Wirtinger Calculus Based Gradient Descent and Levenberg-Marquardt Learning Algorithms in Complex-Valued Neural Networks

Md. Faijul Amin, Muhammad Ilias Amin,
A.Y.H. Al-Nuaimi, and Kazuyuki Murase

Department of System Design Engineering,
University of Fukui, Japan
{amin,ahmedyarub,murase}@synapse.his.u-fukui.ac.jp

Abstract. Complex-valued neural networks (CVNNs) bring in nonholomorphic functions in two ways: (i) through their loss functions and (ii) the widely used activation functions. The derivatives of such functions are defined in Wirtinger calculus. In this paper, we derive two popular algorithms—the gradient descent and the Levenberg-Marquardt (LM) algorithm—for parameter optimization in the feedforward CVNNs using the Wirtinger calculus, which is simpler than the conventional derivation that considers the problem in real domain. While deriving the LM algorithm, we solve and use the result of a least squares problem in the complex domain, $\|\mathbf{b} - (\mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{z}^*)\|_{\min_{\mathbf{z}}}$, which is more general than the $\|\mathbf{b} - \mathbf{A}\mathbf{z}\|_{\min_{\mathbf{z}}}$. Computer simulation results exhibit that as with the real-valued case, the complex-LM algorithm provides much faster learning with higher accuracy than the complex gradient descent algorithm.

Keywords: Complex-valued neural networks (CVNNs), Wirtinger calculus, gradient descent, Levenberg-Marquardt, least squares.

1 Introduction

With the advancement in technology, we see that complex-valued data arise in various practical contexts, such as array signal processing [1], radar and magnetic resonance data processing [2,3], communication systems [4], signal representation in complex baseband [5], and processing data in the frequency domain [2]. Since neural networks are very efficient models in adaptive and nonlinear signal processing, the extension of real-valued neural networks (RVNNs) to complex-valued neural networks (CVNNs) has gained a considerable research interest in the recent years. The key features of CVNNs are that their parameters are complex numbers and they use complex algebraic computations. Although complex-valued data can be processed with RVNNs by considering the data as double dimensional real-valued data, several studies have shown that CVNNs are much more preferable in terms of nonlinear mapping ability, learning convergence, number of parameters, and generalization ability [6].

B.-L. Lu, L. Zhang, and J. Kwok (Eds.): ICONIP 2011, Part I, LNCS 7062, pp. 550–559, 2011.

An important fact in the CVNNs is that they bring in *nonholomorphic* functions in two ways: (i) with the loss function to be minimized over the complex parameters and (ii) the most widely used activation functions. The former is completely unavoidable as the loss function is necessarily real-valued. The second source of nonholomorphism arises because boundedness and analiticity cannot be achieved at the same time in the complex domain, and it is the boundedness that is often preferred over analyticity for the activation functions [6]. Although some researchers have proposed some holomorphic activation functions having singularities [7], a general consideration is that the activation functions can be nonholomorphic. In such a scenario, optimization algorithms are unable to use standard complex derivatives since the derivatives do not exist (i.e., the *Cauchy-Riemann equations* do not hold). As an alternative, conventional approaches cast the optimization problem in the real domain and use the real derivatives, which often requires a tedious computational labor. Here computational labor is meant for human efforts associated with the calculation of derivatives in analytic form.

An elegant approach that can save computational labor in dealing with nonholomorphic functions is to use Wirtinger calculus [8], which uses conjugate coordinate system. A pioneering work that utilizes the concept of conjugate coordinates is by Brandwood [9]. The author formally defines complex gradient and the condition for stationary point. The work is further extended by van dan Bos showing that complex gradient and Hessian are related to their real counterparts by a simple linear transformation [10]. However, neither of the authors has cited the contribution of Wilhelm Wirtinger, a German mathematician, who originally developed the the central idea. Today the Witinger calculus is well appreciated and has been fruitfully exploited by several recent works [11,12].

Although the Wirtinger calculus can be a useful tool in adapting well known first- and second-order optimization algorithms used in the RVNN to the CVNN framework, only few studies can be found in the literature [13]. In [13], the Wirtinger calculus has been utilized to derive a gradient descent algorithm for a feedforward CVNN. The authors employ holomorphic activation functions and show that the derivation is simplified only because of the holomorphic functions. It is further stated that the evaluation of gradient in nonholomorphic case has to be performed in the real domain as it is done traditionally.

