilon > 0$ に対して、最急降下法を使うことにより、学習パラメータは次の式にしたがって修正すべきであることがわかる。 Δx は学習パラメータ x の修正量を表す。

$$\Delta V_{kj} = H_j \times \Delta \Gamma_k, \quad (3.123)$$

$$\Delta \Gamma_k = \varepsilon D_k \Delta_k, \quad (3.124)$$

$$\Delta W_{ji} = I_i \times \Delta \Theta_j, \quad (3.125)$$

$$\Delta \Theta_j = D_j \sum_k (\Delta \Gamma_k \times V_{kj}), \quad (3.126)$$

ただし、

$$D_k = \begin{bmatrix} (1 - O_k^x)O_k^x & 0 & 0 \\ 0 & (1 - O_k^y)O_k^y & 0 \\ 0 & 0 & (1 - O_k^z)O_k^z \end{bmatrix}, \quad (3.127)$$

$$D_j = \begin{bmatrix} (1 - H_j^x)H_j^x & 0 & 0 \\ 0 & (1 - H_j^y)H_j^y & 0 \\ 0 & 0 & (1 - H_j^z)H_j^z \end{bmatrix}. \quad (3.128)$$

さらに、新田は、外積BPの学習特性について調べ、実BPに比べて優れているとの感触を得た。しかし、明確な結論を得るためには、より体系的な実験を要するようである。また、複素BPには2次元アフィン変換学習能力が、3次元ベクトルBPには3次元アフィン変換学習能力がそれぞれ備わっていたように、外積BPにも何らかの固有の性質が備わっていることが予想される。

3.2.3 コンピュータビジョンへの応用

宮内らは、3.2.1項で述べた3次元ベクトルBPをコンピュータビジョンに応用した[85-87,91]。彼らは、3次元ベクトルBPネットワークに3次元の動きの解釈を行なわせ、ノイズの影響にも強いことを含めて、その有効性の確認を行なった。以下、そのベースになった考え方を述べる。

3.1.5項において述べた複素BPの応用は、単眼視(1つのカメラ)によりオプティカル・フローから動物体の(画像平面上の)動きを解釈する方法であった。ここで述べる3次元ベクトルBPの応用は、3.1.5項で述べた方法を基礎にして3眼視(3つのカメラ)による動物体の(3次元空間内の)動きの解釈を行なう方法である。ここでは、3.1.5項で述べた単眼視により得られる「画像平面上の動きを規定するパラメータ」を2次元動きパラメータと呼ぶことにする。そして、この2次元動きパラメータを出力する複素BPネットワーク(図3.21)を2次元動き解釈ネットワークと呼ぶことにする。

図3.26に3次元ベクトルBPを利用した3次元空間内の動きの解釈を行なう処理系の構成例を示す。まず、各カメラ画像のオプティカル・フローをそれぞれ2次元動き解釈ネットワークを通して解釈を行ない、2次元動きパラメータを出力する。その結果を3次元ベクトルBPネットワーク(3次元動き解釈ネットワークと呼ぶ)に入力し、その出力を3次元空間内の動きを表すパラメータ(3次元動きパラメータと呼ぶ)の解釈値とする。ここで、3次元動き解釈ネットワークには、1つの並進運動や1つの回転といった基本的な3次元動きに関するパターンを教師データとして予め学習させておく。基本的な3次元動きを学習した3次元動き解釈ネットワークは、学習していない3次元動きに対してもうまく解釈する能力を示すことが期待される。図3.27に示すように、3次元

動き解釈ネットワークへの入力となるのは、各カメラ視点ごとに得られる2次元動きパラメータである。つまり、 $\{(dx_2, dy_2, dz_2)\}$ および $(\omega z_2(1), \omega z_2(2), \omega z_2(3))$ といった4つの3次元ベクトルである。

ここで、 dx_2 は x 軸方向の並進成分、 dy_2 は y 軸方向の並進成分、 dz_2 は拡大・縮小成分、そして $\omega z_2(i)$ はカメラ i により得られた回転成分である ($i = 1, 2, 3$)。いわば、 $\{(dx_2, dy_2, dz_2)\}$ は並進成分、 $(\omega z_2(1), \omega z_2(2), \omega z_2(3))$ は回転成分である。そして、3次元動き解釈ネットワークは、3次元の動きを表す3次元動きパラメータ、すなわち、対象物体の3次元空間における x, y, z 軸方向への並進成分 (dx, dy, dz) と各軸毎の回転成分 $(\omega x, \omega y, \omega z)$ の2つの3次元ベクトルを出力する。このように、3次元

ベクトルBPネットワークは3次元ベクトルから構成される入出力パターンを想定しているので、3次元情報を自然に扱うことができる。

3組のカメラ画像の動きから連立方程式を解くことにより、3次元の動きを計算する一般的な手法の場合、計算量が多く非効率的であるのに対し、3次元ベクトルBPネットワークを使用するこの手法は、学習後の処理に要する時間が短く、しかも雑音に強い。

