Published in *Proceedings of IEEE International Conference on Neural Networks*, ICNN'95-Perth, Nov. 27-Dec. 1, Vol.5, pp.2753-2756 (1995).

(a) **The title of the article**
A Quaternary Version of the Back-propagation Algorithm

(b) **The authors' full names**
Tohru Nitta

(c) **Current Affiliations**
Neuroscience Research Institute,
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

A Quaternary Version of the Back-propagation Algorithm

Tohru Nitta
Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba Science City, Ibaraki, 305 Japan
Email: tnitta@etl.go.jp

ABSTRACT

A quaternary version of the back-propagation algorithm is proposed for multi-layered neural networks whose weights, threshold values, input and output signals are all quaternions. This new algorithm can be used to learn patterns consisted of quaternions in a natural way. An example was used to successfully test the new formulation.

1 INTRODUCTION

Recently several neural network models with two- (complex-valued) or three-dimensional parameters have been proposed [4, 6, 7, 8, 9] and demonstrated to have the inherent properties such as the abilities to learn 2D or 3D affine transformations [6, 8, 10, 11, 12], and particularly the Complex-BP [6, 8] and the 3DV-BP [7] have been successfully applied to computer vision [5, 14].

This paper presents a quaternary (four-dimensional) version of the back-propagation algorithm (called "Quaternary-BP"), which can be applied to multi-layered neural networks whose weights, threshold values, input and output signals are all quaternions, where a quaternion is a four-dimensional number and was invented by W. R. Hamilton in 1843 [2]. We expect that Quaternary-BP can be effectively used in the fields such as robotics and computer vision in which quaternions have been found useful [1, 3]. This new algorithm was applied to a simulated example on quaternary patterns. Results suggest that the new method is superior to standard BP [13].

Section 2 presents the new Quaternary-BP algorithm. The rest of the paper presents experimental results.

2 THE "QUATERNARY-BP" ALGORITHM

2.1 A Quaternary Neuron

There appear to be several approaches for extending the standard neural networks to higher dimensions. One approach is to extend the number field, i.e. from real numbers x (1 dimension), to complex numbers $z = x + yi$ (2 dimensions; [4, 6, 8]), to quaternions $q = a + bi + cj + dk$ (4 dimensions), to octaves (8 dimensions), to sedenions (16 dimensions),

$\cdots$. Another approach is to extend the dimensionality of the weights and threshold values from 1 dimension to n dimensions using n-dimensinal real valued vectors. Moreover, the latter approach has two varieties : (a) weights are n-dimensional matrices [7], (b) weights are n-dimensional vectors [9]. In this paper we use quaternions in the former approach to extend neural networks to 4 dimensions.

A model neuron used in the Quaternary-BP algorithm is as follows. The input signals, weights, thresholds and output signals are all quaternions. The activity A_n (analogous to the real activity in the standard BP) of neuron n is defined to be :

$$A_n = \sum_m S_m W_{nm} + T_n, \tag{1}$$

where S_m is the quaternary input signal coming from the output of neuron m, W_{nm} is the quaternary weight connecting neuron m and n, T_n is the quaternary threshold value of neuron n. To obtain the quaternary output signal, convert the activity value A_n into its four parts as follows.

$$A_n = x_1 + x_2 i + x_3 j + x_4 k = x, \tag{2}$$

where $i^2 = j^2 = k^2 = -1,\ ij = -ji = k,\ jk = -kj = i,\ ki = -ik = j$.
The output signal $f_4(x)$ is defined to be

$$\begin{aligned} f_4(x) \ &= \ f(x_1) + f(x_2)i + f(x_3)j + f(x_4)k, \\ &\text{where} \quad f(x_l) = \frac{1}{1+\exp(-x_l)}. \end{aligned} \tag{3}$$

The multiplication $S_m W_{nm}$ in eqn (1) should be carefully treated, because the equation $S_m W_{nm} = W_{nm} S_m$ does not hold (the non-commutative property of quternions on multiplication), which produces two kinds of quaternary neurons: one is called "normal quaternary neuron" which calculates $A_n = \sum_m S_m W_{nm} + T_n$, the other is called "inverse quaternary neuron" which calculates $A_n = \sum_m W_{nm} S_m + T_n$.

2.2 A Quaternary Neural Network

In this subsection, we introduce the network used in the Quaternary-BP algorithm. It has 3 layers and consists of only "normal quaternary neurons", for the sake of simplicity.