In this paper, we argue that the Wirtinger calculus can simplify the gradient evaluation in nonholomorphic activation functions too, which is the original motivation of the Wirtinger calculus. Our gradient evaluation is more general and the CVNN with holomorphic activation function becomes a special case. A major contribution of this paper is that we derive a popular second-order learning method, Levenberg-Marquardt (LM) algorithm [14], for CVNN parameter optimization. We find that a key step of LM algorithm is a solution to the least squares problem $\|\mathbf{b} - (\mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{z}^*)\|_{\min_{\mathbf{z}}}$ in the complex domain, which is more general than the $\|\mathbf{b} - \mathbf{A}\mathbf{z}\|_{\min_{\mathbf{z}}}$. Here $\mathbf{z}^*$ denotes the complex conjugate of a column vector $\mathbf{z}$. A solution to the least squares problem has been carried out with a proof in this paper. All computations regarding gradient descent and LM algorithm are carried out in complex matrix-vector form that can be easily

implemented in any computing environment where computations are optimized for matrix operations.

An important aspect of our derivations is that we use functional dependency graph for a visual evaluation method of derivatives, which is particularly useful in multilayer CVNNs. Because the Wirtinger calculus essentially employs conjugate coordinates, a coordinate transformation matrix between the real and conjugate coordinates system plays an important role in adapting optimization algorithms in the RVNNs to the CVNNs. It turns out that the Wirtinger calculus, the coordinate transformation matrix, and the functional dependency graph are three useful tools for deriving algorithms in the CVNN framework.

The remainder of the paper is organized as follows. Section 2 presents complex domain derivations of two popular algorithms—the gradient descent and the LM algorithm —for CVNN parameter optimization, along with a brief discussion of the Wirtinger calculus. Computer simulation results are discussed in Section 3. Finally, concluding remarks are given in Section 4.

2 Complex Gradient Descent and Complex-LM Algorithm Using Wirtinger Calculus

2.1 Wirtinger Calculus

Any function of a complex variable z can be defined as $f(z) = u(x, y) + jv(x, y)$, where $z = x + jy$ and $j = \sqrt{-1}$. The function is said to be holomorphic (complex derivative exists) if the Cauchy-Riemann equations holds, i.e., $u_x = v_y,\ v_x = -u_y$; otherwise, it is nonholomorphic (complex derivative does not exist). Nonholomorphic functions, however, can be dealt with conjugate coordinates, which are related to the real coordinates by

$$\begin{pmatrix} z \\ z^* \end{pmatrix} = \begin{pmatrix} 1 & j \\ 1 & -j \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad . \tag{1}$$

From the inverse relations, $x = (z + z^*)/2$ and $y = -j(z - z^*)/2$, Wirtinger defines the following pair of derivatives for a function $f(z) = f(z, z^*)$:

$$\frac{\partial f}{\partial z} = \frac{1}{2}\left(\frac{\partial f}{\partial x} - j\frac{\partial f}{\partial y}\right), \quad \frac{\partial f}{\partial z^*} = \frac{1}{2}\left(\frac{\partial f}{\partial x} + j\frac{\partial f}{\partial y}\right) \quad . \tag{2}$$

The derivatives are called $\mathbb{R}$-derivative and conjugate $\mathbb{R}$-derivative, respectively.

Wirtinger calculus generalizes the concept of derivatives in the complex domain. It is easy to see that the Cauchy-Riemann equations are equivalent to $\frac{\partial f}{\partial z^*} = 0$. Rigorous description of the Wirtinger calculus with applications can be found in [16,17]. The attractiveness of Wirtinger calculus is that it enables us to perform all computations directly in the complex domain, and the derivatives obey all rules of conventional calculus, including the chain rule, differentiation of products and quotients. In the evaluation of $\frac{\partial f}{\partial z}$, z^* is considered as a constant

and vice versa. Here are some useful identities that we use extensively in the derivation of learning algorithms presented in the next subsections.