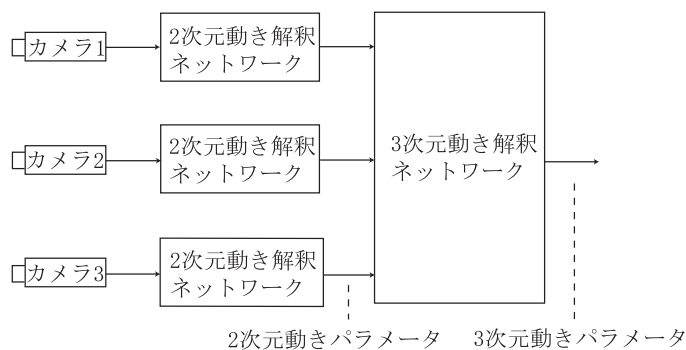


図 3.26 3次元動き解釈処理系の構成例

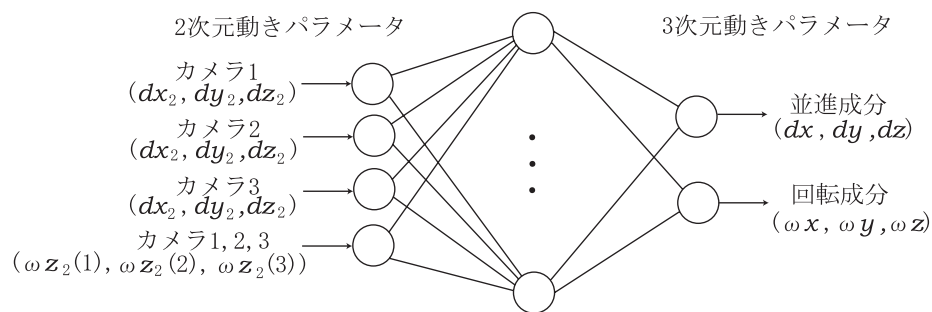


図 3.27 3次元動き解釈ネットワーク構成例

3.3 多層型 4 元数ニューラルネットワーク

本節では、多層型の 4 元数ニューラルネットワークについて述べる。

3.3.1 新田モデル

新田は、通常の実数型のニューラルネットワークを 4 元数に基づいて 4 次元パラメータ化した 4 元数バックプロパゲーション学習アルゴリズム(以下、4 元数 B P と呼ぶ)を提案した [65,68]。4 元数 B P は、重み、閾値、入力信号および出力信号がすべて 4 元数である階層型ニューラルネットワークに適用される。4 元数とは、W. R. Hamilton によって 1843 年に発見された 4 次元の数である [15]。ロボットの動作計画(ロボットを目標配置に至らしめる問題)においては、ロボットの配置(姿勢)の表現に 4 元数が有効に利用されている [12,20,21,35]。

(a) 4 元数ニューロンと逆 4 元数ニューロン

次のようなニューロンを考える。入力信号、重み、閾値および出力信号はすべて 4 元数である。ニューロン n の内部ポテンシャル A_n は、実 B P において用いられるニューロンの内部ポテンシャルの類推で、

$$A_n = \sum_m S_m W_{nm} + T_n \quad (3.129)$$

と定義する。ここで、 W_{nm} はニューロン n とニューロン m との間の重みを表す 4 元数、 S_m はニューロン m から出力されたニューロン n への入力信号を表す 4 元数、 T_n はニューロン n の閾値を表す 4 元数である。出力信号 $f_4(A_n)$ は次のように定義される：

$$f_4(A_n) = f(x_1) + f(x_2)i + f(x_3)j + f(x_4)k, \quad (3.130)$$

$$A_n = x_1 + x_2i + x_3j + x_4k, \quad (3.131)$$

$$f(x_l) = \frac{1}{1 + \exp(-x_l)}. \quad (3.132)$$

ここで、 $i^2 = j^2 = k^2 = -1$, $ij = -ji = k$, $jk = -kj = i$, $ki = -ik = j$.

式 (3.129) における $S_m W_{nm}$ の扱いには注意する必要がある。なぜならば、4 元数は乗法に関して非可換である

ため、一般に $S_m W_{nm} \neq W_{nm} S_m$ であるからである。このことにより、2 種類のニューロンが存在することになる。つまり、 $A_n = \sum_m S_m W_{nm} + T_n$ という計算を行なうニューロン(単に、4 元数ニューロンと呼ぶ)と $A_n = \sum_m W_{nm} S_m + T_n$ という計算を行なうニューロン(逆 4 元数ニューロンと呼ぶ)の 2 種類である。

(b) 4 元数ニューラルネットワーク

次に、4 元数 B P が適用されるネットワークを構成する。簡単のために、(a) で定義した 4 元数ニューロンだけを使って、3 層ネットワークを構成する。

$w_{ml} = w_{ml}^a + w_{ml}^b i + w_{ml}^c j + w_{ml}^d k \in \mathbf{H}$ を入力ニューロン l と中間ニューロン m との間の重みを表す 4 元数とする(\mathbf{H} は 4 元数全体の集合)。 $v_{nm} = v_{nm}^a + v_{nm}^b i + v_{nm}^c j + v_{nm}^d k \in \mathbf{H}$ を中間ニューロン m と出力ニューロン n との間の重みを表す 4 元数、 $\theta_m = \theta_m^a + \theta_m^b i + \theta_m^c j + \theta_m^d k \in \mathbf{H}$ を中間ニューロン m の閾値を表す 4 元数、 $\gamma_n = \gamma_n^a + \gamma_n^b i + \gamma_n^c j + \gamma_n^d k \in \mathbf{H}$ を出力ニューロン n の閾値を表す 4 元数とする。 $I_l = I_l^a + I_l^b i + I_l^c j + I_l^d k \in \mathbf{H}$ を入力ニューロン l への入力信号を表す 4 元数とし、 $H_m = H_m^a + H_m^b i + H_m^c j + H_m^d k \in \mathbf{H}$ および、 $O_n = O_n^a + O_n^b i + O_n^c j + O_n^d k \in \mathbf{H}$ をそれぞれ中間ニューロン m の出力値、出力ニューロン n の出力値を表す 4 元数とする。 $\Delta_n = \Delta_n^a + \Delta_n^b i + \Delta_n^c j + \Delta_n^d k = T_n - O_n \in \mathbf{H}$ を、 O_n と出力ニューロン n に対する教師信号 $T_n = T_n^a + T_n^b i + T_n^c j + T_n^d k \in \mathbf{H}$ との間の誤差とする。パターン p に対する 2 乗誤差を $E_p = (1/2) \sum_{n=1}^N |\Delta_n|^2$ と定義する。ここで、 N は出力ニューロンの総数、 $|x|^{\text{def}} \sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2}$, $x = x_1 + x_2 i + x_3 j + x_4 k \in \mathbf{H}$.

