On Processing Three Dimensional Data by Quaternionic Neural Networks

Noriyuki Muramoto, Teijiro Isokawa, Haruhiko Nishimura, and Nobuyuki Matsui

Abstract—The performance of layered neural networks with quaternionic encoding variables is investigated in this paper. The form of local analyticity with Wirtinger representation is adopted for a backpropagation learning algorithm in this network. A quaternionic version of tanh function is used for the activation function in neuron states’ updates. As tasks of the performance evaluation of the presented networks, two types of three dimensional data processing problem are used; the prediction of the Lorentz attractor and affine transformations in three dimensional space.

I. INTRODUCTION

QUATERNION is a four-dimensional hypercomplex number system discovered by Hamilton. This is extensively used in several fields, such as modern mathematics, physics, computer graphics, and so on [1], [2]. One of the benefits by the use of quaternions is that it can treat and operate three or four dimensional vector as one entity, so the effective information processing can be achieved by the operations for quaternionic variables. In this respect, there have been a growing number of studies concerning the introduction of quaternions into neural networks. They cover from a single quaternionic neuron [3] to networks such as multilayer perceptron models with the applications [4]–[8] and the Hopfield-type networks [7], [9], [10].

There are only a few investigations concerning the activation functions in quaternionic domain. The most popular function is so-called “split type” activation function, in which a real valued function is applied to each of elements of a quaternion [11], [12]. However, this function is a not analytic one in the quaternionic domain. Recently, another type of quaternionic function has been proposed by introducing the idea of local analyticity in quaternionic domain [13], [14], and quaternionic neural networks based on this have also been proposed [8], [15], [16]. In these networks, a complex plane is locally defined in the quaternion space, and quaternionic activation functions can be used as complex-valued ones for updating the states of neurons. This implies that various types of complex-valued functions can be used also for the quaternionic neural networks.

In [16], a quaternionic multilayered neural network is proposed and a backpropagation algorithm as its learning scheme for this network is formulated. This network has the same basis, i.e. local analyticity, to the network [8], but the formulation of the learning scheme is based on Wirtinger derivatives [17]. It is theoretically proven that the learning in this network can be performed, however, the learning capabilities on the practical problems remain unexplored. In this paper, the performances of this quaternionic network are experimentally investigated. As tasks for the quaternionic network, three dimensional data processings are adopted, such as the prediction for the Lorenz system that consists of three variables/equations with chaotic behaviors, learning affine transformations.

II. PRELIMINARIES

A. Quaternionic Algebra

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 \( \mathbf{x} \) in a four-dimensional vector space,

\[
\mathbf{x} = x^e + x^i \mathbf{i} + x^j \mathbf{j} + x^k \mathbf{k}
\]

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 \( k \).

Eq.(1) can also be written using 4-tuple or 2-tuple notation as

\[
\mathbf{x} = (x^e, x^i, x^j, x^k) = (x^e, \bar{x}),
\]

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

\[
\mathbf{x}^* = (x^e, -\bar{x}) = x^e - (x^i \mathbf{i} + x^j \mathbf{j} + x^k \mathbf{k}).
\]

Quaternion bases satisfy the following identities,

\[
i^2 = j^2 = k^2 = ijk = -1, \]

\[
i j = -ji = k, \quad jk = -kj = i, \quad ki = -ik = j, \quad (5)
\]

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 \( p = (p^e, \bar{p}) = (p^e, p^i, p^j, p^k) \) and \( q = (q^e, \bar{q}) = (q^e, q^i, q^j, q^k) \). The addition and subtraction of quaternions are defined in a similar manner as for complex-valued numbers or vectors, i.e.,

\[
\mathbf{p} \pm \mathbf{q} = (p^e \pm q^e, \mathbf{p} \pm \mathbf{q})
\]

where

\[
(p^e \pm q^e, p^i \pm q^i, p^j \pm q^j, p^k \pm q^k).
\]
The product of \( p \) and \( q \) is determined by Eq.(5) as
\[
pq = (p^\ell q^\ell) - \vec{p} \cdot \vec{q}, \quad (p^\ell q + q^\ell \vec{p} + \vec{p} \times \vec{q}),
\]
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
\[
(pq)^* = q^* p^*.
\]

The quaternion norm of \( x \), denoted by \(|x|\), is defined as
\[
|x| = \sqrt{xx^*} = \sqrt{x(\cdot)^2 + x(i)^2 + x(j)^2 + x(k)^2}.
\]

B. Local Analyticity on Quaternion

The Cauchy-Riemann-Fueter (CRF) equation, which is the analytic condition for the quaternionic domain, yields that only linear functions and constant can satisfy this condition [8], [13], [18], thus alternative approaches to assure analyticity have been explored. One approach is called local analyticity, which is different from the standard analyticity, i.e. global analyticity. In the following, local derivatives and analytic condition with the Wirtinger representation are briefly described [16].

