Published in *Neural Networks*, Vol.10, No.8, pp.1391-1415, 1997 as contributed article.
(Submitted March 15, 1991; revised March 10, 1997)

(a) **The title of the article**

An Extension of the Back-Propagation Algorithm to Complex Numbers

(b) **The authors' full names**

Tohru Nitta

(c) **Affiliations**

Neuroscience Research Institute,
National Institute of Advanced Industrial Science and Technology (AIST),
AIST Tsukuba Central 2, 1-1-1 Umezono, Tsukuba-shi, Ibaraki, 305-8568 Japan.
E-mail: tohru-nitta@aist.go.jp

(d) **Acknowledgements**

We wish to thank Drs. A.Tojoh, T.Yuba, K.Ohta, T.Furuya, and T.Higuchi for providing the opportunity for this study. We are also grateful to Dr. B.Manderick for his help with the English language and to the anonymous reviewers for their many helpful suggestions.

(a) **The title of the article**

An Extension of the Back-Propagation Algorithm to Complex Numbers

(b) **Abstract**

This paper presents a complex-valued version of the back-propagation algorithm (called *Complex-BP*), which can be applied to multi-layered neural networks whose weights, threshold values, input and output signals are all complex numbers. Some inherent properties of this new algorithm are studied. The results may be summarized as follows. The updating rule of the Complex-BP is such that the probability for *a standstill in learning* is reduced. The average convergence speed is superior to that of the real-valued back-propagation, whereas the generalization performance remains unchanged. In addition, the number of weights and thresholds needed is only about the half of real-valued back-propagation, where a complex-valued parameter $z = x + iy$ (where $i = \sqrt{-1}$) is counted as two because it consists of a real part x and an imaginary part y. The Complex-BP can transform geometric figures, e.g. rotation, similarity transformation and parallel displacement of straight lines, circles, etc., whereas the real-valued back-propagation cannot. Mathematical analysis indicates that a Complex-BP network which has learned a transformation, has the ability to generalize that transformation with an error which is represented by the sine. It is interesting that the above characteristics appear only by extending neural networks to complex numbers.

(c) **Key words**

Neural networks, Learning, Back-Propagation, Convergence, Complex number, Figure transformation, Pattern transformation, Pattern classification.

(d) **A list of symbols**

$\boldsymbol{N}$	the set of natural numbers
$\boldsymbol{R}$	the set of real numbers
$\boldsymbol{R}^+$	the set of nonnegative real numbers
$\boldsymbol{C}$	the set of complex numbers
i	$\sqrt{-1}$
$Re[z]$	the real part x of a complex number $z = x + iy$
$Im[z]$	the imaginary part y of a complex number $z = x + iy$
$\overline{z}$	the conjugate complex number $x - iy$ of a complex number $z = x + iy$, that is, $\overline{x + iy} = x - iy$
Δz^R	the real part of the magnitude of change of a complex-valued weight or a complex-valued threshold z, that is, $\Delta z^R = Re[\Delta z]$
Δz^I	the imaginary part of the magnitude of change of a complex-valued weight or a complex-valued threshold z, that is, $\Delta z^I = Im[\Delta z]$
∇	a gradient operator with respect to a real-valued parameter
∇^{Re}	a gradient operator with respect to a real part of a complex-valued parameter
∇^{Im}	a gradient operator with respect to an imaginary part of a complex-valued parameter
ϕ	the difference between the argument of a test point and that of an input training point
$E^R(\phi)$	a complex number which denotes the error between the actual output test point and the expected output test point in the case of rotation
$E^S(\phi)$	a complex number which denotes the error between the actual output test point and the expected output test point in the case of similarity transformation
$E^P(\phi)$	a complex number which denotes the error between the actual output test point and the expected output test point in the case of parallel displacement
$E^R_{re}(\phi)$	the real part of the complex number $E^R(\phi)$
$E^R_{im}(\phi)$	the imaginary part of the complex number $E^R(\phi)$

1 INTRODUCTION

In recent years there has been a great deal of interest in artificial neural networks and their applications. One of the most popular neural network models is the multi-layer network and the related back-propagation training algorithm (called *Real-BP* here in the meaning of treating real-valued signals) (Rumelhart et al., 1986). Real-BP has been applied to various fields such as image processing and speech recognition. Examining those fields, we can find that complex numbers are often used according to fields. This indicates that complex-valued neural networks may be useful. In addition, in the human brain, an action potential may have different pulse patterns, and the distance between pulses may be different. This suggests that introducing complex numbers representing phase and amplitude into neural networks is appropriate.

In this paper, we will present a complex-valued version of the back-propagation algorithm (*Complex-BP*), which can be applied to multi-layered neural networks whose weights, threshold values (called *learnable parameters* here), input and output signals are all complex numbers (Nitta et al., 1991, 1993c). This new algorithm enables the network to learn complex-valued patterns in a natural way. The learning convergence theorem can be obtained by extending the theory of adaptive pattern classifiers (Amari, 1967) to complex numbers. We have studied some inherent properties of the Complex-BP algorithm (Nitta et al., 1991, 1993a, 1993b, 1993c, 1994a, 1994b), the results of which may be summarized as follows: (a) The error back propagation has a structure which is concerned with two-dimensional motion. A unit of learning is complex-valued signals flowing through neural networks. The learning rule is structured to avoid a *standstill in learning*. Ultimately, the average convergence speed is superior to that of the Real-BP, whereas the generalization performance remains unchanged. In addition, the required number of learnable parameters is only about the half of the Real-BP, where a complex-valued parameter $z = x + iy$ (where $i = \sqrt{-1}$) is counted as two because it consists of a real part x and an imaginary part y. Thus it seems that the Complex-BP algorithm is well suited for learning complex-valued patterns. (b) The Complex-BP can transform geometric figures, e.g. rotation, similarity transformation and parallel displacement of straight lines, circles,

etc., whereas the Real-BP cannot. Numerical experiments suggest that the behavior of a Complex-BP network which has learned the transformation of geometric figures is related to the Identity Theorem in complex analysis. Mathematical analysis indicates that a Complex-BP network which has learned a rotation, a similarity transformation or a parallel displacement has the ability to generalize the transformation with an error which is represented by the sine. It is interesting that the above characteristics appear only by extending neural networks to complex numbers. In this connection, the Complex-BP algorithm has already been applied to the interpretation of optical flow (motion vector field calculated from images) and estimation of motion which are important tasks in computer vision (Miyauchi et al., 1992a, 1992b, 1993; Watanabe et al., 1994).

Section 2 of this paper presents the Complex-BP algorithm. Section 3 experimentally investigates the learning characteristics: the learning speed and the generalization performance, and how the algorithm can be applied to the transformation of geometric figures. Section 4 analyses these results of Section 3 mathematically. Section 5 discusses the results of Sections 3 and 4. This is followed by our conclusion in Section 6. Proofs of the technical results are contained in a appendix.

2 THE COMPLEX-BP ALGORITHM

2.1 A Complex Adaptive Pattern Classifiers Model

Real-BP is based on the Adaptive Pattern Classifiers Model, or APCM (Amari, 1967) which guarantees that the Real-BP converges. In this section, we will formulate a complex-valued version of the APCM (called *Complex APCM*) for the Complex-BP by introducing *complex numbers* to the APCM, which will guarantee that the Complex-BP converges.

Let us consider two information sources of complex-valued patterns. Two complex-valued patterns $\boldsymbol{x} \in \boldsymbol{C}^n$ and $\boldsymbol{y} \in \boldsymbol{C}^m$ occur from information sources 1 and 2 with the unknown joint probability $P(\boldsymbol{x}, \boldsymbol{y})$, respectively ($\boldsymbol{C}$ denotes the set of complex numbers). We will assume that the number of patterns is finite. Note that the set of pairs of complex-valued patterns $\{(\boldsymbol{x},\boldsymbol{y})\}$ corresponds to the set of learning patterns in neural networks.

The purpose of learning is to estimate a complex-valued pattern $\boldsymbol{y}$ that occurs from information source 2 given a complex-valued pattern $\boldsymbol{x}$ that occurred from information source 1. Let $\boldsymbol{z}(\boldsymbol{w}, \boldsymbol{x}) : \boldsymbol{C}^p \times \boldsymbol{C}^n \to \boldsymbol{C}^m$ be a complex function which gives an estimate of $\boldsymbol{y}$ where $\boldsymbol{w} \in \boldsymbol{C}^p$ is a parameter which corresponds to all weights and thresholds in neural networks, and $\boldsymbol{z}(\boldsymbol{w}, \boldsymbol{x})$ corresponds to the actual output pattern of neural networks. Let $r(\boldsymbol{y}', \boldsymbol{y}) : \boldsymbol{C}^m \times \boldsymbol{C}^m \to \boldsymbol{R}^+$ be an error function which represents an error that occurs when we give an estimate $\boldsymbol{y}'$ for the true complex-valued pattern $\boldsymbol{y}$ ($\boldsymbol{R}^+$ denotes the set of nonnegative real numbers). Note that r is a nonnegative real function and not a complex function. We will define an average error $R(\boldsymbol{w})$ as

$$R(\boldsymbol{w}) \stackrel{\text{def}}{=} \sum_{\boldsymbol{x}} \sum_{\boldsymbol{y}} r(\boldsymbol{z}(\boldsymbol{w}, \boldsymbol{x}), \boldsymbol{y}) P(\boldsymbol{x}, \boldsymbol{y}). \tag{1}$$

$R(\boldsymbol{w})$ corresponds to the error between the actual output pattern and the target output pattern of neural networks, and the smaller $R(\boldsymbol{w})$ is, the better the estimation.

2.2 Learning Convergence Theorem

This section will present a learning algorithm for the Complex APCM described in Section 2.1 and prove that it convergences. The algorithm is a complex-valued version of the probabilistic-descent method (Amari, 1967).

We will introduce a parameter n for discrete time. Let $(\boldsymbol{x}_n, \boldsymbol{y}_n)$ be a complex-valued pattern that occurs at time n. Moreover, we will assume that the (complex-valued) parameter $\boldsymbol{w}$ is modified by

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \Delta \boldsymbol{w}_n, \tag{2}$$

where $\boldsymbol{w}_n$ denotes a (complex-valued) parameter at time n. Eqn (2) can be rewritten as follows:

$$Re[\boldsymbol{w}_{n+1}] = Re[\boldsymbol{w}_n] + Re[\Delta \boldsymbol{w}_n], \tag{3}$$

$$Im[\boldsymbol{w}_{n+1}] = Im[\boldsymbol{w}_n] + Im[\Delta \boldsymbol{w}_n], \tag{4}$$

where Re[z], Im[z] denote the real and imaginary parts of a complex number z, respectively. By definition we say that a parameter $\boldsymbol{w}$ is optimal if and only if the average error

$R(\boldsymbol{w})$ is the local or global minimum. Then, the following theorem holds.

