

Article

Quaternionic Multilayer Perceptron with Local Analyticity

Tejiro Isokawa ^{1,*}, Haruhiko Nishimura ² and Nobuyuki Matsui ¹

¹ Graduate School of Engineering, University of Hyogo, 2167 Shosha, Himeji, Hyogo 671-2280, Japan; E-Mail: matsui@eng.u-hyogo.ac.jp

² Graduate School of Applied Informatics, University of Hyogo, 7-1-28 Minatojima-minamimachi, Chuo-ku, Kobe, Hyogo 650-0047, Japan; E-Mail: haru@ai.u-hyogo.ac.jp

* Author to whom correspondence should be addressed; E-Mail: isokawa@eng.u-hyogo.ac.jp; Tel.: +81-79-267-4952; Fax: +81-79-267-4975.

Received: 11 September 2012; in revised form: 13 November 2012 / Accepted: 20 November 2012 / Published: 28 November 2012

Abstract: A multi-layered perceptron type neural network is presented and analyzed in this paper. All neuronal parameters such as input, output, action potential and connection weight are encoded by quaternions, which are a class of hypercomplex number system. Local analytic condition is imposed on the activation function in updating neurons' states in order to construct learning algorithm for this network. An error back-propagation algorithm is introduced for modifying the connection weights of the network.

Keywords: quaternion; local analyticity; Wirtinger calculus; multilayer perceptron; error back-propagation

1. Introduction

Processing multi-dimensional data is an important problem for artificial neural networks. A single neuron can take only one real value as its input, thus a network should be configured so that several neurons are used for accepting multi-dimensional data. This type of configuration is sometimes unnatural in applications of artificial neural networks to engineering problem, such as processing of acoustic signals or coordinates in the plane. Thus, complex number systems have been utilized to represent two-dimensional data elements as a single entity. Application of complex numbers to neural networks have been extensively investigated, as summarized in the references [1–3].

Though complex values can treat two-dimensional data elements as a single entity, what we should treat data with more than two-dimension in artificial neural networks? Although this problem can of course be solved by applying several real-valued or complex-valued neurons, it would be useful to introduce a number system with higher dimensions, the so-called hypercomplex number systems.

Quaternion is a four-dimensional hypercomplex number system introduced by Hamilton [4,5]. This number system has been extensively employed in several fields, such as modern mathematics, physics, control of satellites, computer graphics, *etc.* [6–8]. One of the benefits provided by quaternions is that affine transformations of geometric figures in three-dimensional spaces, especially spatial rotations, can be represented compactly and efficiently. Applying quaternions to the field of neural networks has been recently explored in an effort to naturally represent high-dimensional information, such as color and three-dimensional body coordinates, by a quaternionic neuron, rather than complex-valued or real-valued neurons.

Thus, there has been a growing number of studies concerning the use of quaternions in neural networks. Multilayer perceptron (MLP) models have been developed in [9–14]. The use of quaternion in MLP models has been applied to several engineering problems such as control problems [10], color image compression [12], color night vision [15,16], and predictions for the output of chaos circuits and winds in three-dimensional space [13,14]. Other types of network models has also been explored, such as the computational ability of a single quaternionic neuron [17] and the existence condition of an energy function in continuous-time and continuous-state recurrent networks [18]. There are several types of quaternionic Hopfield-type networks with discrete-time driven, such as bipolar state [19,20], continuous state [21,22], multistate by phase representation [23,24]. Learning schemes for these networks have also been proposed in [25].

One of the difficulties in constructing neural networks in the quaternionic domain is about the introduction of suitable functions for the activation function in updating the neurons' states. A typical type of activation function is the so-called "split" type function, in which a real-valued function is applied to update each component of a quaternionic value [10]. Real-valued sigmoidal function and hyperbolic function, which are analytic (differentiable) functions, are often used for this purpose. However, as an activation function, the split-type quaternionic function is not appropriate due to lack of analyticity. Thus, it is necessary to define other types of differential functions in order to construct learning schemes such as the error back-propagation algorithm. The Cauchy–Riemann–Fueter (CRF) equation defines the analytic condition for the quaternionic functions, which corresponds to the Cauchy–Riemann equation for the complex-valued functions. The functions satisfying the CRF equation turn out to be only linear functions or constants; therefore it is impossible to introduce non-linearity into the updates of the neurons' states.

Recently, another class of analyticity for the quaternionic functions has been developed [26–28]. Called "*local analyticity*", this analytic condition is derived at a quaternionic point with its local coordinate, rather than in a quaternionic space with a global coordinate. The derivation of local analytic condition shows that a quaternion in the local coordinate system is isomorphic to the complex number system and thus it can be treated as a complex value. A neural network with an activation function with local analyticity has been first proposed and analyzed in [13,14] in terms of MLP-type network. It is shown that the performance for this network is superior to the network with a split-type activation function through several applications. Another application of analytic activation function to

quaternionic neural networks has been investigated in terms of Hopfield-type network [22]. The local analytic condition is constructed based on [28], and the stability conditions are derived in the case of quaternionic tanh function being used for an activation function is deduced in this network, in which the complex-valued tanh function used in [29] can be used as a quaternionic function.

This paper presents an MLP-type quaternionic neural network with locally analytic activation function. All variables in this network, such as input, output, action potential and connection weights, are encoded by quaternions. A learning scheme, a quaternionic equivalent of error back-propagation algorithm, is presented and theoretically explored. The derivation of the learning scheme in this paper adopts the Wirtinger calculus [30], which has been invented in the field of complex analysis, where a quaternionic value and its conjugate are treated independent of each other. This calculus enables the derivations to be more straightforward of its description, than by using the conventional description, *i.e.*, Cartesian representation.