(c) 学習アルゴリズム

次に、(b) で定義した 4 元数 B P ネットワークに対する学習アルゴリズムを導出する。十分小さい学習率 $\varepsilon > 0$ に対して、最急降下法を使うことにより、重みと閾値の値は次の式に従って修正すれば良いことがわかる。 Δx はパラメータ x の修正量を表す。

$$\Delta v_{nm} = \bar{H}_m \Delta \gamma_n, \quad (3.133)$$

$$\begin{aligned} \Delta\gamma_n = & \varepsilon\{\Delta_n^a(1-O_n^a)O_n^a + \Delta_n^b(1-O_n^b)O_n^b i \\ & + \Delta_n^c(1-O_n^c)O_n^c j + \Delta_n^d(1-O_n^d)O_n^d k\}, \end{aligned} \quad (3.134)$$

$$\Delta w_{ml} = \bar{I}_l \Delta \theta_m, \quad (3.135)$$

$$\begin{aligned} \Delta \theta_m = & (1-H_m^a)H_m^a \cdot \text{Re} \left[\sum_n (\Delta\gamma_n \bar{v}_{nm}) \right] \\ & + (1-H_m^b)H_m^b \cdot \text{Im}^i \left[\sum_n (\Delta\gamma_n \bar{v}_{nm}) \right] i \\ & + (1-H_m^c)H_m^c \cdot \text{Im}^j \left[\sum_n (\Delta\gamma_n \bar{v}_{nm}) \right] j \\ & + (1-H_m^d)H_m^d \cdot \text{Im}^k \left[\sum_n (\Delta\gamma_n \bar{v}_{nm}) \right] k. \end{aligned} \quad (3.136)$$

ここで, $\bar{x} \stackrel{\text{def}}{=} x_1 - x_2 i - x_3 j - x_4 k$, $\text{Re}[x] \stackrel{\text{def}}{=} x_1$, $\text{Im}^i[x] \stackrel{\text{def}}{=} x_2$, $\text{Im}^j[x] \stackrel{\text{def}}{=} x_3$, $\text{Im}^k[x] \stackrel{\text{def}}{=} x_4$, $x = x_1 + x_2 i + x_3 j + x_4 k \in \mathbf{H}$.

さらに, 新田は, 4元数BPの学習特性について調べ, 実BPに比べて優れているとの感触を得ている。しかし, 明確な結論を得るためには, より体系的な実験を要するようである。また, 4元数BP固有の性質を調べることは興味深い研究課題である。

3.3.2 関数近似能力と時系列データ予測への応用

Arenaらは, 新田とは独立に多層型4元数ニューラルネットワークを提案し, その関数近似能力を解析的に調べた[8]。また, 電気回路のカオス的ふるまいの予測に応用した。彼らが提案したモデルは, 新田モデルと同じく, 重み, 閾値, 入力信号および出力信号がすべて4元数である階層型ニューラルネットワークである。

(a) 密度定理 (density theorem)

X を \mathbf{H}^n のコンパクト部分集合とし, \mathbf{H} 上の4元数値シグモイド関数 f を,

$$f(q) = \sigma(q_0) + \sigma(q_1)i + \sigma(q_2)j + \sigma(q_3)k, \quad (3.137)$$

$$q = q_0 + q_1 i + q_2 j + q_3 k \in \mathbf{H}, \quad (3.138)$$

$$\sigma(t) = \frac{1}{1 + \exp(-t)}, \quad t \in \mathbf{R}^1 \quad (3.139)$$

と定義する。この f を中間ニューロンの出力関数として持つような3層の4元数ニューラルネットワークに対す

る関数近似定理(密度定理)を次に示す(証明は省略)。

定理 6. $g: X \rightarrow \mathbf{H}$ を連続関数とする。このとき, 任意の $\varepsilon > 0$ に対して, 実数 $\alpha_1, \dots, \alpha_n \in \mathbf{R}^1$ と4元数ベクトル $\bar{y}_1, \dots, \bar{y}_N \in \mathbf{H}^n$, 4元数 $\theta_1, \dots, \theta_N \in \mathbf{H}$ が存在して,

$$\sup_{\bar{x} \in X} \left| g(\bar{x}) - \sum_{i=1}^N \alpha_i f(\bar{y}_i^T \cdot \bar{x} + \theta_i) \right| < \varepsilon \quad (3.140)$$

が成り立つ。言い換えると,

$$S = \left\{ \sum_{i=1}^N \alpha_i f(\bar{y}_i^T \cdot \bar{x} + \theta_i) \right\} \quad (3.141)$$

は, $C^0(X, \mathbf{H})$ において稠密である。ここで, $C^0(X, \mathbf{H})$ は, X 上の4元数値連続関数からなる空間で, ノルム $\|g\| = \sup_{\bar{x} \in X} |g(\bar{x})|$ を持ったものである。□

この定理が対象としている4元数ニューラルネットワークは, $n-N-1$ ネットワークであり, 入力ニューロンと中間ニューロンとの間の重み $\bar{y}_1, \dots, \bar{y}_N$ および中間ニューロンの閾値 $\theta_1, \dots, \theta_N$ は4元数であるが, 中間ニューロンと出力ニューロンとの間の重み $\alpha_1, \dots, \alpha_N$ は実数である。そして, ただ一つの出力ニューロンの出力関数は $h(x) = x$ であり, 出力ニューロンの閾値はゼロである。

