Neural Information Processing - Letters and Reviews Vol.10, No.10, October 2006

LETTER

Three-Dimensional Vector Valued Neural Network and its Generalization Ability

Tohru Nitta

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

(Submitted on May 29, 2006; Accepted on September 25, 2006)

Abstract - This letter introduces a novel neural network whose input and output signals, and threshold values are all 3-dimensional real-valued vectors and whose weights are all 3-dimensional orthogonal matrices, and the related back-propagation learning algorithm. The algorithm allows new spatial characteristics to be treated.

Keywords - back-propagation learning algorithm, three-dimensional vector, figure transformation, generalization, orthogonal matrix

1. Introduction

Complex-valued neural networks whose parameters (weights and threshold values) are all complex numbers, are useful in fields dealing with complex numbers such as telecommunications, speech recognition and image processing with Fourier transformation. Indeed, we can find some applications of complex-valued neural networks to various fields in the literature [1]. On the other hand, three-dimensional vectors are often used in the engineering field, for example, an object in 3D space can be represented by a 3D vector consisting of width, breadth and height. Thus, neural network models dealing with three signals as one cluster are desired.

In this letter, we propose a novel neuron model whose input and output signals, and threshold values are all 3D real-valued vectors, and whose weights are all 3D orthogonal matrices, and derive a three-dimensional vector version of the back-propagation algorithm (3DV-BP) which can be applied to multi-layered neural networks consisting of the above three-dimensional vector valued neurons. This new algorithm is a natural extension of the complex-valued back-propagation learning algorithm (Complex-BP) [2, 3] which can be applied to multi-layered neural networks whose input and output signals, weights and threshold values are all complex numbers. Furthermore, we show in computational experiments that the 3DV-BP network has the ability to learn 3D affine transformations, whereas the neural network trained with the usual real-valued back-propagation learning algorithm (called Real-BP) [4] does not. The ability of the 3DV-BP network to learn 3D affine transformations corresponds to the ability of the Complex-BP network to learn 2D affine transformations (i.e., the ability to transform geometric figures) [2, 3] which has been applied to computer vision [6].

2. Three-Dimensional Vector Valued Neural Network

There appear to be several approaches for extending the standard real-valued neuron [4] or the complex-valued neuron [2, 3] to higher dimensions. One approach is to extend the number field, i.e., from real numbers x (one dimension), to complex numbers $z = x + iy$ (two dimensions), to quaternions $q = a + ib + jc + kd$ (four dimensions), to sedenions (sixteen dimensions), $\cdots$ [5]. Another approach is to extend the dimensionality of the threshold values and weights from two dimensions to three dimensions using 3D real-valued vectors. We use the latter approach in this letter.

The three-dimensional vector valued neuron is defined as follows. The input and output signals, and thresholds are all 3D real-valued vectors, and the weights are all 3D orthogonal matrices. The activity $\mathbf{A_j}$ of neuron j is defined to be $\mathbf{A_j} = \sum_{\mathbf{k}} \mathbf{W_{jk}S_k} + \mathbf{T_j}$ where $\mathbf{W_{jk}}$ is the 3D orthogonal weight matrix connecting neurons j and k, $\mathbf{S_k}$ is the 3D real-valued vector input signal coming from the output of neuron k, and $\mathbf{T_j}$ is the 3D real-valued vector threshold value of neuron j. The output signal $F(\mathbf{A_j})$ is defined to be:

$$F(\mathbf{A_j}) \quad = \quad \begin{bmatrix} f(a_1) \\ f(a_2) \\ f(a_3) \end{bmatrix}, \text{ where } \mathbf{A_j} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \text{ and } f(a_s) = \frac{1}{1+\exp(-a_s)}. \tag{1}$$