THEOREM 1. *Let* $\boldsymbol{A}$ *be a positive definite matrix. Then, by using the update rules*

$$Re[\Delta \boldsymbol{w}_n] = -\varepsilon \boldsymbol{A} \nabla^{Re} r(\boldsymbol{z}(\boldsymbol{w}_n, \boldsymbol{x}_n), \boldsymbol{y}_n), \tag{5}$$

$$Im[\Delta \boldsymbol{w}_n] = -\varepsilon \boldsymbol{A} \nabla^{Im} r(\boldsymbol{z}(\boldsymbol{w}_n, \boldsymbol{x}_n), \boldsymbol{y}_n), \qquad n = 0, 1, \cdots, \tag{6}$$

the (complex-valued) parameter $\boldsymbol{w}$ *approaches the optimum as near as desired by choosing a sufficiently small learning constant* $\varepsilon > 0$ *(*∇^{Re} *is a gradient operator with respect to the real part of* $\boldsymbol{w}$*, and* ∇^{Im} *with respect to the imaginary part).*

Proof. The theory of APCM (Amari, 1967) is applicable to this case. The differences are that $\boldsymbol{w} \in \boldsymbol{R}^p$, $\boldsymbol{x} \in \boldsymbol{R}^n$ and $\boldsymbol{y} \in \boldsymbol{R}^m$ are real-valued variables in the APCM ($\boldsymbol{R}$ denotes the set of real numbers) while $\boldsymbol{w} \in \boldsymbol{C}^p$, $\boldsymbol{x} \in \boldsymbol{C}^n$ and $\boldsymbol{y} \in \boldsymbol{C}^m$ are complex-valued variables in the Complex APCM which influence $\boldsymbol{z}(\boldsymbol{w}, \boldsymbol{x}) : \boldsymbol{C}^p \times \boldsymbol{C}^n \to \boldsymbol{C}^m$, $r(\boldsymbol{y}', \boldsymbol{y}) : \boldsymbol{C}^m \times \boldsymbol{C}^m \to \boldsymbol{R}^+$ and $R(\boldsymbol{w}) : \boldsymbol{C}^p \to \boldsymbol{R}^1$.

In one training step of the Complex APCM, the real part $Re[\boldsymbol{w}] \in \boldsymbol{R}^p$ and the imaginary part $Im[\boldsymbol{w}] \in \boldsymbol{R}^p$ of the (complex-valued) parameter $\boldsymbol{w}$ are independently changed according to eqns (5) and (6). Thus, the way of changing the parameter $\boldsymbol{w}$ in both models are identical in the sense of updating reals. Hence, there is no need to take into account the change of $\boldsymbol{w}$ from a real-valued variable to a complex-valued variable.

Next, $\boldsymbol{x} \in \boldsymbol{C}^n$ and $\boldsymbol{y} \in \boldsymbol{C}^m$ appear in the functions $\boldsymbol{z}(\boldsymbol{w}, \boldsymbol{x})$ and $r(\boldsymbol{y}', \boldsymbol{y})$ which are manipulated only in the form of the mathematical expectation with respect to the complex-valued random variables $(\boldsymbol{x}, \boldsymbol{y})$, e.g. $E_{(\boldsymbol{x},\boldsymbol{y})}[\boldsymbol{z}(\boldsymbol{w}, \boldsymbol{x})]$ and $E_{(\boldsymbol{x},\boldsymbol{y})}[r(\boldsymbol{y}', \boldsymbol{y})]$. Generally, a complex-valued random variable can be manipulated in the same manner as a real-valued random variable. Hence, we can manipulate the functions $\boldsymbol{z}(\boldsymbol{w}, \boldsymbol{x})$ and $r(\boldsymbol{y}', \boldsymbol{y})$, just as the corresponding real functions in the APCM. Therefore, there is no need to change the logic of the proof in Amari's APCM theory. ■

Amari (1967) has proved that the performance of the classifier in the APCM depends on the constant ε and the components of $\boldsymbol{A}$ (see Appendix for the learning convergence

theorem in the APCM). Similarly, it is assumed that the constant ε and the components of $\boldsymbol{A}$ influence the performance of the classifier in the Complex APCM (it should be rigorously proved).

2.3 Learning Rule

2.3.1 *Generalization of Real-BP*

In this section, we will apply the theory of the Complex APCM to a multi-layer (complex-valued) neural network.

We will first describe the Complex-BP model. Fig. 1 shows a model neuron used in the Complex-BP algorithm. The input signals, weights, thresholds and output signals are all complex numbers. The activity Y_n (analogous to the activity in the Real-BP) of neuron n is defined as:

$$Y_n = \sum_m W_{nm} X_m + V_n, \tag{7}$$

where W_{nm} is the (complex-valued) weight connecting neuron n and m, X_m is the (complex-valued) input signal from neuron m, and V_n is the (complex-valued) threshold value of neuron n. To obtain the (complex-valued) output signal, convert the activity value Y_n into its real and imaginary parts as follows.

$$Y_n = x + iy = z, \tag{8}$$

where i denotes $\sqrt{-1}$. Although various output functions of each neuron can be considered, we will use the output function defined by the following equation.

$$f_C(z) = f_R(x) + i f_R(y) \tag{9}$$

where $f_R(u) = 1/(1 + \exp(-u))$ and is called the sigmoid function. It is obvious that $0 \leq Re[f_C],\ Im[f_C] \leq 1$ and $|f_C(z)| \leq \sqrt{2}$. Note also that $f_C(z)$ is not holomorphic, because the Cauchy-Riemann equation does not hold: $\partial f_C(z)/\partial x + i\partial f_C(z)/\partial y = (1 - f_R(x))f_R(x) + i(1 - f_R(y))f_R(y) \neq 0$, where $z = x + iy$.

For the sake of simplicity, the networks used in the analysis and experiments will have 3 layers. We will use w_{ml} for the weight between the input neuron l and the hidden

neuron m, v_{nm} for the weight between the hidden neuron m and the output neuron n, θ_m for the threshold of the hidden neuron m, and γ_n for the threshold of the output neuron n. Let I_l, H_m, O_n denote the output values of the input neuron l, the hidden neuron m, and the output neuron n, respectively. Let also U_m and S_n denote the internal potentials of the hidden neuron m and the output neuron n, respectively. That is, $U_m = \sum_l w_{ml} I_l + \theta_m$, $S_n = \sum_m v_{nm} H_m + \gamma_n$, $H_m = f_C(U_m)$ and $O_n = f_C(S_n)$. Let $\delta^n = T_n - O_n$ denote the error between the actual pattern O_n and the target pattern T_n of output neuron n. We will define the square error for the pattern p as $E_p = (1/2)\sum_{n=1}^{N} |T_n - O_n|^2$, where N is the number of output neurons.

Next, we define a learning rule for the Complex-BP model described above. For a sufficiently small learning constant (learning rate) $\varepsilon > 0$ and a unit matrix $\boldsymbol{A}$, using Theorem 1, we can show that the weights and the thresholds should be modified according to the following equations.

$$\Delta v_{nm} = -\varepsilon \frac{\partial E_p}{\partial Re[v_{nm}]} - i\varepsilon \frac{\partial E_p}{\partial Im[v_{nm}]}, \tag{10}$$

$$\Delta \gamma_n = -\varepsilon \frac{\partial E_p}{\partial Re[\gamma_n]} - i\varepsilon \frac{\partial E_p}{\partial Im[\gamma_n]}, \tag{11}$$

$$\Delta w_{ml} = -\varepsilon \frac{\partial E_p}{\partial Re[w_{ml}]} - i\varepsilon \frac{\partial E_p}{\partial Im[w_{ml}]}, \tag{12}$$

$$\Delta \theta_m = -\varepsilon \frac{\partial E_p}{\partial Re[\theta_m]} - i\varepsilon \frac{\partial E_p}{\partial Im[\theta_m]} \quad . \tag{13}$$

The above eqns (10)-(13) can be expressed as:

$$\Delta v_{nm} = \overline{H}_m \Delta \gamma_n, \tag{14}$$

$$\Delta \gamma_n = \varepsilon\Big(Re[\delta^n](1 - Re[O_n]) Re[O_n] + i Im[\delta^n](1 - Im[O_n]) Im[O_n] \Big), \tag{15}$$

$$\Delta w_{ml} = \overline{I}_l \Delta \theta_m, \tag{16}$$

$$\begin{aligned} \Delta \theta_m = \varepsilon \Bigg[& (1 - Re[H_m]) Re[H_m] \\ & \sum_n \Big(Re[\delta^n](1 - Re[O_n]) Re[O_n] Re[v_{nm}] + Im[\delta^n](1 - Im[O_n]) Im[O_n] Im[v_{nm}] \Big) \\ & -i(1 - Im[H_m]) Im[H_m] \\ & \sum_n \Big(Re[\delta^n](1 - Re[O_n]) Re[O_n] Im[v_{nm}] - Im[\delta^n](1 - Im[O_n]) Im[O_n] Re[v_{nm}] \Big) \Bigg], \end{aligned} \tag{17}$$

where $\overline{z}$ denotes the complex conjugate of a complex number z.

In this connection, the updating rule of the Real-BP are as follows:

$$\Delta v_{nm} = H_m \Delta \gamma_n, \tag{18}$$

$$\Delta \gamma_n = \varepsilon(1 - O_n) O_n \delta^n, \tag{19}$$

$$\Delta w_{ml} = I_l \Delta \theta_m, \tag{20}$$

$$\Delta \theta_m = (1 - H_m) H_m \sum_n v_{nm} \Delta \gamma_n, \tag{21}$$

where $\delta^n,\ I_l,\ H_m, O_n,\ v_{nm},\ \gamma_n,\ w_{ml},\ \theta_m$ are all real numbers. Eqns (14) to (17) resemble those of the Real-BP.

2.3.2 *Discussion*

We first formulated a complex-valued neuron with a *holomorphic* complex function

$$f_C(z) = \frac{1}{1 + \exp(-z)}, \tag{22}$$

where $z = x + iy$, because the holomorphy of a complex function seemed to be natural and to produce many interesting results. Kim et al. independently proposed the complex-valued neuron with this output function (eqn (22)) (Kim et al., 1990). However, the complex-valued back-propagation algorithm with this holomorphic complex function never converged in our experiments. We considered that the cause was non-boundedness of the complex function (eqn (22)) and decided to adopt *bounded* functions. Here, from the Liouville's Theorem (e.g., Derrick, 1984), an holomorphic complex function cannot be bounded on all of $\boldsymbol{C}$ unless it is a constant. Thus, we could not keep the holomorphy of complex functions, because we required boundedness for complex functions. Therefore, we adopted eqn (9) as a solution, which was bounded but non-holomorphic.

