Published in *Proceedings of IEEE/INNS International Joint Conference on Neural Networks*, IJCNN'92-Beijing, Nov. 3-6, Vol.2, pp. 511-516 (1992).

(a) **The title of the article**
A 3D Vector Version of the Back-propagation Algorithm

(b) **The authors' full names**
Tohru Nitta and Hugo de Garis

(c) **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 3D Vector Version of the Back-propagation Algorithm

Tohru Nitta and Hugo de Garis
Computational Models Section, Computer Science Division, Electrotechnical Laboratory,
1-1-4 Umezono, Tsukuba Science City, Ibaraki, 305, Japan
email : tnitta@etl.go.jp

Abstract

This paper introduces a 3D vector version of the back-propagation algorithm (called "3DV-BP"), which can be applied to multi-layered neural networks whose threshold values, input and output signals are all 3D real numbered vectors, and whose weights are all 3D orthogonal matrices. This new variant of the standard back-propagation algorithm allows new "spatial" characteristics to be treated. The algorithm is a natural extention of the authors' Complex-BP algorithm which was a complex numbered version of the back-propagation algorithm ([2], [3]). The XOR problem was used to successfully test the new formulation.

1 INTRODUCTION

We present a three-dimensional vector version of the back-propagation algorithm (called "3DV-BP"), which can be applied to multi-layered neural networks whose threshold values, input and output signals are all 3D real valued vectors, and whose weights are all 3D orthogonal matrices. This new algorithm is a natural extention of the authors' Complex-BP algorithm ([2], [3]) which can be applied to multi-layered neural networks whose weights, threshold values, input and output signals are all complex numbers. Moreover, this new algorithm was applied to the XOR problem. Results suggest that the new method is superior to standard BP [4].

Section 2 presents the new 3DV-BP algorithm. The rest of the paper presents experimental results.

2 THE "3DV-BP" ALGORITHM

2.1 A 3D VECTOR NEURON

There appear to be several approaches for extending the standard BP or the authors' Complex-BP ([2], [3]) to higher dimensions. One approach is to extend the number field, i.e. from real numbers x (1 dimension), to complex numbers $z = x + iy$ (2 dimensions), to quaternions $q = a + ib + jc + kd$ (4 dimensions), to sedenions (16 dimensions), $\cdots$ ([5]). Another approach is to extend the dimensionality of the threshold values and weights from 2 dimensions to 3 dimensions using 3D real valued vectors. It is this alternative approach that we use in this paper.

A model neuron used in the 3DV-BP algorithm is as follows. The input signals, thresholds and output signals are all 3D real valued vectors, and the weights are all 3D orthogonal matrices. The activity $\boldsymbol{A}_j$ (analogous to the activity in the standard BP) of neuron j is defined to be :

$$\boldsymbol{A}_j = \sum_k \boldsymbol{W}_{jk}\boldsymbol{S}_k + \boldsymbol{T}_j, \tag{1}$$

where $\boldsymbol{W}_{jk}$ is the 3D orthogonal weight matrix connecting neuron j and k, $\boldsymbol{S}_k$ is the 3D real valued vector input signal coming from the output of neuron k, and $\boldsymbol{T}_j$ is the 3D real valued vector threshold value of neuron j. The output signal $F(\boldsymbol{A}_j)$ is defined to be :

$$F(\boldsymbol{A}_j) = \begin{bmatrix} f(a_1) \\ f(a_2) \\ f(a_3) \end{bmatrix}, \qquad \text{where} \quad \boldsymbol{A}_j = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad \text{and} \quad f(a_i) = \frac{1}{1+\exp(-a_i)}. \tag{2}$$