In the above formulation, various restrictions can be imposed on the 3D matrix, e.g. it can be regular, symmetric or orthogonal, which will influence the behavioral characteristics of the neuron. In this letter, the weights are assumed to be orthogonal matrices because this assumption is a natural extension of the weights used in the Complex-BP network [2, 3]. We demonstrate this natural extension as follows. Consider a n-input complex-valued neuron with weights $w_k = w_k^r + iw_k^i \in \mathbf{C}^1$ $(1 \le k \le n)$ ($\mathbf{C}$ denotes the set of complex numbers, $i = \sqrt{-1}$) and a threshold value $\theta = \theta^r + i\theta^i \in \mathbf{C}^1$. Given an input pattern $x_k + iy_k \in \mathbf{C}^1$ $(1 \le k \le n)$, the neuron generates a complex-valued output value $X + iY$, where

$$\begin{bmatrix} X \\ Y \end{bmatrix} = F_C\left(\left[\begin{array}{cc|c|cc} w_1^r & -w_1^i & \cdots & w_n^r & -w_n^i \\ w_1^i & w_1^r & \cdots & w_n^i & w_n^r \end{array}\right] \left[\begin{array}{c} x_1 \\ y_1 \\ \hline \vdots \\ \hline x_n \\ y_n \end{array}\right] + \begin{bmatrix} \theta^r \\ \theta^i \end{bmatrix} \right) \tag{2}$$

where $F_C({}^t[x \;\; y]) = {}^t[f(x) \;\; f(y)]$. In Eq.(2), $\begin{bmatrix} w_k^r & -w_k^i \\ w_k^i & w_k^r \end{bmatrix}$ is an element of the two-dimensional orthogonal group $O_2(\mathbf{R})$ (where $\mathbf{R}$ denotes the set of real numbers). Then, by extending the 2D orthogonal matrix $\begin{bmatrix} w_k^r & -w_k^i \\ w_k^i & w_k^r \end{bmatrix} \in O_2(\mathbf{R})$ to a 3D orthogonal matrix and ${}^t[X \;\; Y], {}^t[x_k \;\; y_k], {}^t[\theta^r \;\; \theta^i]$ in Eq. (2) to 3D vectors, the 3D vector valued neuron described above can be obtained. Thus, the formulation of the 3D vector valued neuron is natural.

Next, we introduce a multi-layered 3D vector valued neural network consisting of the 3D vector valued neuron described above, which is used in the simulation of the next section. It has three layers, for the sake of simplicity. We use $\mathbf{W}_{ji} \in O_3(\mathbf{R})$ for the weight between the input neuron i and the hidden neuron j (where $O_3(\mathbf{R})$ denotes the three-dimensional orthogonal group.), $\mathbf{V}_{kj} \in O_3(\mathbf{R})$ for the weight between the hidden neuron j and the output neuron k, $\mathbf{\Theta_j} = {}^{\mathbf{t}}[\theta_{\mathbf{j}}^{\mathbf{x}} \;\; \theta_{\mathbf{j}}^{\mathbf{y}} \;\; \theta_{\mathbf{j}}^{\mathbf{z}}] \in \mathbf{R^3}$ for the threshold of the hidden neuron j, $\mathbf{\Gamma}_k = {}^t[\gamma_k^x \;\; \gamma_k^y \;\; \gamma_k^z] \in \mathbf{R^3}$ for the threshold of the output neuron k. Let $\mathbf{I_i} = {}^{\mathbf{t}}[\mathbf{I_i^x} \;\; \mathbf{I_i^y} \;\; \mathbf{I_i^z}] \in \mathbf{R^3}$ denote the input signal to the input neuron i, and let $\mathbf{H}_j = {}^t[H_j^x \;\; H_j^y \;\; H_j^z] \in \mathbf{R^3}$ and $\mathbf{O}_k = {}^t[O_k^x \;\; O_k^y \;\; O_k^z] \in \mathbf{R^3}$ denote the output signals of the hidden neuron j, and the output neuron k, respectively. Let $\mathbf{Delta}^k = {}^t[\delta_k^x \;\; \delta_k^y \;\; \delta_k^z] = \mathbf{T}_k - \mathbf{O_k} \in \mathbf{R^3}$ denote the error between $\mathbf{O_k}$ and the target output signal $\mathbf{T}_k = {}^t[T_k^x \;\; T_k^y \;\; T_k^z] \in \mathbf{R^3}$ of the pattern to be learned for the output neuron k. We express the weights as:

$$\mathbf{W}_{ji} \quad = \quad \begin{bmatrix} w_{ji}^x & -w_{ji}^y & 0 \\ w_{ji}^y & w_{ji}^x & 0 \\ 0 & 0 & w_{ji}^z \end{bmatrix} \in O_3(\mathbf{R}), \qquad \text{where } w_{ji}^z = \sqrt{(w_{ji}^x)^2 + (w_{ji}^y)^2}, \tag{3}$$

$$\mathbf{V}_{kj} \quad = \quad \begin{bmatrix} v_{kj}^x & 0 & 0 \\ 0 & v_{kj}^y & -v_{kj}^z \\ 0 & v_{kj}^z & v_{kj}^y \end{bmatrix} \in O_3(\mathbf{R}), \qquad \text{where } v_{kj}^x = \sqrt{(v_{kj}^y)^2 + (v_{kj}^z)^2}. \tag{4}$$

$\mathbf{W}_{ji}$ in Eq. (3) denotes a rotation about the Z-axis, and $\mathbf{V}_{kj}$ in Eq. (4) a rotation about the X-axis in the 3D space. We then derived a 3D vector version of the back-propagation algorithm (*3DV-BP*) for the multi-layered 3D vector neural network described above using a steepest descent method. For a sufficiently small learning constant $\varepsilon > 0$, the weights and the thresholds should be modified according to the following equations.

$$\begin{bmatrix} \Delta v_{kj}^{y} \\ \Delta v_{kj}^{z} \end{bmatrix} = \begin{cases} \begin{bmatrix} (v_{kj}^{y}/v_{kj}^{x})H_j^x & H_j^y & H_j^z \\ (v_{kj}^{z}/v_{kj}^{x})H_j^x & -H_j^z & H_j^y \end{bmatrix} \begin{bmatrix} \Delta\gamma_k^x \\ \Delta\gamma_k^y \\ \Delta\gamma_k^z \end{bmatrix} & (\text{if} \quad v_{kj}^x \neq 0) \\ \begin{bmatrix} 0 & H_j^y & H_j^z \\ 0 & -H_j^z & H_j^y \end{bmatrix} \begin{bmatrix} \Delta\gamma_k^x \\ \Delta\gamma_k^y \\ \Delta\gamma_k^z \end{bmatrix} & (\text{if} \quad v_{kj}^x = 0), \end{cases} \tag{5}$$

$$\begin{bmatrix} \Delta\gamma_k^x \\ \Delta\gamma_k^y \\ \Delta\gamma_k^z \end{bmatrix} = \varepsilon \begin{bmatrix} (1-O_k^x)O_k^x & 0 & 0 \\ 0 & (1-O_k^y)O_k^y & 0 \\ 0 & 0 & (1-O_k^z)O_k^z \end{bmatrix} \begin{bmatrix} \delta_k^x \\ \delta_k^y \\ \delta_k^z \end{bmatrix}, \tag{6}$$

$$\begin{bmatrix} \Delta w_{ji}^{x} \\ \Delta w_{ji}^{y} \end{bmatrix} = \begin{cases} \begin{bmatrix} I_i^x & I_i^y & (w_{ji}^{x}/w_{ji}^{z})I_i^z \\ -I_i^y & I_i^x & (w_{ji}^{y}/w_{ji}^{z})I_i^z \end{bmatrix} \begin{bmatrix} \Delta\theta_j^x \\ \Delta\theta_j^y \\ \Delta\theta_j^z \end{bmatrix} & (\text{if} \quad w_{ji}^z \neq 0) \\ \begin{bmatrix} I_i^x & I_i^y & 0 \\ -I_i^y & I_i^x & 0 \end{bmatrix} \begin{bmatrix} \Delta\theta_j^x \\ \Delta\theta_j^y \\ \Delta\theta_j^z \end{bmatrix} & (\text{if} \quad w_{ji}^z = 0), \end{cases} \tag{7}$$

