

# On the learning machine for three dimensional mapping

Bipin K. Tripathi · Prem K. Kalra

Received: 17 June 2009 / Accepted: 11 February 2010 / Published online: 5 March 2010  
© Springer-Verlag London Limited 2010

**Abstract** In this paper, we investigate the neural network with three-dimensional parameters for applications like 3D image processing, interpretation of 3D transformations, and 3D object motion. A 3D vector represents a point in the 3D space, and an object might be represented with a set of these points. Thus, it is desirable to have a 3D vector-valued neural network, which deals with three signals as one cluster. In such a neural network, 3D signals are flowing through a network and are the unit of learning. This article also deals with a related 3D back-propagation (3D-BP) learning algorithm, which is an extension of conventional back-propagation algorithm in the single dimension. 3D-BP has an inherent ability to learn and generalize the 3D motion. The computational experiments presented in this paper evaluate the performance of considered learning machine in generalization of 3D transformations and 3D pattern recognition.

**Keywords** 3D Back-propagation algorithm · 3D Real-valued vector · Orthogonal matrix · 3D Face

## 1 Introduction

In computer vision system, the interpretations of 3D motion [9], 3D transformations, and 3D face or object matching [6, 10] important tasks. There have been many methodologies to solve them. However, these methods are time-consuming, computation extensive, and weak to noise. A small complexity, robust performance, and quick convergence of artificial neural network (ANN) are vital for its wide applicability. This study is devoted to investigate the solution using neural network with 3D parameters. Any ANN designed must have small complexity for real-life applications, and at the same time, it should have adequate generalization and adequate functional mapping capabilities. The proposed learning machine consists of multi-layer network of 3D vector-valued neurons. In 3D vector-valued neural network, the parameters such as threshold values, input–output signals are all 3D real-valued vectors and weights associated with connections are 3D orthogonal matrices. All the operations in such neural models are scalar matrix operations. The corresponding 3D vector-valued back-propagation training algorithm is the natural extension of complex-valued back-propagation algorithm [1, 5] and has natural ability to learn *3D motion* as complex-BP learns 2D motion.

In this paper, we explore the characteristics of 3D vector-valued neural network through variety of computational experiments. Section 2 explains the architecture of a 3D vector-valued neuron. Section 3 and Appendix define 3D vector-valued version of back-propagation algorithm for a multi-layer network. The learning and generalization ability of above mentioned neural network is investigated using diverse test patterns in Sect. 4. Section 5 works out the neural network and learning algorithm for 3D face

---

B. K. Tripathi (✉) · P. K. Kalra  
Indian Institute of Technology, Kanpur, India  
e-mail: abkt@iitk.ac.in

P. K. Kalra  
e-mail: kalra@iitk.ac.in

*Present Address:*

P. K. Kalra  
Indian Institute of Technology, Rajasthan, India

matching. Section 6 presents the concluding remarks and suggestions for future works.

## 2 3D Vector-valued neuron

The complexity of ANN depends on the number of neurons and learning algorithm. The higher the complexity of an ANN is the more computations and memory intensive it can be. The number of neurons to be used in an ANN is a function of the mapping or the classifying power of the neuron itself [7, 8]. Therefore, in case of high-dimensional problem, it is imperative to look for higher dimensional neuron model that can directly process the high-dimensional information. It will serve as a building block for a powerful ANN with fewer neurons. Various researchers have independently proposed extension of real-valued neuron (one dimension) to higher dimension [3, 4]. Most of them have followed natural extension of number field like real number (one dimension), complex number (two dimension), 3D real-valued vectors (three dimension), quaternion (four dimension) for representation of higher dimension neurons. Therefore, it will be worthwhile to explore the capabilities of the 3D vector-valued neurons in function mapping and pattern classification problems in 3D space. The activation function for 3D vector-valued neuron can be defined as 3D extension of real activation function. Let  $V = [V_x, V_y, V_z]^T$  be the net internal potential of a neuron, then its output is defined as:

$$Y = f(V) = [f(V_x), f(V_y), f(V_z)]^T \quad (1)$$

## 3 3D Back-propagation network

The learning rules correlates the input and output values of the neurons by adjusting the interconnection weights. In neural network literature, most popular back-propagation learning algorithm (for single dimension) has been extended to higher dimensional neuron models [1, 2, 7]. Appendix presents a 3D vector version of back-propagation (3D-BP) algorithm for a three layer network ( $L - M - N$ ) of vector-valued neurons, based on scalar product operation among network parameters where,  $L$  is the number of inputs,  $M$  is the number of neurons in hidden layer, and  $N$  is the number of neurons in output layer. The synaptic weights are three dimensional orthogonal matrices. The learning rules have been obtained using steepest descent on error function, which enables the neuron or network to learn 3D vector patterns naturally by minimizing mean square error:

$$E = \frac{1}{2} |e|^2, \quad (2)$$

where  $|e| = \sqrt{e_x^2 + e_y^2 + e_z^2}$  and  $e = [e_x, e_y, e_z]^T$ .

## 4 Generalized mapping in 3D

The three basic classes of transformations (*scaling, rotation, and translation*) are used to convey the dominant geometric characteristics of mapping [7]. They are used to solve many mathematical and practical engineering problems. This article presents the mappings for different combinations of above mentioned three transformations. In our computational experiments, we have considered a three layer network (2-6-2) of vector-valued neurons. It transforms every input point  $(x, y, z)$  into another point  $(x', y', z')$  in 3D space. All points in training and testing patterns are within a unit sphere centered at origin.

For all experiments considered in this section, the training input–output points lie on a straight line in the 3D space. The line has a defined reference point (mid of line). Similarly, test patterns take a set of points lying on the surface of curve (3D structure), which has a well-defined reference point, for example, center of sphere or nose tip of face. Thus, the first input of considered network (2-6-2) takes a set of points from the surface of input curve, and second input is its reference point. Similarly, the first output neuron gives the surface of transformed curve, and second output gives its reference point. In different experiments, it is observed that considering the reference point of curves yields better accuracy in results. In all transformation experiments, the training is done with 21 equal interval points lying on a line and generalization of trained network is tested over different 3D geometric structures. The considered network is run till error ‘E’ converges to 0.00001 using 3D-BP algorithm. The transformations of different 3D structures are graphically shown in various figures of this section.

In our first experiment, the considered network is trained for input–output mapping over a straight line in 3D for similarity transformation (scaling reduction of 1/2). The generalization ability of such a trained network is tested over sphere (1,681 data points) and 3D face of point cloud data (6,397 data points). The transformation result in Figs. 1 and 2 shows excellent generalization with proposed methodology.

In second experiment, Fig. 3 presents the generalization performance of considered network, which learned the composition of scaling and translation transformations. There are 451 test data points on cylinder. All patterns in 3D are contracted by factor 1/2 and displaced by  $(0, 0.2, 0)$ .

In another experiment, the input training points along the straight line are rotated over  $\pi/2$  radians clockwise and displaced by  $(0, 0, 0.2)$ . Thus, considered network has learned the composition of rotation and translation.