2. Quaternionic Algebra

2.1. Definition of Quaternion

Quaternions form a class of hypercomplex numbers consisting of a real number and three imaginary numbers—*i*, *j*, and *k*. Formally, a quaternion number is defined as a vector *x* in a four-dimensional vector space,

$$\mathbf{x} = x^{(e)} + x^{(i)}\mathbf{i} + x^{(j)}\mathbf{j} + x^{(k)}\mathbf{k} \tag{1}$$

where $x^{(e)}$, $x^{(i)}$, $x^{(j)}$, and $x^{(k)}$ are real numbers. The division ring of quaternions, *H*, constitutes the four-dimensional vector space over the real numbers with bases 1, *i*, *j*, and *k*. Equation (1) can also be written using 4-tuple or 2-tuple notation as

$$\mathbf{x} = (x^{(e)}, x^{(i)}, x^{(j)}, x^{(k)}) = (x^{(e)}, \vec{x}) \tag{2}$$

where $\vec{x} = \{x^{(i)}, x^{(j)}, x^{(k)}\}$. In this representation, $x^{(e)}$ is the scalar part of *x*, and \vec{x} forms the vector part. The quaternion conjugate is defined as

$$\mathbf{x}^* = (x^{(e)}, -\vec{x}) = x^{(e)} - x^{(i)}\mathbf{i} - x^{(j)}\mathbf{j} - x^{(k)}\mathbf{k} \tag{3}$$

Quaternion bases satisfy the following identities,

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \tag{4}$$

$$\mathbf{ij} = -\mathbf{ji} = \mathbf{k}, \mathbf{jk} = -\mathbf{kj} = \mathbf{i}, \mathbf{ki} = -\mathbf{ik} = \mathbf{j} \tag{5}$$

known as the Hamilton rule. From these rules, it follows immediately that multiplication of quaternions is not commutative.

Next, we define the operations between quaternions $\mathbf{p} = (p^{(e)}, \vec{p}) = (p^{(e)}, p^{(i)}, p^{(j)}, p^{(k)})$ and $\mathbf{q} = (q^{(e)}, \vec{q}) = (q^{(e)}, q^{(i)}, q^{(j)}, q^{(k)})$. The addition and subtraction of quaternions are defined in a similar manner as for complex-valued numbers or vectors, *i.e.*,

$$\mathbf{p} \pm \mathbf{q} = (p^{(e)} \pm q^{(e)}, \vec{p} \pm \vec{q}) \tag{6}$$

$$= (p^{(e)} \pm q^{(e)}, p^{(i)} \pm q^{(i)}, p^{(j)} \pm q^{(j)}, p^{(k)} \pm q^{(k)}) \tag{7}$$

The product of \mathbf{p} and \mathbf{q} is determined by Equation (5) as

$$\mathbf{pq} = (p^{(e)}q^{(e)} - \vec{p} \cdot \vec{q}, p^{(e)}\vec{q} + q^{(e)}\vec{p} + \vec{p} \times \vec{q}) \tag{8}$$

where $\vec{p} \cdot \vec{q}$ and $\vec{p} \times \vec{q}$ denote the dot and cross products, respectively, between three-dimensional vectors \vec{p} and \vec{q} . The conjugate of the product is given as

$$(\mathbf{pq})^* = \mathbf{q}^* \mathbf{p}^* \tag{9}$$

The quaternion norm of \mathbf{x} , denoted by $|\mathbf{x}|$, is defined as

$$|\mathbf{x}| = \sqrt{\mathbf{xx}^*} = \sqrt{x^{(e)2} + x^{(i)2} + x^{(j)2} + x^{(k)2}} \tag{10}$$

2.2. Quaternionic Analyticity

It is important to introduce an analytic function (or differentiable function) to serve as the activation function in the neural network. This section describes the required analyticity of the function in the quaternionic domain, in order to construct activation functions for quaternionic neural networks.

The condition for differentiability of the quaternionic function \mathbf{f} is given by

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x^{(e)}} = -\mathbf{i} \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x^{(i)}} = -\mathbf{j} \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x^{(j)}} = -\mathbf{k} \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x^{(k)}} \tag{11}$$

The analytic condition for the quaternionic function, called the Cauchy–Riemann–Fueter (CRF) equation, yields:

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x^{(e)}} + \mathbf{i} \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x^{(i)}} + \mathbf{j} \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x^{(j)}} + \mathbf{k} \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x^{(k)}} = 0 \tag{12}$$

This is an extension of the Cauchy–Riemann (CR) equations defined for the complex domain. However, only linear functions and constants satisfy the CRF equation [14,26,27].

An alternative approach to assure analyticity in the quaternionic domain has been explored in [26–28]. This approach is called *local analyticity* and is distinguished from the standard analyticity, *i.e.*, global analyticity. In the following, we introduce local derivatives with Wirtinger representation and analytic conditions for quaternionic functions, with reference to [28].

A quaternion \mathbf{x} can be alternatively represented as:

$$\mathbf{x} = x^{(e)} + \mathbf{u}_x r \tag{13}$$

$$r = \sqrt{x^{(i)2} + x^{(j)2} + x^{(k)2}} \tag{14}$$

$$\mathbf{u}_x = \frac{x^{(i)} \mathbf{i} + x^{(j)} \mathbf{j} + x^{(k)} \mathbf{k}}{r} \tag{15}$$

From the definition in Equation (15), we deduce that $\mathbf{u}_x^2 = -1$. If \mathbf{u}_x holds a commutative property against a difference of \mathbf{x} , then the system with \mathbf{u}_x can be regarded as locally isomorphic to the complex number system.

