Natural Gradient Descent for Training Stochastic Complex-Valued Neural Networks

Tohru Nitta
National Institute of Advanced Industrial Science and Technology (AIST),
AIST Tsukuba Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan
Email: tohru-nitta@aist.go.jp

***Abstract*—In this paper, the natural gradient descent method for the multilayer stochastic complex-valued neural networks is considered, and the natural gradient is given for a single stochastic complex-valued neuron as an example. Since the space of the learnable parameters of stochastic complex-valued neural networks is not the Euclidean space but a curved manifold, the complex-valued natural gradient method is expected to exhibit excellent learning performance.**

***Keywords*—Neural network; Complex number; Learning; Singular point**

I. Introduction

Complex-valued neural networks whose parameters (weights and threshold values) are all complex numbers, are useful in fields dealing with complex numbers or two-dimensional vectors such as telecommunications, speech recognition and image processing with Fourier transformation. Indeed, we can find some applications of complex-valued neural networks to various fields in the literature [6], [9].

The multilayer complex-valued neural network is usually trained using the gradient descent learning method [5], [10], [11], [12], as in the case of the multilayer real-valued neural network. The space of the learnable parameters of *stochastic* complex-valued neural networks is, however, not the Euclidean space but a curved manifold. For stochastic complex-valued neural networks, the ordinary gradient does not give the steepest direction of a target function, and the steepest direction is given by the natural gradient [2], [3]. It has been shown in [4] that the natural gradient method could avoid singular points of the real-valued parameter space which is a cause of standstill in learning, and the natural gradient method could improve the learning performance of the real-valued neural networks as a result. Similarly, there exist many singular points in the complex-valued neural networks [7]. Thus, the natural gradient method would be useful for the complex-valued neural networks, too. In this paper, we extend the natural gradient descent method for the multilayer stochastic real-valued neural networks to the complex domain, and give the natural gradient for a single stochastic complex-valued neuron as an example.

Section II describes the complex-valued neural network. Section III is devoted to the explanation of the natural gradient method, and Section IV presents the natural gradient method in complex-valued neural networks, which is followed by our conclusion in Section V.

II. Complex-Valued Neural Network Model

This section describes the complex-valued neural network model used in this paper. First, we will consider the following complex-valued neuron. The input signals, weights, thresholds and output signals are all complex numbers. The net input U_n to a complex-valued neuron n is defined as: $U_n = \sum_m W_{nm} X_m + V_n$, where W_{nm} is the complex-valued weight connecting the complex-valued neurons n and m, X_m is the complex-valued input signal from the complex-valued neuron m, and V_n is the complex-valued threshold value of the complex-valued neuron n. To obtain the complex-valued output signal, convert the net input U_n into its real and imaginary parts as follows: $U_n = x + iy = z$, where i denotes $\sqrt{-1}$. The complex-valued output signal is defined to be

$$f_C(z) = \varphi(x) + i\varphi(y), \tag{1}$$

where $\varphi : \boldsymbol{R} \to \boldsymbol{R}$, ($\boldsymbol{R}$ denotes the set of real numbers). Eq. (1) is often called a *split-type* complex-valued activation function. Note that the activation function f_C is not a regular complex-valued function because the Cauchy-Riemann equations do not hold.

The complex-valued neural network used in this paper consists of such complex-valued neurons described above.

Note that various types of activation functions other than Eq. (1) can be considered naturally (for examples, the non-split-type (fully) one [10]).

III. Natural Gradient Method

This section briefly describes the natural gradient proposed in [2], [3]. Let $S = \{\boldsymbol{w} \in \boldsymbol{R}^N\}$ be a Riemannian space with the Riemannian metric tensor $\boldsymbol{G}(\boldsymbol{w}) = (g_{ij}(\boldsymbol{w}))$ on which a function $L(\boldsymbol{w})$ is defined. If

$$g_{ij}(\boldsymbol{w}) = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}, \tag{2}$$

that is, $\boldsymbol{G}(\boldsymbol{w})$ is the unit matrix, then S is an Euclidean space. Amari proved the following theorem [3].

Theorem 1: *The steepest descent direction of $L(\boldsymbol{w})$ in a Riemannian space is given by*

$$-\tilde{\nabla} L(\boldsymbol{w}) = -\boldsymbol{G}^{-1}(\boldsymbol{w}) \nabla L(\boldsymbol{w}) \tag{3}$$

where $\boldsymbol{G}^{-1}(\boldsymbol{w}) = (g^{ij}(\boldsymbol{w}))$ is the inverse of the metric $\boldsymbol{G}(\boldsymbol{w}) = (g_{ij}(\boldsymbol{w}))$ and ∇L is the conventional gradient,

$$\nabla L(\boldsymbol{w}) = \left(\frac{\partial}{\partial w_1} L(\boldsymbol{w}), \cdots, \frac{\partial}{\partial w_N} L(\boldsymbol{w}) \right)^T, \tag{4}$$

where the superscript T denotes the transposition. □

$\tilde{\nabla} L(\boldsymbol{w}) = \boldsymbol{G}^{-1}\nabla L(\boldsymbol{w})$ is called the *natural gradient* of L in the Riemannian space. The natural gradient descent algorithm is given by

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \varepsilon_t \tilde{\nabla} L(\boldsymbol{w}_t), \tag{5}$$

where ε_t is the learning rate.

Amari derived the natural gradients explicitly in the case of the space of real-valued perceptrons for neural learning, the space of matrices for blind source separation, and the space of linear dynamical systems for blind multichannel source deconvolution [3].