(b) カオス時系列データ予測への応用

さらに, 4元数ニューラルネットワークをカオス電気回路から得られる時系列データの短期予測に応用した。対象としたカオス電気回路は, 超カオス Saito 回路と呼ばれるもので, 4つの変数 x_1, x_2, y_1, y_2 によって特徴付けられる。 t を時間パラメータとして, この4つの変数を4元数 $q(t) = x_1(t) + iy_1(t) + jx_2(t) + ky_2(t)$ として表現した。学習パターンは, $\{q(t) \mid t = 1, \dots, 200\}$ を入力とし, $\{q(t+\tau) \mid \tau = 1, \dots, 10\}$ を出力とした。テストパターンは400個用意した。実ニューラルネットワークは4-6-4ネットワーク, 4元数ニューラルネットワークは1-3-1ネットワークを比較対象として選んだ。これらは中間ニューロン数をいろいろ変化させて実験した結果, それぞれ最も良い予測精度を得たものである。そして, 予測精度は4元数ニューラルネットワークの方が実数型のニューラルネットワークに比べて優れたものであ

た。しかも、そのときの重みなどのパラメータ数は、実数ニューラルネットワークは58個であったのに対して、4元数ニューラルネットワークは28個と少ないものであった。

第 4 章 相互結合型複素ニューラルネットワーク

相互結合型の高次元ニューラルネットワークは、今のところ複素数(2次元)しか存在しない。そこで、本章では相互結合型の複素ニューラルネットワークの研究の現状について、その概要を述べる。詳細については、それぞれの参考文献を参照されたい。

(a) 根本ら [42]

複素数化した自己関連連想記憶型のニューラルネットワークを提案し、それによるメリットや回路の性質をシミュレーションなどにより調べた。特に記憶容量が増加することを実験により示唆した。

(b) 木ノ内ら [27,29]

時系列データの学習は、ニューラルネットワークにとって困難な課題である。木ノ内らは、中間ニューロンに自分自身へのフィードバックループを付加した複素ニューラルネットワークとその学習アルゴリズムを提案した。そして、時系列データの学習に関して、従来の時系列学習を行なうことのできる実数型のニューラルネットワーク(Elmanのネットワーク[16]など)との比較を行ない、学習性能が優れていることを計算機実験により示した。この学習性能の優位性は、自己フィードバックループを持った複素ニューロンが、自己フィードバックループを持った実数ニューロンに比べて、複素平面上に位相(角度)を持つために過去の情報を保持しやすいからではないかと指摘している。

(c) 木ノ内ら [28,30]

木ノ内らは(b)で述べた自己フィードバックループを持った複素ニューラルネットワークを用いて、メロディの記憶と想起を行なうシステム MUSIC(Multi layer network for Sequential Inputs using Complex neurons)を提案し、Elmanのネットワークによって実現したシステムとの計算機実験による比較を行ない、木ノ内らのシステムの方が学習速度、学習精度に関して優れていることを確認した。さらに、MUSICを複数個用意して、統合

することによって、曲数を多く含む大規模な問題にも適用することができるメロディ検索システムを構築した。これは、従来の音楽検索システムが必要としていたデータベースとのマッチング処理はいっさい行なわないという特長を持つ。また、検索の手がかりとして、曲名や作曲者などの文字情報ではなく、メロディの一部を用いるため、ユーザにとって利用しやすいものとなっている。さらに想起が逐次的に行なわれるため、入力に対する対話的な処理を行なうことが可能である。

(d) Aguら [3]

Aguらは、相互結合型の複素ニューラルネットワークの性質を解析した。それぞれの複素ニューロンは、複素平面上の原点を中心とした単位円上の値を取り、複素ニューロンの値の更新は非同期に行なわれる。また、複素数値のエネルギー関数が定義されている。このようなモデルに対して、まずエネルギー関数が単調減少することを証明し、この事実を使って、複素ニューラルネットワークの平衡状態は安定なもの(persistent)、不安定なもの(fragile)の2つに分けられることを指摘し、安定であるための十分条件を求めた。

(e) 青木 [6]

通常の2値のホップフィールド連想記憶モデルが、自然な形で複素数体上のモデルへと拡張できることを示している。また、この拡張したモデルにおいて、クリステンゼン関数を用いることにより記録パターンと平衡状態の関係性を明らかにしている。

(f) 青木ら [7]

彼等は、ネットワークに現れるすべての変数を複素数値化した多値の複素連想記憶モデルのエネルギー関数にペナルティ項を導入したモデルの性質について報告している。彼等が用いたモデルは、記録パターンを作る際に何らかの規則を設けて、その規則をエネルギー関数の中にペナルティ項として取り込んだものである。これを用いることで想起の際に、設定した規則を先見情報と見なすことによって、記憶容量や想起能力が向上することを示している。また、複素連想記憶モデルを用いた通信符号の誤り訂正機能などへの応用について示唆を与えている。

(g) Hirose[19]

ニューラルネットワークの入力ベクトルとアトラクターが複素数であるようなものを提案している。このネットワークでは、データの実部と虚部を同等に取り扱うことができるので、本来、複素数値でデータが与えられる場合に対して有効に機能することが期待される。彼の提案したネットワークモデルは、実数値だけをとる通常のネットワークモデルとは違い、2倍の自由度を持つ空間でネットワークが動作するため局所安定解を取り除くことができるという利点を持っている。

(h) Jankowski ら [23]

多値の連想記憶モデルを提案している。このモデルは、各ニューロンが複素数値の複数の状態をとり得るような全結合型ニューラルネットワークである。彼等は、この全結合型複素ニューラルネットワークモデルは、実数値型のホップフィールドネットワークモデルの一般化として見ることができることを示している。また、非同期型のダイナミクスに対するネットワークの安定性についての評価およびニューロンの状態数に対する記憶容量の変化についても考察している。