A quaternionic difference of \mathbf{x} , denoted by $d\mathbf{x} = (dx^{(e)}, dx^{(i)}, dx^{(j)}, dx^{(k)})$, can be decomposed by using:

$$d\mathbf{x} = d\mathbf{x}_{\parallel} + d\mathbf{x}_{\perp} \tag{16}$$

where,

$$d\mathbf{x}_{\parallel} = \frac{1}{2} (d\mathbf{x} - \mathbf{u}_x d\mathbf{x} \mathbf{u}_x), \quad d\mathbf{x}_{\perp} = \frac{1}{2} (d\mathbf{x} + \mathbf{u}_x d\mathbf{x} \mathbf{u}_x)$$

Then, the following relations hold:

$$d\mathbf{x}_{\parallel} \mathbf{x} = \mathbf{x} d\mathbf{x}_{\parallel}, \quad d\mathbf{x}_{\perp} \mathbf{x} = \mathbf{x}^* d\mathbf{x}_{\perp}$$

when we set $d\mathbf{x}_{\perp} = 0$, i.e., $d\mathbf{x} + \mathbf{u}_x d\mathbf{x} \mathbf{u}_x = 0$, which results in $\mathbf{u}_x d\mathbf{x} = d\mathbf{x} \mathbf{u}_x$. This leads to $\mathbf{u}_x \times d\vec{x} = 0$, because \mathbf{u}_x is a quaternion without a real part. Thus, \mathbf{u}_x and $d\vec{x}$ are parallel to each other. Then, $d\vec{x} = \mathbf{u}_x \delta$ can be obtained, where δ is a real-valued constant. From Equation (14), it follows that

$$\begin{aligned} dr &= \frac{\partial r}{\partial x^{(i)}} dx^{(i)} + \frac{\partial r}{\partial x^{(j)}} dx^{(j)} + \frac{\partial r}{\partial x^{(k)}} dx^{(k)} \\ &= \frac{1}{2} \frac{1}{r} 2x^{(i)} dx^{(i)} + \frac{1}{2} \frac{1}{r} 2x^{(j)} dx^{(j)} + \frac{1}{2} \frac{1}{r} 2x^{(k)} dx^{(k)} \\ &= \frac{1}{r} x^{(i)} dx^{(i)} + \frac{1}{r} x^{(j)} dx^{(j)} + \frac{1}{r} x^{(k)} dx^{(k)} \end{aligned}$$

Then,

$$\begin{aligned} r dr &= x^{(i)} dx^{(i)} + x^{(j)} dx^{(j)} + x^{(k)} dx^{(k)} \\ &= \vec{x} \cdot d\vec{x} \end{aligned}$$

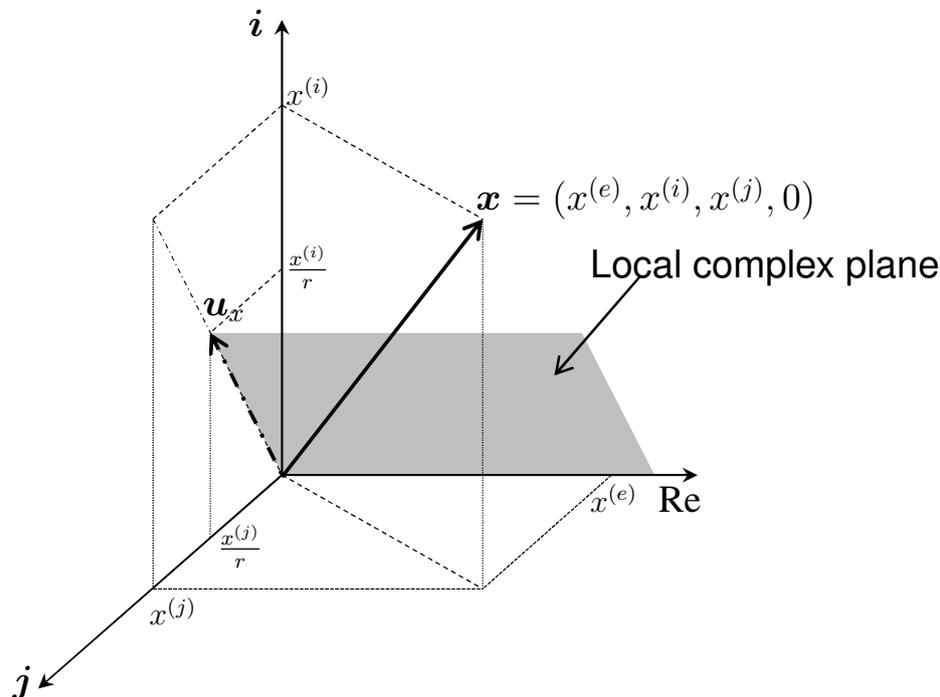
Considering $\vec{x} = \mathbf{u}_x r$ and $d\vec{x} = \mathbf{u}_x \delta$, we obtain

$$\begin{aligned} r dr &= r \delta \mathbf{u}_x \cdot \mathbf{u}_x \\ &= r \delta \end{aligned}$$

Hence, $\delta = dr$ is derived and $d\vec{x} = \mathbf{u}_x dr$ is obtained. $d\mathbf{x}$ is represented as $d\mathbf{x} = d\mathbf{x}_{\parallel} = dx^{(e)} + d\vec{x} = dx^{(e)} + \mathbf{u}_x dr$.

Figure 1 shows a schematic coordinate system for defining a local complex plane. The component \mathbf{k} is omitted ($x^{(k)} = 0$) in this figure due to difficulties in representing a four-dimensional vector space. In this example, for a given quaternion \mathbf{x} , its unit vector \mathbf{u}_x is defined in the i - j plane. Then, a complex plane is defined by spanning the components $x^{(e)}$ (real axis) and $x^{(r)}$ in the quaternionic space, and the analytic condition is constrained in this plane.

Figure 1. A schematic illustration of local complex plane in a quaternionic space, where the component k is omitted for simplicity.



The local derivative operators are introduced, corresponding to the form of dx_{\parallel} , as follows:

$$\frac{\partial}{\partial x_{\parallel}} = \frac{1}{2} \left(\frac{\partial}{\partial x^{(e)}} - \mathbf{u}_x \frac{\partial}{\partial x^{(r)}} \right)$$

$$\frac{\partial}{\partial x_{\parallel}^*} = \frac{1}{2} \left(\frac{\partial}{\partial x^{(e)}} + \mathbf{u}_x \frac{\partial}{\partial x^{(r)}} \right)$$

where

$$\frac{\partial}{\partial x^{(r)}} \equiv \frac{x^{(i)}}{r} \frac{\partial}{\partial x^{(i)}} + \frac{x^{(j)}}{r} \frac{\partial}{\partial x^{(j)}} + \frac{x^{(k)}}{r} \frac{\partial}{\partial x^{(k)}}$$

with the properties

$$\frac{\partial x}{\partial x_{\parallel}} = \frac{\partial x^*}{\partial x_{\parallel}^*} = 1 \quad \text{and} \quad \frac{\partial x^*}{\partial x_{\parallel}} = \frac{\partial x}{\partial x_{\parallel}^*} = 0$$

Note that the variables x and x^* turn out to be independent of each other. These derivative operators are quaternionic equivalents to the well-known Wirtinger derivative in the complex domain [30].