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

$$\begin{aligned} X + iY &= f_C\Big(\sum_{k=1}^{n}(w_k^r + iw_k^i)(x_k + iy_k) + (\theta^r + i\theta^i)\Big) \\ &= f\Big(\sum_{k=1}^{n}(w_k^r x_k - w_k^i y_k) + \theta^r\Big) + if\Big(\sum_{k=1}^{n}(w_k^i x_k + w_k^r y_k) + \theta^i\Big), \end{aligned} \tag{3}$$

and $f_C(z) = f(x) + if(y)$, $z = x + iy$. Furthermore,

$$\begin{aligned} \begin{bmatrix} X \\ Y \end{bmatrix} &= F_C\left(\left[\begin{array}{cc|c|cc} w_1^r & -w_1^i & \cdots & w_n^r & -w_n^i \\ w_1^i & w_1^r & \cdots & w_n^i & w_n^r \end{array}\right] \left[\begin{array}{c} x_1 \\ y_1 \\ \hline \vdots \\ \hline x_n \\ y_n \end{array}\right] + \begin{bmatrix} \theta^r \\ \theta^i \end{bmatrix}\right) \\ &= F_C\left(|w_1| \begin{bmatrix} \cos\alpha_1 & -\sin\alpha_1 \\ \sin\alpha_1 & \cos\alpha_1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \cdots \right. \\ &\qquad \left. + |w_n| \begin{bmatrix} \cos\alpha_n & -\sin\alpha_n \\ \sin\alpha_n & \cos\alpha_n \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \begin{bmatrix} \theta^r \\ \theta^i \end{bmatrix}\right), \end{aligned} \tag{4}$$

where $F_C\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} f(x) \\ f(y) \end{bmatrix}$, $\alpha_k = \arctan(w_k^i / w_k^r)$ $(1 \le k \le n)$. In the expression (4), $\begin{bmatrix} w_k^r & -w_k^i \\ w_k^i & w_k^r \end{bmatrix}$ is an element of the two-dimensional orthogonal group $O_2(\boldsymbol{R})$ (where

$\boldsymbol{R}$ denotes the set of real numbers). Considering (4), the formulation of a neuron as given in (1) and (2) above is natural. Moreover, we can see that $\begin{bmatrix} \cos\alpha_k & -\sin\alpha_k \\ \sin\alpha_k & \cos\alpha_k \end{bmatrix}$ is an element of the two-dimensional rotation group $SO_2(\boldsymbol{R})$, and $|w_k|$ is an element of $\boldsymbol{R}^+ = \{x \in \boldsymbol{R} | x > 0\}$ which is a group on multiplication. Therefore, $\begin{bmatrix} w_k^r & -w_k^i \\ w_k^i & w_k^r \end{bmatrix}$ is also an element of the group $\boldsymbol{R}^+ \times SO_2(\boldsymbol{R})$.

Note that there is another formulation of the complex valued version [1] in which the output function is a complex valued function $f(z) = 1/(1+\exp(-z))$, where $z = x + iy$.

2.2 A 3D VECTOR NEURAL NETWORK

In this subsection, we introduce the network used in the 3DV-BP algorithm. It has 3 layers, for the sake of simplicity. We use $\boldsymbol{W}_{ji} \in O_3(\boldsymbol{R})$ for the weight between the input neuron i and the hidden neuron j (where $O_3(\boldsymbol{R})$ denotes the three-dimensional orthogonal group.), $\boldsymbol{V}_{kj} \in O_3(\boldsymbol{R})$ for the weight between the hidden neuron j and the output neuron k, $\boldsymbol{\Theta}_j = {}^t[\theta_j^x \ \ \theta_j^y \ \ \theta_j^z] \in \boldsymbol{R}^3$ for the threshold of the hidden neuron j, $\boldsymbol{\Gamma}_k = {}^t[\gamma_k^x \ \ \gamma_k^y \ \ \gamma_k^z] \in \boldsymbol{R}^3$ for the threshold of the output neuron k. Let $\boldsymbol{I}_i = {}^t[I_i^x \ \ I_i^y \ \ I_i^z] \in \boldsymbol{R}^3$ denote the input signal to the input neuron i, and let $\boldsymbol{H}_j = {}^t[H_j^x \ \ H_j^y \ \ H_j^z] \in \boldsymbol{R}^3$ and $\boldsymbol{O}_k = {}^t[O_k^x \ \ O_k^y \ \ O_k^z] \in \boldsymbol{R}^3$ denote the output signals of the hidden neuron j, and the output neuron k, respectively. Let $\boldsymbol{\Delta}^k = {}^t[\delta_k^x \ \ \delta_k^y \ \ \delta_k^z] = \boldsymbol{T}_k - \boldsymbol{O}_k \in \boldsymbol{R}^3$ denote the error between $\boldsymbol{O}_k$ and the target output signal $\boldsymbol{T}_k = {}^t[T_k^x \ \ T_k^y \ \ T_k^z] \in \boldsymbol{R}^3$ of the pattern to be learned for the output neuron k. We define the square error for the pattern p as $E_p = (1/2)\sum_{k=1}^{N} ||\boldsymbol{T}_k - \boldsymbol{O}_k||^2$, where N is the number of output neurons, $||x|| \stackrel{\text{def}}{=} \sqrt{x_1^2+x_2^2+x_3^2}$, $x = {}^t[x_1 \ \ x_2 \ \ x_3]$. We express the weights as :-