第 5 章 おわりに

高次元ニューラルネットワーク関連研究に関するサーベイを行なった。

高次元ニューラルネットワークは、主として90年代に提案があり、画像処理などへの応用が行なわれてきたが、本格的な取り組みについてはこれからの課題である。また、高次元ニューラルネットワークの性質は、今までに調べられたもの以外にも有益なものが多く備わっている可能性があり、高次元自己回帰モデルの性質と対比させながら、十分に調べ尽す必要がある。高次元ニューラルネットワークの性質の解明と応用は、相互に良い影響を与えながら進めてゆく必要がある。性質の解明が進めば、応用が容易になるだけでなく、応用の適用範囲も広がる。反対に応用サイドからの要請により、高次元ニューラルネットワークの問題点も浮き彫りにされる。

ところで、雲の動きは複雑であり、降雨量や降雪量の予測は非常に難しく、一種の複雑系と言える。また、近年、気象予報業務の自由化に伴い、時間・空間分解能が高い詳細な気象情報の提供が望まれている。落合らは通常のニューラルネットワークを用いて30分～3時間先までの降雨・降雪量を予測する方法を提案している [70]。3.1.2.3 項で述べた複素ニューラルネットワークの2次元アフィン変換学習能力は、このような短期天気予報に有効に活用できるのではないかと考えている。2次元平面上の任意の点の雲の動きは、2次元ベクトルとして表現できる。そして、多くのそのような2次元ベクトルを複素ニューラルネットワークに予め学習させておけば、2次元アフィン学習能力により、一定時間後の雲の動きをうまく予測でき、結果として降雨・降雪量を精度良く予測できるものと考えられる。このような応用を行なっていくに当たっては、複素ニューラルネットワークの2次元アフィン変換学習能力がどの程度のものであるのか、どのようなクラスの関数まで有効であるのか、その限界を理論的、実験的に調べておく必要もあるだろう。恐らく必要となるのは高度な非線形能力だと思われるが、それを持たせるための方策も考える必要がある。また、一般に出力関数に Radial Basis Function(RBF) を用いるとニューラルネットワークの学習能力が向上すると言われているが [72]、高次元ニューラルネットワークを RBF 化

することも考えられる。

一般にユークリッド空間 R^n においては、体積が1の超立方体に含まれている超球は n が増加するに従って、その超立方体をはみ出るという直観には合わない奇妙な現象が起きることが知られている。このように一般に高次元化はその概念に興味深い性質を与える。ニューラルネットワークのパラメータを5次元以上に高次元化した場合に、どのような特性が現れてくるのかについても興味のあるところである。

本調査研究を行なっている最中に、多層型の複素ニューラルネットワーク、3次元ニューラルネットワーク、4次元ニューラルネットワークに関する研究成果をまとめた本が出版された [9]。その内容には本調査報告書に記載されていないものもあるが、逆に、本調査報告書には彼らの本に記載されていないものもある。時間の関係で詳しく調査できなかったが、今後の研究の展開に応じて参考にされることを期待したい。

謝辞

本調査の機会を与えられ、終始多大のご支援とご理解、適切なご助言とご指導をいただきました、大蒔和仁 情報アーキテクチャ部長、中島秀之 情報科学部長に感謝の意を表します。

参考文献

- [1] Adiv, G. : “Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects”, IEEE Trans. Pattern Anal. & Machine Intell., **PAMI-7** (4), pp.384–401 (1985).
- [2] Aggarwal, J. K. and Nandhakumar, N. : “On the Computation of Motion from Sequences of Images - A Review”, Proceedings of IEEE, 76, 8, pp.917–935 (1988).
- [3] Agu, M., Yamanaka, K. and Takahashi, H.: “A Local Property of the Phasor Model of Neural Networks”, IEICE Trans. Inf. & Syst., Vol.E79-D, No.8, pp.1209–1211 (1996).
- [4] 合原 一幸 : “カオスの数理と技術”, 放送大学教育振興会 (1997).
- [5] Amari, S. : “A Theory of Adaptive Pattern Classifiers”, IEEE Trans. Electronic Computers, **EC-16** (3), pp.299–307 (1967).
- [6] 青木 宏之 : “複素数体上に拡張したホップフィールド連想記憶の平衡状態の解析”, 電子情報通信学会論文誌, Vol.J78-A, No.9, pp.1238–1241 (1995).
- [7] 青木 宏之, 小杉 幸夫 : “ペナルティ項を有する複素連想記憶モデルの性質”, 電子情報通信学会 信学技報, NC97-33, pp.61–68 (1997).
- [8] Arena, P., Fortuna, L., Muscato, G. and Xibilia, M. G. : “Multilayer Perceptrons to Approximate Quaternion Valued Functions”, Neural Networks, Vol.10, No.2, pp. 335–342 (1997).
- [9] Arena, P., Fortuna, L., Muscato, G. and Xibilia, M. G. : “Neural Networks in Multidimensional Domains”, Lecture Notes in Control and Information Sciences 234, Springer, p. 165 (1998).
- [10] Ballard, D. H. and Brown, C. M. : “Computer Vision”, Rrentice-Hall, Inc. (1982).
- [11] Benvenuto, N. and Piazza, F. : “On the Complex Backpropagation Algorithm”, IEEE Trans. Signal Processing, Vol.40, No.4, pp.967–969, April (1992).
- [12] Canny, J. F. : “The Complexity of Robot Motion Planning”, The MIT Press (1988).
- [13] Durbin, J. : “The Fitting of Time-series Models”, Rev. Inst. Int. de Stat., Vol.28, No.3, pp.233–244 (1960).
- [14] Dubois, S.R. and Glanz, F.H. : “An Autoregressive Model Approach to Two-dimensional Shape Classification”, IEEE Trans. Pattern Anal. & Machine Intelli., PAMI-8, No.1, pp.55–66 (1986).
- [15] エビングハウス他(成木 勇夫訳): “数(下)”, シュプリンガー・フェアラー東京 (1993).
- [16] Elman, J. L.: “Finding Structure in Time”, Cognitive Science, 14, pp.179–211 (1990).
- [17] Georgiou, G. M. and Koutsougeras, C. : “Complex Domain Backpropagation”, IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing, Vol.39, No.5, pp.330–334, May (1992).
- [18] 葉原 耕平編集: “ニューラルネットワーク応用”, オーム社 (1995).
- [19] Hirose, A. : “Proposal of Fully Complex-valued Neural Networks”, Proceeding of IJCNN’92, Vol.IV, pp.152–157 (1992).
- [20] Hirukawa, H., Papegay, Y. and Matsui, T. : “A Motion Planning Algorithm for Convex Polyhedra in Contact under Translation and Rotation”, Proceedings of IEEE International Conference on