IV. Natural Gradient in Complex-Valued Neural Networks

In this section, the natural gradient is applied to the complex-valued neural networks and the natural gradient descent algorithm is explicitly derived for a single complex-valued neuron.

A. Natural Gradient Learning in Complex-Valued Neural Networks

Let us consider a stochastic complex-valued multilayer feedforward neural network with N input neurons, one output neuron, and a learnable complex-valued vector parameter $\boldsymbol{w} = (w_1, \cdots, w_N)^T \in \boldsymbol{C}^N$ which consists of all the weights and thresholds ($\boldsymbol{C}$ denotes the set of complex numbers). Assume that the complex-valued input signal $\boldsymbol{z} = (z_1, \cdots, z_N)^T \in \boldsymbol{C}^N$ is subject to an unknown probability distribution $q(\boldsymbol{z})$, and the complex-valued output signal $y \in \boldsymbol{C}$ is given by

$$y = g_C(\boldsymbol{z}, \boldsymbol{w}) + n, \tag{6}$$

where g_C is a complex function, and $n = n_R + in_I$ is a complex-valued random variable subject to a complex normal distribution (or bivariate normal distribution) $N(\mu, \boldsymbol{\Sigma})$. The model specifies the probability density of the input-output pair as

$$p(\boldsymbol{z}, y; \boldsymbol{w}) = q(\boldsymbol{z}) \cdot p(y|\boldsymbol{z}; \boldsymbol{w}). \tag{7}$$

Define a loss function $l(\boldsymbol{z}, y; \boldsymbol{w})$ when input signal $\boldsymbol{z}$ is processed by the stochastic complex-valued neural network having parameter $\boldsymbol{w}$ as:

$$\begin{aligned} l(\boldsymbol{z}, y; \boldsymbol{w}) &\stackrel{\text{def}}{=} -\log p(\boldsymbol{z}, y; \boldsymbol{w}) \\ &= -\log q(\boldsymbol{z}) - \log p(y|\boldsymbol{z}; \boldsymbol{w}). \end{aligned} \tag{8}$$

Given the training set $\{(\boldsymbol{z}_t, y_t), t = 1, \cdots, T\}$, minimizing the loss function (Eq. (8)) is equivalent to maximizing the probability that the stochastic complex-valued neural network outputs the training output signal y_t.

The space of all the probability distributions which the above stochastic complex-valued neural network realizes, can be regarded as a $2N$-dimensional Riemannian space because the complex-valued parameter consists of the two real-valued parameters: the real-part and the imaginary part. Thus, the information geometry [1] can be applied to the complex-vauled case, too.

The natural gradient descent algorithm for the complex-valued neural network is given by

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t - \varepsilon_t \tilde{\nabla} l(\boldsymbol{z}_t, y_t; \boldsymbol{v}_t), \tag{9}$$

where $\{(\boldsymbol{z}_t, y_t) \in \boldsymbol{C}^N \times \boldsymbol{C},\ t = 1, 2, \cdots\}$ is the sequence of the complex-valued training signals, and

$$\begin{aligned} \boldsymbol{v} &= (v_1, \cdots, v_{2N})^T \\ &= (Re[w_1], \cdots, Re[w_N], Im[w_1], \cdots, Im[w_N])^T, \end{aligned} \tag{10}$$

$$\tilde{\nabla} l(\boldsymbol{z}, y, \boldsymbol{v}) \stackrel{\text{def}}{=} \boldsymbol{G}^{-1}(\boldsymbol{v}) \cdot \nabla l(\boldsymbol{z}, y, \boldsymbol{v}). \tag{11}$$

Eq. (11) is the natural gradient of $l(\boldsymbol{z}, y, \boldsymbol{v})$, and the usual gradient $\nabla l(\boldsymbol{z}, y, \boldsymbol{v})$ is given by

$$\begin{aligned} \nabla l(\boldsymbol{z}, y, \boldsymbol{v}) &\stackrel{\text{def}}{=} \left(\frac{\partial l(\boldsymbol{z}, y, \boldsymbol{v})}{\partial v_1}, \cdots, \frac{\partial l(\boldsymbol{z}, y, \boldsymbol{v})}{\partial v_{2N}} \right)^T \\ &= \left(\frac{\partial l(\boldsymbol{z}, y, \boldsymbol{w})}{\partial Re[w_1]}, \cdots, \frac{\partial l(\boldsymbol{z}, y, \boldsymbol{w})}{\partial Re[w_N]}, \right. \\ &\quad \left. \frac{\partial l(\boldsymbol{z}, y, \boldsymbol{w})}{\partial Im[w_1]}, \cdots, \frac{\partial l(\boldsymbol{z}, y, \boldsymbol{w})}{\partial Im[w_N]} \right)^T. \end{aligned} \tag{12}$$

The Riemannian metric tensor $\boldsymbol{G}(\boldsymbol{v})$ is the Fisher information matrix [3], and is given by

$$\boldsymbol{G}(\boldsymbol{v}) = (g_{ij}(\boldsymbol{v})), \tag{13}$$

$$g_{ij}(\boldsymbol{v}) = E\left[\frac{\partial \log p(\boldsymbol{z}, y; \boldsymbol{v})}{\partial v_i} \cdot \frac{\partial \log p(\boldsymbol{z}, y; \boldsymbol{v})}{\partial v_j} \right]. \tag{14}$$

B. Natural Gradient Learning in a Single Complex-Valued Neuron

In this section, the natural gradient descent learning algorithm for a single complex-valued neuron is given.