We use $w_{ml} = w^a_{ml} + w^b_{ml} i + w^c_{ml} j + w^d_{ml} k \in \boldsymbol{H}$ for the weight between the input neuron l and the hidden neuron m (where $\boldsymbol{H}$ denotes the set of quaternions), $v_{nm} = v^a_{nm} + v^b_{nm} i + v^c_{nm} j + v^d_{nm} k \in \boldsymbol{H}$ for the weight between the hidden neuron m and the output neuron n, $\theta_m = \theta^a_m + \theta^b_m i + \theta^c_m j + \theta^d_m k \in \boldsymbol{H}$ for the threshold of the hidden neuron m, $\gamma_n = \gamma^a_n + \gamma^b_n i + \gamma^c_n j + \gamma^d_n k \in \boldsymbol{H}$ for the threshold of the output neuron n. Let $I_l = I^a_l + I^b_l i + I^c_l j + I^d_l k \in \boldsymbol{H}$ denote the input signal to the input neuron l, and let $H_m = H^a_m + H^b_m i + H^c_m j + H^d_m k \in \boldsymbol{H}$ and $O_n = O^a_n + O^b_n i + O^c_n j + O^d_n k \in \boldsymbol{H}$ denote the output signals of the hidden neuron m, and the output neuron n, respectively. Let $\Delta_n = \Delta^a_n + \Delta^b_n i + \Delta^c_n j + \Delta^d_n k = T_n - O_n \in \boldsymbol{H}$ denote the error between O_n and the target output signal $T_n = T^a_n + T^b_n i + T^c_n j + T^d_n k \in \boldsymbol{H}$ of the pattern to be learned for the output neuron n. We define the square error for the pattern p as $E_p = (1/2) \sum_{n=1}^{N} |\ T_n - O_n|^2$, where

N is the number of output neurons, $|x| \stackrel{\text{def}}{=} \sqrt{x_1^2+x_2^2+x_3^2+x_4^2}$, $x = x_1+x_2 i+x_3 j+x_4 k \in \boldsymbol{H}$.

2.3 The Learning Algorithm

Next, we define a learning rule for the Quaternary-BP model described above. For a sufficiently small learning constant $\varepsilon > 0$, and using a steepest descent method, we can show that the weights and the thresholds should be modified according to the following equations.

$$\begin{aligned}\Delta v_{nm} &\stackrel{\text{def}}{=} \Delta v_{nm}^a+\Delta v_{nm}^b i+\Delta v_{nm}^c j+\Delta v_{nm}^d k \\ &= -\varepsilon\left(\frac{\partial E_p}{\partial v_{nm}^a}+\frac{\partial E_p}{\partial v_{nm}^b}i+\frac{\partial E_p}{\partial v_{nm}^c}j+\frac{\partial E_p}{\partial v_{nm}^d}k\right),\end{aligned} \tag{4}$$

$$\begin{aligned}\Delta \gamma_n &\stackrel{\text{def}}{=} \Delta \gamma_n^a+\Delta \gamma_n^b i+\Delta \gamma_n^c j+\Delta \gamma_n^d k \\ &= -\varepsilon\left(\frac{\partial E_p}{\partial \gamma_n^a}+\frac{\partial E_p}{\partial \gamma_n^b}i+\frac{\partial E_p}{\partial \gamma_n^c}j+\frac{\partial E_p}{\partial \gamma_n^d}k\right),\end{aligned} \tag{5}$$

$$\begin{aligned}\Delta w_{ml} &\stackrel{\text{def}}{=} \Delta w_{ml}^a+\Delta w_{ml}^b i+\Delta w_{ml}^c j+\Delta w_{ml}^d k \\ &= -\varepsilon\left(\frac{\partial E_p}{\partial w_{ml}^a}+\frac{\partial E_p}{\partial w_{ml}^b}i+\frac{\partial E_p}{\partial w_{ml}^c}j+\frac{\partial E_p}{\partial w_{ml}^d}k\right),\end{aligned} \tag{6}$$

$$\begin{aligned}\Delta \theta_m &\stackrel{\text{def}}{=} \Delta \theta_m^a+\Delta \theta_m^b i+\Delta \theta_m^c j+\Delta \theta_m^d k \\ &= -\varepsilon\left(\frac{\partial E_p}{\partial \theta_m^a}+\frac{\partial E_p}{\partial \theta_m^b}i+\frac{\partial E_p}{\partial \theta_m^c}j+\frac{\partial E_p}{\partial \theta_m^d}k\right),\end{aligned} \tag{7}$$

where Δx denotes the amount of the correction of a parameter x. The above equations (4) – (7) can be expressed as:

$$\Delta v_{nm} = \overline{H}_m \Delta \gamma_n, \tag{8}$$

$$\begin{aligned}\Delta \gamma_n = \varepsilon\{&\Delta_n^a(1-O_n^a)O_n^a+\Delta_n^b(1-O_n^b)O_n^b i \\ &+\Delta_n^c(1-O_n^c)O_n^c j+\Delta_n^d(1-O_n^d)O_n^d k\},\end{aligned} \tag{9}$$