$$\left(\frac{\partial f}{\partial z}\right)^* = \frac{\partial f^*}{\partial z^*}; \quad \text{for } f \text{ is real, } \left(\frac{\partial f}{\partial z}\right)^* = \frac{\partial f}{\partial z^*} \tag{3}$$

$$df = \frac{\partial f}{\partial z}dz + \frac{\partial f}{\partial z^*}dz^* \quad \text{Differential rule} \tag{4}$$

$$\frac{\partial h(g)}{\partial z} = \frac{\partial h}{\partial g}\frac{\partial g}{\partial z} + \frac{\partial h}{\partial g^*}\frac{\partial g^*}{\partial z} \quad \text{Chain rule} \tag{5}$$

$$\frac{\partial h(g)}{\partial z^*} = \frac{\partial h}{\partial g}\frac{\partial g}{\partial z^*} + \frac{\partial h}{\partial g^*}\frac{\partial g^*}{\partial z^*} \quad \text{Chain rule} \tag{6}$$

2.2 The Complex Gradient Descent Algorithm

The gradient of a real-valued scalar function of several complex variables can evaluated in both real and conjugate coordinates systems. In fact, there is a one to one correspondence between the coordinate systems. Let $\mathbf{z}$ be a n-dimensional column vector, i.e., $\mathbf{z} = (z_1, z_2, \ldots, z_n)^T \in \mathbb{C}^n$, where $z_i = x_i + jy_i$, $i = 1, \ldots, n$ and $j = \sqrt{-1}$. Then

$$\mathbf{c} \triangleq \begin{pmatrix} \mathbf{z} \\ \mathbf{z}^* \end{pmatrix} \Leftrightarrow \mathbf{r} \triangleq \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \Re(\mathbf{z}) \\ \Im(\mathbf{z}) \end{pmatrix}$$

and they are related as follows

$$\mathbf{c} = \begin{pmatrix} \mathbf{z} \\ \mathbf{z}^* \end{pmatrix} = \begin{pmatrix} \mathbf{I} & j\mathbf{I} \\ \mathbf{I} & -j\mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \mathbf{J} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \mathbf{J}\mathbf{r} \tag{7}$$

Note that $\mathbf{J}^{-1} = \frac{1}{2}\mathbf{J}^H$, where H denotes the Hermitian transpose. Consequently, the real gradient and the complex gradient are related by a linear transformation. The relation guides the algorithm derivation directly in the complex domain. It is established that the complex-gradient of a real-valued function is evaluated as $\nabla_{\mathbf{z}^*} f = 2\dfrac{\partial f}{\partial \mathbf{z}^*}$ [13].

In the following derivation of complex gradient descent algorithm, we will consider a single hidden layer CVNN for the sake of notational convenience only. The forward equations for signal passing through the network are as follows

$$\mathbf{y} = \mathbf{V}\mathbf{x} + \mathbf{a}; \quad \mathbf{h} = \phi(\mathbf{y}); \quad \mathbf{v} = \mathbf{W}\mathbf{h} + \mathbf{b}; \quad \mathbf{g} = \phi(\mathbf{v}) \ . \tag{8}$$

Here $\mathbf{x}$ is the input signal and $\mathbf{h}$ and $\mathbf{g}$ are the outputs at hidden and output layer, respectively; the weight matrix $\mathbf{V}$ connects the input units to the hidden units, while the matrix $\mathbf{W}$ connects the hidden units to the output units; the column vectors $\mathbf{a}$ and $\mathbf{b}$ are the biases to the hidden and output units, respectively; and ϕ is any activation function (*holomorphic* or *nonholomorphic*) having real partial derivatives. When the function takes a vector as its argument, each component

is mapped individually yielding another vector. The gradient descent algorithm minimizes a real-valued loss function

$$l(\mathbf{z}, \mathbf{z}^*) = \frac{1}{2}\sum_k e_k^* e_k = \frac{1}{2}\mathbf{e}^H\mathbf{e} \ , \tag{9}$$

where $\mathbf{e} = \mathbf{d} - \mathbf{g}$ denotes the complex error and $\mathbf{d}$ the desired signal. The negative of complex gradient can be written as

$$-2\frac{\partial l}{\partial \mathbf{z}^*} = -2\left(\frac{\partial l}{\partial \mathbf{z}}\right)^* = -2\left(\frac{\partial l}{\partial \mathbf{z}^T}\right)^H \ . \tag{10}$$