Consider a stochastic complex-valued neuron with N-inputs, weights $w_k = u_k + iv_k \in \boldsymbol{C}$ $(1 \le k \le N)$, and a threshold value $\gamma = c + id \in \boldsymbol{C}$. Then, for N input signals $z_k = x_k + iy_k \in \boldsymbol{C}$ $(1 \le k \le N)$, the stochastic complex-valued neuron generates

$$\begin{aligned} y &= f_C\left(\sum_{k=1}^{N} w_k z_k + \gamma \right) + n \\ &= X + iY \end{aligned} \tag{15}$$

where $f_C : \boldsymbol{C} \to \boldsymbol{C}$ is a so-called *split-type* complex-valued activation function which is defined to be

$$f_C(a + ib) = \varphi(a) + i\varphi(b) \tag{16}$$

for any $a + ib \in \boldsymbol{C}$, and $\varphi : \boldsymbol{R} \to \boldsymbol{R}$ is suitably chosen, for examle, the sigmoid function

$$\varphi(s) = \frac{1}{1 + e^{-s}} \tag{17}$$

was used in [11], and the scaled error function

$$\varphi(s) = \frac{2}{\sqrt{\pi}} \int_0^{\frac{s}{\sqrt{2}}} e^{-t^2} dt \tag{18}$$

was used in [13]. $n = n_R + in_I$ is a complex-valued random variable subject to the complex normal distribution (or bivariate normal distribution) $N(\mu, \mathbf{\Sigma})$ where

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{19}$$

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma & 0 \\ 0 & \sigma \end{bmatrix}. \tag{20}$$

We assume that the input signal $\boldsymbol{z} = \boldsymbol{x} + i\boldsymbol{y}$ where $\boldsymbol{x} \stackrel{\text{def}}{=} (x_1, \cdots, x_N)^T, \boldsymbol{y} \stackrel{\text{def}}{=} (y_1, \cdots, y_N)^T$ is subject to the multivariate complex normal distribution (or $2N$-dimensional normal distribution) $N(\boldsymbol{0}, \boldsymbol{I})$ where the variance covariance matrix $\boldsymbol{I}$ is the unit matrix; denote its joint probability density function by $q(\boldsymbol{z})$. The loss function $l(\boldsymbol{z}, y; \theta)$ is defined as

$$\begin{aligned} l(\boldsymbol{z}, y; \theta) &\stackrel{\text{def}}{=} -\log p(\boldsymbol{z}, y; \theta) \\ &= -\log q(\boldsymbol{z}) - \log p(y|\boldsymbol{z}; \theta), \end{aligned} \tag{21}$$

where $\theta \stackrel{\text{def}}{=} (\boldsymbol{u}^T, \boldsymbol{v}^T, c,\ d)^T,\ \boldsymbol{u} = (u_1, \cdots, u_N)^T,\ \boldsymbol{v} = (v_1, \cdots, v_N)^T$.

Given the sequence of the complex-valued training signals $\{(\boldsymbol{z}_t, y_t) \in \boldsymbol{C}^N \times \boldsymbol{C},\ t = 1, 2, \cdots\}$, the the natural gradient descent algorithm for the stochastic complex-valued neuron is given by

$$\theta_{t+1} = \theta_t - \varepsilon_t \cdot \boldsymbol{G}^{-1}(\theta_t) \cdot \nabla l(\boldsymbol{z}_t, y_t; \theta_t). \tag{22}$$

We shall calculate the Fisher information matirx $\boldsymbol{G}(\theta) = (g_{ij}(\theta))$. For any $1 \le i, j \le 2N + 2$,

$$\begin{aligned} g_{ij}(\theta) &= E\left[\frac{\partial \log p(\boldsymbol{z}, y; \theta)}{\partial \theta_i} \cdot \frac{\partial \log p(\boldsymbol{z}, y; \theta)}{\partial \theta_j}\right] \\ &= E\left[\frac{\partial \log p(y|\boldsymbol{z}; \theta)}{\partial \theta_i} \cdot \frac{\partial \log p(y|\boldsymbol{z}; \theta)}{\partial \theta_j}\right]. \end{aligned} \tag{23}$$

(from Eq. (21))

Here, since n_R is independent of n_I,

$$\begin{aligned} \log p(y|\boldsymbol{z}; \theta) &= \log p((\varphi(S) + n_R) + i(\varphi(T) + n_I)|\boldsymbol{z}; \theta) \\ &= \log p(\varphi(S) + n_R|\boldsymbol{z}; \theta) \\ &\quad + \log p(\varphi(T) + n_I|\boldsymbol{z}; \theta) \\ &= \log p(X|\boldsymbol{z}; \theta) + \log p(Y|\boldsymbol{z}; \theta), \end{aligned} \tag{24}$$

where $S = Re\left[\sum_{k=1}^N w_k z_k + \gamma\right]$, $T = Im\left[\sum_{k=1}^N w_k z_k + \gamma\right]$. Thus, for any $1 \le i, j \le 2N + 2$,

$$\begin{aligned} g_{ij}(\theta) &= E\left[\frac{\partial \log p(X|\boldsymbol{z}; \theta)}{\partial \theta_i} \cdot \frac{\partial \log p(X|\boldsymbol{z}; \theta)}{\partial \theta_j}\right] \\ &\quad + 2E\left[\frac{\partial \log p(X|\boldsymbol{z}; \theta)}{\partial \theta_i} \cdot \frac{\partial \log p(Y|\boldsymbol{z}; \theta)}{\partial \theta_j}\right] \\ &\quad + E\left[\frac{\partial \log p(Y|\boldsymbol{z}; \theta)}{\partial \theta_i} \cdot \frac{\partial \log p(Y|\boldsymbol{z}; \theta)}{\partial \theta_j}\right]. \end{aligned} \tag{25}$$