Note that there is another formulation of the complex-valued version: the output function is a holomorphic complex function $f_C(z) = z$, where $z = x + iy$, and the number of layers is two, i.e., the network has no hidden layers (Widrow et al., 1975).

3 EXPERIMENTS

In this section, the characteristics of the Complex-BP algorithm given in the previous section are discussed experimentally.

3.1 Learning Speed

In this section, we study the learning speed of the Complex-BP algorithm on a number of examples using complex-valued patterns. And we compare its performance with that of the Real-BP.

We examine its learning speed in terms of a computational complexity perspective (i.e., time and space complexities). Here, time complexity means the sum of four operations for real numbers, and space complexity the sum of learnable parameters (weights and thresholds), where a complex-valued parameter $w = w^R + iw^I$ is counted as two because it consists of a real part w^R and an imaginary part w^I.

The average number of learning cycles needed to converge by the proposed technique was compared with that of the conventional back-propagation technique. In the comparison, the neural network structures such that the time complexity per learning cycle of the Complex-BP was almost equal to that of the Real-BP were used. In addition, the space complexity was also examined.

In the experiments, the initial real and imaginary components of the weights and thresholds were chosen to be random numbers between $-$ 0.3 and + 0.3. The stopping criteria used for learning was

$$\sqrt{\sum_{p}\sum_{n=1}^{N}|T_n^{(p)} - O_n^{(p)}|^2} \quad = \quad 0.10, \tag{23}$$

where $T_n^{(p)},\ O_n^{(p)} \in \boldsymbol{C}$ denote the desired output value, the actual output value of the neuron n for the pattern p, i.e., the left side of eqn (23) denotes the error between the desired output pattern and the actual output pattern; N denotes the number of neurons in the output layer. The presentation of one set of learning patterns to the neural network was regarded as one learning cycle.

Experiment 1

First, a set of simple (complex-valued) learning patterns shown in Table 1 was used to compare the performance of the Complex-BP algorithm with that of the Real-BP algorithm. We used a 1-3-1 three-layered network for the Complex-BP, and a 2-7-2 three-layered network for the Real-BP, because their time complexities per learning cycle were almost equal as shown in Table 2.

In the experiment with the Real-BP, the real component of a complex number was input into the first input neuron, and the imaginary component was input into the second input neuron. The output from the first output neuron was interpreted to be the real component of a complex number; the output from the second output neuron was interpreted to be the imaginary component.

The average convergence of 50 trials for each of the 6 learning rates (0.1, 0.2, $\cdots$, 0.6) was used as the evaluation criterion. Although we stopped learning at the 50,000th iteration, all trials succeeded in converging. The result of the experiments is shown in Fig. 2.

Experiment 2

Next, we conducted an experiment using the set of (complex-valued) learning patterns shown in Table 3. The learning patterns were defined according to the following two rules:- (a) the real part of Complex Number 3 (output) is 1 if Complex Number 1 (input) is equal to Complex Number 2 (input), otherwise it is 0; (b) the imaginary part of Complex Number 3 is 1 if Complex Number 2 is equal to either 1 or i, otherwise it is 0.

The experimental task was the same as in Experiment 1 except for the layered network structure: a 2-4-1 three-layered network was used for the proposed method while a 4-9-2 three-layered network was used for the Real-BP, because their time complexities per learning cycle were equal as shown in Table 4.

In the experiment with the Real-BP, the real and imaginary components of Complex Number 1 and the real and imaginary components of Complex Number 2 were input into the first, second, third and fourth input neurons, respectively. The output from the first output neuron was interpreted to be the real component of a complex number; the output

from the second output neuron was interpreted to be the imaginary component.

We stopped learning at the 100,000th iteration. The results of the experiments are shown in Fig. 3. For reference, we show the rate of convergence in Table 5.

We can conclude from these experiments that the Complex-BP exhibits the following characteristics in learning complex-valued patterns :- the learning speed is several times faster than that of the conventional technique (Figs. 2 and 3), while the space complexity (i.e., the number of learnable parameters) is only about the half of Real-BP (Tables 2 and 4).

3.2 Generalization Ability

In this section, we study the generalization ability of the two sets of the Real-BP and Complex-BP networks and the algorithms. The learning patterns and the networks for the *Experiments 1 and 2* in Section 3.1 were also used to test the generalization ability on unseen data-inputs. The learning constant used in these experiments was 0.5. The *Experiments 1 and 2* in the following correspond to the ones in Section 3.1, respectively.

Experiment 1

After training (using a 1-3-1 network, and the Complex-BP) with the 4 training points (Table 1, Figs. 4(a) and (b)), by presenting the 12 test points shown in Fig. 4(c), the Complex-BP network generated the points as shown in Fig. 5(a). Fig. 5(b) shows the case in which the 2-7-2 Real-BP network was used.

Experiment 2

After training with the 8 training points shown in Table 3, Figs. 6(a) and (b), the 2-4-1 Complex-BP network formed the set of points as shown in Fig. 7(a) for the 8 test points (Fig. 6(c)). The results for the 4-9-2 Real-BP network appear in Fig. 7(b). Here, we need to know the distances between the input training points and the test points to evaluate the generalization performance of the Real-BP and the Complex-BP. But, Figs. 6(a) and (c) do not always express the exact distances between the input training points and the test points. In order to clarify this, for any input training point $\boldsymbol{x} = (x_1, x_2) \in \boldsymbol{C}^2$ and

test point $\boldsymbol{y} = (y_1, y_2) \in \boldsymbol{C}^2$, we define a distance-measure as

$$\begin{aligned} \|\boldsymbol{x}-\boldsymbol{y}\|^2 &\stackrel{\text{def}}{=} |x_1 - y_1|^2 + |x_2 - y_2|^2 \\ &= (Re[x_1 - y_1])^2 + (Im[x_1 - y_1])^2 + (Re[x_2 - y_2])^2 + (Im[x_2 - y_2])^2 \quad (24) \end{aligned}$$

and show the distances between the input training points and the test points in Table 6 using the distance-measure (eqn (24)). For example, the closest input training point to the test point 6 is 5 (Table 6).

The above simulation results clearly suggest that the Complex-BP algorithm has the same degree of the generalization performance as compared to the Real-BP.

3.3 Transforming Geometric Figures

In this section, we will show that the Complex-BP can transform geometric figures in a natural way.

We used a 1-6-1 three-layered network, which transformed a point $(x,\ y)$ into (x', y') in the complex plane. Although the Complex-BP network generates a value z within the range $0 \leq Re[z], Im[z] \leq 1$, for the sake of convenience, we will present it in the figures given below as having a transformed value within the range $-1 \leq Re[z],\ Im[z] \leq 1$.

We also carried out some experiments with a 2-12-2 network with real-valued weights and thresholds, to compare the Complex-BP with the Real-BP. As before, the real component of a complex number was input into the first input neuron, and the imaginary component was input into the second input neuron. The output from the first output neuron was interpreted as the real component of a complex number; the output from the second output neuron was interpreted as the imaginary component.

The learning constant used in these experiments was 0.5. The initial real and imaginary components of the weights and the thresholds were chosen to be random real numbers between 0 and 1. The experiments described in this section consisted of two parts - a training step, followed by a test step.

3.3.1 *Some Examples*

To begin with, we will now present some examples on transformation in which the training input and output pairs were presented 1,000 times (in the usual back-prop manner) in the training step.

Rotation

In the first experiment (using a 1-6-1 network, and the Complex-BP), the training step consisted of learning a set of (complex-valued) weights and thresholds, such that the input set of (straight line) points (indicated by black circles in Fig. 8(a)) gave as output, the (straight line) points (indicated by white circles) rotated counterclockwise over $\pi/2$ radians. These complex-valued weights and thresholds were then used in a second (test) step, in which the input points lying on two straight lines (indicated by black triangles in Figs. 8(a) and (b)) would hopefully be mapped to an output set of points lying on the straight lines (indicated by white triangles) rotated counterclockwise over $\pi/2$ radians. The actual output test points for the Complex-BP did, indeed, lie on the straight lines (indicated by white squares).

It appears that the complex-valued network has learned to generaize the transformation of each point $Z_k (= r_k \exp[i\theta_k])$ into $Z_k \exp[i\alpha] (= r_k \exp[i(\theta_k + \alpha)])$, i.e., the angle of each complex-valued point is updated by a complex-valued factor $\exp[i\alpha]$, but the absolute length of each input point is preserved.

To compare how a real-valued network would perform, the 2-12-2 (real-valued) network mentioned above was trained using the linear pairs of points, i.e., the (input) black circles and (desired output) white circles of Fig. 8. The black triangle points of Fig. 8 were then input with this real-valued network. The outputs were the black squares. Obviously, the Real-BP did not preserve each input point's absolute length. All points were mapped onto straight lines, as shown in Fig. 8.

In the above experiments, the 11 training input points lay on the line $y = -x + 1$ ($0 \leq x \leq 1$) and the 11 training output points lay on the line $y = x + 1$ ($-1 \leq x \leq 0$). The 13 test input points lay on the lines $y = 0.2$ ($-0.9 \leq x \leq 0.3$) (Fig. 8(a)) and $y = -x + 0.5$ ($0 \leq x \leq 0.5$) (Fig. 8(b)). The desired output test points should lie on the

lines $x = -0.2$ and $y = x + 0.5$.

Next, we made an experiment on rotation of the word ISO which consisted of three characters (Fig. 9). The training set of points was as follows: the input set of points lay on the slanted character I (indicated by black circles in Fig. 9(a)), and the output set of points lay on the character I which was straight up (indicated by white circles). The angle between the input points and the output points was $\pi/4$ radians. In a test step, we gave the network some points (indicated by black triangles in Figs. 9(b) and (c)) on two slanted characters S and O as the test input points. The Complex-BP rotated the slanted characters S and O counterclockwise over $\pi/4$ radians, whereas the Real-BP destroyed them (Fig. 9).

Similarity Transformation

We examined a similarity transformation with scaling factor $\alpha = 1/2$ from one circle $x^2 + y^2 = 1$ to another circle $x^2 + y^2 = 0.5^2$ (Fig. 10(a)). The training step consisted of learning a set of (complex-valued) weights and thresholds, such that the input set of (straight line) points (indicated by black circles in Fig. 10(a)) gave as output, the half-scaled straight line points (indicated by white circles). In a second (test) step, the input points lying on a circle (indicated by black triangles) would hopefully be mapped to an output set of points (indicated by white triangles) lying on a half-scaled circle. The actual output test points for the Complex-BP did, indeed, lie on the circle (indicated by white squares).

It appears that the complex-valued network has learned to generalize the transformation of each point Z_k $(= r_k \exp[i\theta_k])$ into αZ_k $(= \alpha r_k \exp[i\theta_k])$, i.e., the absolute length of each complex-valued point is shrunk by a real-valued factor α, but the angle of each input point is preserved.