$$\begin{bmatrix} \Delta\theta_j^x \\ \Delta\theta_j^y \\ \Delta\theta_j^z \end{bmatrix} = \begin{bmatrix} (1-H_j^x)H_j^x & 0 & 0 \\ 0 & (1-H_j^y)H_j^y & 0 \\ 0 & 0 & (1-H_j^z)H_j^z \end{bmatrix} \sum_k \begin{bmatrix} v_{kj}^x & 0 & 0 \\ 0 & v_{kj}^y & v_{kj}^z \\ 0 & -v_{kj}^z & v_{kj}^y \end{bmatrix} \begin{bmatrix} \Delta\gamma_k^x \\ \Delta\gamma_k^y \\ \Delta\gamma_k^z \end{bmatrix}. \tag{8}$$

3. Ability to Learn 3D Affine Transformations

We will present an illustrative example to show that an adaptive network of 3D vector valued neurons can be used to learn 3D affine transformations. Due to space limitations, we will restrict the presentation of our results to similarity transformation, although similar work has been carried out on rotation and parallel displacement.

We used a 1-6-1 3DV-BP three-layered network, which transformed a point $(x_1,\ x_2,\ x_3)$ into another point $(x'_1,\ x'_2,\ x'_3)$ in 3D space. Although the 3DV-BP network generates a value $\mathbf{X} = {}^t[x_1\ x_2\ x_3]$ within the range $0 \leq x_1, x_2, x_3 \leq 1$, for the sake of convenience, we present it in the figure given below as having a transformed value within the range $-1 \leq x_1, x_2, x_3 \leq 1$. The experiment consisted of two parts - a training step, followed by a test step. Two kinds of learning patterns were used in the experiment (Fig. 1): Learning Pattern 1 and Learning Pattern 2. Learning Pattern 1 was on the similarity transformation with a scale reduction rate of 0.5, and Learning Pattern 2 was on that with a scale reduction rate of 0.1. Those 11 input training points (white circles) with equal intervals lying along the straight line ${}^t[x\ \ y\ \ z] = u\,{}^t[1\ \ 1\ \ 1]$ $(0.0 \leq u \leq 1.0)$ mapped onto points along the same line, but with the scale reduction rate of 0.5 (Learning Pattern 1). The corresponding 11 output training points (white squares) lay along the straight line ${}^t[x\ \ y\ \ z] = u\,{}^t[1\ \ 1\ \ 1]$ $(0.0 \leq u \leq 0.5)$. For example, the 1-6-1 3DV-BP network received an input training point ${}^t[1\ \ 1\ \ 1]$ and output an output training point ${}^t[0.5\ \ 0.5\ \ 0.5]$, that is, the distance between the input training point and the origin was reduced to a half. Those 11 input training points (black circles) with equal intervals lying along the straight line ${}^t[x\ \ y\ \ z] = u\,{}^t[1\ \ 1\ \ 1]$ $(-1.0 \leq u \leq 0.0)$ mapped onto points along the same line, but with the scale reduction rate of 0.1 (Learning Pattern 2). The corresponding 11 output training points lay along the straight line ${}^t[x\ \ y\ \ z] = u\,{}^t[1\ \ 1\ \ 1]$ $(-0.1 \leq u \leq 0.0)$ (only three black squares are presented in Fig. 1 because it is not easy to understand 11 crowded black squares). In the test step, by presenting the 622 points (black circles) lying on the unit sphere $x^2 + y^2 + z^2 = 1$, the actual output points (white squares) took the patterns as shown in Fig. 2. It appears that this 3DV-BP network has learned to generalize the scale reduction rate α as a function of the position in 3D space, i.e., a point ${}^t[x\ \ y\ \ z]$ is transformed into another point $\alpha\,{}^t[x\ \ y\ \ z]$, where $\alpha({}^t[x\ \ y\ \ z]) \approx 0.5$ for $x, y, z \geq 0$, and $\alpha({}^t[x\ \ y\ \ z]) \approx 0.1$ for $x, y, z \leq 0$.