We find that it is convenient to take derivative of a scalar or a column vector with respect to a row vector as it gives the Jacobian naturally. Now

$$-2\frac{\partial l}{\partial \mathbf{z}^T} = -\frac{\partial\left(\mathbf{e}^H\mathbf{e}\right)}{\partial \mathbf{e}^T}\frac{\partial \mathbf{e}}{\partial \mathbf{z}^T} - \frac{\partial\left(\mathbf{e}^H\mathbf{e}\right)}{\partial\left(\mathbf{e}^T\right)^*}\frac{\partial \mathbf{e}^*}{\partial \mathbf{z}^T} \quad \text{[Eq. (5) in vector form]} \tag{11}$$

$$= \mathbf{e}^H\mathbf{J}_{\mathbf{z}} + \mathbf{e}^T\left(\mathbf{J}_{\mathbf{z}^*}\right)^* \quad \text{[Eq. (3)]} \tag{12}$$

Here, we define two Jacobian matrices, $\mathbf{J}_{\mathbf{z}} = \dfrac{\partial \mathbf{g}}{\partial \mathbf{z}^T}$ and $\mathbf{J}_{\mathbf{z}^*} = \dfrac{\partial \mathbf{g}}{\partial\left(\mathbf{z}^*\right)^T}$. Taking Hermitian transpose to both side of Eq. (12) yields the negative of complex-gradient

$$-\nabla_{\mathbf{z}^*} l = \mathbf{J}_{\mathbf{z}}^H\mathbf{e} + \left(\mathbf{J}_{\mathbf{z}^*}^H\mathbf{e}\right)^* \tag{13}$$

It is clear from Eq. (13) that in order to evaluate complex-gradient all we need is to compute a pair of Jacobians, $\mathbf{J}_{\mathbf{z}}$ and $\mathbf{J}_{\mathbf{z}^*}$. It should be noted that the Jacobians have the form of $\mathbf{P} + j\mathbf{Q}$ and $\mathbf{P} - j\mathbf{Q}$, respectively, because of the definition of derivatives in the Wirtinger calculus. Here $\mathbf{P}$ and $\mathbf{Q}$ are complex matrices since $\mathbf{g}$ is complex-valued. Consequently, we can compute the other one while computing one of the Jacobians. In the feedforward CVNN, parameters are structured in layers. We find that it is very much efficient to compute the Jacobians visually if we draw a functional dependency graph, where each node denotes a vector-valued functional and each edge connecting two nodes is labeled with the Jacobian of one node w.r.t. the other. Figure 1 depicts the functional dependency graph for a single hidden layer CVNN. To evaluate the Jacobian of the right most node (i.e., network's output) w.r.t. any other node to its left, we just need to follow the possible paths from the right most node to that node. Then the desired Jacobian is the sum of all possible paths, where for each path the labeled Jacobians from right to left are multiplied successively. For example, in Fig. 1,

$$\mathbf{J}_{\mathbf{y}} = \frac{\partial \mathbf{g}}{\partial \mathbf{y}^T} = \Lambda_{\mathbf{v}}\mathbf{W}\Lambda_{\mathbf{y}} + \Lambda_{\mathbf{v}^*}\mathbf{W}^*\left(\Lambda_{\mathbf{y}^*}\right)^* \tag{14}$$

where $\Lambda_{\mathbf{v}}$ denotes the Jacobian of right node $\mathbf{g}$ w.r.t. its left node $\mathbf{v}$. Fortunately, we can reuse the computation from right to left. We only need to look for Jacobian to the immediate rightmost nodes, presumably the Jacobian are already computed there. Thus Eq. (14) can be alternatively computed as

$$\mathbf{J}_{\mathbf{y}} = \mathbf{J}_{\mathbf{h}}\Lambda_{\mathbf{y}} + \mathbf{J}_{\mathbf{h}^*}\left(\Lambda_{\mathbf{y}^*}\right)^*$$