To compare how a real-valued network would perform, the (real-valued) network was trained using the linear pairs of points, i.e., the (input) black circles and (desired output) white circles of Fig. 10(a). The black triangle points of Fig. 10(a) were then input with this real-valued network. The outputs were the black squares . Obviously, the Real-BP did not preserve each input point's angle. All angles were mapped onto a straight line, as

shown in Fig. 10(a).

We also made a further experiment. Fig. 10(b) shows the result of the responses of the networks to presentation of the points on an arbitrary curved line. We can find that the curved line was halved by the Complex-BP, holding its shape, as in the case of the circle, whereas such a thing did not occur in the Real-BP.

In the two above experiments, the 11 training input points lay on the line $y = x$ ($0 \leq x \leq 1$), and the 11 training output points lay on the line $y = x$ ($0 \leq x \leq 0.5$). In the case of Fig. 10(a), the 12 test input points lay on the circle with equation $x^2 + y^2 = 1$, and the desired output test points should lie on the circle with equation $x^2 + y^2 = 0.5^2$.

In addition, we carried out an experiment on the magnification of a square. The 11 training input points (indicated by black circles in Fig. 10(c)) lay on the line $y = x$ ($0 \leq x \leq 0.3$), and the training output points (indicated by white circles) lay on the straight line $y = x$ ($0 \leq x \leq 0.99$) which could be generated by magnifying the line $y = x$ ($0 \leq x \leq 0.3$) with a scale magnification factor of 3.3. For a square whose side was 0.3 (indicated by black triangles), the Complex-BP generated a square whose side was nearly 1.0 (indicated by white squares), whereas the Real-BP generated points (indicated by black squares) on the straight line $y = x$.

Parallel Displacement

Fig. 11(a) shows the results of an experiment on parallel displacement of a straight line. The training points used in the experiment were as follows: the input set of (straight line) points (indicated by black circles in Fig. 11(a)) gave as output, the straight line points displaced in parallel (indicated by white circles). The distance of the parallel displacement was $1/\sqrt{2}$, and the direction was a $-\pi/4$-radian angle. In a test step, the input points lying on a straight line (indicated by black triangles in Fig. 11(a)) would hopefully be mapped to an output set of points (indicated by white triangles) lying on a straight line displaced in parallel. The actual output test points for the Complex-BP did, indeed, lie on the straight line (indicated by white squares).

It appears that the complex-valued network has learned to generalize the transformation of each point Z_k into $Z_k + \alpha$, where α is a complex number.

To compare how a real-valued network would perform, the 2-12-2 (real-valued) network was trained using the linear pairs of points, i.e., the (input) black circles and (desired output) white circles of Fig. 11(a). The black triangle points of Fig. 11(a) were then input with this real-valued network. The outputs were the black squares. Obviously, the Real-BP did not displace them in parallel.

In the above experiments, the 11 training input points lay on the line $y = x + 1$ ($-1 \leq x \leq 0$) and the 11 training output points lay on the line $y = x$ ($-0.5 \leq x \leq 0.5$). The 11 test input points lay on the straight line $y = x$ ($-0.5 \leq x \leq 0.5$). The desired output test points should lie on the straight line $y = x - 1$ ($0 \leq x \leq 1$).

We also conducted an experiment on parallel displacement of an arbitrary curved line. As shown in Fig. 11(b), only the Complex-BP moved it in parallel.

3.3.2 *Systematic Evaluation*

Next, we systematically investigated the generalization ability of the Complex-BP algorithm on the transformation of geometric figures.

In the experiments (using 1-1-1 and 1-6-1 networks and the Complex-BP), training input and output pairs were as follows:

1. Rotation

   The input set of (straight line) points (indicated by black circles in Fig. 12(a)) gave as output the (straight line) points (indicated by white circles) rotated counterclockwise over $\pi/2$ radians.

2. Similarity Transformation

   The input set of (straight line) points (indicated by black circles in Fig. 12(b)) gave as output the half-scaled straight line points (indicated by white circles).

3. Parallel Displacement

   The input set of (straight line) points (indicated by black circles in Fig. 12(c)) gave as output the straight line points displaced in parallel (indicated by white circles), where the distance of the parallel displacement was $1/\sqrt{2}$, and the direction was a $-\pi/4$-radian angle.

Note that the purpose of the use of a 1-1-1 network is to investigate the degree of the approximation of the results of the mathematical analysis which will be presented in Section 5.

For each of the above three cases, the input (test) points lying on a circle (indicated by black triangles in Fig. 12) were presented in a second (test) step. We then evaluated the performance of the generalization ability of the Complex-BP on *a rotation angle*, *a similitude ratio* and *a parallel displacement vector*. The evaluation results appear in Fig. 13. The vertical line of Fig. 13 denotes the generalization performance, and the horizontal line the difference ϕ between the argument of a test point and that of an input training point. *Error* in Fig. 13 refers to the left side of eqn (23), i.e., the error between the desired output patterns and the actual output patterns in the training step. As a evaluation criteria of the generalization performance, we used $|E^R(\phi)|$, $|E^S(\phi)|$ and $|E^P(\phi)|$ which denoted the Euclidean distances between the actual output test point and the expected output test point (see Fig. 13). As shown in Fig. 13, the generalization error (generalization performance) increased as the distance between the test point and the input training point became larger (i.e., ϕ became larger), and it showed the maximum value around the point which gave the largest distance ($\phi \approx 180$). Furthermore, it decreased again as the test point approached the input training point. Fig. 13 also suggests that the generalization error on the transformation of geometric figures decreases as the number of hidden neurons increases, where only one hidden neuron was used in the three experiments of Figs. 13(a)-(c) and six hidden neurons were used in the three experiments of Figs. 13(d)-(f).

In the above experiments, the 11 training input points lay on the line $x = 0$ $(0 \leq y \leq 1)$ and the 11 training output points lay on the line $y = 0$ $(-1 \leq x \leq 0)$ for the rotation, the 11 training input points lay on the line $x = 0$ $(0 \leq y \leq 1)$ and the 11 training output points lay on the line $x = 0$ $(0 \leq y \leq 0.5)$ for the similarity transformation, and the 11 training input points lay on the line $x = 0$ $(0 \leq y \leq 1)$ and the 11 training output points lay on the line $x = 0.5$ $(-0.5 \leq y \leq 0.5)$ for the parallel displacement. The 8 test input points lay on the circle $x^2 + y^2 = 0.5^2$ for all three cases.

3.3.3 *Discussion*

As we have seen in Section 3.3.1, 1-n-1 Complex-BP networks have the ability to generalize the transformation of geometric figures. This brings us to the second point, that is, do the 1-n-1 Complex-BP networks have such a usual generalization ability that the Real-BP networks have? At first sight, the Real-BP and Complex-BP algorithms appear to have two different abilities of the generalization. In order to investigate whether this is true or not, we tested the generalization ability of the 1-n-1 Complex-BP network for a continuous mapping task which 2-m-2 Real-BP networks could solve, which appeared in (Tsutsumi, 1989).

In the experiments, a set of 25 training points shown in Fig. 14 was used for a 1-6-1 Complex-BP network and a 2-12-2 Real-BP network. After the sufficient training, by presenting the 252 test points on the 12 dotted lines shown in Fig. 15, the actual output points formed the solid lines as shown in Figs. 15(a) and (b). Fig. 16 shows the case in which the input training points are Fig. 14(b) and the target training points are Fig. 14(a).

It is suggested from Figs. 15 and 16 that both the 1-6-1 Complex-BP network and the 2-12-2 Real-BP network can obtain the same degree of generalization.

4 MATHEMATICAL ANALYSIS

In this section, the learning characteristics and the ability to generalize the transformation of the Complex-BP algorithm described in the previous section are analyzed.

4.1 The Geometry of Learning

First, we investigated the structure of the learning rule of the Complex-BP algorithm, using the three-layered (complex-valued) neural network defined in Section 2.3 as an example.

Let Δz^R, Δz^I be the real part and the imaginary part of the magnitude of change of a learnable parameter z, respectively; i.e., $\Delta z^R = Re[\Delta z], \quad \Delta z^I = Im[\Delta z]$. Then, the

learning rule in eqns (14)-(17) can be expressed as:

$$\begin{bmatrix}\Delta v_{nm}^R\\ \Delta v_{nm}^I\end{bmatrix} = \begin{bmatrix} Re[H_m] & Im[H_m]\\ -Im[H_m] & Re[H_m]\end{bmatrix}\begin{bmatrix}\Delta\gamma_n^R\\ \Delta\gamma_n^I\end{bmatrix} = |H_m|\begin{bmatrix} \cos\beta_m & \sin\beta_m\\ -\sin\beta_m & \cos\beta_m\end{bmatrix}\begin{bmatrix}\Delta\gamma_n^R\\ \Delta\gamma_n^I\end{bmatrix}, \quad (25)$$

$$\Big(\Delta v_{nm} = |H_m|e^{-i\beta_m}\Delta\gamma_n\Big), \quad (26)$$

$$\begin{bmatrix}\Delta\gamma_n^R\\ \Delta\gamma_n^I\end{bmatrix} = \varepsilon\begin{bmatrix} A_n & 0\\ 0 & B_n\end{bmatrix}\begin{bmatrix} Re[\delta^n]\\ Im[\delta^n]\end{bmatrix}, \quad (27)$$

$$\begin{bmatrix}\Delta w_{ml}^R\\ \Delta w_{ml}^I\end{bmatrix} = \begin{bmatrix} Re[I_l] & Im[I_l]\\ -Im[I_l] & Re[I_l]\end{bmatrix}\begin{bmatrix}\Delta\theta_m^R\\ \Delta\theta_m^I\end{bmatrix} = |I_l|\begin{bmatrix} \cos\phi_l & \sin\phi_l\\ -\sin\phi_l & \cos\phi_l\end{bmatrix}\begin{bmatrix}\Delta\theta_m^R\\ \Delta\theta_m^I\end{bmatrix}, \quad (28)$$

$$\begin{aligned}\begin{bmatrix}\Delta\theta_m^R\\ \Delta\theta_m^I\end{bmatrix} &= \begin{bmatrix}C_m & 0\\ 0 & D_m\end{bmatrix}\sum_n\begin{bmatrix} Re[v_{nm}] & Im[v_{nm}]\\ -Im[v_{nm}] & Re[v_{nm}]\end{bmatrix}\begin{bmatrix}\Delta\gamma_n^R\\ \Delta\gamma_n^I\end{bmatrix}\\ &= \begin{bmatrix}C_m & 0\\ 0 & D_m\end{bmatrix}\sum_n|v_{nm}|\begin{bmatrix} \cos\varphi_{nm} & \sin\varphi_{nm}\\ -\sin\varphi_{nm} & \cos\varphi_{nm}\end{bmatrix}\begin{bmatrix}\Delta\gamma_n^R\\ \Delta\gamma_n^I\end{bmatrix}, \quad (29)\end{aligned}$$

where $A_n = (1-Re[O_n])Re[O_n]$, $B_n = (1-Im[O_n])Im[O_n]$, $C_m = (1-Re[H_m])Re[H_m]$, $D_m = (1-Im[H_m])Im[H_m]$, $\beta_m = \arctan(Im[H_m]/Re[H_m])$, $\phi_l = \arctan(Im[I_l]/Re[I_l])$, and $\varphi_{nm} = \arctan(Im[v_{nm}]/Re[v_{nm}])$.