$F(x + dx)$ can be expanded using the above-mentioned representations as

$$F(x + dx) = F(x) + F^{(1)} + F^{(2)} + O(dx^3) \tag{17}$$

where,

$$F^{(1)} = F'(x)dx_{\parallel} + \frac{F(x) - F(x^*)}{(x - x^*)} dx_{\perp},$$

$$F^{(2)} = \frac{1}{2} F''(x)dx_{\parallel}^2 + \frac{F(x) - F(x^*)}{(x - x^*)^2} (dx_{\perp}dx_{\parallel} - dx dx_{\perp})$$

$$+ \frac{F'(x)}{x - x^*} dx dx_{\perp} + \frac{F'(x^*)}{x^* - x} dx_{\perp} dx_{\parallel}$$

When $dx_{\perp} = 0$, the local derivative of $F(\mathbf{x})$ is written as

$$F'(\mathbf{x}) = \frac{\partial F(\mathbf{x})}{\partial \mathbf{x}_{\parallel}}$$

and the local analytic condition for the function $F(\mathbf{x})$ is given by

$$\frac{\partial F(\mathbf{x})}{\partial \mathbf{x}_{\parallel}^*} = 0 \quad \text{i.e.,} \quad \frac{\partial F}{\partial x^{(e)}} + \mathbf{u}_x \frac{\partial F}{\partial x^{(r)}} = 0$$

in the corresponding local complex plane. This result corresponds to the one presented in [27], where $dx_{\perp} = 0$ always holds.

Moreover, if F is a function with the two arguments, \mathbf{x} and \mathbf{x}^* , it becomes:

$$\begin{aligned} & F(\mathbf{x} + d\mathbf{x}, \mathbf{x}^* + d\mathbf{x}^*) \\ &= F(\mathbf{x}, \mathbf{x}^*) + \frac{\partial F}{\partial \mathbf{x}_{\parallel}} d\mathbf{x}_{\parallel} + \frac{\partial F}{\partial \mathbf{x}_{\parallel}^*} d\mathbf{x}_{\parallel}^* + \frac{1}{2} \left(\frac{\partial^2 F}{\partial \mathbf{x}_{\parallel}^2} d\mathbf{x}_{\parallel}^2 + \frac{\partial}{\partial \mathbf{x}_{\parallel}^*} \left(\frac{\partial F}{\partial \mathbf{x}_{\parallel}} \right) d\mathbf{x}_{\parallel} d\mathbf{x}_{\parallel}^* \right. \\ & \quad \left. + \frac{\partial}{\partial \mathbf{x}_{\parallel}} \left(\frac{\partial F}{\partial \mathbf{x}_{\parallel}^*} \right) d\mathbf{x}_{\parallel}^* d\mathbf{x}_{\parallel} + \frac{\partial^2 F}{\partial \mathbf{x}_{\parallel}^*{}^2} d\mathbf{x}_{\parallel}^*{}^2 \right) + O(d\mathbf{x}_{\parallel}^3) \end{aligned} \tag{18}$$

with \mathbf{x} and \mathbf{x}^* being independent of each other. As a result, we can treat quaternionic functions in the same manner as complex-valued functions under the condition of local analyticity.

3. Quaternionic Multilayer Perceptron

3.1. Network Model

The structure of the network assumed in this paper is shown in Figure 2. This network is a so-called multilayer perceptron network with one hidden layer, and the parameters in the network are encoded by quaternionic values.

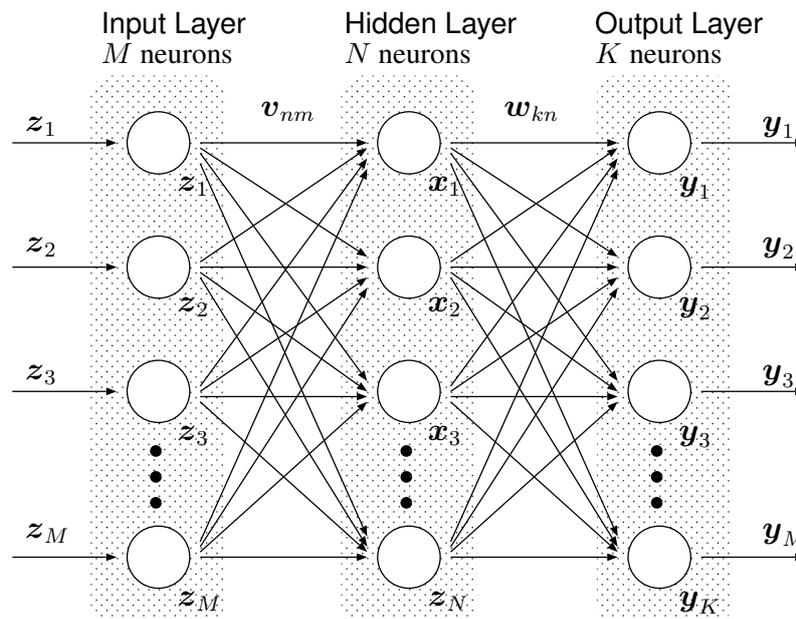
The numbers of neurons in the input, hidden and output layers are set to M , N and K , respectively. A set of quaternionic signals denoted by \mathbf{z} is input to the neurons in the input layer of the network. The outputs of the neurons in the input layer are the same as input \mathbf{z} 's. In the hidden layer, each neuron takes the weighted sum of the output signals from the input layer. The (connection) weight from the n -th neuron in the input layer to the m -th neuron in the hidden layer is denoted by v_{nm} . The output of the neuron in the hidden layer, denoted by \mathbf{x}_n , is determined by

$$\mathbf{x}_n = \mathbf{g} \left(\sum_{m=1}^M v_{nm} \mathbf{z}_m \right) \tag{19}$$

where \mathbf{g} is a quaternionic activation function introducing non-linearity between the action potential and output in the neuron. This function satisfies the following condition:

$$(\mathbf{g}(\mathbf{z}))^* = \mathbf{g}(\mathbf{z}^*) \tag{20}$$

Figure 2. The structure of the multilayer perceptron in this paper.