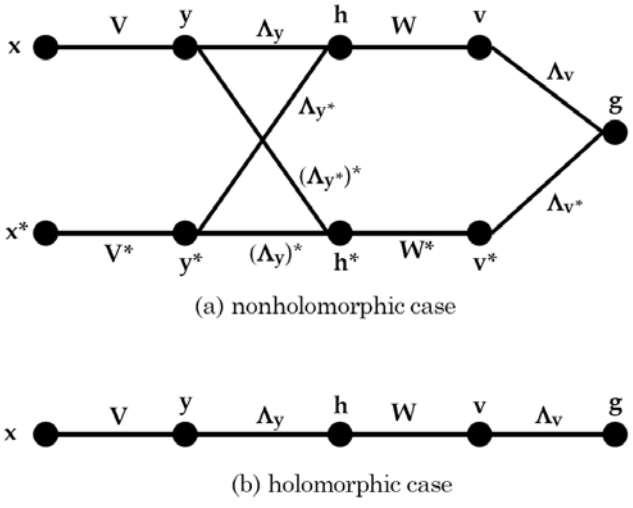

Fig. 1. Functional dependency graph of a single hidden layer CVNN

It is now a simple task to find the update rule for the CVNN parameters. We note from Eq. (8) that $\mathbf{J_b} = \mathbf{J_v}$ and $\mathbf{J_a} = \mathbf{J_y}$. Thus update rules for the biases at hidden and output layer are

$$\Delta\mathbf{a} = \alpha\left(\mathbf{J}_{\mathbf{a}}^H\mathbf{e} + \left(\mathbf{J}_{\mathbf{a}^*}^H\mathbf{e}\right)^*\right); \quad \Delta\mathbf{b} = \alpha\left(\mathbf{J}_{\mathbf{b}}^H\mathbf{e} + \left(\mathbf{J}_{\mathbf{b}^*}^H\mathbf{e}\right)^*\right) \tag{15}$$

Here α is the learning rate. Extending the notation for vector gradient to matrix gradient of a real-valued scalar function [13] and using Eq. (8), the update rules for hidden and output layer weight matrices are given by

$$\Delta\mathbf{V} = (\Delta\mathbf{a})\mathbf{x}^H; \quad \Delta\mathbf{W} = (\Delta\mathbf{b})\mathbf{h}^H \ . \tag{16}$$

2.3 The Complex LM Algorithm

The LM algorithm is a widely used batch-mode fast learning algorithm in the neural networks that also yields higher accuracy in function approximation than the gradient descent algorithm [15]. Basically, it is an extension of Gauss-Newton algorithm. Therefore, the Gauss-Newton algorithm will be first derived in the complex-domain.

The Gauss-Newton method iteratively *re-linearizes* the nonlinear model and updates the current parameter set according to a least squares solution to the linearized model. In the CVNN, the linearized model of network output $\mathbf{g}(\mathbf{z}, \mathbf{z}^*)$ around $(\hat{\mathbf{z}}, \hat{\mathbf{z}}^*)$ is given by

$$\mathbf{g}(\hat{\mathbf{z}} + \Delta\mathbf{z}, \hat{\mathbf{z}}^* + \Delta\mathbf{z}^*) \approx \hat{\mathbf{g}} + \mathbf{J}_{\mathbf{z}}\Delta\mathbf{z} + \mathbf{J}_{\mathbf{z}^*}\Delta\mathbf{z}^* \quad \text{[Eq. (4) in vector form]} \tag{17}$$

The error associated with the linearized model is $\mathbf{e} = \hat{\mathbf{e}} - (\mathbf{J}_{\mathbf{z}}\Delta\mathbf{z} + \mathbf{J}_{\mathbf{z}^*}\Delta\mathbf{z}^*)$ Then the Gauss-Newton update rule is given by the least squares solution to

$\|\hat{\mathbf{e}} - (\mathbf{J}_{\mathbf{z}}\Delta\mathbf{z} + \mathbf{J}_{\mathbf{z}^*}\Delta\mathbf{z}^*)\|$. So we encounter a more general least squares problem, $\|\mathbf{b} - (\mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{z}^*)\|_{\min_{\mathbf{z}}}$, than the well known problem, $\|\mathbf{b} - \mathbf{A}\mathbf{z}\|_{\min_{\mathbf{z}}}$.

Proposition 1. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *are arbitrary complex matrices of same dimension. Then a solution to the least squares problem,* $\|\mathbf{b} - (\mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{z}^*)\|_{\min_{\mathbf{z}}}$, *is given by the following normal equation*

$$\mathbf{C}^H \begin{pmatrix} \mathbf{b} \\ \mathbf{b}^* \end{pmatrix} = \mathbf{C}^H\mathbf{C} \begin{pmatrix} \mathbf{z} \\ \mathbf{z}^* \end{pmatrix}; \quad \text{where } \mathbf{C} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^* & \mathbf{A}^* \end{pmatrix} \ . \tag{18}$$