By simple calculations, we obtain

$$\begin{aligned} g_{ij}(\theta) &= \frac{1}{\sigma^2}\left\{E[(\varphi'(S))^2 x_i x_j] \right. \\ &\qquad \left. + E[(\varphi'(T))^2 y_i y_j]\right\} \\ &\qquad (1 \le i, j \le N), \end{aligned} \tag{26}$$

$$\begin{aligned} g_{ij}(\theta) &= \frac{1}{\sigma^2}\left\{E[(\varphi'(S))^2 (-y_{i-N}) x_j] \right. \\ &\qquad \left. + E[(\varphi'(T))^2 x_{i-N} y_j]\right\} \\ &= g_{ji}(\theta) \\ &\qquad (N + 1 \le i \le 2N,\ 1 \le j \le N), \end{aligned} \tag{27}$$

$$\begin{aligned} g_{ij}(\theta) &= \frac{1}{\sigma^2}\left\{E[(\varphi'(S))^2 y_{i-N} y_{j-N}] \right. \\ &\qquad \left. + E[(\varphi'(T))^2 x_{i-N} x_{j-N}]\right\} \\ &\qquad (N + 1 \le i, j \le 2N), \end{aligned} \tag{28}$$

$$\begin{aligned} g_{2N+1,j}(\theta) &= \frac{1}{\sigma^2} E[(\varphi'(S))^2 x_j] \\ &= g_{j,2N+1}(\theta)\ \ (1 \le j \le N), \end{aligned} \tag{29}$$

$$\begin{aligned} g_{2N+1,j}(\theta) &= \frac{1}{\sigma^2} E[(\varphi'(S))^2 (-y_{j-N})] \\ &\qquad (N + 1 \le j \le 2N), \end{aligned} \tag{30}$$

$$\begin{aligned} g_{2N+2,j}(\theta) &= \frac{1}{\sigma^2} E[(\varphi'(T))^2 y_j] \qquad (1 \le j \le N), \end{aligned} \tag{31}$$

$$\begin{aligned} g_{2N+2,j}(\theta) &= \frac{1}{\sigma^2} E[(\varphi'(T))^2 x_{j-N}] \\ &\qquad (N + 1 \le j \le 2N), \end{aligned} \tag{32}$$

$$g_{2N+1,2N+1}(\theta) = \frac{1}{\sigma^2} E[(\varphi'(S))^2], \tag{33}$$

$$\begin{aligned} g_{2N+2,2N+1}(\theta) &= 0 \\ &= g_{2N+1,2N+2}(\theta), \end{aligned} \tag{34}$$

$$g_{2N+2,2N+2}(\theta) = \frac{1}{\sigma^2} E[(\varphi'(T))^2]. \tag{35}$$

From Eqs. (26) – (35), we can rewrite $\boldsymbol{G}(\theta)$ as

$$\boldsymbol{G}(\theta) = \frac{1}{\sigma^2} \boldsymbol{A}(\theta), \tag{36}$$

where

$$\boldsymbol{A}(\theta) \stackrel{\text{def}}{=} \left(\begin{array}{c|c|c|c} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \boldsymbol{a}_{13} & \boldsymbol{a}_{14} \\ \hline \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \boldsymbol{a}_{23} & \boldsymbol{a}_{24} \\ \hline \boldsymbol{a}_{31} & \boldsymbol{a}_{32} & \multicolumn{2}{c}{\multirow{2}{*}{$\boldsymbol{A}_{44}$}} \\ \cline{1-2} \boldsymbol{a}_{41} & \boldsymbol{a}_{42} & \multicolumn{2}{c}{} \end{array}\right), \tag{37}$$

$\boldsymbol{A}_{11} = \sigma^2 \cdot$ Eq. (26), $\boldsymbol{A}_{21} = \boldsymbol{A}_{12}^T = \sigma^2 \cdot$ Eq. (27), $\boldsymbol{A}_{22} = \sigma^2 \cdot$ Eq. (28), $\boldsymbol{a}_{13} = \boldsymbol{a}_{31}^T = \sigma^2 \cdot$ Eq. (29), $\boldsymbol{a}_{32} = \boldsymbol{a}_{23}^T = \sigma^2 \cdot$ Eq. (30), $\boldsymbol{a}_{14} = \boldsymbol{a}_{41}^T = \sigma^2 \cdot$ Eq. (31), $\boldsymbol{a}_{42} = \boldsymbol{a}_{24}^T = \sigma^2 \cdot$ Eq. (32), and

$$\boldsymbol{A}_{44} = \begin{pmatrix} \sigma^2 \cdot \text{Eq.(33)} & \text{Eq.(34)} \\ \text{Eq.(34)} & \sigma^2 \cdot \text{Eq.(35)} \end{pmatrix}. \tag{38}$$

In what follows, each submatrix of $\boldsymbol{A}(\theta)$ (Eq. (37)) is calculated. Let $u = ||\boldsymbol{u}|| = \sqrt{u_1^2 + \cdots + u_N^2}$, $v = ||\boldsymbol{v}|| = \sqrt{v_1^2 + \cdots + v_N^2}$, $\boldsymbol{u}_1 = \boldsymbol{u}/u$, $\boldsymbol{v}_1 = \boldsymbol{v}/v$, and extend $\boldsymbol{u}_1$, $\boldsymbol{v}_1$ to orthonormal bases $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_N\}$, $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N\}$ for $\boldsymbol{R}^N$, respectively. Then, the real part $\boldsymbol{x} \in \boldsymbol{R}^N$ and the imaginary

part $\boldsymbol{y} \in \boldsymbol{R}^N$ of the input signal $\boldsymbol{z} = \boldsymbol{x} + i\boldsymbol{y} \in \boldsymbol{C}^N$ can be decomopsed as