Processing the neurons' outputs in the output layer can be defined in the same manner as in the hidden layer. The output of the neuron in the output layer, y_k , is defined as

$$y_k = h \left(\sum_{n=1}^N w_{kn} x_n \right) \tag{21}$$

where the function h is a quaternionic activation function from the action potential to the output, and w_{kn} is the connection weight between the n -th neuron in the hidden layer and the k -th neuron in the output layer. The function h also satisfies

$$(h(x))^* = h(x^*) \tag{22}$$

The connection weights should be modified by the so-called learning algorithms, in order to obtain the desired output signals with respect to the input signals. One of the learning algorithms for MLP-type networks is the error back-propagation (EBP) algorithm. The following section describes our derivation of this algorithm for the presented network.

3.2. Learning Algorithm

An EBP algorithm works so that the output error, calculated by the neurons' outputs at the output layer and the desired output signals, is minimized. In the case of networks with three layers as shown in Figure 2, the connection weights between the hidden and output layers are first modified, and then the weights between the input and hidden layers are modified. In general (networks with n layers), EBP algorithms first modify the connection weights between the n -th layer (the output layer) and $(n - 1)$ -th layer, and then between the $(n - 1)$ -th layer and the $(n - 2)$ -th layer, *etc.* This section only describes the three-layers case.

First, let \mathbf{d}_k be a quaternionic desired signal for the k -th output neuron when \mathbf{z} 's are input to the network. The connection weights affect the output signals with respect to a set of input signals, thus the error E is regarded as a function with arguments \mathbf{w}_{kn} 's and \mathbf{w}_{kn}^* 's. The output error E at the time t is then defined as

$$E(t) = E(\mathbf{w}_{kn}(t), \mathbf{w}_{kn}^*(t)) = \sum_{k=1}^K (\mathbf{d}_k - \mathbf{y}_k(t)) (\mathbf{d}_k - \mathbf{y}_k(t))^* \tag{23}$$

The output error should be real-valued so that it can be minimized.

Suppose that the connection weights are updated at the time $(t + 1)$ by

$$\mathbf{w}_{kn}(t + 1) = \mathbf{w}_{kn}(t) + \Delta\mathbf{w}_{kn} \tag{24}$$

where $\Delta\mathbf{w}_{kn}$ is a quantity in updating. Then, the output error at the time $(t + 1)$ can be written as

$$\begin{aligned} E(t + 1) &= E(\mathbf{w}_{kn}(t + 1), \mathbf{w}_{kn}^*(t + 1)) \\ &= E(\mathbf{w}_{kn}(t), \mathbf{w}_{kn}^*(t)) + \sum_{k,n} \frac{\partial E}{\partial \mathbf{w}_{kn\parallel}} \Delta\mathbf{w}_{kn} + \sum_{k,n} \frac{\partial E}{\partial \mathbf{w}_{kn}^*} \Delta\mathbf{w}_{kn}^* \end{aligned} \tag{25}$$

Note that the local analytic condition in quaternionic domain should be satisfied in calculating the derivatives. Thus, if we set $\Delta\mathbf{w}_{kn}$ as

$$\Delta\mathbf{w}_{kn} = -\mu \frac{\partial E}{\partial \mathbf{w}_{kn}^*}, \quad \left(i.e., \quad \Delta\mathbf{w}_{kn}^* = -\mu^* \frac{\partial E}{\partial \mathbf{w}_{kn\parallel}} \right) \tag{26}$$

where μ is a quaternionic constant, the temporal difference of the output error, ΔE , becomes

$$\begin{aligned} \Delta E &= E(t + 1) - E(t) \\ &= \sum_{k,n} \frac{\partial E}{\partial \mathbf{w}_{kn\parallel}} \Delta\mathbf{w}_{kn} + \sum_{k,n} \frac{\partial E}{\partial \mathbf{w}_{kn}^*} \Delta\mathbf{w}_{kn}^* \\ &= -\mu \sum_{k,n} \frac{\partial E}{\partial \mathbf{w}_{kn\parallel}} \frac{\partial E}{\partial \mathbf{w}_{kn}^*} - \mu^* \sum_{k,n} \frac{\partial E}{\partial \mathbf{w}_{kn}^*} \frac{\partial E}{\partial \mathbf{w}_{kn\parallel}} \\ &= -(\mu + \mu^*) \sum_{k,n} \left| \frac{\partial E}{\partial \mathbf{w}_{kn\parallel}} \right|^2 \\ &= -2Re(\mu) \sum_{k,n} \left| \frac{\partial E}{\partial \mathbf{w}_{kn\parallel}} \right|^2 \end{aligned} \tag{27}$$

If the real part of μ is positive, $\Delta E \leq 0$ holds. This indicates that the output error would decrease upon updating weights according to Equations (24) and (26). For calculating the updated quantity in Equation (26), the component $\partial E / \partial \mathbf{w}_{kn}^*$ are expanded by using chain rule of derivative and $\partial \mathbf{y} / \partial \mathbf{w}_{kn}^* = 0$ from the local analytic condition is applied:

$$\begin{aligned}
 \frac{\partial E}{\partial \mathbf{w}_{kn}^*} &= \frac{\partial E}{\partial \mathbf{y}_{k\parallel}^*} \frac{\partial \mathbf{y}_{k\parallel}^*}{\partial \mathbf{w}_{kn}^*} + \frac{\partial E}{\partial \mathbf{y}_k} \frac{\partial \mathbf{y}_k}{\partial \mathbf{w}_{kn}^*} \\
 &= \frac{\partial E}{\partial \mathbf{y}_{k\parallel}^*} \frac{\partial \mathbf{y}_{k\parallel}^*}{\partial \mathbf{w}_{kn}^*} \\
 &= -(\mathbf{d}_k - \mathbf{y}_k) \mathbf{h}' \left(\sum_{n=1}^N \mathbf{w}_{kn}^* \mathbf{x}_n^* \right) \mathbf{x}_n^* \\
 &= \boldsymbol{\delta}_k \mathbf{x}_n^*
 \end{aligned} \tag{28}$$

where \mathbf{h}' is the (local) derivative of the activation function \mathbf{h} , and $\boldsymbol{\delta}_k$ is defined as $-(\mathbf{d}_k - \mathbf{y}_k) \mathbf{h}'(\sum_{n=1}^N \mathbf{w}_{kn}^* \mathbf{x}_n^*)$.