Proof. From Eq. (7), we know that the conjugate coordinates are related to the real coordinates by the transformation matrix $\mathbf{J}$, while $\mathbf{J}^{-1} = \frac{1}{2}\mathbf{J}^H$. The error equation and its complex conjugate associated to the least squares problem are

$$\mathbf{e} = \mathbf{b} - (\mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{z}^*) \tag{19}$$

$$\mathbf{e}^* = \mathbf{b}^* - (\mathbf{A}^*\mathbf{z}^* + \mathbf{B}^*\mathbf{z}) \tag{20}$$

Combining Eqs. (19) and (20) to form a single matrix equation and applying the coordinate transformation, the problem can be transformed to real coordinate system, for which the normal equation for least squares problem is well known. This gives the following equation

$$\mathbf{J}^H \begin{pmatrix} \mathbf{e} \\ \mathbf{e}^* \end{pmatrix} = \mathbf{J}^H \begin{pmatrix} \mathbf{b} \\ \mathbf{b}^* \end{pmatrix} - \mathbf{J}^H\mathbf{C} \left(\frac{1}{2}\mathbf{J}\mathbf{J}^H\right) \begin{pmatrix} \mathbf{z} \\ \mathbf{z}^* \end{pmatrix}; \quad \left[\frac{1}{2}\mathbf{J}\mathbf{J}^H = I\right] \tag{21}$$

It can be shown that $\mathbf{J}^H\mathbf{C}\frac{1}{2}\mathbf{J}$ is a real-valued matrix. Because it is now completely in the real coordinate system, we can readily apply the normal equation of the form $\mathbf{P}^T\mathbf{q} = \mathbf{P}^T\mathbf{P}\mathbf{x}$, for real-valued matrices. Noting that the ordinary transpose and Hermitian transpose is the same in the real-valued matrices, the normal equation for the least squares problem of Eq. (21) is

$$\frac{1}{2}\mathbf{J}^H\mathbf{C}^H\mathbf{J}\mathbf{J}^H \begin{pmatrix} \mathbf{b} \\ \mathbf{b}^* \end{pmatrix} = \frac{1}{2}\mathbf{J}^H\mathbf{C}^H\mathbf{J}\mathbf{J}^H\mathbf{C} \left(\frac{1}{2}\mathbf{J}\mathbf{J}^H\right) \begin{pmatrix} \mathbf{z} \\ \mathbf{z}^* \end{pmatrix} \tag{22}$$

Equation (22) immediately yields the following complex normal equation

$$\mathbf{C}^H \begin{pmatrix} \mathbf{b} \\ \mathbf{b}^* \end{pmatrix} = \mathbf{C}^H\mathbf{C} \begin{pmatrix} \mathbf{z} \\ \mathbf{z}^* \end{pmatrix}$$ □

According to Proposition 1, the least squares solution to $\|\hat{\mathbf{e}} - (\mathbf{J}_{\mathbf{z}}\Delta\mathbf{z} + \mathbf{J}_{\mathbf{z}^*}\Delta\mathbf{z}^*)\|$ gives the the following Gauss-Newton update rule

$$\begin{pmatrix} \Delta\mathbf{z} \\ \Delta\mathbf{z}^* \end{pmatrix} = \mathbf{H}^{-1}\mathbf{G}^H \begin{pmatrix} \hat{\mathbf{e}} \\ \hat{\mathbf{e}}^* \end{pmatrix} \tag{23}$$

$$\text{where } \mathbf{G} = \begin{pmatrix} \mathbf{J}_{\mathbf{z}} & \mathbf{J}_{\mathbf{z}^*} \\ (\mathbf{J}_{\mathbf{z}^*})^* & (\mathbf{J}_{\mathbf{z}})^* \end{pmatrix} \text{ and } \mathbf{H} = \mathbf{G}^H\mathbf{G} = \begin{pmatrix} \mathbf{H}_{\mathbf{z}\mathbf{z}} & \mathbf{H}_{\mathbf{z}^*\mathbf{z}} \\ \mathbf{H}_{\mathbf{z}\mathbf{z}^*} & \mathbf{H}_{\mathbf{z}^*\mathbf{z}^*} \end{pmatrix}$$