A quaternion \( x \) can be alternatively represented as:
\[
x = x(\cdot) + u_x r, \quad r = \sqrt{x(\cdot)^2 + x(i)^2 + x(j)^2 + x(k)^2}, \quad u_x = \frac{x(\cdot)i + x(i)j + x(j)k}{r}.
\]

The numbers of neurons in the input, hidden, and the output layer are the same as input \( \mathbf{z} \). In the hidden layer, each neuron takes the weighted sum of the output signals. 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 \(x_n\), is determined by
\[
x_n = g \left( \sum_{m=1}^{M} v_{nm} z_m \right),
\]
where \(g\) is a quaternionic activation function introducing non-linearity between the action potential and output in the neuron. 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
\[
y_k = h \left( \sum_{n=1}^{N} w_{kn} x_n \right),
\]
where the function \(h\) is a quaternionic activation function from the action potential to the output, and the \(w_{kn}\) is a connection weight between the \(n\)-th neuron in the hidden layer and the \(k\)-th neuron in the output layer.

2) Back-propagation algorithm: A learning scheme for the presented network, so-called Error Back-Propagation algorithm, is recapitulated [16]. Let \(d_k\) be a quaternionic desired signal for the \(k\)-th output neuron for the \(z\) that are inputs to the network. The connection weights affects the output signals with respect to a set of input signals, thus the error \(E\) is regarded as a function with arguments \(w_{kn}\)’s and \(w_{kn}’s\). The output error \(E\) at the time \(t\) is then defined as
\[
E(t) = \sum_{k=1}^{K} (d_k - y_k(t)) (d_k - y_k(t))^*,
\]

Suppose that the connection weights are updated at the time \((t+1)\) by
\[
w_{kn}(t+1) = w_{kn}(t) + \Delta w_{kn},
\]
where \(\Delta w_{kn}\) is an updating quantity. The output error at the time \((t+1)\) can be written as
\[
E(t+1) = E(w_{kn}(t), v_{nm}^*(t)) + \sum_{k,n} \frac{\partial E}{\partial w_{kn}^*} \Delta w_{kn}^* + \sum_{k,n} \frac{\partial E}{\partial w_{kn}} \Delta w_{kn},
\]
Thus, if we set \(\Delta w_{kn}\) as
\[
\Delta w_{kn} = -\mu \frac{\partial E}{\partial w_{kn}^*},
\]
where \(\mu\) is a quaternionic constant. The temporal difference of the output error, \(\Delta E = E(t+1) - E(t)\), can be calculated:
\[
\Delta E = -2 \text{Re} (\mu) \sum_{k,n} \left| \frac{\partial E}{\partial w_{kn}} \right|^2.
\]

For calculating the updated quantity in Eq. (20), the component \(\partial E/\partial w_{kn}^*\) are expanded by using chain rule of derivative and \(\partial y/\partial w^*_{kn} = 0\) from the local analytic condition is applied:
\[
\frac{\partial E}{\partial w_{kn}^*} = -(d_k - y_k) h' \left( \sum_{n=1}^{N} w_{kn}^* x_n^* \right) x_n^* = \delta_k x_n^*,
\]
where \(h'\) is the first order derivative function for the activation function \(h\). \(\delta_k\) is defined as \(-(d_k - y_k) h' (\sum_{n=1}^{N} w_{kn}^* x_n^*)\).

The update quantities for the connection weights \(v\)’s can be obtained in a similar way to \(w\)’s. The output error at the time \(t\) can be represented by
\[
E(t+1) = E(v_{nm}(t), v_{nm}^*(t)) + \sum_{n,m} \frac{\partial E}{\partial v_{nm}^*} \Delta v_{nm} + \sum_{n,m} \frac{\partial E}{\partial v_{nm}} \Delta v_{nm}^*,
\]
\(v_{nm}\) is updated with the quantity \(\Delta v_{nm}\),
\[
v_{nm}(t+1) = v_{nm}(t) + \Delta v_{nm},
\]
\[
\Delta v_{nm} = -\mu \frac{\partial E}{\partial v_{nm}^*}.
\]
The component \(\partial E/\partial v_{nm}^*\) can be expanded by chain rules and local analytic conditions, \(\partial x/\partial v_{nm}^* = 0\), \(\partial y/\partial v_{nm}^* = 0\), and \(\partial x/\partial v_{nm} = 0\) are applied. Finally,
\[
\frac{\partial E}{\partial v_{nm}^*} = \left( \sum_{k=1}^{K} \delta_k w_{kn}^* \right) g' \left( \sum_{m=1}^{M} v_{nm}^* z_m^* \right) z_m^*,
\]
can be obtained. Thus the connection weights can be updated according to the output error \(E\).