$$\boldsymbol{x} = \sum_{i=1}^{N} \chi_i \boldsymbol{u}_i, \tag{39}$$

$$\boldsymbol{y} = \sum_{i=1}^{N} \psi_i \boldsymbol{v}_i. \tag{40}$$

Then, noticing that $\chi_i, \psi_i \sim N(0,1)$, we have

$$\begin{aligned} E[\chi_i \chi_j] &= E[\boldsymbol{u}_i^T \boldsymbol{x}\boldsymbol{x}^T \boldsymbol{u}_j] = \boldsymbol{u}_i^T E[\boldsymbol{x}\boldsymbol{x}^T]\boldsymbol{u}_j \\ &= \boldsymbol{u}_i^T \boldsymbol{u}_j = \delta_{ij}, \end{aligned} \tag{41}$$

$$\begin{aligned} \boldsymbol{x}\boldsymbol{x}^T &= \chi_1^2 \boldsymbol{u}_1 \boldsymbol{u}_1^T + \sum_{i=2}^{N} \chi_1 \chi_i (\boldsymbol{u}_1 \boldsymbol{u}_i^T + \boldsymbol{u}_i \boldsymbol{u}_1^T) \\ &\quad + \sum_{i,j=2}^{N} \chi_i \chi_j \boldsymbol{u}_i \boldsymbol{u}_j^T, \end{aligned} \tag{42}$$

$$\boldsymbol{u}^T \cdot \boldsymbol{x} = \boldsymbol{u}^T (\chi_1 \boldsymbol{u}_1 + \cdots + \chi_N \boldsymbol{u}_N) = \chi_1 u, \tag{43}$$

$$\boldsymbol{v}^T \cdot \boldsymbol{y} = \boldsymbol{v}^T (\psi_1 \boldsymbol{v}_1 + \cdots + \psi_N \boldsymbol{v}_N) = \psi_1 v. \tag{44}$$

From Eqs. (41) – (44), we find that the first term of Eq. (26) $(\boldsymbol{A}_{11})$ is given by

$$\begin{aligned} E[(\varphi'(S))^2 \boldsymbol{x}\boldsymbol{x}^T] &= E[(\varphi'(\boldsymbol{u}^T\boldsymbol{x} - \boldsymbol{v}^T\boldsymbol{y} + c))^2 \boldsymbol{x}\boldsymbol{x}^T] \\ &= E[(\varphi'(\chi_1 u - \psi_1 v + c))^2 \chi_1^2] \boldsymbol{u}_1 \boldsymbol{u}_1^T \\ &\quad + E[(\varphi'(\chi_1 u - \psi_1 v + c))^2] \sum_{i=2}^{N} \boldsymbol{u}_i \boldsymbol{u}_i^T. \end{aligned} \tag{45}$$

Next, in order to calculate the second term of Eq. (26) $(\boldsymbol{A}_{11})$, decompose the real part $\boldsymbol{x} \in \boldsymbol{R}^N$ and the imaginary part $\boldsymbol{y} \in \boldsymbol{R}^N$ of the input signal $\boldsymbol{z} = \boldsymbol{x} + i\boldsymbol{y} \in \boldsymbol{C}^N$ as

$$\boldsymbol{x} = \sum_{i=1}^{N} \chi_i' \boldsymbol{v}_i, \tag{46}$$

$$\boldsymbol{y} = \sum_{i=1}^{N} \psi_i' \boldsymbol{u}_i. \tag{47}$$

Then, we have

$$E[\psi_i' \psi_j'] = \delta_{ij}, \tag{48}$$

$$\begin{aligned} \boldsymbol{y}\boldsymbol{y}^T &= (\psi_1')^2 \boldsymbol{u}_1 \boldsymbol{u}_1^T + \sum_{i=2}^{N} \psi_1' \psi_i' (\boldsymbol{u}_1 \boldsymbol{u}_i^T + \boldsymbol{u}_i \boldsymbol{u}_1^T) \\ &\quad + \sum_{i,j=2}^{N} \psi_i' \psi_j' \boldsymbol{u}_i \boldsymbol{u}_j^T, \end{aligned} \tag{49}$$

$$\boldsymbol{v}^T \cdot \boldsymbol{x} = \boldsymbol{v}^T (\chi_1' \boldsymbol{v}_1 + \cdots + \chi_N' \boldsymbol{v}_N) = \chi_1' v, \tag{50}$$

$$\boldsymbol{u}^T \cdot \boldsymbol{y} = \boldsymbol{u}^T (\psi_1' \boldsymbol{u}_1 + \cdots + \psi_N' \boldsymbol{u}_N) = \psi_1' u. \tag{51}$$

From Eqs. (48) – (51), we find that the second term of Eq. (26) $(\boldsymbol{A}_{11})$ is given by

$$\begin{aligned} E[(\varphi'(T))^2 \boldsymbol{y}\boldsymbol{y}^T] &= E[(\varphi'(\boldsymbol{v}^T\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{y} + d))^2 \boldsymbol{y}\boldsymbol{y}^T] \\ &= E[(\varphi'(\chi_1' v + \psi_1' u + d))^2 (\psi_1')^2] \boldsymbol{u}_1 \boldsymbol{u}_1^T \\ &\quad + E[(\varphi'(\chi_1' v + \psi_1' u + d))^2] \sum_{i=2}^{N} \boldsymbol{u}_i \boldsymbol{u}_i^T. \end{aligned} \tag{52}$$