The matrix $\mathbf{H}$ can be considered as an approximation to the Hessian matrix that would result from the Newton method. Note that when $\mathbf{H} = \mathbf{I}$, the Gauss-Newton update rule reduces to the gradient descent algorithm. There is also a pseudo-Gauss-Newton algorithm [17], where the off-diagonal block matrices of $\mathbf{H}$ are $\mathbf{0}$. The pseudo-Gauss-Newton update rule then takes a simpler form

$$\Delta\mathbf{z}^{pseudo-Gauss-Newton} = \mathbf{H}_{\mathbf{zz}}^{-1}\left(\mathbf{J}_{\mathbf{z}}^{H}\hat{\mathbf{e}} + \left(\mathbf{J}_{\mathbf{z}^*}^{H}\hat{\mathbf{e}}\right)^*\right) \quad . \tag{24}$$

When the activation functions in the CVNN are holomorphic, the output function $\mathbf{g}(\mathbf{z})$ is also holomorphic. Then the Gauss-Newton update rule resembles the real-valued case

$$\Delta\mathbf{z}^{holomorphic} = \left(\mathbf{J}_{\mathbf{z}}^{H}\mathbf{J}_{\mathbf{z}}\right)^{-1}\mathbf{J}_{\mathbf{z}}^{H}\hat{\mathbf{e}} \tag{25}$$

It can be observed that all the computations use the Jacobian matrices extensively, which can be evaluated visually and efficiently from the functional dependency graph of Fig. 1. It thus shows the simplicity of our derivation using the Wirtinger calculus.

The LM algorithm makes a simple modification to the Gauss-Newton algorithm of Eq. (23) in the following way

$$\begin{pmatrix}\Delta\mathbf{z}\\ \Delta\mathbf{z}^*\end{pmatrix} = \left(\mathbf{G}^H\mathbf{G} + \mu\mathbf{I}\right)^{-1}\mathbf{G}^H\begin{pmatrix}\hat{\mathbf{e}}\\ \hat{\mathbf{e}}^*\end{pmatrix} \tag{26}$$

The parameter μ is varied over the iterations. Whenever a step increases the error rather than decreasing, μ is multiplied by a factor β. Otherwise, μ is divided by β. A popular choice for the initial value of μ is 0.01 and $\beta = 10$. The algorithmic steps of complex-LM is same as the steps of real-LM [15], except for the parameter update rule derived above.

3 Computer Simulation Results

Computer experiments were performed in order to verify the algorithms presented in the previous section. The algorithms were implemented in Matlab to utilize its matrix computing environment. We took a problem for experiment from [6], where the task is to learn a set of given patterns (see Table 3 of [6]). We trained a 2-4-1 CVNN using the complex gradient descent learning algorithm and two variants of complex-LM algorithm. Note that the LM algorithm uses Gauss-Newton update rule as its basic constituent, which has a variant called pseudo-Gauss-Newton method. Accordingly, we call the variants as complex-LM and pseudo-complex-LM. For all learning algorithms, the training was stopped when the error (mean squared error (MSE)) goal was met, or a maximum number of iteration has been passed. We used the same activation function of [6] in the hidden and output layer, which is nonholomorphic.

The number of iterations required to meet error goal (MSE = 0.001) were 32 and 82 for the complex-LM and the pseudo-complex-LM, respectively.

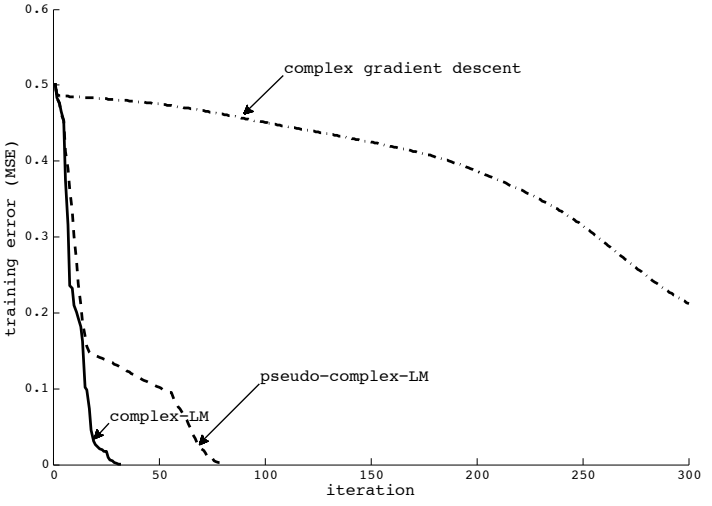

Fig. 2. Learning curves of complex gradient descent, complex-LM, and pseudo-complex-LM algorithm

The complex gradient descent, however, failed to reach the error goal. Figure 2 shows the learning curves for first 300 iterations. Note that the LM algorithms stopped long before. The experimental result suggests that the complex-LM along with its variant has very fast learning convergence with much lower MSE than the complex gradient descent algorithm, which resembles the learning characteristics in the RVNNs. Furthermore, the complex-LM is faster than the pseudo-complex-LM since the latter is an approximation to the former. But the pseudo-complex-LM involves less computation.