In this paper, a quaternionic \(\text{tanh}\) function is applied for the activation functions, i.e. \(g'()\) and \(h'()\). As in the case of complex-valued \(\text{tanh}\) function, it is necessary to consider the singular points scattered in the domain of this function. There exist several types of singular points, that are analyzed in [19], and a quaternionic \(\text{tanh}\) function could be categorized in the essential singularities that cannot be removed, as in the case of complex-valued \(\text{tanh}\) function. This type of functions can also be used as activation functions, with restricting the regions for their domains so that their regions never cover their singularities.

In addition to the singularities, it is also necessary to consider the condition of local analyticity in using the presented networks. This can be conducted by choosing appropriate values in the learning coefficient \(\mu\).

III. EXPERIMENTS ON THREE-DIMENSIONAL DATA PROCESSING

The performances of the presented quaternionic neural networks are investigated in this section. The tasks for these networks are the prediction of three-dimensional chaotic signals called the Lorenz system and the learning of affine (geometrical) transformation in three-dimensional space.
where \( x, y, \) and \( z \) are the states of the system with time \( t \), and \( \sigma, \rho, \) and \( \beta \) are the parameters of the system. This system shows chaotic behaviors for particular sets of parameters, e.g. for a set of parameters \( \sigma = 10, \rho = 28, \) and \( \beta = 8/3, \) the system has two fixed attractors, called the Lorenz attractors.

To predict the states of the system, a three-layered quaternionic neural network is prepared in which the numbers of neurons in the input, hidden, and output layers are one, four, and one, respectively. The network is trained in an online manner so that the prediction state at time \( t + 1 \) is come up from the predicted state at time \( t \) presented on the input of the network. A set of 1000 training data is prepared from a Lorenz system evolved by Euler method with the time step 0.004 and with the initial configuration of \( x = 10.0, y = 12.0, \) and \( z = 15.0. \) Each component for the input data is normalized in the range \([-0.3, 0.3]. \) For training a network, \( \mu = (0.5, 0.1, 0.1, 0.1) \) is used for the learning coefficient.

Figure 1 shows a squared error with respect to the learning epoch, which is measured at the output layer of the network. The estimated error decreases with the learning epoch, thus the prediction of chaotic signals can be successfully conducted. A trajectory for the predicted states for the system, with the training data, is shown in Fig. 2. Also from this result, the network can predict the system states with two attractors.

B. Affine transformations

Three kinds of the affine transformations, such as enlargement, shift, and rotation, are used for learning tasks for quaternionic neural networks. In these tasks, a set of three-dimensional coordinates are used for the input to the network, and their enlarged, shifted, or rotated coordinates constitute to the desired output for the network. Figure 3 shows the learning data sets. For an example, in Fig. 3(a), there are nine points in an x-y plane with \( z = 0.4, \) which are used for the input to the network, and there are also nine points in an x-y plane with \( z = 0.2 \) for the desired output. The network is trained so that the transformation \( z \rightarrow z - 0.2 \) can be conducted without changing \( x \) and \( y \) coordinates. Note that, the training data sets are distributed in a plane, i.e. two-dimensional space, but the data sets for the testing the networks are three-dimensional.

The network contains three layers, and the numbers of the input, hidden, and output layers are one, four, and one, called a \( 1 - 4 - 1 \) network. The learning coefficient is set to \( \mu = (5 \times 10^{-2}, 1.0 \times 10^{-3}, 1.0 \times 10^{-3}, 1.0 \times 10^{-3}). \) The number of epochs for learning is set to 200,000. For the learning data sets, each coordinates is normalized in the range \([0, 0.3]. \)

The network outputs for these tasks are shown in Fig. 4. It is shown that the network can learn each task of affine transformations though some transformation errors still remain. The main reason for these transformation errors may be from ambiguousness in the training sets, rather than the learning capabilities of the networks. For an example, the training set in the enlarging task, the outputs from the network are slightly shifted as well as enlargement of the input data (Fig. 4(b)). In the training data for this task, as shown in Fig. 3(b), only the point at \((0.5, 0.5, 0.5), \) which is the origin in enlargement, is common in the input and desired data sets, and the other eight points in the desired output set are far from the input data. Thus the network is trained so that the input points are put in shifted coordinates rather than the enlarged coordinates.

IV. CONCLUSION

In this paper, the performances of the quaternionic neural networks are investigated through the task of processing of three-dimensional data. The learning algorithm defined for the presented network adopts local analyticity that introduces derivative operators in quaternionic domain. The derivative is defined on a complex plane at a quaternionic point, thus the functions isomorphic to the complex functions can be used for activation function of neurons. The derivation of learning
algorithm, i.e. error back-propagation algorithm, has shown briefly by using the local analyticity.

The learning tasks for the presented network are the prediction of the Lorenz system of which the state is represented in three variables and the affine transformation in three dimensional space. It is shown that for these tasks the network can learn the input-output relations.

There are several types of quaternionic neural networks such as split-type [4]–[6] and fully-type [8]. A comprehensive comparisons with these networks will be important. Also, explorations of the presented network in other types of applications, such as image processing tasks [6], [7], are very challenging problems. These remain for our future work.

REFERENCES