- Robotics and Automation, Vol.4, pp.3020-3027 (1994).
- [21] Horn, B. K. P. : “*Robot Vision*”, The MIT Press (1986).
- [22] 井庭 崇, 福原義久: “複雑系入門”, NTT 出版 (1998).
- [23] Jankowski, S., Lozowski, A. and Zurada, J.M. : “Complex-Valued Multistate Neural Associative Memory”, IEEE Transactions on Neural Networks, Vol.7, No.6, pp.1491-1496 (1996).
- [24] 笠原 乾吉 : “複素解析”, 実教出版 (1980).
- [25] Kashap, R. L. and Chellappa, R.: “Stochastic Models for Closed Boundary Analysis: Representation and Reconstruction”, IEEE Trans. Inform. Theory, IT-27, No.5, pp.627-637 (1981).
- [26] Kim, M. S. et al.: “Modification of Backpropagation Networks for Complex-valued Signal Processing in Frequency Domain”, Proceedings of IEEE/INNS International Joint Conference on Neural Networks, IJCNN'90, Vol.3, 27-31 (1990).
- [27] Kinouchi, M. and Hagiwara, M.: “Learning Temporal Sequences by Complex Neurons with Local Feedback”, Proceedings of IEEE International Conference on Neural Networks, ICNN'95-Perth, Nov. 27-Dec. 1, Vol.6, pp.3165-3169 (1995).
- [28] Kinouchi, M. and Hagiwara, M.: “Memorization of Melodies by Complex-valued Recurrent Network”, Proceedings of IEEE International Conference on Neural Networks, ICNN'96-Washington, D.C., Vol.2, pp.1324-1328 (1996).
- [29] 木ノ内 誠, 荻原 将文 : “複素ニューロンによる時系列の学習”, 電学論, Vol.116-C, No.7, pp.748-754 (1996).
- [30] 木ノ内 誠, 荻原 将文: “複素リカレントニューラルネットワークを用いたメロディの記憶と想起”, 情報処理学会論文誌, Vol.39, No.5, pp.1232-1239 (1998).
- [31] Koford, J. S. and Groner, G. F. : “The Use of an Adaptive Threshold Element to Design a Linear Optimal Pattern Classifier”, IEEE Trans. Inform. Theory, Vol.IT-12, pp.42-50, Jan. (1966).
- [32] 栗田 多喜夫, 大津 展之, 関田 巖 : “複素自己回帰係数および複素 PARCOR 係数の高速計算法”, 電子情報通信学会総合全国大会予稿集, D-497 (1989).
- [33] 栗田 多喜夫, 関田 巖, 大津 展之 : “複素自己回帰モデルに基づく輪郭形状間の距離”, 電子情報通信学会論文誌, Vol.J73-D-II, No.9, pp.1493-1503 (1990).
- [34] 栗田 多喜夫 : “柔らかな情報処理のための統計的手法の応用に関する研究”, 電子技術総合研究所研究報告, 第 957 号 (1993).
- [35] Latombe, J. C. : “*Robot Motion Planning*”, Kluwer Academic Publishers (1991).
- [36] Leung, H. and Haykin, S. : “The Complex Backpropagation Algorithm”, IEEE Trans. Signal Processing, Vol.39, No.9, pp.2101-2104, Sep. (1991).
- [37] 大津 展之, 栗田 多喜夫 : “Shape Descriptor としての複素自己回帰モデルの提案”, 電子情報通信学会総合全国大会予稿集, D-496 (1989).
- [38] Miyauchi, M. and Seki, M. : “Interpretation of Optical Flow through Neural Network Learning”, Proceedings of IEEE International Conference on Communication Systems/International Symposium on Information Theory and its Applications, Singapore, Nov., pp.1247-1251 (1992).
- [39] Miyauchi, M., Seki, M., Watanabe, A. and Miyauchi, A. : “Interpretation of Optical Flow through Neural Network Learning”, Proceedings of IAPR Workshop on Machine Vision Applications, Tokyo, Dec., pp.523-528 (1992).
- [40] Miyauchi, M., Seki, M., Watanabe, A. and Miyauchi, A. : “Interpretation of Optical Flow through Complex Neural Network”, Proceedings of International Workshop on Artificial Neural