In eqn (25), $|H_m|$ is a similarity transformation (reduction, magnification), and $\begin{bmatrix} \cos\beta_m & \sin\beta_m\\ -\sin\beta_m & \cos\beta_m\end{bmatrix}$ is a clockwise rotation by β_m radians around the origin. Thus, eqn (25) performs the linear transformation called two-dimensional motion. Hence, we find that the magnitude of change in the weight between the hidden and output neurons $(\Delta v_{nm}^R, \Delta v_{nm}^I)$ can be obtained via the above linear transformation (two-dimensional motion) of $(\Delta\gamma_n^R, \Delta\gamma_n^I)$ which is the magnitude of change in the threshold of the output neuron (Fig. 17). Similarly, the magnitude of change in the threshold of the hidden neuron $(\Delta\theta_m^R, \Delta\theta_m^I)$ can be obtained by applying the two-dimensional motion concerning v_{nm} (the weight between the hidden and output neurons) to $(\Delta\gamma_n^R, \Delta\gamma_n^I)$ which is the magnitude of change in the threshold of the output neuron (eqn (29)). Finally, $(\Delta w_{ml}^R, \Delta w_{ml}^I)$ can be obtained by applying the two-dimensional motion concerning I_l to $(\Delta\theta_m^R, \Delta\theta_m^I)$ (eqn (28)). Thus, it seems that the error propagation in the Complex-BP is based on

two-dimensional motion.

Next, we will contrast this with the geometry of the Real-BP. Representing the magnitude of the learnable parameter updates (real number) as a point on a real line (1 dimension), we can interpret Δv_{nm} as the product of $|\Delta\gamma_n|$ and H_m (eqn (18), Fig. 18). Similarly, the product of $|\Delta\gamma_n|$ and v_{nm} produces $\Delta\theta_m$ (eqn (21)), and the product of $|\Delta\theta_m|$ and I_l leads Δw_{ml} (eqn (20)). Hence, the error propagation in the Real-BP is based on one-dimensional motion.

Therefore, we see that extending the Real-BP to complex numbers varies the structure of the error propagation from one to two dimensions.

The two-dimensional structure of the error propagation described above also means that the units of learning in the Complex-BP algorithm are complex-valued signals flowing through neural networks. For example, both Δv_{nm}^R and Δv_{nm}^I are functions of the real parts ($Re[H_m]$, $Re[O_n]$) and the imaginary parts ($Im[H_m]$, $Im[O_n]$) of the complex-valued signals (H_m, O_n) flowing through the neural networks (eqn (25)). That is, there is a relation between Δv_{nm}^R and Δv_{nm}^I through ($Re[H_m]$, $Re[O_n]$) and ($Im[H_m]$, $Im[O_n]$). Similarly, there are relations between Δw_{ml}^R and Δw_{ml}^I (eqn (28)), and between $\Delta\theta_m^R$ and $\Delta\theta_m^I$ (eqn (29)). Eqn (27) indicates no relation between $\Delta\gamma_n^R$ and $\Delta\gamma_n^I$. However, we can find one since $Re[O_n]$ is a function of $Re[H_m]$ and $Im[H_m]$, because

$$Re[O_n] = f_R(Re[S_n]), \tag{30}$$

where

$$Re[S_n] = \sum_m \Big(Re[v_{nm}]Re[H_m] - Im[v_{nm}]Im[H_m] \Big) + Re[\gamma_n]. \tag{31}$$

Similarly, $Im[O_n]$ is also a function of $Re[H_m]$ and $Im[H_m]$, because

$$Im[O_n] = f_R(Im[S_n]), \tag{32}$$

where

$$Im[S_n] = \sum_m \Big(Re[v_{nm}]Im[H_m] + Im[v_{nm}]Re[H_m] \Big) + Im[\gamma_n]. \tag{33}$$

Thus, both $\Delta\gamma_n^R$ and $\Delta\gamma_n^I$ are functions of $Re[H_m]$ and $Im[H_m]$. Hence, there is also a relation between $\Delta\gamma_n^R$ and $\Delta\gamma_n^I$ through $Re[H_m]$ and $Im[H_m]$. Therefore, in the Complex-BP algorithm, both the real part and the imaginary part of learnable parameters are

modified as a function of the real part and the imaginary part of complex-valued signals, respectively (Fig. 19). From these facts, we may conclude that complex-valued signals flowing through neural networks are the unit of learning in the Complex-BP algorithm.

4.2 Improving Learning Speed

In this section, we will show how the error propagation of the Complex-BP algorithm is based on two-dimensional geometry and how this improves learning speed.

In the learning rule of the real-valued back-propagation (eqns (18)-(21)), $(1-H_m)H_m \in \boldsymbol{R}$ and $(1-O_n)O_n \in \boldsymbol{R}$ are the derivative $(1 - f_R(u))\, f_R(u)$ of the sigmoid function $f_R(u) = 1/(1+\exp(-u))$ which is the output function of each neuron. The value of the derivative asymptotically approaches 0 as the absolute value of the net input u to a neuron increases (Fig. 20). Hence, as $|u|$ increases to make the output value of a neuron exactly approach 0.0 or 1.0, the derivative $(1 - f_R(u))\, f_R(u)$ shows a small value, which causes *a standstill in learning*. This phenomenon is called *getting stuck in a local minimum* if it continuously takes place for a considerable length of time, and the error between the actual output value and the desired output value remains large. As is generally known, this is the mechanism of *standstill in learning* in the Real-BP.

On the other hand, two kinds of derivatives of the sigmoid function appear in the learning rule of the Complex-BP algorithm (eqns (14)-(17)): one is the derivative of the real part of an output function $((1 - Re[O_n])Re[O_n],\ \ (1 - Re[H_m])Re[H_m])$, the other is that of the imaginary part $((1 - Im[O_n])Im[O_n],\ \ (1 - Im[H_m])Im[H_m])$. The learning rule of the Complex-BP algorithm basically consists of two linear combinations of them:

$$\alpha_1(1-Re[O_n])Re[O_n] + \beta_1(1-Im[O_n])Im[O_n], \tag{34}$$

$$\alpha_2(1-Re[H_m])Re[H_m] + \beta_2(1-Im[H_m])Im[H_m], \tag{35}$$

where $\alpha_k, \beta_k \in \boldsymbol{R}$ $(k = 1, 2)$. Note that eqn (34) has a very small value only when both $(1 - Re[O_n])Re[O_n]$ and $(1 - Im[O_n])Im[O_n]$ are very small. Hence, eqn (34) does not show an extremely small value even if $(1 - Re[O_n])Re[O_n]$ is very small, because $(1 - Im[O_n])Im[O_n]$ is not always small in the Complex-BP algorithm (whereas the magnitude of learnable parameter updates inevitably becomes quite small if $(1 - O_n)O_n \in$

$\boldsymbol{R}$ is quite small in the Real-BP algorithm (eqns (18)-(21))). In this sense, the real factor $((1 - Re[O_n])Re[O_n],\ \ (1 - Re[H_m])Re[H_m])$ makes up for the imaginary factor $((1 - Im[O_n])Im[O_n],\ \ (1 - Im[H_m])Im[H_m])$ showing an abnormally small value, and vice versa. Thus, compared with the updating rule of the Real-BP, the Complex-BP is such that the probability for a standstill in learning is reduced. This indicates that the learning speed of the Complex-BP is faster than that of the Real-BP, which has been confirmed by computational experiments on complex-valued patterns in Section 3.1.

We can assume that the structure of reducing standstill in learning by the linear combinations (eqns (34) and (35)) of the real component and the imaginary component of the derivative of an output function causes the learning speed of the Complex-BP algorithm on a number of examples using complex-valued patterns described in Section 3.1.

4.3 Transforming Geometric Figures

This section presents a mathematical analysis of the behavior of a complex-valued neural network which has learned the concept of rotation, similarity transformation or parallel displacement using the Complex-BP algorithm.

We will introduce a simple 1-1-1 three-layered (complex-valued) network for the analysis. We will use $v\exp[iw] \in \boldsymbol{C}$ for the weight between the input and hidden neurons, $c\exp[id] \in \boldsymbol{C}$ for the weight between the hidden and output neurons, $s\exp[it] \in \boldsymbol{C}$ for the threshold of the hidden neuron, and $r\exp[il] \in \boldsymbol{C}$ for the threshold of the output neuron. Let $v^0\exp[iw^0]$, $c^0\exp[id^0]$, $s^0\exp[it^0]$ and $r^0\exp[il^0]$ denote the learned values of the learnable parameters.

First, we investigate the behavior of the Complex-BP network which has learned *rotation*. Learning patterns are as follows: p points with equal intervals on a straight line which forms an angle of x radians with the real axis, are rotated counterclockwise over α radians in the complex plane (Fig. 21). That is, there are p training points such that for any $1 \leq k \leq p$, an input point can be expressed as

$$ka\exp[ix], \tag{36}$$

and the corresponding output point as

$$\frac{1}{2}ka\exp[i(x+\alpha)]+\frac{1}{\sqrt{2}}\exp\left[i\frac{\pi}{4}\right], \tag{37}$$

where $k,p\in\boldsymbol{N}$ ($\boldsymbol{N}$ denotes the set of natural numbers), $a,x,\alpha\in\boldsymbol{R}^{+}$, $0<pa\leq 1$ which limits the values of learning patterns to the range from -1 to 1 in the complex plane, and the constant a denotes the interval between points. Note that although the output points take a value z within the range $-1\leq Re[z],\ Im[z]\leq 1$, we transformed them as having a value z within the range $0\leq Re[z],\ Im[z]\leq 1$, because the Complex-BP network generates a value z within the range $0\leq Re[z],\ Im[z]\leq 1$. For this reason, eqn (37) seem to be somewhat complicated.

The following theorem will explain the qualitative properties of the generalization ability of the Complex-BP on a rotation angle.