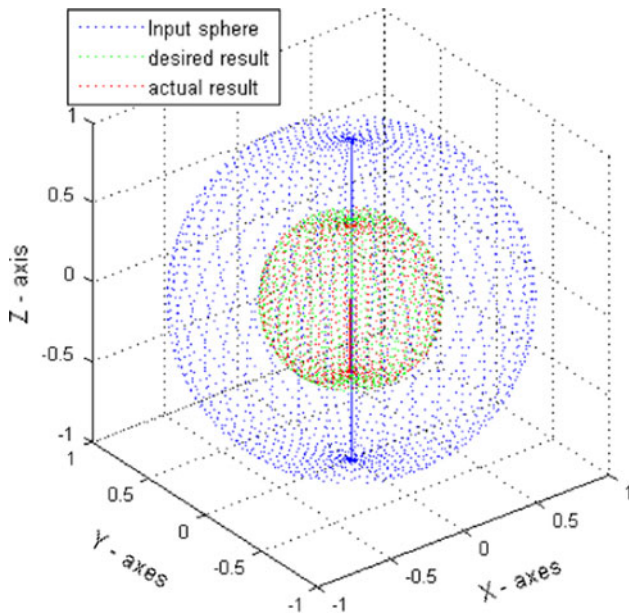


Fig. 1 Similarity transformation in 3D

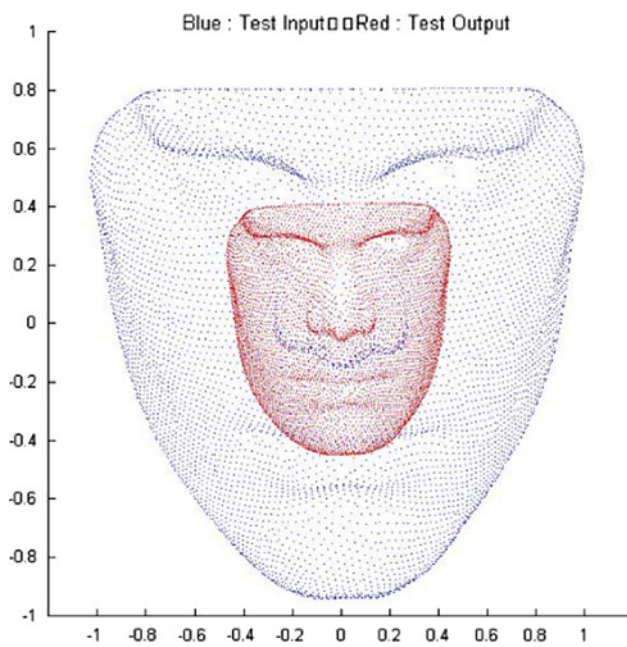


Fig. 2 Similarity transformation in 3D

Figure 4 presents the generalization capability of this network over cylinder containing 451 data points.

In last experiment, Fig. 5 presents the generalization performance of neural network which learned the composition of all three transformations. There are 451 data points on surface of a cylinder. They are contracted by factor 1/2, rotated over  $\pi/2$  radians clockwise, and displaced by (0, 0, 0.2).