Similarly, the updates for the connection weights \mathbf{v} 's can be deduced. The output error function E is a function with arguments \mathbf{v}_{nm} 's and \mathbf{v}_{nm}^* 's, thus the output error at the time $(t + 1)$ can be represented by

$$\begin{aligned}
 E(t + 1) &= E(\mathbf{v}_{nm}(t + 1), \mathbf{v}_{nm}^*(t + 1)) \\
 &= E(\mathbf{v}_{nm}(t), \mathbf{v}_{nm}^*(t)) + \sum_{n,m} \frac{\partial E}{\partial \mathbf{v}_{nm\parallel}} \Delta \mathbf{v}_{nm} + \sum_{n,m} \frac{\partial E}{\partial \mathbf{v}_{nm\parallel}^*} \Delta \mathbf{v}_{nm}^*
 \end{aligned} \tag{29}$$

Hence, \mathbf{v}_{nm} is updated with the quantity $\Delta \mathbf{v}_{nm}$,

$$\mathbf{v}_{nm}(t + 1) = \mathbf{v}_{nm}(t) + \Delta \mathbf{v}_{nm}, \tag{30}$$

$$\Delta \mathbf{v}_{nm} = -\boldsymbol{\mu} \frac{\partial E}{\partial \mathbf{v}_{nm\parallel}^*} \tag{31}$$

This leads to $\Delta E \leq 0$ under the condition of the real part of $\boldsymbol{\mu}$ being positive. The component $\partial E / \partial \mathbf{v}_{nm\parallel}^*$ can be expanded by chain rules and local analytic conditions, $\partial \mathbf{x} / \partial \mathbf{v}_{\parallel}^* = 0$, $\partial \mathbf{y} / \partial \mathbf{x}_{\parallel}^* = 0$, and $\partial \mathbf{x} / \partial \mathbf{v}_{\parallel}^* = 0$ are applied:

$$\begin{aligned}
 \frac{\partial E}{\partial \mathbf{v}_{nm\parallel}^*} &= \sum_{k=1}^K \left(\frac{\partial E}{\partial \mathbf{y}_{k\parallel}^*} \frac{\partial \mathbf{y}_{k\parallel}^*}{\partial \mathbf{x}_{n\parallel}^*} \frac{\partial \mathbf{x}_{n\parallel}^*}{\partial \mathbf{v}_{nm\parallel}^*} + \frac{\partial E}{\partial \mathbf{y}_{k\parallel}^*} \frac{\partial \mathbf{y}_{k\parallel}^*}{\partial \mathbf{x}_{n\parallel}} \frac{\partial \mathbf{x}_{n\parallel}}{\partial \mathbf{v}_{nm\parallel}^*} \right. \\
 &\quad \left. + \frac{\partial E}{\partial \mathbf{y}_k} \frac{\partial \mathbf{y}_k}{\partial \mathbf{x}_{n\parallel}^*} \frac{\partial \mathbf{x}_{n\parallel}^*}{\partial \mathbf{v}_{nm\parallel}^*} + \frac{\partial E}{\partial \mathbf{y}_k} \frac{\partial \mathbf{y}_k}{\partial \mathbf{x}_{n\parallel}} \frac{\partial \mathbf{x}_{n\parallel}}{\partial \mathbf{v}_{nm\parallel}^*} \right) \\
 &= \sum_{k=1}^K \left(\frac{\partial E}{\partial \mathbf{y}_{k\parallel}^*} \frac{\partial \mathbf{y}_{k\parallel}^*}{\partial \mathbf{x}_{n\parallel}^*} \frac{\partial \mathbf{x}_{n\parallel}^*}{\partial \mathbf{v}_{nm\parallel}^*} \right) \\
 &= \frac{\partial \mathbf{x}_{n\parallel}^*}{\partial \mathbf{v}_{nm\parallel}^*} \sum_{k=1}^K \left(\frac{\partial E}{\partial \mathbf{y}_{k\parallel}^*} \frac{\partial \mathbf{y}_{k\parallel}^*}{\partial \mathbf{x}_{n\parallel}^*} \right)
 \end{aligned}$$

Using the derivatives $\partial \mathbf{x}_{n\parallel}^* / \partial \mathbf{v}_{nm\parallel}^* = \mathbf{g}'(\sum_{m=1}^M \mathbf{v}_{nm}^* \mathbf{z}_m^*) \mathbf{z}_m^*$, $\partial \mathbf{y}_{k\parallel}^* / \partial \mathbf{x}_{n\parallel}^* = \mathbf{h}'(\sum_{n=1}^N \mathbf{w}_{kn}^* \mathbf{x}_n^*) \mathbf{w}_{kn}^*$, and $\partial E / \partial \mathbf{y}_{k\parallel}^* = -(\mathbf{d}_k - \mathbf{y}_k)$, we finally obtain

$$\frac{\partial E}{\partial \mathbf{v}_{nm\parallel}^*} = \left(\sum_{k=1}^K \boldsymbol{\delta}_k \mathbf{w}_{kn}^* \right) \left(\sum_{m=1}^M \mathbf{v}_{nm}^* \mathbf{z}_m^* \right) \mathbf{z}_m^* \tag{32}$$

Once a set of network output $\{y_k\}$ is obtained for a set of network input, the output error with respect to a target set $\{d_k\}$ can be calculated by Equation (23). Then, the connection weights between hidden and output layers are modified by Equations (24–28). The connection weights between input and hidden layers are finally modified by Equations (30–32).