$$\Delta w_{ml} = \overline{I}_l \Delta \theta_m, \tag{10}$$

$$\begin{aligned}\Delta \theta_m = &(1-H_m^a)H_m^a \cdot Re\Big[\sum_n(\Delta \gamma_n \overline{v}_{nm})\Big] \\ &+(1-H_m^b)H_m^b \cdot Im^i\Big[\sum_n(\Delta \gamma_n \overline{v}_{nm})\Big] i \\ &+(1-H_m^c)H_m^c \cdot Im^j\Big[\sum_n(\Delta \gamma_n \overline{v}_{nm})\Big] j \\ &+(1-H_m^d)H_m^d \cdot Im^k\Big[\sum_n(\Delta \gamma_n \overline{v}_{nm})\Big] k,\end{aligned} \tag{11}$$

where $\overline{x} \stackrel{\text{def}}{=} x_1-x_2 i-x_3 j-x_4 k$, $Re[x] \stackrel{\text{def}}{=} x_1$, $Im^i[x] \stackrel{\text{def}}{=} x_2$, $Im^j[x] \stackrel{\text{def}}{=} x_3$ and $Im^k[x] \stackrel{\text{def}}{=} x_4$

for a quaternion $x = x_1 + x_2 i + x_3 j + x_4 k \in \boldsymbol{H}$.

3 SIMULATION

An example on quaternary patterns was used to compare the performance of the new Quaternary-BP algorithm with the standard back-propagation algorithm.

We used a 1-2-1 three-layered network for the Quaternary-BP, and a 4-9-4 three-layered network for the standard BP. Table 1 shows that their time complexities per learning cycle are almost equal. The learning constant used in the experiment was 0.5. The initial first, second, third and fourth parts of the weights and the thresholds were chosen to be random real numbers between $-$ 0.3 and + 0.3. The input data were presented in sequence, together with the desired output, to the net as shown in Table 2.

The results of the simulation are plotted in Fig.1. The new algorithm converged in 400 iterations, whereas the original algorithm required 600. Furthermore, the space complexity (i.e. the number of parameters) is almost one-third of that of the standard BP, as seen in Table 1.

4 CONCLUSIONS

We have proposed a quaternary version of the back-propagation learning algorithm, where the input signals, weights, thresholds, and output signals are all quaternions. An simple example was used to test the presented method and it showed excellent performance. We expect that this new algorithm has the inherent properties such as the abilities of the Complex-BP to learn "2D affine transformation" [6, 8] and the 3DV-BP "3D affine transformation"[12], and will demonstrate its real ability in the areas dealing with quaternions. The extension of the Quaternary-BP algorithm to fully connected neural networks will be presented in a future paper.

ACKNOWLEDGEMENTS

The author expresses his thanks to Dr.K.Ohta, Director of the Computer Science Division, and Dr.T.Higuchi, Chief of the Computational Models Section, for having an opportunity to do this study and their continual encouragement. He is also grateful to the anonymous reviewers for many helpful suggestions.

References

[1] Canny, J. F. (1988). *The Complexity of Robot Motion Planning*, MIT Press.

[2] Ebbinghaus, H.-D. etl al. (Eds.). (1988). *Zahlen*, Springer-Verlag Berlin Heidelderg (in German).

[3] Faugeras, O. (1993). *Three-Dimensional Computer Vision*, MIT Press.

[4] Kim, M. S. and Guest, C. C. (1990). Modification of Backpropagation Networks for Complex-valued Signal Processing in Frequency Domain. *Proc. IEEE/INNS International Joint Conference on Neural Networks*, IJCNN'90-SanDiego, June, Vol.3, pp.27–31.

[5] Miyauchi, M., Seki, M., Watanabe, A. and Miyauchi, A. (1993). Interpretation of Optical Flow through Complex Neural Network. *Proc. International Workshop on*

Artificial Neural Networks, IWANN'93-Barcelona, Lecture Notes in Computer Science, Vol.686, Springer-Verlag, pp.645–650.

[6] Nitta, T. and Furuya, T. (1991). A Complex Back-propagation Learning. *Transactions of Information Processing Society of Japan*, Vol. 32, No. 10, pp.1319–1329 (in Japanese).

[7] Nitta, T. and deGaris, H. (1992). A 3D Vector Version of the Back-propagation Algorithm. *Proc. IEEE/INNS International Joint Conference on Neural Networks*, IJCNN'92-Beijing, Nov.3-6, Vol.2, pp.511–516.