- Networks, IWANN'93-Barcelona, Lecture Notes in Computer Science, Vol.686, Springer-Verlag, pp.645-650 (1993).
- [41] 宮嶋 照行, 長谷川 孝明: “複素ニューラルネットワークの高速学習アルゴリズム”, 電子情報通信学会論文誌, Vol.J76-D-II, No.7, pp.1468-1470 (1993).
- [42] 根本 幾, 河野 知志: “神経回路網の複素数値化の試み”, 電子情報通信学会論文誌, Vol.J74-D-II, No.9, pp.1282-1288 (1991).
- [43] 新田 徹, 古谷 立美: “複素バックプロパゲーション学習”, 情報処理学会論文誌, Vol.32, No.10, pp.1319-1329 (1991).
- [44] 新田 徹, 赤穂 昭太郎, 秋山 泰, 古谷 立美: “複素バックプロパゲーション・ネットワークにおける重みパラメータと決定表面の構造”, 情報処理学会論文誌, Vol.33, No.11, pp.1306-1313 (1992).
- [45] Nitta, T. and deGaris, H. : “A 3D Vector Version of the Back-Propagation Algorithm”, Proceedings of IEEE/INNS International Joint Conference on Neural Networks, IJCNN'92-Beijing, Nov.3-6, Vol.2, pp.511-516 (1992).
- [46] 新田 徹, 古谷 立美: “複素バックプロパゲーション学習アルゴリズムの学習特性”, 情報処理学会論文誌, Vol.34, No.1, pp.29-38 (1993).
- [47] 新田 徹: “回転を学習した複素 BP ネットワークのふるまい”, 情報処理学会論文誌, Vol.34, No.1, pp.39-51 (1993).
- [48] Nitta, T. : “A Complex Numbered Version of the Back-Propagation Algorithm”, Proceedings of INNS World Congress on Neural Networks, WCNN'93-Portland, July 11-15, Vol.3, pp.576-579 (1993).
- [49] Nitta, T. : “A Three-Dimensional Back-Propagation”, Proceedings of INNS World Congress on Neural Networks, WCNN'93-Portland, July 11-15, Vol.3, pp.572-575 (1993).
- [50] Nitta, T. : “A Back-Propagation Algorithm for Complex Numbered Neural Networks”, Proceedings of IEEE/INNS International Joint Conference on Neural Networks, IJCNN'93-Nagoya, Oct. 25-29, Vol.2, pp.1649-1652 (1993).
- [51] Nitta, T. : “A Back-Propagation Algorithm for Neural Networks Based on 3D Vector Product”, Proceedings of IEEE/INNS International Joint Conference on Neural Networks, IJCNN'93-Nagoya, Oct. 25-29, Vol.1, pp.589-592 (1993).
- [52] Nitta, T. : “A Supervised Learning Algorithm for Neural Networks with Two-dimensional Weights”, Proceedings of CIE/IEEE International Conference on Neural Networks and Signal Processing, ICNNSP'93-Guangzhou, Nov. 2-5, pp.425-430 (1993).
- [53] Nitta, T. : “Proposal of Neural Networks Based on Vector Product”, Proceedings of CIE/IEEE International Conference on Neural Networks and Signal Processing, ICNNSP'93-Guangzhou, Nov. 2-5, pp.397-402 (1993).
- [54] Nitta, T. : “An Extension of the Back-Propagation Algorithm to Three Dimensions by Vector Product”, Proceedings of 5th IEEE International Conference on Tools with Artificial Intelligence, TAI'93-Boston, Nov. 8-11, pp.460-461 (1993).
- [55] Nitta, T. : “Fundamental Structures of the Complex Back-Propagation Algorithm”, Proceedings of International Symposium on Artificial Neural Networks, ISANN'93-Taiwan, Dec. 20-22, pp.30-38 (1993).
- [56] 新田 徹: “ニューラルネットワークの3次元への拡張”, 情報処理学会論文誌, Vol.35, No.7, pp.1300-1310 (1994).
- [57] Nitta, T. : “Ability of the 3D Vector Version of the Back-Propagation to Learn 3D Motion”, Proceedings of INNS World Congress on Neural Networks,

- WCNN'94-SanDiego, June 4-9, Vol.3, pp.262-267 (1994).
- [58] Nitta, T. : "An Analysis on the Learning Rule in the Complex Back-Propagation Algorithm", Proceedings of INNS World Congress on Neural Networks, WCNN'94-SanDiego, June 4-9, Vol.3, pp.702-707 (1994).
- [59] Nitta, T. : "Decision Boundaries of the Complex Valued Neural Networks", Proceedings of INNS World Congress on Neural Networks, WCNN'94-SanDiego, June 4-9, Vol.4, pp.727-732 (1994).
- [60] Nitta, T. : "Behavior of the Complex Numbered Back-Propagation Network which has Learned Similar Transformation", Proceedings of INNS World Congress on Neural Networks, WCNN'94-SanDiego, June 4-9, Vol.4, pp.765-770 (1994).
- [61] Nitta, T. : "Structure of Learning in the Complex Numbered Back-Propagation Network", Proceedings of IEEE International Conference on Neural Networks, ICNN'94-Orlando, June 28-July 2, Vol.1, pp.269-274 (1994).
- [62] Nitta, T. : "An Analysis on Decision Boundaries in the Complex Back-Propagation Network", Proceedings of IEEE International Conference on Neural Networks, ICNN'94-Orlando, June 28-July 2, Vol.2, pp.934-939 (1994).
- [63] Nitta, T. : "Generalization Ability of the Three-Dimensional Back-Propagation Network", Proceedings of IEEE International Conference on Neural Networks, ICNN'94-Orlando, June 28-July 2, Vol.5, pp.2895-2900 (1994).
- [64] Nitta, T. : "Ability of the Complex Back-propagation Algorithm to Learn Similar Transformation", Proceedings of IEEE International Conference on Neural Networks, ICNN'95-Perth, Nov. 27-Dec. 1, Vol.3, pp.1513-1516 (1995).
- [65] Nitta, T. : "A Quaternary Version of the Back-propagation Algorithm", Proceedings of IEEE International Conference on Neural Networks, ICNN'95-Perth, Nov. 27-Dec. 1, Vol.5, pp.2753-2756 (1995).
- [66] 新田 徹 : "高次元化されたパラメータを持つ階層型ニューラルネットワークの研究", 電子技術総合研究所研究報告 第976号 p. 98 (1995).
- [67] 新田 徹, 古谷 立美 : "ニューラルネットワークおよびその信号処理方法", 登録番号 : 特許第 2090565号 (1996年9月18日登録).
- [68] Nitta, T. : "An Extension of the Back-propagation Algorithm to Quaternions", Proceedings of International Conference on Neural Information Processing, ICONIP'96-HongKong, Sep. 25-27, Vol.1, pp.247-250 (1996).
- [69] Nitta, T. : "An Extension of the Back-Propagation Algorithm to Complex Numbers", Neural Networks, Vol.10, No.8, pp. 1392-1415 (1997).
- [70] Ochiai, K. and Sonehara, N. : "Precipitation Nowcast with Artificial Neural Networks", Proceedings of International Conference on Neural Information Processing, ICONIP'98-Kitakyushu, Vol.3, pp.1343-1346 (1998).
- [71] Pavlidis, T. : "Algorithm for Shape Analysis of Contours and Waveforms", IEEE Trans. Pattern Anal. & Machine Intelli. , PAMI-2, No.4, pp.301-312 (1980).
- [72] Poggio, T. and Edelman, S. : "A Network that Learns to Recognize Three-Dimensional Objects", Nature, Vol.343:18, pp.263-266 (1990).
- [73] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. : "*Parallel Distributed Processing*", Vol.1, The MIT press (1986).
- [74] Sanger, T. D.: "Optimal Unsupervised Learning in a Single-layer Linear Feedforward Neural Network", Neural Networks, Vol.2, pp.459-473 (1992).

