Signal Processing 136 (2017) 59–68

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Feed forward neural network with random quaternionic neurons

Toshifumi Minemoto[a,*], Teijiro Isokawa[a], Haruhiko Nishimura[b], Nobuyuki Matsui[a]

[a] *Graduate School of Engineering, University of Hyogo, 2167 Shosha, Himeji 671-2280, Japan*
[b] *Graduate School of Applied Informatics, University of Hyogo, Computational Science Center Building 5-7F 7-1-28 Minatojima-minamimachi, Chuo-ku, Kobe, Hyogo 650-0047, Japan*

ARTICLE INFO

Keywords:
Quaternion
Feed forward neural network
Extreme learning machine
Classification
Autoencoder
Affine transformation

ABSTRACT

A quaternionic extension of feed forward neural network, for processing multi-dimensional signals, is proposed in this paper. This neural network is based on the three layered network with random weights, called Extreme Learning Machines (ELMs), in which iterative least-mean-square algorithms are not required for training networks. All parameters and variables in the proposed network are encoded by quaternions and operations among them follow the quaternion algebra. Neurons in the proposed network are expected to operate multi-dimensional signals as single entities, rather than real-valued neurons deal with each element of signals independently. The performances for the proposed network are evaluated through two types of experiments: classifications and reconstructions for color images in the CIFAR-10 dataset. The experimental results show that the proposed networks are superior in terms of classification accuracies for input images than the conventional (real-valued) networks with similar degrees of freedom. The detailed investigations for operations in the proposed networks are conducted.

1. Introduction

Processing multi-dimensional signals, such as color images, is an important problem in artificial neural networks. Artificial neural networks consist of many neurons, interconnected to each other, that accept only real-valued signals for their input, internal states, and output. Of course, these neural networks cope with high dimensional signals by configuring neurons so that each of them covers each element in these signals. But this type of configuration would be unnatural because each of elements in multi-dimensional signals is not independent to each other and these signals should be processed as a single entity. Thus, for over two decades, applications of complex values to neural networks have been extensively investigated, as summarized in the references [1–3]. Besides these studies, neural networks with dimensions more than two have also been explored: one motivation is inspired by a natural extension from real-valued neural networks to complex-valued ones. Another motivation and necessity arise from engineering applications in which multi-dimensional signals, such as three-dimensional components in color images (red, green, and blue) or body coordinates in three dimensional space (X, Y, Z), should be processed. Although neural networks for these applications can be composed by real-valued or complex-valued neurons, it would be useful to introduce a number system with high dimensions, the so-called hypercomplex number systems.

Quaternion is a four-dimensional hypercomplex number system introduced by W.R. Hamilton [4,5]. This number system has been extensively employed in the fields of modern mathematics, physics, control of satellites, computer graphics, signal processing, and so on [6–10]. One of the benefits provided by quaternions is that operators in quaternions efficiently accomplish the affine transformations in three-dimensional space, especially spatial rotations, with their compact representations. Thus, it is expected that neurons with quaternionic representation and operations would be useful for processing three- and four-dimensional signals.

Feed forward neural networks, or multilayer perceptron (MLP) neural networks, are most popular neural networks where input-output relations can be constructed by adjusting the connection weights in the network. Adjusting process is also called learning and typically incorporates least-mean-square (LMS) method. Feed forward neural network models based on quaternions have been explored in [11–18].

Applications of quaternionic neural networks and quaternionic LMS have also been explored, such as control problem [12], signal classification [14], color image processing [13,19], prediction of chaotic time series [20,21,15,17], forecasting three-dimensional wind signals [15,17]. Error back-propagation algorithms for training neural networks and LMS methods need to calculate gradients in the error surface in multi-dimensional space, spanned by connection weights or filter coefficients, in order to minimize the output error. In quaternio-

* Corresponding author.
E-mail address: toshifumi.minemoto@gmail.com (T. Minemoto).

http://dx.doi.org/10.1016/j.sigpro.2016.11.008
Received 30 April 2016; Received in revised form 4 November 2016; Accepted 7 November 2016
Available online 14 November 2016

nic domain, it is necessary to develop gradient operators with satisfying analytic (differentiable) conditions, as in the domains of real values or complex values. Despite the Cauthy-Riemann-Fueter (CRF) equation claims that only linear functions or constants are analytic in the quaternionic domain, other classes of analytic conditions and derivatives have been developed [22,23,16,24].