THEOREM 2. *Fix $1\leq k\leq p$ arbitrarily. To the Complex-BP network which has learned the training points* (eqns (36) and (37)), *a test point $ka\exp[i(x+\phi)]$ is given which can be obtained by a counterclockwise rotation of an input training point $ka\exp[ix]$ by arbitrary ϕ radians around the origin* (Fig. 21). *Then, the network generates the following value*:

$$\left[\frac{1}{2}ka\exp\left[i(x+\phi+\alpha)\right]+\frac{1}{\sqrt{2}}\exp\left[i\frac{\pi}{4}\right]\right]+E^{R}(\phi)\in\boldsymbol{C}. \tag{38}$$

The first term of eqn (38) *refers to the point which can be obtained by the counterclockwise rotation of the test point $ka\exp[i(x+\phi)]$ by α radians around the origin* (Fig. 21). *Note that α is the angle which the network has learned. Also, the second term $E^{R}(\phi)$ is a complex number which denotes the error, and the absolute value called Generalization Error on Angle is given in the following expression*:

$$|E^{R}(\phi)|=M\left|\sin\left(\frac{\phi}{2}\right)\right|, \tag{39}$$

where M is a constant (*see Appendix for M*).

REMARK. The value of M differs with each learning because M depends on the values of the learnable parameters after learning (see Appendix). That is, the value of M is a

constant in the world after learning in which the learnable parameters are fixed. Therefore, M is a constant, not a function of α, in Theorem 2 where a situation after one learning with a fixed value of α is assumed.

The proof will be given in the Appendix.

The above theorem tells us that the *Generalization Error on Angle* $|E^R(\phi)|$ increases as the distance between the test point and the input training point increases (i.e., ϕ becomes larger), and it shows the maximum value M at the point which gives the largest distance ($\phi = 180$). Furthermore, it decreases as the test point becomes closer to the input training point.

Next, we will explain the behavior of the Complex-BP network which has learned a *similarity transformation*. We will use the following learning patterns: p points with equal intervals a on a straight line which forms an angle of x radians with the real axis, are transformed into the points obtained by the similarity transformation with the similitude ratio β in the complex plane, respectively (Fig. 22). That is, there are p training points; for any $1 \leq k \leq p$, an input point can be expressed as

$$ka \exp[ix], \tag{40}$$

and the corresponding output point as

$$\frac{1}{2}ka\beta \exp[ix] + \frac{1}{\sqrt{2}} \exp\left[i\frac{\pi}{4}\right], \tag{41}$$

where $k, p \in \boldsymbol{N}$, $a, x, \beta \in \boldsymbol{R}^+$, $0 < pa\beta \leq 1$ which limits the values of learning patterns to the range from -1 to 1 in the complex plane. Note that although the output points take a value z within the range $-1 \leq Re[z],\ Im[z] \leq 1$, we transformed them as having a value z within the range $0 \leq Re[z],\ Im[z] \leq 1$ as in the case of rotation.

The following theorem shows the qualitative property of the generalization ability of the Complex-BP on a similitude ratio.

THEOREM 3. *Fix* $1 \leq k \leq p$ *arbitrarily. To the Complex-BP network which has learned the training points* (eqns (40) and (41)), *a test point* $ka \exp[i(x+\phi)]$ *is given which can be*

obtained by a counterclockwise rotation of an input training point $ka\exp[ix]$ *by arbitrary* ϕ *radians around the origin* (Fig. 22). *Then, the network generates the following value*:

$$\left[\frac{1}{2}ka\beta\exp\left[i(x+\phi)\right]+\frac{1}{\sqrt{2}}\exp\left[i\frac{\pi}{4}\right]\right]+E^S(\phi)\in\boldsymbol{C}. \tag{42}$$

The first term of eqn (42) *refers to the point which can be obtained by the similarity transformation of the test point* $ka\exp[i(x+\phi)]$ *on the distance from the origin with the similitude ratio* β *in the complex plane* (Fig. 22). *Note that* β *is the similitude ratio which the network has learned. Also, the second term* $E^S(\phi)$ *is a complex number which denotes the error, and the absolute value called Generalization Error on Similitude Ratio is given in the following expression*:

$$|E^S(\phi)|=M\left|\sin\left(\frac{\phi}{2}\right)\right|, \tag{43}$$

where M *is a constant* (*see Appendix for* M).

REMARK. For the same reason as Theorem 2, M is a constant, not a function of β, in Theorem 3 where a situation after one learning with a fixed value of β is assumed.

The proof is omitted, because it can be done in the same way as Theorem 2.

We derive from Theorem 3 that the *Generalization Error on Similitude Ratio* $|E^S(\phi)|$ increases as the distance between the test point and the input training point increases (i.e., ϕ becomes larger), and it takes the maximum value M at the point which gives the largest distance ($\phi=180$). Furthermore, it decreases as the test point approaches the input training point.

Finally, we will show the behavior of the Complex-BP network which has learned *parallel displacement*. We will use the following learning patterns: p points with equal intervals a on a straight line which forms an angle of x radians with the real axis, are transformed into the points which can be obtained by the parallel displacement with a complex number $\gamma=\tau\exp[i\omega]$ (called *parallel displacement vector* here) determining the direction and distance in the complex plane, respectively (Fig. 23). That is, there are p training points; for any $1\leq k\leq p$, an input point can be expressed as

$$ka\exp[ix], \tag{44}$$

and the corresponding output point as

$$\frac{1}{2}(ka\exp[ix]+\gamma)+\frac{1}{\sqrt{2}}\exp\left[i\frac{\pi}{4}\right], \tag{45}$$

where $k,p\in\boldsymbol{N}$, $a,x\in\boldsymbol{R}^{+}$, $\gamma\in\boldsymbol{C}$ and

$$-1\leq Re[pa\exp[ix]+\gamma],\ Im[pa\exp[ix]+\gamma]\leq 1 \tag{46}$$

which limits the values of learning patterns to the range from -1 to 1 in the complex plane. Note that although the output points take a value z within the range $-1\leq Re[z],\ Im[z]\leq 1$, we transformed them as having a value z within the range $0\leq Re[z],\ Im[z]\leq 1$ as in the previous cases.

We can obtain the following theorem which clarifies the qualitative property of the generalization ability of the Complex-BP on a parallel displacement vector.

THEOREM 4. *Fix $1\leq k\leq p$ arbitrarily. To the Complex-BP network which has learned the training points* (eqns (44) and (45)), *a test point $ka\exp[i(x+\phi)]$ is given which can be obtained by a counterclockwise rotation of an input training point $ka\exp[ix]$ by arbitrary ϕ radians around the origin* (Fig. 23). *Then, the network generates the following value*:

$$\left[\frac{1}{2}\Big(ka\exp\left[i(x+\phi)\right]+\gamma\Big)+\frac{1}{\sqrt{2}}\exp\left[i\frac{\pi}{4}\right]\right]+E^{P}(\phi)\in\boldsymbol{C}. \tag{47}$$

The first term of eqn (47) *refers to the point which can be obtained by the parallel displacement of the test point $ka\exp[i(x+\phi)]$ with the parallel displacement vector γ* (Fig. 23). *Note that γ is the parallel displacement vector which the network has learned. Also, the second term $E^{P}(\phi)$ is a complex number which denotes the error, and the absolute value called the Generalization Error on Parallel Displacement Vector is given in the following expression*:

$$|E^{P}(\phi)|=M'\left|\sin\left(\frac{\phi}{2}\right)\right|, \tag{48}$$

where M' is a constant (*see Appendix for M'*).

REMARK. For the same reason as Theorem 2, M' is a constant, not a function of γ, in Theorem 4 where a situation after one learning with a fixed value of γ is assumed.

We can obtain this theorem in the same manner as Theorem 2. Therefore the proof is omitted.

Theorem 4 indicates that the *Generalization Error on Parallel Displacement Vector* $|E^P(\phi)|$ increases as the distance between the test point and the input training point becomes larger (i.e., ϕ becomes larger), and it shows the maximum value M' at the point which gives the largest distance ($\phi = 180$). Furthermore, it decreases as the test point approaches the input training point.

5 DISCUSSION

In this section, we will discuss the experimental results and the mathematical results described in Sections 3 and 4 on the learning characteristics, the usual generalization abiliby, and the ability to generalize the transformation of the Complex-BP algorithm.

We conducted the experiments on the learning characteristics using the comparatively small number of learning patterns and the comparatively small networks and showed the superiority of the Complex-BP algorithm in terms of a computational complexity perspective in Section 3.1. We believe that the Complex-BP algorithm can be more and more superior to the Real-BP algorithm when it tackles larger problems with larger networks such as massive real world applications. Because the experiment results in Section 3.1 suggest that the difference of the learning speed between the Complex-BP and the Real-BP shown in *Experiment 2* is larger than that shown in *Experiment 1* where the network size and the number of learning patterns used in *Experiment 2* are larger than those used in *Experiment 1* (see Figs. 2 and 3, Tables 1 and 3). On the other hand, from the experiment results in Section 3.2 one may say that the generalization performance will not change according to the network size and the number of the learning patterns (see Figs. 5 and 7). Systematic experiments would be needed to clarify the above statements.

It is clear from Section 3.3 that the Complex-BP can transform geometric figures in a way that the Real-BP cannot. We will first discuss some examples on transformation described in Section 3.3.1. The counterclockwise rotation of a point (x, y) in the complex plane by θ radians around the origin corresponds to multiplying that complex number $z_1 = x + iy$ by the complex number $z_2 = \exp[i\theta]$, which has a radius of 1 and an argument of θ radians. That is, $z_1 z_2$ denotes the point generated by the counterclockwise rotation of a point (x, y) by θ radians around the origin. Furthermore, similarity transformations (reduction and magnification), and parallel displacement of a point (x, y) in the complex plane correspond respectively to:- a) multiplying a complex number $z_1 = x + iy$ by the real number α, b) adding a complex number w to $z_1 = x + iy$. We therefore believe that the complex-valued neural network has learned the complex function $g(z) = z \exp[i\theta]$, $g(z) = \alpha z$ or $g(z) = z + w$ (for rotation, similarity transformation, and translation, respectively) in the experiments of Section 3.3.1. For example, $\theta = \pi/2$ in Fig. 8 (rotation), $\alpha = 0.5$ in Fig. 10(a) (reduction), and $w = 0.5 - 0.5i$ in Fig. 11(a) (parallel displacement). One should note that the neural network has learned nothing but some points on a certain straight line in the domain; nevertheless the domain of the complex function g is $[-1, 1] \times [-1, 1]$. The neural network presented a sequence of some points in the domain, the responses of the neural network to presentation of all points in the domain were nearly the values of the learned complex function g. This behavior of complex-valued neural networks closely resembles the Identity Theorem in complex analysis, which we will now state.