$$\boldsymbol{W}_{ji} = \begin{bmatrix} w_{ji}^x & -w_{ji}^y & 0 \\ w_{ji}^y & w_{ji}^x & 0 \\ 0 & 0 & w_{ji}^z \end{bmatrix} \in O_3(\boldsymbol{R}), \qquad \text{where} \quad w_{ji}^z = \sqrt{(w_{ji}^x)^2 + (w_{ji}^y)^2}, \tag{5}$$

$$\boldsymbol{V}_{kj} = \begin{bmatrix} v_{kj}^x & 0 & 0 \\ 0 & v_{kj}^y & -v_{kj}^z \\ 0 & v_{kj}^z & v_{kj}^y \end{bmatrix} \in O_3(\boldsymbol{R}), \qquad \text{where} \quad v_{kj}^x = \sqrt{(v_{kj}^y)^2 + (v_{kj}^z)^2}. \tag{6}$$

$\boldsymbol{W}_{ji}$, $\boldsymbol{V}_{kj}$ in the expressions (5) and (6) are also elements of the group $\boldsymbol{R}^+ \times SO_3(\boldsymbol{R})$, because they can be rewritten as follows :

$$\boldsymbol{W}_{ji} = w_{ji}^z A(\alpha_{ji}) = w_{ji}^z \begin{bmatrix} \cos\alpha_{ji} & -\sin\alpha_{ji} & 0 \\ \sin\alpha_{ji} & \cos\alpha_{ji} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7}$$

$$\boldsymbol{V}_{kj} = v_{kj}^x B(\beta_{kj}) = v_{kj}^x \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta_{kj} & -\sin\beta_{kj} \\ 0 & \sin\beta_{kj} & \cos\beta_{kj} \end{bmatrix}, \tag{8}$$

where $B(\beta_{kj})$ is the canonical form of the elements of the three-dimensional rotation group $SO_3(\boldsymbol{R})$, i.e. for any $X \in SO_3(\boldsymbol{R})$, there exists an orthogonal matrix $Y \in O_3(\boldsymbol{R})$, $YXY^{-1} = B(\beta_{kj})$. $B(\beta_{kj})$ denotes a rotation about the X-axis, and $A(\alpha_{ji})$ a rotation about the Z-axis.

2.3 THE LEARNING ALGORITHM

Next, we define a learning rule for the 3DV-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.

$$\begin{aligned}
\begin{bmatrix} \Delta v_{kj}^y \\ \Delta v_{kj}^z \end{bmatrix} &= -\varepsilon \begin{bmatrix} \frac{\partial E_p}{\partial v_{kj}^y} \\ \frac{\partial E_p}{\partial v_{kj}^z} \end{bmatrix}, \qquad \begin{bmatrix} \Delta \gamma_k^x \\ \Delta \gamma_k^y \\ \Delta \gamma_k^z \end{bmatrix} = -\varepsilon \begin{bmatrix} \frac{\partial E_p}{\partial \gamma_k^x} \\ \frac{\partial E_p}{\partial \gamma_k^y} \\ \frac{\partial E_p}{\partial \gamma_k^z} \end{bmatrix}, \\
\begin{bmatrix} \Delta w_{ji}^x \\ \Delta w_{ji}^y \end{bmatrix} &= -\varepsilon \begin{bmatrix} \frac{\partial E_p}{\partial w_{ji}^x} \\ \frac{\partial E_p}{\partial w_{ji}^y} \end{bmatrix}, \qquad \begin{bmatrix} \Delta \theta_j^x \\ \Delta \theta_j^y \\ \Delta \theta_j^z \end{bmatrix} = -\varepsilon \begin{bmatrix} \frac{\partial E_p}{\partial \theta_j^x} \\ \frac{\partial E_p}{\partial \theta_j^y} \\ \frac{\partial E_p}{\partial \theta_j^z} \end{bmatrix} \quad . \qquad (9)
\end{aligned}$$