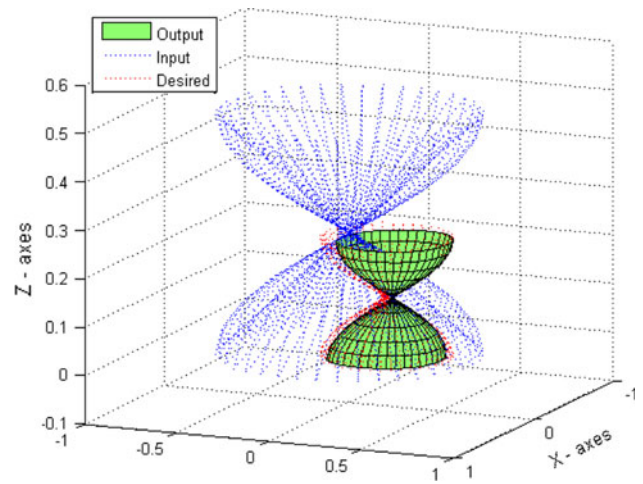


Fig. 3 Scaling and translation in 3D

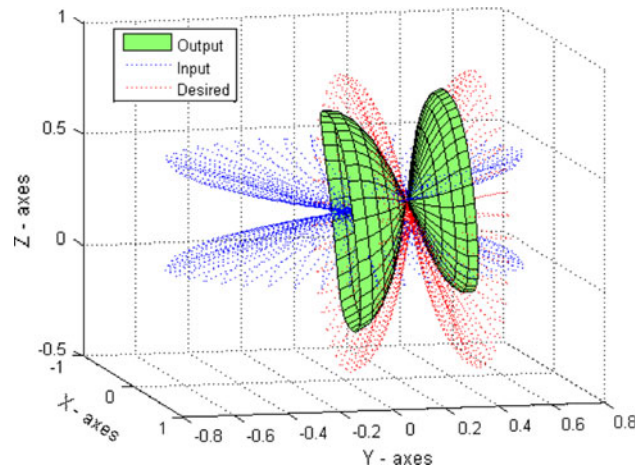


Fig. 4 Rotation and translation in 3D

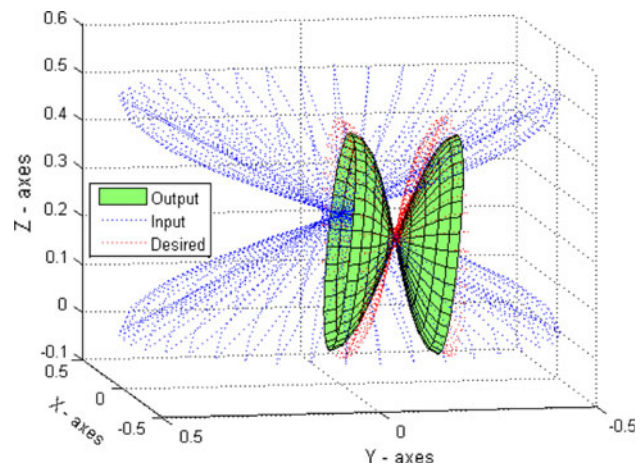
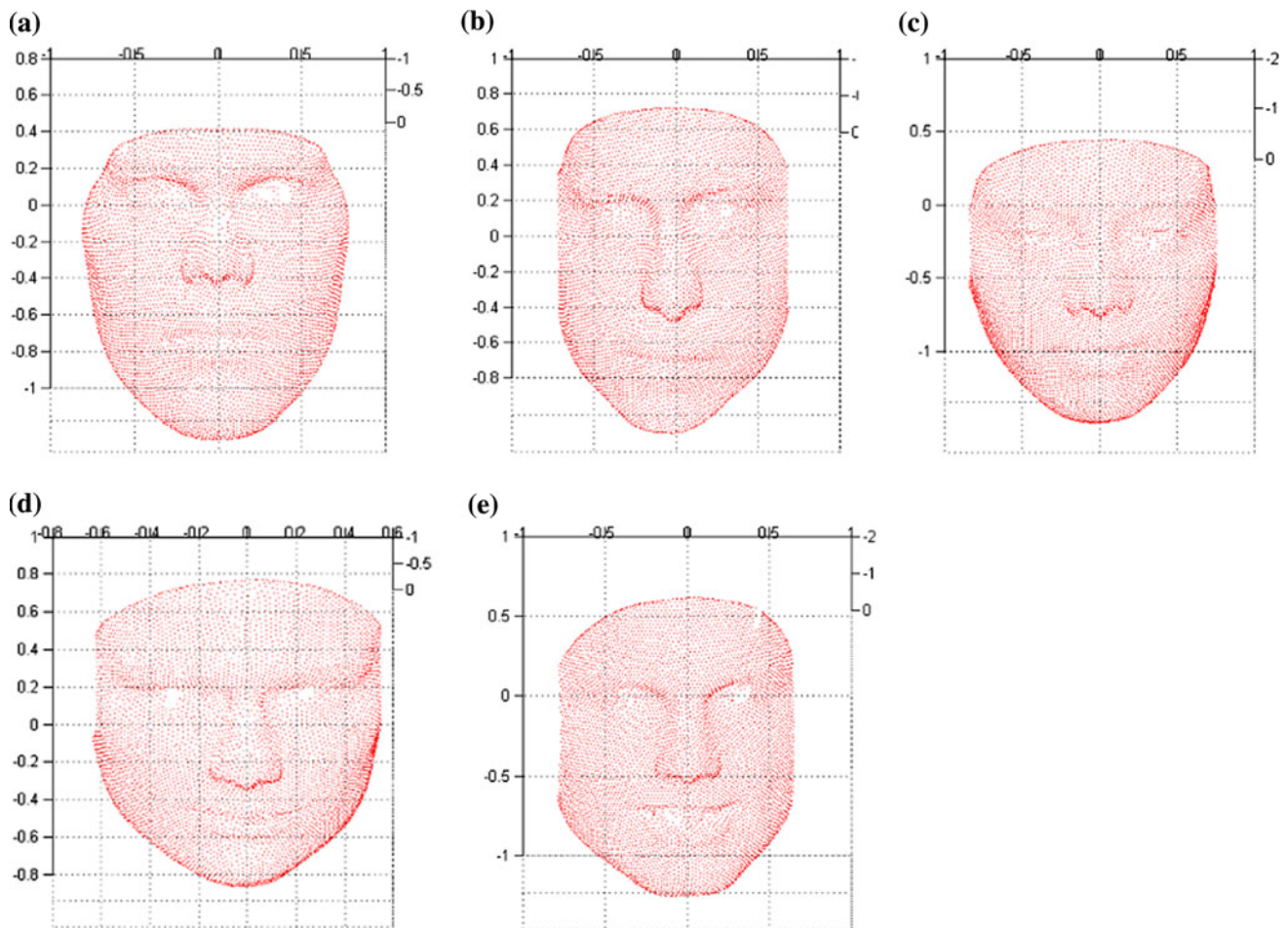