THE IDENTITY THEOREM. *Let F and G be holomorphic functions over the complex domain D. If $F(z) = G(z)$ on a given line in D, then $F(z) = G(z)$ over D identically.*

This theorem is indicative of a phenomenon found in complex analysis which does not exist in real analysis. We interpret the behavior of the Complex-BP, as shown in Section 3.3.1, in terms of the Identity Theorem. We will assume that the training points are obtained from the points on a given (straight) line in the domain of the true complex function $F : [-1, 1] \times [-1, 1] \rightarrow [-1, 1] \times [-1, 1]$. The neural network approximates

F on the basis of the given training points, resulting in a complex function G (where $G(z)$ should equal $F(z)$, at least on the training points). Then, for all z in the complex domain, the neural network generates the point $G(z)$ which is close to $F(z)$, as though it had satisfied the Identity Theorem. We believe that Complex-BP networks satisfy the Identity Theorem, that is, Complex-BP networks can approximate complex functions just by training them only over a part of the domain of the complex functions.

On the other hand, as we have seen in Section 4.3, the Generalization Error of the Complex-BP on transformation of geometric figures can be represented by the sine of the difference between the argument of the test point and that of the input training point. This mathematical results agree qualitatively with the simulation results in Section 3.3.2 which state that the Generalization Error increases as the distance between the test point and the input training point increases, and it takes the maximum value M around the point which gives the largest distance; furthermore, it decreases as the test point approaches the input training point. Here, we investigated the theoretical values and the experimental values of M based on the simulation results (using 1-1-1 networks) in Section 3.3.2. Table 7 shows that there are some errors beween the theoretical values and the experimental values of M. It is assumed that the cause of such errors is that Theorems 2, 3 and 4 are approximations, i.e., the sigmoid function in the output function of each neuron was approximated by the piecewise linear function. Thus, we can conclude that although there are approximation errors, Theorems 2, 3 and 4 clarify the qualitative property of the generalization ability of the Complex-BP on the transformation of geometric figures.

The mathematical analysis in Section 4.3 is restricted to a class of transformation of geometric figures such that the training points are lying on a line starting from the origin, and the test points are obtained by an arbitrary counterclockwise rotation of the input training points. Thus, Theorems 2, 3 and 4 are applicable to only Figs. 9 and 10 of the simulation results (Figs. 8-11) presented in Section 3.3.1 (note that the training points in Fig. 9 are not precise). Judging from the method of the analysis, it seems reasonable to suppose that similar theorems not only for Figs. 8 and 11 but also for more general networks such as $1-n-1$ and $1-n_1-n_2-\cdots-n_k-1$ networks can be proved using the approach of Theorems 2, 3 and 4. The following problems may, however, occur: (a)

Unlike Theorem 2, 3 and 4, the generalization error on the transformation of geometric figures cannot be simply represented by the sine. (b) The approximation errors increase as the number of neurons increases, because the sigmoid function in the output function of each neuron is approximated by piecewise linear functions. To solve these problems, another approach would be needed.

We have discovered that the Complex-BP has the ability to transform geometric figures in a way that the Real-BP cannot as its *inherent* property. On the other hand, we have experimentally clarified in Sections 3.2 and 3.3.3 that the Complex-BP algorithm has the same degree of the *usual* generalization performance as compared to the Real-BP. The question now arises: while the Complex-BP has the inherent generalization ability such as the ability to transform geometric figures, why can the Complex-BP have the same degree of the usual generalization performance as compared to the Real-BP? Our answer to this question is given below. The learning patterns used in the experiments on transforming geometric figures in Sections 3.3.1 and 3.3.2 were all very peculiar in the meaning that some 10 learning patterns with narrow intervals (i.e., high density) massed on part of the plane (see Figs. 8-12). On the other hand, the learning patterns used in the experiments on the learning speed and the usual generalization ability in Sections 3.2 and 3.3.3 all had only low peculiarity. That is,

1. Only four learning patterns with wide intervals (i.e., low density) were scattered on the plane in the *Experiment 1* of Section 3.2, that is, only a few learning patterns were used and the distances between the learning patterns were large (see Table 1, Figs. 4(a)(b)).

2. The learning patterns were evenly distributed on the plane, although many learning patterns with high density were used in the experiment of Section 3.3.3 (see Fig. 14).

And we cannot discuss the reason why the Complex-BP could have the same degree of the usual generalization performance as compared to the Real-BP in the *Experiment 2* of Section 3.2 bacause the 2-4-1 Complex-BP network was used in the *Experiment 2* of Section 3.2, which was substantially different from the 1-n-1 Complex-BP network used

for transforming geometric figures. We can be fairly certain that the inherent generalization ability to transform geometric figures is unique to the 1-n-1 network structure. Thus we believe that the structures of the network and the learning patterns caused the experimental results that the Complex-BP could have the same degree of the usual generalization performance as compared to the Real-BP, in other words, the use of the very peculiar learning patterns in the particular 1-n-1 network made the inherent ability to generalize emerge.

The search for other various tasks in which the behavior of the Complex-BP network is dramatically different from that of the Real-BP network (for example, the Real-BP can do, whereas the Complex-BP cannot) is a future research topic, which will strongly open the complex-valued neural network world. And it should be noted here that we cannot always say that other inherent abilities of the Complex-BP algorithm which will be probably discovered in a future are superior to that of the Real-BP because the superiority of the Complex-BP as compared to the Real-BP depends on which problems and how the algorithms will be applied to.

6 CONCLUSIONS

We have proposed a complex-valued version of the back-propagation learning algorithm, where the input signals, weights, thresholds, and output signals are all complex numbers. Furthermore, we have investigated the fundamental characteristics of the Complex-BP algorithm and found that this new algorithm had some inherent properties. The error back propagation has a structure which is concerned with two-dimensional motion. A unit of learning is complex-valued signals flowing through the neural network. Compared with the updating rule of the Real-BP, the Complex-BP updating rule is such that it reduces the probability for a standstill in learning. As a result, the average convergence speed is superior to that of the Real-BP (whereas the generalization performance remains unchanged). In addition, the number of learnable parameters needed is almost half of the Real-BP, where a complex-valued parameter $z = x + iy$ was counted as two because it consisted of a real part x and an imaginary part y. Thus it seems that the Complex-BP

algorithm is well suited for learning complex-valued patterns. We must emphasize the point that the Complex-BP can transform geometric figures in a way that the Real-BP cannot. Numerical experiments suggest that the behavior of a Complex-BP network which has learned the transformation of geometric figures is related to the Identity Theorem in complex analysis. Mathematical analysis indicates that a Complex-BP network which has learned a transformation, has the ability to generalize that transformation with an error which is represented by the sine of the difference between the argument of the test point and that of the training point. This mathematical result agrees qualitatively with simulation results. Futhermore, we experimentally confirmed that the 1-n-1 type Complex-BP network which had the ability to transform geometric figures, could also solve a continuous mapping task on the usual generalization ability very well which the 2-m-2 type Real-BP network could. We believe that the structure of the learning patterns caused the successful experimental result that the 1-n-1 type Complex-BP network could solve the continuous mapping task.

It is interesting that such characteristics appear only by extending neural networks to complex numbers. We feel that the work presented in this paper is probably only scratching the surface of what might be possible by extending the back-propagation algorithm to the complex number domain. We expect that this new algorithm will demonstrate its real ability in the areas dealing with complex numbers.

APPENDIX

Convergence Theorem (APCM)

We describe below the learning convergence theorem in APCM (Amari, 1967) whose parameters and functions take on real values: $\boldsymbol{x} \in \boldsymbol{R}^n$, $\boldsymbol{y} \in \boldsymbol{R}^m$, $\boldsymbol{w} \in \boldsymbol{R}^p$, $\boldsymbol{z}(\boldsymbol{w}, \boldsymbol{x}) : \boldsymbol{R}^p \times \boldsymbol{R}^n \to \boldsymbol{R}^m$, $r(\boldsymbol{y}', \boldsymbol{y}) : \boldsymbol{R}^m \times \boldsymbol{R}^m \to \boldsymbol{R}^+$ and $R(\boldsymbol{w}) : \boldsymbol{R}^p \to \boldsymbol{R}^1$.

CONVERGENCE THEOREM (APCM). *Let $\boldsymbol{A}$ be a positive definite matrix. Then, by using the update rule*

$$\Delta \boldsymbol{w}_n = -\varepsilon \boldsymbol{A} \nabla r(\boldsymbol{z}(\boldsymbol{w}_n, \boldsymbol{x}_n), \boldsymbol{y}_n), \qquad n = 0, 1, \cdots, \tag{$A-1$}$$

the (real-valued) parameter $\boldsymbol{w}$ approaches the optimum as near as desired by choosing a sufficiently small learning constant $\varepsilon > 0$ (∇ is a gradient operator with respect to $\boldsymbol{w}$).

Proof of Theorem 2

We will define the following constants:

$$K = \frac{1}{(1+\sqrt{2})c^0 + 2r^0}, \quad G = \frac{kac^0 v^0}{2(kav^0 + s^0)}, \quad A = \frac{c^0 s^0}{2(kav^0 + s^0)}, \quad B = \frac{c^0}{\sqrt{2}}, \quad C = r^0,$$

$$H_R = A\cos(t^0 + d^0) + B\cos\left(d^0 + \frac{\pi}{4}\right) + C\cos(l^0),$$

$$H_I = A\sin(t^0 + d^0) + B\sin\left(d^0 + \frac{\pi}{4}\right) + C\sin(l^0),$$

$$M = 2K\sqrt{H_R^2 + H_I^2},$$

$$M' = 2\sqrt{K^2\left[A^2 + B^2 + C^2 + 2AB\cos\left(t^0 - \frac{\pi}{4}\right) + 2AC\cos(t^0 + d^0 - l^0) + 2BC\cos\left(d^0 + \frac{\pi}{4} - l^0\right)\right]}$$
$$+ \frac{\tau^2}{4} - \tau K\left[A\cos(t^0 + d^0 - \omega) + B\cos\left(d^0 + \frac{\pi}{4} - \omega\right) + C\cos(l^0 - \omega)\right]. \tag{$A-2$}$$

In order to prove Theorem 2, a technical result is needed:

LEMMA. *For any* $1 \leq k \leq p$, *the following approximate equations hold:*

$$K\Big[G\cos(x + w^0 + d^0) + H_R\Big] + \frac{1}{2} = \frac{1}{2}ka\cos(x + \alpha) + \frac{1}{2}, \qquad (A-3)$$

$$K\Big[G\sin(x + w^0 + d^0) + H_I\Big] + \frac{1}{2} = \frac{1}{2}ka\sin(x + \alpha) + \frac{1}{2}. \qquad (A-4)$$

Proof. For any $1 \leq k \leq p$, by computing the output value of the Complex-BP network for the input training point $ka\exp[ix]$, we find that the real part of the output value is equal to the left side of eqn (A-3), and the imaginary part the left side of eqn (A-4). In the above computations, the sigmoid function in the output function (eqn (9)) of each neuron was approximated by the following piecewise linear functions:

$$g(x) = \begin{cases} \frac{1}{2(kav^0 + s^0)}x + \frac{1}{2} & (-(kav^0 + s^0) \leq x \leq kav^0 + s^0) \\ 1 & (kav^0 + s^0 < x) \\ 0 & (x < -(kav^0 + s^0)) \end{cases} \qquad (A-5)$$

for the hidden neuron, and

$$h(x) = \begin{cases} \frac{1}{(1+\sqrt{2})c^0 + 2r^0}x + \frac{1}{2} & \Big(-(\frac{1+\sqrt{2}}{2}c^0 + r^0) \leq x \leq \frac{1+\sqrt{2}}{2}c^0 + r^0\Big) \\ 1 & \Big(\frac{1+\sqrt{2}}{2}c^0 + r^0 < x\Big) \\ 0 & \Big(x < -(\frac{1+\sqrt{2}}{2}c^0 + r^0)\Big) \end{cases} \qquad (A-6)$$

for the output neuron. On the other hand, the real part and the imaginary part of the output value of the complex-valued neural network for the input training point should be equal to the real part and the imaginary part of the output training point $(1/2)ka\exp[i(x+\alpha)] + (1/\sqrt{2})\exp[i(\pi/4)]$, respectively. This concludes the proof of Lemma. ■

Proof of Theorem 2. Theorem 2 will be proved according to the following policy: using eqns (A-3) and (A-4) in the previous Lemma, we compute the output value of the Complex-BP network for the test point $ka\exp[i(x + \phi)]$, and transform it into [

The point generated by the counterclockwise rotation over α radians of the test point $ka\exp[i(x+\phi+\alpha)]$] + [The error].