The above equations (9) can be expressed as :

$$\begin{bmatrix} \Delta v_{kj}^y \\ \Delta v_{kj}^z \end{bmatrix} = \begin{cases} \begin{bmatrix} (v_{kj}^y/v_{kj}^x)H_j^x & H_j^y & H_j^z \\ (v_{kj}^z/v_{kj}^x)H_j^x & -H_j^z & H_j^y \end{bmatrix} \begin{bmatrix} \Delta \gamma_k^x \\ \Delta \gamma_k^y \\ \Delta \gamma_k^z \end{bmatrix} & (\text{if} \quad v_{kj}^x \neq 0) \\ \begin{bmatrix} 0 & H_j^y & H_j^z \\ 0 & -H_j^z & H_j^y \end{bmatrix} \begin{bmatrix} \Delta \gamma_k^x \\ \Delta \gamma_k^y \\ \Delta \gamma_k^z \end{bmatrix} & (\text{if} \quad v_{kj}^x = 0), \end{cases} \qquad (10)$$

$$\begin{bmatrix} \Delta \gamma_k^x \\ \Delta \gamma_k^y \\ \Delta \gamma_k^z \end{bmatrix} = \varepsilon \begin{bmatrix} (1-O_k^x)O_k^x & 0 & 0 \\ 0 & (1-O_k^y)O_k^y & 0 \\ 0 & 0 & (1-O_k^z)O_k^z \end{bmatrix} \begin{bmatrix} \delta_k^x \\ \delta_k^y \\ \delta_k^z \end{bmatrix}, \qquad (11)$$

$$\begin{bmatrix} \Delta w_{ji}^x \\ \Delta w_{ji}^y \end{bmatrix} = \begin{cases} \begin{bmatrix} I_i^x & I_i^y & (w_{ji}^x/w_{ji}^z)I_i^z \\ -I_i^y & I_i^x & (w_{ji}^y/w_{ji}^z)I_i^z \end{bmatrix} \begin{bmatrix} \Delta \theta_j^x \\ \Delta \theta_j^y \\ \Delta \theta_j^z \end{bmatrix} & (\text{if} \quad w_{ji}^z \neq 0) \\ \begin{bmatrix} I_i^x & I_i^y & 0 \\ -I_i^y & I_i^x & 0 \end{bmatrix} \begin{bmatrix} \Delta \theta_j^x \\ \Delta \theta_j^y \\ \Delta \theta_j^z \end{bmatrix} & (\text{if} \quad w_{ji}^z = 0), \end{cases} \qquad (12)$$

$$\begin{bmatrix} \Delta \theta_j^x \\ \Delta \theta_j^y \\ \Delta \theta_j^z \end{bmatrix} = \begin{bmatrix} (1-H_j^x)H_j^x & 0 & 0 \\ 0 & (1-H_j^y)H_j^y & 0 \\ 0 & 0 & (1-H_j^z)H_j^z \end{bmatrix} \sum_k \begin{bmatrix} v_{kj}^x & 0 & 0 \\ 0 & v_{kj}^y & v_{kj}^z \\ 0 & -v_{kj}^z & v_{kj}^y \end{bmatrix} \begin{bmatrix} \Delta \gamma_k^x \\ \Delta \gamma_k^y \\ \Delta \gamma_k^z \end{bmatrix}. \qquad (13)$$

3 SIMULATION

The XOR problem was used to compare the performance of the new 3DV-BP algorithm with the standard back-propagation algorithm. We used a 1-6-1 three-layered network for the 3DV-BP, and a 3-15-3 three-layered network for the standard BP. Table 2 shows that their time complexities per learning cycle are almost equal. The learning constant used in the experiment was 0.5. The initial X-, Y- and Z-components of the weights and the thresholds were chosen to be random real numbers between -0.3 and 0.3. The input data are presented in sequence, together with the desired output, to the net as shown in Table 1.