- [75] 関田 巖, 栗田 多喜夫, 大津 展之: “複素自己回帰モデルによる形の認識実験”, 電子情報通信学会総合全国大会予稿集, D-495 (1989).
- [76] 関田 巖, 栗田 多喜夫, 大津 展之: “複素自己回帰モデルによる類似形状の検索”, 電子情報通信学会技術報告, IE89-7 (1989).
- [77] 関田 巖, 栗田 多喜夫, 大津 展之: “複素自己回帰モデルによる形の識別”, 電子情報通信学会論文誌, Vol.J73-D-II, No.6, pp.804-811 (1990).
- [78] Sekita, I., Kurita, T. and Otsu, N.: “Complex Autoregressive Model for Shape Recognition”, IEEE Trans. Pattern Anal. & Machine Intell., PAMI-14, No.4 (1992).
- [79] 白井 良明編: “パターン理解, 知識工学講座9”, オーム社 (1987).
- [80] Subbarao, M.: “Interpretation of Image Flow: A Spatio-Temporal Approach”, IEEE Trans. Pattern Anal. & Machine Intell., PAMI-11 (3), pp.266-278 (1989).
- [81] 高橋 祥兼: “正則写像を近似する複素ニューラルネットの有限学習データと最小構造”, 電子情報通信学会論文誌, Vol.J77-A, No.3, pp.593-598 (1994).
- [82] Tanaka, M.: “3D Autoregressive Model with Proper Transformation Property under Rotation”, SPIE VISION GEOMETRY IV, Vol.2826, pp.183-189 (1995).
- [83] Tanaka, M.: “Three Dimensional Autoregressive Model under Rotation”, SPIE VISION GEOMETRY V, Vol.2826, pp.172-179 (1996).
- [84] Tsai, R. Y. and Huang, T. S.: “Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces”, IEEE Trans. Pattern Anal. & Machine Intell., PAMI-6 (1), pp.13-27 (1984).
- [85] 渡部 朗, 宮内 新, 宮内 ミナミ: “ニューラルネットワークによるオプティカル・フローの3次元動き解釈のための一提案”, 第46回情報処理学会全国大会 論文集 (分冊2), pp.167-168 (1993).
- [86] 渡部 朗, 宮内 新, 宮内 ミナミ: “ニューラルネットワークを用いた物体の3次元動き解釈手法の研究”, 第16回情報理論とその応用シンポジウム (SITA'93) 予稿集 (2), pp.529-532 (1993).
- [87] Watanabe, A., Yazawa, N., Miyauchi, A. and Miyauchi, M.: “A Method to Interpret 3D Motions Using Neural Networks”, IEICE Trans. Fundamentals, Vol.E77-A, No.8, pp.1363-1370 (1994).
- [88] Widrow, B. and Hoff, M. E.: “Adaptive Switching Circuits”, in 1960 IRE WESCON Conv. Rec., pt.4, pp.96-104 (1960).
- [89] Widrow, B.: “Adaptive Filters”, in *Aspects of Network and System Theory*, Kalman, R. E. and DeClaris, N. Eds. New York: Holt, Rinehart and Winston, pp.563-587 (1971).
- [90] Widrow, B., McCool, J. and Ball, M.: “The Complex LMS Algorithm”, Proceedings of the IEEE, Vol.63, No.4, pp.719-720 (1975).
- [91] 矢沢 伸行, 全 へい東, 宮内 新, 宮内 ミナミ: “ニューラルネットワークを用いた3次元動き解釈システムのカメラ配置に関する検討”, 第49回情報処理学会全国大会 論文集 (分冊2), pp.129-130 (1994).
- [92] 米沢 豊美子: “複雑さを科学する”, 岩波書店 (1995).
- [93] Zhang, Y. and Ma, Y.: “CGHA for Principal Component Extraction in the Complex Domain”, IEEE Trans. on Neural Networks, Vol.8, No.5, pp.1031-1036 (1997).