Recently another type of feed forward neural networks with their training algorithms have been a focus of attention [25–27].

Called Extreme Learning Machine (ELM), this type of networks does not require gradient-base iterative LMS methods for their training. ELMs are typically three-layered networks, i.e., one input layer with neurons that accept external signals, one output layer with neurons in which output signals can be obtained, and one hidden layer with neurons that interconnect neurons in the input layer and ones in the output layer. Some parts of connection weights in the network are fixed randomly and remaining weights are calculated by solving a so-called least squares optimization problem. This one-shot learning scheme has advantages of fast calculation and of no gradient operators. There are several kinds of extensions investigated, such as many layered network (so-called deep learning) [28], ELM autoencoder [29,30], regularized ELM [31], time series prediction [32], and complex-valued network [33].

Motivated by the work for complex-valued ELM [33], we present a quaternionic extension of ELM, called Q-ELM, in this paper. Our Q-ELM is a three-layered network where all parameters, such as inputs, outputs, and connection weights, are encoded by quaternions, and operators in calculating neurons' states follow quaternion algebra. The basic structure of the proposed Q-ELM is the same as the existing quaternionic multilayer perceptron networks [12], and also has similar features in quaternionic adaptive filters and quaternionic Kalman filters [34]. Also, the proposed network resembles so-called quaternionic echo state network [35] for the point that a part of connection weights are randomly chosen and the rest of weights is determined by learning algorithms. But the proposed Q-ELM does not utilize gradients along the error surface produced by the networks, like error back-propagation algorithm. The performances of our Q-ELM are evaluated through the classification and autoencoding tasks on CIFAR-10 dataset [36]. These evaluations include the comparisons with the conventional (real-valued) ELMs and the quaternionic multilayer perceptron network with a gradient-based learning algorithm [12].

This paper is organized as follows. In Section 2, we first recapitulate the conventional ELM model and its learning algorithm. Quaternionic extension of ELM, Q-ELM, is described in Section 3 with the basic of quaternion algebra. Two types of experimental results on CIFAR-10 dataset are given in Section 4. In Section 5, we discuss the superior performances for Q-ELM with the results for several tasks. We finish with conclusions in Section 6.

2. Preliminaries

We first recapitulate a learning algorithm for single hidden layer feed forward neural networks called the Extreme learning machine (ELM) [25,26]. ELM is a layered network with three layers, i.e., one input layer, one hidden layer, and one output layer. Each layer has neurons. The output of a neuron in the input layer connects to the neurons' inputs in the hidden layer and the output of a neuron in the hidden layer connects to the neurons' inputs in the output layer. There are no connections among neurons within each layer.

We assume the network trains a series of N samples $\{\boldsymbol{x}_i, \boldsymbol{t}_i\}$, $i = 1, 2, \ldots, N$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is a d-dimensional real-valued input signal and $\boldsymbol{t}_i \in \mathbb{R}^m$ is an m-dimensional real-valued signal that is the desired output signal for the input signal. The output of ELM, denoted by $\boldsymbol{y}_i \in \mathbb{R}^m$, is given as

$$\boldsymbol{y}_i = \sum_{j=1}^{L} f(\boldsymbol{x}_i, \boldsymbol{w}_j, b_j)\boldsymbol{\beta}_j, \quad i = 1, \ldots, N, \tag{1}$$

where $\boldsymbol{w}_j \in \mathbb{R}^d$ is a set of connection weights between the j-th neuron in the hidden layer and each neuron in the input layer, b_j is the bias for j-th neuron in hidden layer, and $\boldsymbol{\beta}_j = [\beta_{j1}, \beta_{j2}, \ldots, \beta_{jm}]^{\mathrm{T}}$ is a set of connection weights between the j-th neuron in the hidden layer and each neuron in the output layer. The function $f(\cdot)$ denotes a nonlinear activation function for neurons in the hidden layer, such as a sigmoidal function given by

$$f(\boldsymbol{x}, \boldsymbol{w}, b) = \frac{1}{1 + \exp(-\boldsymbol{w}\cdot\boldsymbol{x} + b)}. \tag{2}$$