[8] Nitta, T. (1993). A Complex Numbered Version of the Back-propagation Algorithm. *Proc. INNS World Congress on Neural Networks*, WCNN'93-Portland, Vol. 3, pp. 576–579.

[9] Nitta, T. (1993). A Back-propagation Algorithm for Neural Networks Based on 3D Vector Product. *Proc. IEEE/INNS International Joint Conference on Neural Networks*, IJCNN'93-Nagoya, Oct. 25–29, Vol.1, pp.589–592.

[10] Nitta, T. (1994). Structure of Learning in the Complex Numbered Back-propagation Network. *Proc. IEEE International Conference on Neural Networks*, ICNN'94-Orlando, June 28-July 2, Vol.1, pp.269–274.

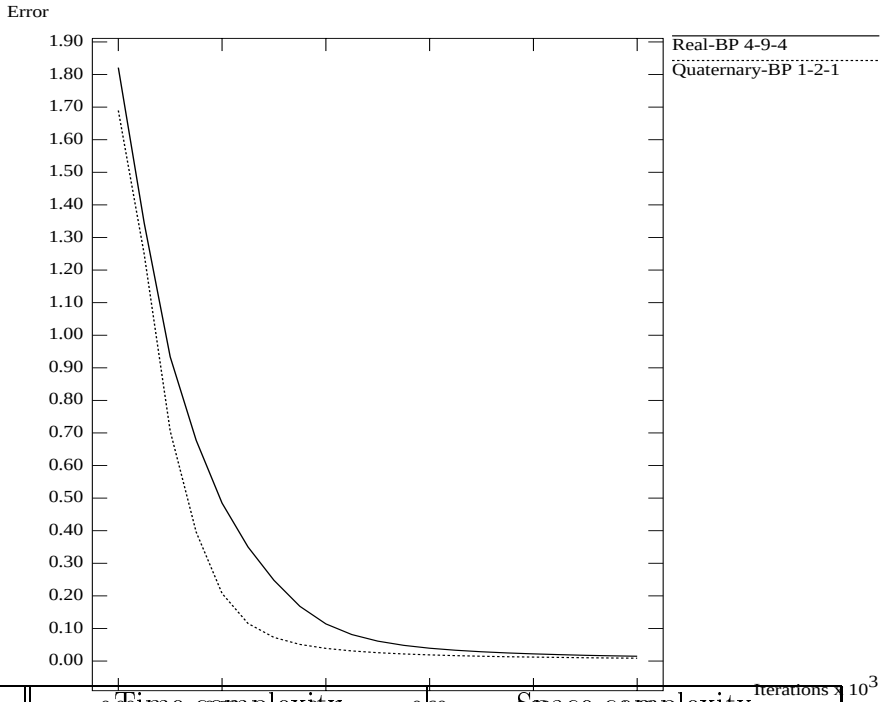
[11] Nitta, T. (1994). An Analysis on Decision Boundaries in the Complex Back-propagation Network. *Proc. IEEE International Conference on Neural Networks*, ICNN'94-Orlando, June 28-July 2, Vol.2, pp.934–939.

[12] Nitta, T. (1994). Generalization Ability of the Three-dimensional Back-propagation Network. *Proc. IEEE International Conference on Neural Networks*, ICNN'94-Orlando, June 28-July 2, Vol.5, pp.2895–2900.

[13] Rumelhart, D. E. et al. (1986). *Parallel Distributed Processing*, Vol. 1, MIT press.

[14] Watanabe, A., Yazawa, N., Miyauchi, A. and Miyauchi, M. (1994). A Method to Interpret 3D Motions Using Neural Networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol.E77-A, No.8, pp.1363–1370.

Figure 1: Learning Curves for the Simulation.



Network	Time complexity			Space complexity		
	$\times$ and $\div$	$+$ and $-$	Sum	Weights	Thresholds	Sum
Quaternary-BP 1-2-1	185	152	337	16	12	28
Standard BP 4-9-4	210	116	326	72	13	85

Table 1 : The Computational Complexity of the Quaternary-BP and the Standard 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).

Input	Output
$1+i+j+k$	1
$1+2i+j+2k$	i
$2+i+2j+k$	j
$2+2i+2j+2k$	k

Table 2 : The Input Patterns and the Corresponding Desired Output Patterns for the Simulation. The first component of a quaternion is given to the first component of the input/output neuron 1, the second is the second component, the third is the third component, and the fourth is the fourth component in the Quaternary-BP network. The first component of a quaternion is given to the input/output neuron 1, the second is the neuron 2, the third is the neuron 3, and the fourth is the neuron 4 in the standard BP network.