Thus, from Eqs. (45) and (52),

$$\begin{aligned} \boldsymbol{A}_{11} &= E[(\varphi'(S))^2 \boldsymbol{x}\boldsymbol{x}^T] + E[(\varphi'(T))^2 \boldsymbol{y}\boldsymbol{y}^T] \\ &= \big\{ E[(\varphi'(\chi_1 u - \psi_1 v + c))^2 \chi_1^2] \\ &\quad + E[(\varphi'(\chi_1' v + \psi_1' u + d))^2 \psi_1^2] \big\} \boldsymbol{u}_1 \boldsymbol{u}_1^T \\ &\quad + \big\{ E[(\varphi'(\chi_1 u - \psi_1 v + c))^2] \\ &\quad + E[(\varphi'(\chi_1' v + \psi_1' u + d))^2] \big\} \sum_{i=2}^{N} \boldsymbol{u}_i \boldsymbol{u}_i^T \\ &= d_0(u,v,c,d) \cdot I \\ &\quad + \{ d_2(u,v,c,d) - d_0(u,v,c,d) \} \cdot \frac{\boldsymbol{u}\boldsymbol{u}^T}{u^2}, \end{aligned} \tag{53}$$

where

$$\begin{aligned} d_0(u,v,c,d) &\stackrel{\text{def}}{=} E[\{\varphi'(\chi_1 u - \psi_1 v + c)\}^2] \\ &\quad + E[\{\varphi'(\chi_1' v + \psi_1' u + d)\}^2], \end{aligned} \tag{54}$$

$$\begin{aligned} d_2(u,v,c,d) &\stackrel{\text{def}}{=} E[\{\varphi'(\chi_1 u - \psi_1 v + c)\}^2 \chi_1^2] \\ &\quad + E[\{\varphi'(\chi_1' v + \psi_1' u + d)\}^2 (\psi_1')^2]. \end{aligned} \tag{55}$$

Similarly, we have

$$\begin{aligned} \boldsymbol{A}_{21} &= d_{11}(u,v,c,d) \cdot \frac{\boldsymbol{v}\boldsymbol{u}^T}{uv} \\ &= \boldsymbol{A}_{12}^T, \end{aligned} \tag{56}$$

$$\begin{aligned} \boldsymbol{A}_{22} &= d_0(u,v,c,d) \cdot I \\ &\quad + \{ d_2'(u,v,c,d) - d_0(u,v,c,d) \} \cdot \frac{\boldsymbol{v}\boldsymbol{v}^T}{v^2}, \end{aligned} \tag{57}$$

$$\begin{aligned} \boldsymbol{a}_{31} &= d_{1x}(u,v,c) \cdot \boldsymbol{u}_1^T \\ &= \boldsymbol{a}_{13}^T, \end{aligned} \tag{58}$$

$$\begin{aligned} \boldsymbol{a}_{32} &= -d_{1y}(u,v,c) \cdot \boldsymbol{v}_1^T \\ &= \boldsymbol{a}_{23}^T, \end{aligned} \tag{59}$$

$$\begin{aligned} \boldsymbol{a}_{41} &= d_{1y}'(u,v,d) \cdot \boldsymbol{u}_1^T \\ &= \boldsymbol{a}_{14}^T, \end{aligned} \tag{60}$$

$$\begin{aligned} \boldsymbol{a}_{42} &= d_{1x}'(u,v,d) \cdot \boldsymbol{v}_1^T \\ &= \boldsymbol{a}_{24}^T, \end{aligned} \tag{61}$$

$$\boldsymbol{A}_{44} = \begin{pmatrix} d_0(u,v,c) & 0 \\ 0 & d_0'(u,v,d) \end{pmatrix} \tag{62}$$

where

$$
\begin{aligned}
d_{11}(u,v,c,d) &\stackrel{\text{def}}{=} E[\{\varphi'(\chi_1'v+\psi_1'u+d)\}^2\chi_1'\psi_1'] \\
&- E[\{\varphi'(\chi_1u-\psi_1v+c)\}^2\chi_1\psi_1], \quad (63)\\
d_2'(u,v,c,d) &\stackrel{\text{def}}{=} E[\{\varphi'(\chi_1u-\psi_1v+c)\}^2\psi_1^2] \\
&+ E[\{\varphi'(\chi_1'v+\psi_1'u+d)\}^2(\chi_1')^2], \quad (64)\\
d_{1x}(u,v,c) &\stackrel{\text{def}}{=} E[\{\varphi'(\chi_1u-\psi_1v+c)\}^2\chi_1], \quad (65)\\
d_{1y}(u,v,c) &\stackrel{\text{def}}{=} E[\{\varphi'(\chi_1u-\psi_1v+c)\}^2\psi_1], \quad (66)\\
d_{1y}'(u,v,d) &\stackrel{\text{def}}{=} E[\{\varphi'(\chi_1'v+\psi_1'u+d)\}^2\psi_1], \quad (67)\\
d_{1x}'(u,v,d) &\stackrel{\text{def}}{=} E[\{\varphi'(\chi_1'v+\psi_1'u+d)\}^2\chi_1], \quad (68)\\
d_0(u,v,c) &\stackrel{\text{def}}{=} E[\{\varphi'(\chi_1u-\psi_1v+c)\}^2], \quad (69)\\
d_0'(u,v,d) &\stackrel{\text{def}}{=} E[\{\varphi'(\chi_1'v+\psi_1'u+d)\}^2]. \quad (70)
\end{aligned}
$$

We compute the inverse of $\boldsymbol{A}(\theta)$ (Eq. (37)) using the following formula used in [13].