3.3. Universal Approximation Capability

As an example of activation functions for neurons (g and h), the quaternionic tanh function [22] can be used. Other types of activation functions are also available, because complex-valued functions can be used for the presented network and the properties of several functions have been explored for the activation functions in [29]. It is important to consider the capability of the proposed quaternionic network with these activation functions, *i.e.*, whether the proposed network can approximate given functions.

This concern is known as the universal approximation theorem [31,32]. In the real-valued MLPs with single hidden layer, the universality has been proven with a so-called sigmoidal function being used as an activation function of neurons. Other than sigmoidal functions, the bounded and differentiable functions are also available for activation functions.

This theorem is also discussed in the case of complex-valued networks [29]. The condition for boundedness in the real-valued MLPs is not required in some complex-valued MLPs. There are three types of activation functions discussed in [29], which are categorized by the properties of complex-valued functions, and for each of them, it is shown that universal approximation can be achieved.

The first type of complex-valued functions concerns the functions without any singular points. These functions can be used as activation functions and the networks with this type of activation functions are shown as good approximators. Although some of the functions are not bounded, they can be used by introducing bounding operation for their regions. The second type concerns the functions having the bounded singular points, *e.g.*, the discontinuous functions. These singularities can be removed and thus they can also be used for activation functions and can achieve their universality. The last type is for the functions with the so-called essential singularities, *i.e.*, their singularities cannot be removed. These functions can also be used as activation functions, with the consideration of restricting the regions for them so that their regions never cover their singularities.

In the proposed quaternionic MLPs, for example, a quaternionic tanh function can be used as an activation function. This function is unbounded and may contain several kinds of singularities as in the case of complex-valued functions described above. Thus this quaternionic MLPs would face the same problem, *i.e.*, the existence of singularities, but this can also be handled similarly as in the case of complex-valued MLPs for the removal or avoidance of such singularities. It could be possible to show the universality with handling singularities for the proposed MLPs according to the ways adopted in the complex-valued MLPs [29], but it remains as our future work.

4. Conclusions and Discussion

This paper has proposed a multilayer type neural network and an error back-propagation algorithm for its learning scheme in the quaternionic domain. The neurons in this network adopt locally analytic

activation functions. The quaternionic functions with local analytic conditions are isomorphic to the complex functions, thus several activation functions, such as complex-valued tanh function, can be used extendedly in the quaternionic domain. The Wirtinger calculus, where a quaternion and its conjugate are treated as independent of each other, makes the derivation of the learning scheme clear and compact.

Analytic conditions for quaternionic functions are derived by defining a complex plane at a quaternionic point, which is a kind of reduction from quaternionic domain to complex domain. There exists another type of reduction in quaternionic domains, such as the commutative quaternion, which is a four-dimensional hypercomplex number system with commutativity in its multiplication [33–36]. A principal property is that a commutative quaternion can be decomposed and represented by two complex numbers with two linearly independent bases (called the decomposed form by idempotent bases). Commutative quaternions have been applied to neural networks only in terms of Hopfield-type network [24], but defining multilayer perceptron type networks can also be straightforward. Thus, it will be interesting to explore the relationship between commutative quaternion-based networks and networks with the local analyticity.

Showing the universality of the proposed network is also an important issue. Quaternionic functions, such as tanh function, may contain several kinds of singularities where the values of functions or their differentials are not defined in particular regions. In complex-valued networks with fully complex-valued function [29], the universality of the networks can be shown by dealing with these singularities, so that such singularities are removed or avoided by restricting the regions. It is expected to show the universality of the proposed network, in a similar way to the case of complex-valued networks.

Also, it is necessary to investigate the performances of the proposed network, though in this paper the experimental exploration could not be accomplished. The proposed network is similar to the networks proposed in [13,14], due to the introduction of local analytic function in quaternionic domain, thus the performances for both types of networks may have similar tendencies. Performance comparisons can also be conducted between the proposed network and the ones in [29], because both networks adopt the same representation in their constructions, *i.e.*, Wirtinger calculus. A wide variety of activation functions have been investigated including the split-type function, phase-preserving function [37], and circular-type function [38]. Similar experiments should be conducted for the quaternionic networks.

Application of the presented network to engineering problems is also challenging. The processing of three or four dimensional vector data, such as color/multi-spectral image processing, predictions for three-dimensional protein structures, and controls of motion in three-dimensional space, will be the candidates from now on.

Acknowledgments

This study was financially supported by Japan Society for the Promotion of Science (Grant-in-Aids for Young Scientists (B) 24700227 and Scientific Research (C) 23500286).