The application of complex-LM algorithms, however, is limited by the number of parameters of CVNN, because computation of a matrix inversion is involved in each iteration, while the matrix dimension is in the order of total number of learning parameters. Therefore, the complex-LM algorithms are very useful when the number of parameters are reasonable for matrix inversion and/or a high accuracy in the network mapping is required, such as system identification and time-series prediction problems in the complex domain.

4 Conclusions

In this paper, the complex gradient descent and the complex-LM algorithm have been derived using the Wirtinger calculus, which enables performing all computations directly in the complex domain. The use of the Wirtinger calculus also simplifies the derivation. In course of complex-LM derivation, we have encountered a general least squares problem in the complex domain. We have presented a solution with proof and used the result for derivation in this paper. We identify

that the Wirtinger calculus, the coordinate transformation between conjugate and real coordinates, and the functional dependency graph of Jacobians greatly simplify adaptation of the algorithms known for the RVNN to the CVNN framework. Computer simulation results are provided to verify the derivations, and it is shown that the complex-LM as well as its variant, the pseudo-complex-LM, have much faster learning convergence with higher accuracy in nonlinear mapping by the CVNN.

References

1. van Trees, H.L.: Optimum Array Processing. Wiley Interscience, New York (2002)
2. Hirose, A.: Complex-Valued Neural Networks. Springer, Heidelberg (2006)
3. Calhoun, V.D., Adali, T., Pearlson, G.D., van Zijl, P.C.M., Pekar, J.J.: Independent component analysis of fMRI data in the complex domain. Magnetic Resonance in Medicine 48(1), 180–192 (2002)
4. Stuber, G.L.: Principles of Mobile Communication. Kluwer, Boston (2001)
5. Helstrom, C.W.: Elements of Signal Detection and Estimation. Prentice Hall, New Jersey (1995)
6. Nitta, T.: An Extension of the Back-Propagation Algorithm to Complex Numbers. Neural Networks 10(8), 1391–1415 (1997)
7. Kim, T., Adali, T.: Approximation by Fully-Complex Multilayer Perceptrons. Neural Computation 15(7), 1641–1666 (2003)
8. Wirtinger, W.: Zur Formalen Theorie der Funktionen von mehr Complexen Veränderlichen. Mathematische Annalen 97, 357–375 (1927) (in German)
9. Brandwood, D.H.: Complex Gradient Operator and its Application in Adaptive Array Theory. IEE Proceedings F (Communications, Radar and Signal Processing) 130(1), 11–16 (1983)
10. van den Bos, A.: Complex Gradient and Hessian. IEEE Proceedings (Vision, Image and Signal Processing) 141(6), 380–382 (1994)
11. Bouboulis, P., Theodoridis, S.: Extension of Wirtinger's Calculus to Reproducing Kernel Hilbert Spaces and the Complex Kernel LMS. IEEE Trans. on Signal Processing 59(3), 964–978 (2011)
12. Li, H., Adali, T.: Algorithms for Complex ML ICA and Their Stability Analysis Using Wirtinger Calculus. IEEE Trans. on Signal Processing 58(12), 6156–6167 (2010)
13. Li, H., Adali, T.: Complex-Valued Adaptive Signal Processing Using Nonlinear Functions. EURASIP Journal on Advances in Signal Processing 2008, 1–9 (2008)
14. Marquardt, D.: An Algorithm for Least-Squares Estimation of Nonlinear Parameters. SIAM Journal on Applied Mathematics 11(2), 431–441 (1963)
15. Hagan, M., Menhaj, M.: Training Feedforward Networks with the Marquardt Algorithm. IEEE Trans. on Neural Networks 5(6), 989–993 (1994)
16. Remmert, R.: Theory of Complex Functions. Springer, New York (1991)
17. Kreutz-Delgado, K.: The Complex Gradient Operator and the CR-Calculus, `http://dsp.ucsd.edu/~kreutz/PEI05.htm`