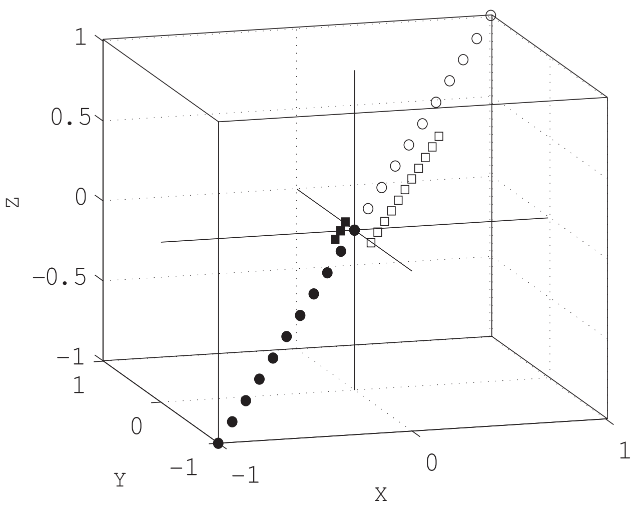

Figure 1. Learning patterns used in the experiment on the ability of the 3DV-BP network to learn 3D affine transformations, which consist of the two kinds of patterns: Learning Patterns 1 and 2. A white circle denotes an input training point and a white square an output training point, which belong to Learning Pattern 1 (the scale reduction rate is 0.5). A black circle denotes an input training point and a black square an output training point, which belong to Learning Pattern 2 (the scale reduction rate is 0.1). Note that the white and black squares are modified to facilitate visualization.

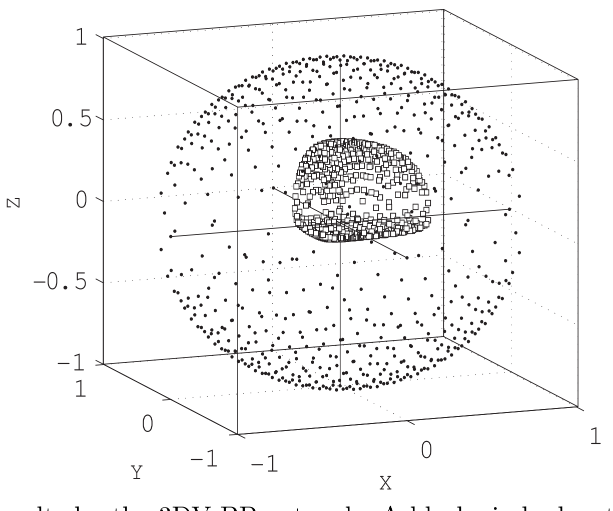

Figure 2. Simulation results by the 3DV-BP network. A black circle denotes an input test point and a white square an actual output point of the 3DV-BP network.