References

1. Hirose, A. *Complex-Valued Neural Networks: Theories and Application*; World Scientific Publishing: Singapore, 2003.

2. Hirose, A. *Complex-Valued Neural Networks*; Springer-Verlag: Berlin, Germany, 2006.
3. Nitta, T. *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*; Information Science Reference: New York, NY, USA, 2009.
4. Hamilton, W.R. *Lectures on Quaternions*; Hodges and Smith: Dublin, Ireland, 1853.
5. Hankins, T.L. *Sir William Rowan Hamilton*; Johns Hopkins University Press: Baltimore, MD, USA, 1980.
6. Mukundan, R. Quaternions: From classical mechanics to computer graphics, and beyond. In *Proceedings of the 7th Asian Technology Conference in Mathematics*, Melaka, Malaysia, 17–21 December 2002; pp. 97–105.
7. Kuipers, J.B. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*; Princeton University Press: Princeton, NJ, USA, 1998.
8. Hoggar, S.G. *Mathematics for Computer Graphics*; Cambridge University Press: Cambridge, MA, USA, 1992.
9. Nitta, T. An extension of the back-propagation algorithm to quaternions. In *Proceedings of International Conference on Neural Information Processing (ICONIP'96)*, Hong Kong, China, 24–27 September 1996; Volume 1, pp. 247–250.
10. Arena, P.; Fortuna, L.; Muscato, G.; Xibilia, M. Multilayer perceptrons to approximate quaternion valued functions. *Neural Netw.* **1997**, *10*, 335–342.
11. Buchholz, S.; Sommer, G. Quaternionic spinor MLP. In *Proceeding of 8th European Symposium on Artificial Neural Networks (ESANN 2000)*, Bruges, Belgium, 26–28 April 2000; pp. 377–382.
12. Matsui, N.; Isokawa, T.; Kusamichi, H.; Peper, F.; Nishimura, H. Quaternion neural network with geometrical operators. *J. Intell. Fuzzy Syst.* **2004**, *15*, 149–164.
13. Mandic, D.P.; Jahanchahi, C.; Took, C.C. A quaternion gradient operator and its applications. *IEEE Signal Proc. Lett.* **2011**, *18*, 47–50.
14. Ujang, B.C.; Took, C.C.; Mandic, D.P. Quaternion-valued nonlinear adaptive filtering. *IEEE Trans. Neural Netw.* **2011**, *22*, 1193–1206.
15. Kusamichi, H.; Isokawa, T.; Matsui, N.; Ogawa, Y.; Maeda, K. A new scheme for color night vision by quaternion neural network. In *Proceedings of the 2nd International Conference on Autonomous Robots and Agents (ICARA2004)*, Palmerston North, New Zealand, 13–15 December 2004; pp. 101–106.
16. Isokawa, T.; Matsui, N.; Nishimura, H. Quaternionic neural networks: Fundamental properties and applications. In *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*; Nitta, T., Ed.; Information Science Reference: New York, NY, USA, 2009; Chapter XVI, pp. 411–439.
17. Nitta, T. A solution to the 4-bit parity problem with a single quaternary neuron. *Neural Inf. Process. Lett. Rev.* **2004**, *5*, 33–39.
18. Yoshida, M.; Kuroe, Y.; Mori, T. Models of hopfield-type quaternion neural networks and their energy functions. *Int. J. Neural Syst.* **2005**, *15*, 129–135.
19. Isokawa, T.; Nishimura, H.; Kamiura, N.; Matsui, N. Fundamental properties of quaternionic hopfield neural network. In *Proceedings of 2006 International Joint Conference on Neural Networks*, Vancouver, BC, USA, 30 October 2006; pp. 610–615.

20. Isokawa, T.; Nishimura, H.; Kamiura, N.; Matsui, N. Associative memory in quaternionic hopfield neural network. *Int. J. Neural Syst.* **2008**, *18*, 135–145.
21. Isokawa, T.; Nishimura, H.; Kamiura, N.; Matsui, N. Dynamics of discrete-time quaternionic hopfield neural networks. In *Proceedings of 17th International Conference on Artificial Neural Networks*, Porto, Portugal, 9–13 September 2007; pp. 848–857.
22. Isokawa, T.; Nishimura, H.; Matsui, N. On the fundamental properties of fully quaternionic hopfield network. In *Proceedings of IEEE World Congress on Computational Intelligence (WCCI2012)*, Brisbane, Australia, 10–15 June 2012; pp. 1246–1249.
23. Isokawa, T.; Nishimura, H.; Saitoh, A.; Kamiura, N.; Matsui, N. On the scheme of quaternionic multistate hopfield neural network. In *Proceedings of Joint 4th International Conference on Soft Computing and Intelligent Systems and 9th International Symposium on Advanced Intelligent Systems (SCIS & ISIS 2008)*, Nagoya, Japan, 17–21 September 2008; pp. 809–813.
24. Isokawa, T.; Nishimura, H.; Matsui, N. Commutative quaternion and multistate hopfield neural networks. In *Proceedings of IEEE World Congress on Computational Intelligence (WCCI2010)*, Barcelona, Spain, 18–23 July 2010; pp. 1281–1286.
25. Isokawa, T.; Nishimura, H.; Matsui, N. An iterative learning scheme for multistate complex-valued and quaternionic hopfield neural networks. In *Proceedings of International Joint Conference on Neural Networks (IJCNN2009)*, Atlanta, GA, USA, 14–19 June 2009; pp. 1365–1371.
26. Leo, S.D.; Rotelli, P.P. Local hypercomplex analyticity. 1997. Available online: <http://arxiv.org/abs/funct-an/9703002> (accessed on 20 November 2012).
27. Leo, S.D.; Rotelli, P.P. Quaternionic analyticity. *Appl. Math. Lett.* **2003**, *16*, 1077–1081.
28. Schwartz, C. Calculus with a quaternionic variable. *J. Math. Phys.* **2009**, *50*, 013523:1–013523:11.
29. Kim, T.; Adalı, T. Approximation by fully complex multilayer perceptrons. *Neural Comput.* **2003**, *15*, 1641–1666.
30. Wirtinger, W. Zur formalen theorie der funktionen von mehr komplexen veränderlichen. *Math. Ann.* **1927**, *97*, 357–375.
31. Cybenko, G. Approximations by superpositions of sigmoidal functions. *Math. Control Signals Syst.* **1989**, *2*, 303–314.
32. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Netw.* **1991**, *4*, 215–257.
33. Segre, C. The real representations of complex elements and extension to bicomplex systems. *Math. Ann.* **1892**, *40*, 322–335.
34. Catoni, F.; Cannata, R.; Zampetti, P. An Introduction to commutative quaternions. *Adv. Appl. Clifford Algebras* **2006**, *16*, 1–28.
35. Davenport, C.M. A commutative hypercomplex algebra with associated function theory. In *Clifford Algebra With Numeric and Symbolic Computation*; Ablamowicz, R., Ed.; Birkhauser: Boston, MA, USA, 1996; pp. 213–227.
36. Pei, S.C.; Chang, J.H.; Ding, J.J. Commutative reduced biquaternions and their Fourier Transform for signal and image processing applications. *IEEE Trans. Signal Proc.* **2004**, *52*, 2012–2031.
37. Hirose, A. Continuous complex-valued back-propagation learning. *Electron. Lett.* **1992**, *28*, 1854–1855.

38. Georgiou, G.M.; Koutsougeras, C. Complex domain backpropagation. *IEEE Trans. Circuits Syst. II* **1992**, *39*, 330–334.

© 2012 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/>).