Namely, the output of ELM is a weighted sum of L non-linear mapping for the weighted input signals. A network with desired input-output relations according to training samples can be obtained by appropriately configuring connection weights.

A learning algorithm for ELM, i.e., a scheme for setting connection weights, is described. The connection weights $\boldsymbol{w}$ and the biases b in ELM are determined by uniform random values at the first stage and never changed, thus the learning is accomplished by setting the values in $\boldsymbol{\beta}$ to produce desired output signals. This can be formulated by solving the following least squares norm minimization problem:

$$\min_{\boldsymbol{B}} \quad \| \boldsymbol{HB} - \boldsymbol{T} \|^2, \tag{3}$$

where

$$\boldsymbol{H} = \begin{bmatrix} f(\boldsymbol{x}_1, \boldsymbol{w}_1, b_1) & \ldots & f(\boldsymbol{x}_1, \boldsymbol{w}_L, b_L) \\ \vdots & \ddots & \vdots \\ f(\boldsymbol{x}_N, \boldsymbol{w}_1, b_1) & \ldots & f(\boldsymbol{x}_N, \boldsymbol{w}_L, b_L) \end{bmatrix}_{N\times L}, \tag{4}$$

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{\beta}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\beta}_L^{\mathrm{T}} \end{bmatrix}_{L\times m}, \quad \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{t}_N^{\mathrm{T}} \end{bmatrix}_{N\times m}, \tag{5}$$

and $\|\cdot\|$ denotes the Frobenius norm. The problem in Eq. (3) is convex and thus the optimal solution $\hat{\boldsymbol{B}}$ is given by

$$\hat{\boldsymbol{B}} = \boldsymbol{H}^{\dagger}\boldsymbol{T}, \tag{6}$$

where $\boldsymbol{H}^{\dagger}$ indicates the Moore-Penrose pseudo inverse matrix of $\boldsymbol{H}$. There are several methods to calculate $\hat{\boldsymbol{B}}$, such as orthogonal projection method and singular value decomposition. In this paper, we employ a closed-form solution which is defined as follows:

$$\hat{\boldsymbol{B}} = \begin{cases} \boldsymbol{H}^{\mathrm{T}}(\boldsymbol{HH}^{\mathrm{T}})^{-1}\boldsymbol{T} & (N \le L) \\ (\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{T} & (N > L) \end{cases}. \tag{7}$$

When an ELM is used for classification problems, it is necessary to encode classes as desired output signals with respect to input signals. In this paper, m neurons are prepared in the output layer for a multiclass classification problem with m classes. In the case of the input signal $\boldsymbol{x}_i$ that belongs to the class $k_i \in \{1, \ldots, m\}$, the encoded desired output signal $\boldsymbol{t}_i$ is given as

$$\boldsymbol{t}_i = (t_{i1}, \ldots, t_{im})^{\mathrm{T}}, \quad t_{ij} = \begin{cases} 1 & j = k_i \\ 0 & j \ne k_i \end{cases}. \tag{8}$$

From the inputs and their corresponding desired output signals, the network can be trained. Also, it is necessary to determine the class from the neurons' outputs in the output layer after training. In this paper, a predicted class from an input $\boldsymbol{x}_i$ is given as

$$\text{Class for } \boldsymbol{x}_i = \operatorname*{argmax}_{j=1,\ldots,m} y_{ij}, \tag{9}$$

where y_{ij} denotes an output signal of the j-th neuron in the output layer.