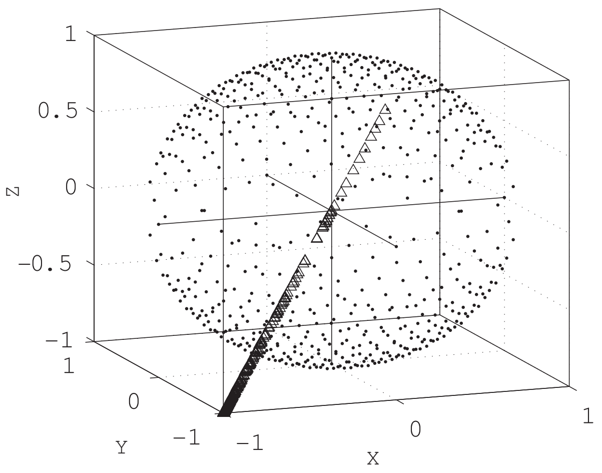

Figure 3. Simulation results by the Real-BP network. A black circle denotes an input test point and a white triangle an actual output point of the Real-BP network.

We also conducted an experiment with a 3-15-3 network with real-valued weights and thresholds, to compare the 3DV-BP with the Real-BP. The first component of a 3D vector was input into the first input neuron, the second component was input into the second input neuron, and the third component was input into the third input neuron. The output from the first output neuron was interpreted as the first component of a 3D vector, and the output from the second output neuron was interpreted as the second component, and the output from the third output neuron was interpreted as the third component. In this connection, time complexity per learning cycle of the 1-6-1 three-layered network for the 3DV-BP was nearly equal to that of the 3-15-3 three-layered network for the Real-BP, as seen in Table 1. Furthermore, the space complexity (i.e. the number of parameters) was almost half that of the Real-BP. To compare how a real-valued network would perform, the 3-15-3 Real-BP network mentioned above was trained using the same pairs of training points described above. The same 622 test points (black circles) lying on the unit sphere were then input with this real-valued neural network. All the actual output points (white triangles) were mapped onto a straight line, as shown in Fig. 3. That is, it appears that the Real-BP network does not have the ability to learn 3D affine transformations.

Table 1 The computational complexities of the 3DV-BP and the Real-BP. Time complexity means the sum of the four operations performed per learning cycle. Space complexity means the sum of the parameters (weights and thresholds).

Network	Time complexity			Space complexity		
	$\times$ and $\div$	$+$ and $-$	Sum	Weights	Thresholds	Sum
3DV-BP 1-6-1	255	141	396	36	21	57
Real-BP 3-15-3	264	141	405	90	18	108

4. Conclusions

We have introduced a three-dimensional vector valued neural network and a learning algorithm, 3DV-BP, which is a natural extension of the Complex-BP algorithm. The 3DV-BP network has the ability to learn 3D affine transformations as its inherent generalization ability, whereas the Real-BP does not.

We expect that applications for the 3DV-BP algorithm will be found in such areas as 3D image processing.

References

[1] A. Hirose (ed.), *Complex-valued neural networks: theories and applications*, World Scientific Publishing, Singapore, 2003.

[2] T. Nitta & T. Furuya, "A complex back-propagation learning," Transactions of Information Proc. Soc. of Japan, Vol.32, No.10, pp.1319-1329, 1991 (in Japanese).

[3] T. Nitta, "An extension of the back-propagation algorithm to complex numbers," Neural Networks, Vol.10, No.8, pp.1392-1415, 1997.

[4] D. E. Rumelhart, G. E. Hinton, & R. J. Williams, *Parallel distributed processing, Vol.1*, MIT Press, Cambridge, 1986.

[5] H. Weyl, *Classical groups*, Princeton, 1946.

[6] A. Watanabe, N. Yazawa, A. Miyauchi & M. Miyauchi, "A method to interpret 3D motions using neural networks," IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E77-A, No.8, pp.1363-1370, 1994.

Tohru Nitta received his Ph.D. in information science from University of Tsukuba in 1995, and is currently a Senior Research Scientist in National Institute of Advanced Industrial Science and Technology (AIST), Japan and is also a Professor at Department of Mathematics, Graduate School of Science, Osaka University, Japan. His research interests include theoretical issues in computational models such as artificial neural networks, especially complex-valued neural networks. (Home page: http://staff.aist.go.jp/tohru-nitta/)