First, we compute the real part of the output value when the test point $ka\exp[i(x+\phi)]$ is fed into the Complex-BP network. Using the equation

$$\cos\theta - \lambda\sin\theta = \sqrt{1+\lambda^2}\cos(\theta+\phi) \qquad (A-7)$$

for any θ, where $\lambda = \tan\phi$, and by computing [eqn (A-3)] $-\ \lambda\cdot$[eqn (A-4)], we get

$$K\left[G\cos(x+\phi+w^0+d^0)+H_R\right]+\frac{1}{2} = \left[\frac{1}{2}ka\cos(x+\phi+\alpha)+\frac{1}{2}\right]+E^R_{re}(\phi), \quad (A-8)$$

where

$$E^R_{re}(\phi) = 2K\sin\left(\frac{\phi}{2}\right)\cdot\left[A\sin\left(t^0+d^0+\frac{\phi}{2}\right)+B\sin\left(d^0+\frac{\pi}{4}+\frac{\phi}{2}\right)+C\sin\left(l^0+\frac{\phi}{2}\right)\right]. \qquad (A-9)$$

Note that the left side of eqn (A-8) refers to the real part of the output value of the Complex-BP network for the test point, and the first term of the right side of eqn (A-8) to the real part of *the point generated by the counterclockwise rotation over α radians of the test point* $ka\exp[i(x+\phi+\alpha)]$. Finally, $E^R_{re}(\phi)$ refers to the real part of a complex number which denotes the error.

Similarly, using the equation

$$\lambda\cos\theta + \sin\theta = \sqrt{1+\lambda^2}\sin(\theta+\phi) \qquad (A-10)$$

for any θ, and by computing $\lambda\cdot$[eqn (A-3)] + [eqn (A-4)], we get

$$K\left[G\sin(x+\phi+w^0+d^0)+H_I\right]+\frac{1}{2} = \left[\frac{1}{2}ka\sin(x+\phi+\alpha)+\frac{1}{2}\right]+E^R_{im}(\phi), \quad (A-11)$$

where

$$E^R_{im}(\phi) = -2K\sin\left(\frac{\phi}{2}\right)\cdot\left[A\cos\left(t^0+d^0+\frac{\phi}{2}\right)+B\cos\left(d^0+\frac{\pi}{4}+\frac{\phi}{2}\right)+C\cos\left(l^0+\frac{\phi}{2}\right)\right]. \qquad (A-12)$$

Note that the left side of eqn (A-11) refers to the imaginary part of the output value of the Complex-BP network for the test point, and the first term of the right side of eqn (A-11) to the imaginary part of *the point generated by the counterclockwise rotation over*

α radians of the test point $ka\exp[i(x+\phi+\alpha)]$. Finally, $E^R_{im}(\phi)$ refers to the imaginary part of a complex number which denotes the error.

Hence, it follows from eqns (A-8) and (A-11) that the output value of the Complex-BP network for the test point can be expressed as

$$\left[\frac{1}{2}ka\exp[i(x+\phi+\alpha)] + \frac{1}{\sqrt{2}}\exp\left[i\frac{\pi}{4}\right]\right] + E^R(\phi), \qquad (A-13)$$

where

$$E^R(\phi) \stackrel{\text{def}}{=} E^R_{re}(\phi) + iE^R_{im}(\phi). \qquad (A-14)$$

That is, the Complex-BP network rotates the test point $ka\exp[i(x+\phi)]$ counterclockwise over α radians with the error $E^R(\phi)$. And eqn (39) follows from eqns (A-9) and (A-12).

■

REFERENCES

Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Transactions on electronic computers*, **EC-16** (3), 299-307.

Derrick, W. R. (1984). *Complex analysis and applications*. Wadsworth, Inc.

Kim, M. S., & Guest, C. C. (1990). Modification of backpropagation networks for complex-valued signal processing in frequency domain. *Proceedings of the IJCNN*, San Diego, June, **3**, 27-31.

Miyauchi, M., & Seki, M. (1992a). 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., 1247-1251.

Miyauchi, M., Seki, M., Watanabe, A., & Miyauchi, A. (1992b). Interpretation of optical flow through neural network learning. *Proceedings of IAPR workshop on machine vision applications*, Tokyo, Dec., 523-528.

Miyauchi, M., Seki, M., Watanabe, A., & Miyauchi, A. (1993). Interpretation of optical flow through complex neural network. *Proceedings of international workshop on artificial neural networks*, Barcelona, Lecture Notes in Computer Science, **686**, Springer-Verlag, 645-650.

Nitta, T., & Furuya, T. (1991). A complex back-propagation learning. *Transactions of information processing society of japan*, **32** (10), 1319-1329 (in Japanese).

Nitta, T., & Furuya, T. (1993a). Characteristics of learning in the complex back-propagation learning algorithm. *Transactions of information processing society of japan*, **34** (1), 29-38 (in Japanese).

Nitta, T. (1993b). Behavior of complex BP network which has learned rotation. *Transactions of information processing society of japan*, **34** (1), 39-51 (in Japanese).

Nitta, T. (1993c). A complex numbered version of the back-propagation algorithm. *Proceedings of the WCNN*, Portland, July, **3**, 576-579.

Nitta, T. (1994a). An analysis on the learning rule in the complex back-propagation algorithm. *Proceedings of the WCNN*, SanDiego, June, **3**, 702-707.

Nitta, T. (1994b). Behavior of the complex numbered back-propagation network which

has learned similar transformation. *Proceedings of the WCNN*, SanDiego, June, **4**, 765-770.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructures of cognition* (**1**, 318-362). Cambridge, MA:MIT Press.

Tsutsumi, K. (1989). A multi-layered neural network composed of backprop. and Hopfield nets and internal space representation. *Proceedings of the IJCNN*, Washington, D.C., June, **1**, 365-371.

Watanabe, A., Yazawa, N., Miyauchi, A., & Miyauchi, M. (1994). A method to interpret 3D motions using neural networks. *IEICE Transactions on fundamentals of electronics, communications and computer sciences*, **E77-A** (8), 1363-1370.

Widrow, B., McCool, J., & Ball, M. (1975). The complex LMS algorithm. *Proceedings of the IEEE*, **63** (4), 719-720.

Table 1

Learning Patterns [Experiment 1]

Input Pattern	Output Pattern
0	0
i	1
1	$1+i$
$1+i$	i

$i = \sqrt{-1}$

Table 2

Computational Complexity of the Complex-BP and the Real-BP [Experiment 1]

Network	Time Complexity			Space Complexity		
	$\times$ and $\div$	$+$ and $-$	Sum	Weights	Thresholds	Sum
Complex-BP 1-3-1	78	52	130	12	8	20
Real-BP 2-7-2	90	46	136	28	9	37

Time complexity means the sum of the four operations performed per learning cycle. Space complexity means the sum of the parameters (weights and thresholds), where a complex-valued parameter $z = x + iy$ (where $i = \sqrt{-1}$) is counted as two because it consists of a real part x and an imaginary part y.

Table 3

Learning Patterns [Experiment 2]

Input Pattern		Output Pattern
Complex Number 1	Complex Number 2	Complex Number 3
0	0	1
0	i	i
i	i	$1+i$
i	1	i
1	1	$1+i$
i	0	0
$1+i$	$1+i$	1
$1+i$	i	i

$$i = \sqrt{-1}$$

Table 4

Computational Complexity of the Complex-BP and the Real-BP [Experiment 2]

Network	Time Complexity			Space Complexity		
	$\times$ and $\div$	$+$ and $-$	Sum	Weights	Thresholds	Sum
Complex-BP 2-4-1	134	92	226	24	10	34
Real-BP 4-9-2	150	76	226	54	11	65

Time complexity means the sum of the four operations performed per learning cycle. Space complexity means the sum of the parameters (weights and thresholds), where a complex-valued parameter $z = x + iy$ (where $i = \sqrt{-1}$) is counted as two because it consists of a real part x and an imaginary part y.

Table 5

Rate of Convergence [Experiment 2]

Network	Learning Rate					
	0.1	0.2	0.3	0.4	0.5	0.6
Complex-BP 2-4-1	100	96	88	92	90	98
Real-BP 4-9-2	0	22	64	78	90	100

Table 6

Distances between the Input Training Points and the Test Points

Test Point	Input Training Point							
	1	2	3	4	5	6	7	8
1	1	1	1	2	2	2	2	3
2	2	1	2	2	3	2	1	2
3	1	2	1	1	1	2	3	3
4	3	2	1	1	3	2	1	1
5	2	2	1	2	1	1	1	2
6	2	3	2	2	1	2	3	2
7	3	3	1	2	2	2	2	1
8	3	2	2	1	2	1	2	1

The distance-measure $\|\boldsymbol{x} - \boldsymbol{y}\|^2$ is used for an input training point $\boldsymbol{x} \in \boldsymbol{C}^2$ and a test point $\boldsymbol{y} \in \boldsymbol{C}^2$, which is defined in eqn (24).

Table 7

Comparison of the Theoretical Values and the Experimental Values of M(and M')

Type of Transformation	Theoretical Value	Experimenal Value
Rotation	0.19	0.35
Similarity Transformation	0.02	0.03
Parallel Displacement	0.49	0.27

Learning was stopped when the error between the desired output pattern and the actual output pattern was equal to 0.06 in the case of rotation, 0.02 in the cases of similarity transformation and parallel displacement.