The results of the simulation are plotted in Fig.1. The new algorithm converged in 900 iterations, whereas the original algorithm required 1,500. Furthermore, the space complexity (i.e. the number of parameters) is almost half that of the standard BP, as seen in Table 2.

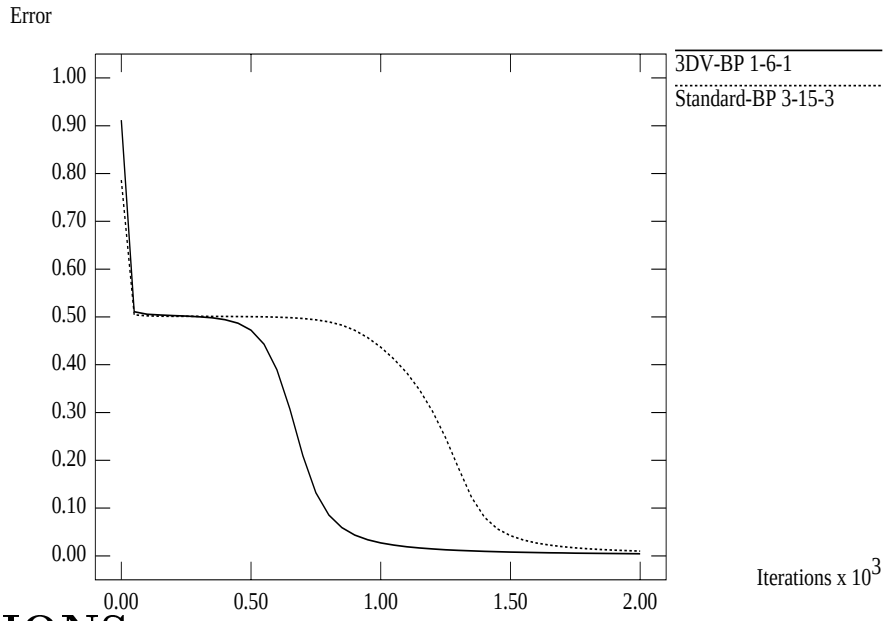
Input			Output		
x	y	z	x	y	z
0	0	0	0	0	0
0	1	0	1	0	0
1	0	0	1	0	0
1	1	0	0	0	0

Table 1 : The Input Patterns and the Corresponding Desired Output Patterns for the XOR Problem. x is given to the X-component of the input/output neuron 1, y is the Y-component, and z is the Z-component in the 3DV-BP network. x is given to the input/output neuron 1, y is the neuron 2, and z is the neuron 3 in the standard BP network.

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

Table 2 : The Computational Complexity of the 3DV-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).

Figure 1: Learning Curves for the XOR problem.



4 CONCLUSIONS

We have introduced a three-dimensional vector version of the back-propagation learning algorithm, where the input signals, thresholds, and output signals are all 3D real valued vectors, and the weights are all 3D orthogonal matrices. This new algorithm is a natural extension of the authors' Complex-BP algorithm. The XOR problem was used to test the presented method and it showed excellent performance. We expect that this new algorithm will find applications in such areas as 3D image processing. The extension of the 3DV-BP algorithm to fully connected neural networks will be presented in a future paper.

ACKNOWLEDGEMENTS

The authors would like to give special thanks to Dr.S.Akaho for valuable comments. He also expresses his thanks to Dr.T.Yuba, Director of the Computer Science Division, and Dr.T.Furuya, Chief of the Computational Models Section, for having an opportunity to do this study and their continual encouragement.

References

[1] Kim, M. S. and Guest, C. C. (1990). Modification of backpropagation networks for complex-valued signal processing in frequency domain. *Proc. IJCNN 1990*, Vol.3, 27-31.

[2] Nitta, T. and Furuya, T. (1991). A complex back-propagation learning and the capability for the transformation of geometrical figures. *Technical report of the professional group on neurocomputing of IEICE Japan*, NC90-60, 45-52 (in Japanese).

[3] Nitta, T., Hugo de Garis and Furuya, T. (1992). A complex numbered version of the back-propagation algorithm. *ETL Technical Report*, TR-92-26, 1-10.

[4] Rumelhart, D. E. et al. (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland(Eds.), *Parallel distributed processing : Explorations in the microstructures of cognition*, Vol.1, Cambridge, MIT Press.

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