Lemma 1:

$$
\begin{pmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{B}^{11} & \boldsymbol{B}^{12} \\ \boldsymbol{B}^{21} & \boldsymbol{B}^{22} \end{pmatrix}, \quad (71)
$$

provided

$$
|\boldsymbol{B}_{11}| \neq 0,\ |\boldsymbol{B}_{22}-\boldsymbol{B}_{21}\boldsymbol{B}_{11}^{-1}\boldsymbol{B}_{12}| \neq 0 \quad (72)
$$

where

$$
\begin{aligned}
\boldsymbol{B}^{11} &= \boldsymbol{B}_{11}^{-1}+\boldsymbol{B}_{11}^{-1}\boldsymbol{B}_{12}\boldsymbol{B}_{22,1}^{-1}\boldsymbol{B}_{21}\boldsymbol{B}_{11}^{-1}, \quad (73)\\
\boldsymbol{B}_{22,1} &= \boldsymbol{B}_{22}-\boldsymbol{B}_{21}\boldsymbol{B}_{11}^{-1}\boldsymbol{B}_{12}, \quad (74)\\
\boldsymbol{B}^{22} &= \boldsymbol{B}_{22,1}^{-1}, \quad (75)\\
\boldsymbol{B}^{12} &= (\boldsymbol{B}^{21})^T = -\boldsymbol{B}_{11}^{-1}\boldsymbol{B}_{12}\boldsymbol{B}_{22,1}^{-1}. \quad (76)
\end{aligned}
$$

□

By using Lemma 1, we have

$$
\boldsymbol{A}(\theta)^{-1} = \begin{pmatrix} \boldsymbol{A}^{11} & \boldsymbol{A}^{12} \\ \boldsymbol{A}^{21} & \boldsymbol{A}^{22} \end{pmatrix}, \quad (77)
$$

where

$$
\begin{aligned}
\boldsymbol{A}^{11} &= \begin{pmatrix} a^{11} & a^{12} \\ a^{21} & a^{22} \end{pmatrix}, \quad (78)\\
a^{11} &= \frac{\boldsymbol{I}}{d_0}+\left\{G+\left(\frac{1}{d_0}+G\right)(m_1l_1+m_2l_2)\right. \\
&\left.+H(m_1l_3+m_2l_4)\right\}\frac{\boldsymbol{u}\boldsymbol{u}^T}{u^2}, \quad (79)\\
a^{12} &= \left\{H+H(m_1l_1+m_2l_2)+\right. \\
&\left.\left(\frac{1}{d_0}+E\right)(m_1l_3+m_2l_4)\right\}\frac{\boldsymbol{u}\boldsymbol{v}^T}{uv}, \quad (80)\\
a^{21} &= \left\{H+\left(\frac{1}{d_0}+G\right)(m_3l_1+m_4l_2)+\right. \\
&\left.H(m_3l_3+m_4l_4)\right\}\frac{\boldsymbol{v}\boldsymbol{u}^T}{vu}, \quad (81)\\
a^{22} &= \frac{\boldsymbol{I}}{d_0}+\{E+H(m_3l_1+m_4l_2) \\
&\left.+\left(\frac{1}{d_0}+E\right)(m_3l_3+m_4l_4)\right\}\frac{\boldsymbol{v}\boldsymbol{v}^T}{v^2}, \quad (82)\\
\boldsymbol{A}^{12} &= \begin{pmatrix} l_1\boldsymbol{u}/u & l_2\boldsymbol{u}/u \\ l_3\boldsymbol{v}/v & l_4\boldsymbol{v}/v \end{pmatrix} \\
&= (\boldsymbol{A}^{21})^T, \quad (83)\\
\boldsymbol{A}^{22} &= \frac{1}{k_1k_2-k^2}\begin{pmatrix} k_2 & -k \\ -k & k_1 \end{pmatrix}, \quad (84)\\
E &= \frac{d_2}{d_2d_2'-d_{11}^2}-\frac{1}{d_0}, \quad (85)\\
F &= \frac{1}{d_2}-\frac{1}{d_0}, \quad (86)\\
G &= Ed_{11}^2\left(\frac{1+2F}{d_0}+F^2\right) \\
&+\frac{d_{11}^2}{d_0}\left(F^2+2F+\frac{1}{d_0^2}\right)+F, \quad (87)\\
H &= -d_{11}\left(\frac{1}{d_0}+E\right)\left(\frac{1}{d_0}+F\right), \quad (88)\\
k &= -d_{1x}d_{1y}'\left(\frac{1}{d_0}+G\right)-H(d_{1y}d_{1y}'-d_{1x}d_{1x}') \\
&+d_{1x}'d_{1y}\left(\frac{1}{d_0}+E\right), \quad (89)\\
k_1 &= d_0-d_{1x}^2\left(\frac{1}{d_0}+G\right) \\
&+2Hd_{1x}d_{1y}-d_{1y}^2\left(\frac{1}{d_0}+E\right), \quad (90)\\
k_2 &= d_0'-d_{1y}'^2\left(\frac{1}{d_0}+G\right) \\
&-2Hd_{1x}'d_{1y}'-d_{1x}'^2\left(\frac{1}{d_0}+E\right), \quad (91)\\
l_1 &= \frac{1}{k_1k_2-k^2}\left\{-k_2\left(\frac{d_{1x}}{d_0}+Gd_{1x}-Hd_{1y}\right)\right. \\
&\left.+k\left(\frac{d_{1y}'}{d_0}+Gd_{1y}'+Hd_{1x}'\right)\right\}, \quad (92)
\end{aligned}
$$

$$l_2 = \frac{1}{k_1k_2-k^2}\left\{k\left(\frac{d_{1x}}{d_0}+Gd_{1x}-Hd_{1y}\right) - k_1\left(\frac{d'_{1y}}{d_0}+Gd'_{1y}+Hd'_{1x}\right)\right\}, \quad (93)$$

$$l_3 = \frac{1}{k_1k_2-k^2}\left\{-k_2\left(Hd_{1x}-\frac{d_{1y}}{d_0}-Ed_{1y}\right) + k\left(Hd'_{1y}+\frac{d'_{1x}}{d_0}+Ed'_{1x}\right)\right\}, \quad (94)$$

$$l_4 = \frac{1}{k_1k_2-k^2}\left\{k\left(Hd_{1x}-\frac{d_{1y}}{d_0}-Ed_{1y}\right) - k_1\left(Hd'_{1y}+\frac{d'_{1x}}{d_0}+Ed'_{1x}\right)\right\}, \quad (95)$$

$$m_1 = \frac{1}{k_1k_2-k^2}\left\{k_2\left(l_1\left(\frac{1}{d_0}+G\right)+Hl_3\right) -k\left(l_2\left(\frac{1}{d_0}+G\right)+Hl_4\right)\right\}, \quad (96)$$

$$m_2 = \frac{1}{k_1k_2-k^2}\left\{-k\left(l_1\left(\frac{1}{d_0}+G\right)+Hl_3\right) +k_1\left(l_2\left(\frac{1}{d_0}+G\right)+Hl_4\right)\right\}, \quad (97)$$

$$m_3 = \frac{1}{k_1k_2-k^2}\left\{k_2\left(Hl_1+l_3\left(\frac{1}{d_0}+E\right)\right) -k\left(Hl_2+l_4\left(\frac{1}{d_0}+E\right)\right)\right\}, \quad (98)$$

$$m_4 = \frac{1}{k_1k_2-k^2}\left\{-k\left(Hl_1+l_3\left(\frac{1}{d_0}+E\right)\right) +k_1\left(Hl_2+l_4\left(\frac{1}{d_0}+E\right)\right)\right\}. \quad (99)$$

V. Concluding Remark

We have developed the natural gradient descent method for the multilayer stochastic complex-valued neural networks, and derived the natural gradient for a single stochastic complex-valued neuron. Since we assume that the variance σ_1^2 for the real part of the complex-valued additive noise is equal to that σ_2^2 for the imaginary, the Riemannian metric tensor is given by $\boldsymbol{G}(\theta) = (1/\sigma^2)\boldsymbol{A}(\theta)$ where $\boldsymbol{A}(\theta)$ does not depend on $\sigma^2 = \sigma_1^2 = \sigma_2^2$. The situation is, however, not the same if $\sigma_1 \neq \sigma_2$. And also, the Riemannian metric tensor will be more complicated if the covariance is not zero.

In future studies, based on the results of this paper, we will derive the natural gradient descent algorithm for the three-layered stochastic complex-valued neural network, and make clear its properties via computer simulations. Since there exist many singular points in the three-layered stochastic complex-valued neural network [7], it is expected that the complex-valued natural gradient method improves its learning performance dramatically. And also, it has been shown that there exist singular points in the polar variable complex-valued neurons [8]. We will apply the natural gradient method to the polar variable complex-valued neurons.

Acknowledgment

The author would like to give special thanks to the anonymous reviewers for valuable comments.

References

[1] S. Amari, *Differential-geometrical methods in statistics*, Lecture Notes in Statistics vol. 28, Springer-Verlag, 1985.

[2] S. Amari, "Neural learning in structured parameter spaces – Natural Riemannian gradient," In M. C. Mozer, M. I. Jordan, & Th. Petsche (Eds.), *Advances in neural processing systems, 9*, Cambridge, MA: MIT Press, 1996.

[3] S. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.

[4] S. Amari, H. Park, and T. Ozeki, "Singularities affect dynamics of learning in neuromanifolds," *Neural Computation*, vol. 18, no. 5, pp. 1007–1065, 2006.

[5] G. M. Georgiou and C. Koutsougeras, "Complex domain backpropagation," *IEEE Trans. Circuits and Systems–II: Analog and Digital Signal Processing*, vol. 39, no. 5, pp. 330–334, 1992.

[6] T. Nitta (Ed.), *Complex-valued neural networks: utilizing high-dimensional parameters*, Information Science Reference, Pennsylvania, USA, 2009.

[7] T. Nitta, "Local minima in hierarchical structures of complex-valued neural networks," *Neural Networks*, vol. 43, pp. 1–7, 2013.

[8] T. Nitta, "Plateau in a polar variable complex-valued neuron," Proceedings of the 6th International Conference on Agents and Artificial Intelligence, ICAART2014-Anger, March 6-8, 2014, pp. 526–531.

[9] A. Hirose (Ed.), *Complex-valued neural networks: adcances and applications in the IEEE press series on computational intelligence*, Wiley-IEEE Press, 2013.

[10] M. S. Kim and C. C. Guest, "Modification of backpropagation networks for complex-valued signal processing in frequency domain," *Proc. International Joint Conference on Neural Networks*, San Diego, June, 1990, vol. 3, pp. 27–31.

[11] T. Nitta, "An extension of the back-propagation algorithm to complex numbers," *Neural Networks*, vol. 10, no. 8, pp. 1392-1415, 1997.

[12] T. Nitta, "Orthogonality of decision boundaries in complex-valued neural networks," *Neural Computation*, vol. 16, no. 1, pp. 73–97, 2004.

[13] H. H. Yang and S. Amari, "Complexity issues in natural gradient descent method for training multilayer perceptrons," *Neural Computation*, vol. 10, no. 8, pp. 2137–2157, 1998.