Fig. 5 Composition of all transformation in 3D





**Fig. 6** The set of five different faces

### 5 3D Face matching

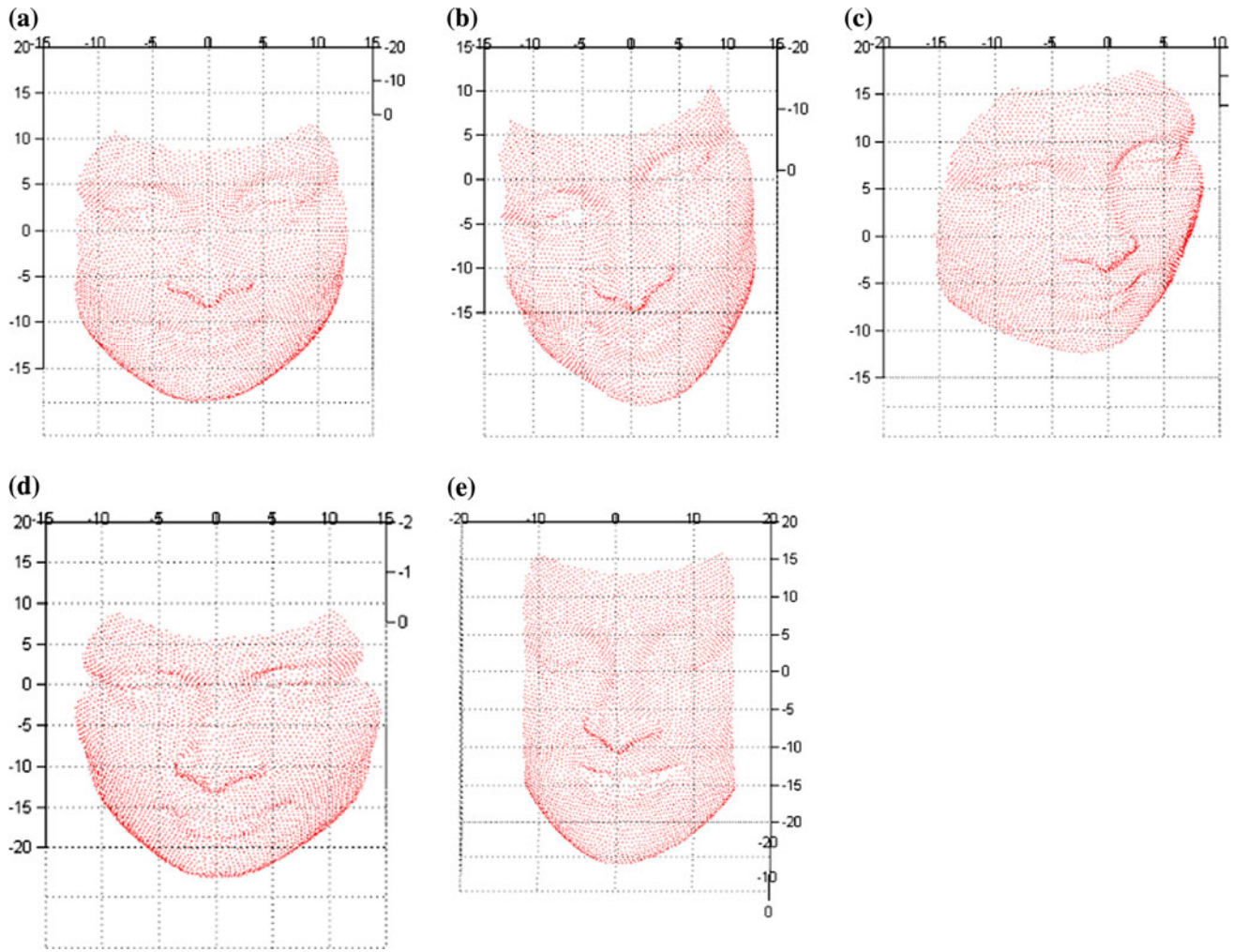
The face recognition technologies have come a long way in the last twenty years for automatic verification of identity information for secure transactions, surveillance, access control, and security task [6, 10]. But little attention was given in design of authentication system which can process 3D patterns. It is one of the main areas of current research to determine the identity of human being. This paper focuses on 3D faces recognition using high-dimensional neural network. The proposed technique is found to be pose and expression invariant. In our experiments, we have performed simulations over point cloud data of two sets of 3D faces. Given 3D faces were pre-sampled but not normalized. They were normalized using standard 3D geometric transformations (Translation, rotation, and finally scaling). It aligns each face on a same scale and at same orientation. In order to make standard alignment, the origin is translated to the nose tip. The scanned 3D face data are of front part of face and almost straight, and accordingly it is translated. A 1-2-1 network of vector-valued neurons was used in following two experiments:

**Table 1** Comparison of testing error of first set (Fig. 6) of face data

| S. No.                      | Face | Test error |
|-----------------------------|------|------------|
| Target error, MSE = 0.00005 |      |            |
| 1                           | 6(a) | 0.260990   |
| 2                           | 6(b) | 0.948520   |
| 3                           | 6(c) | 2.228900   |
| 4                           | 6(d) | 0.057350   |
| 5                           | 6(e) | 0.000078   |

*Example 1* The network was trained by a face, Fig. 6e, from first set of face data shown in Fig. 6. This face data set contains five faces of different persons. Each face contains 6,397 data points. All five faces are tested in trained network. The testing error for four other faces is much higher in comparison with the face which is used for training, as shown in Table 1. It demonstrates that four other faces are different from the face used in training.

*Example 2* The network was trained by a face, Fig. 7a, from second set of face data shown in Fig. 7. This face data



**Fig. 7** The set of five faces of same person with different orientation and poses

**Table 2** Comparison of testing error of second set (Fig. 7) of face data

| S. No.                     | Face | Test error |
|----------------------------|------|------------|
| Target error, MSE = 0.0001 |      |            |
| 1                          | 7(a) | 0.000296   |
| 2                          | 7(b) | 0.006005   |
| 3                          | 7(c) | 0.004560   |
| 4                          | 7(d) | 0.000530   |
| 5                          | 7(e) | 0.000780   |

set contains five faces of same person with different orientation and poses. Each face contains 4,663 data points. All five faces are tested in trained network. The testing error for four other faces is minimum and comparable to the face which is used for training, as shown in Table 2. It demonstrates that four other faces are same as the face used in training.

### 6 Conclusion

The generalization of transformations in different experiments brings out the fact that 3D vector-valued network preserves the angle between oriented curves, and the phase of each point on the curve is also maintained during transformations. This enables to learn 3D motion of signals in the 3D vector-valued neural network by virtue of its inherent property, as complex valued network allows learning 2D motion of signals [7]. In contrast, real-valued neural network administers 1D motion of signals, and such generalization is not possible [3]. Our experiments confirmed that a 3D back-propagation network can learn composition of transformations and has ability to generalize them with small error. The 3D back-propagation network has shown very good performance for 3D face matching, especially in varied orientations and poses of faces.

The work presented in this paper is basic and fundamental, but results are surprisingly encouraging. We expect the application of such learning machine in design of robust authentication system for large data set, which is an urgent need of today’s world scenario. The 3D mapping and 3D face matching presented in this paper will be extended to 3D object motion interpretation and 3D face reconstruction in near future.

### Appendix

#### Derivation of update rule

Consider a three layer network ( $L - M - N$ ) of vector-valued neurons, where  $L$  ( $l = 1 \dots L$ ) is the number of inputs, hidden and output layer has  $M$  ( $m = 1 \dots M$ ) and  $N$  ( $n = 1 \dots N$ ) neurons, respectively. A three-layered network can approximate any continuous non-linear mapping. All input-output signals and threshold are 3D real-valued vectors, and weights are 3D orthogonal matrices. Different symbols used are listed below:

- $W_{lm}$ : weight between  $l$ th input to  $m$ th hidden neuron
- $S_{mn}$ : weight between  $m$ th hidden neuron to  $n$ th output neuron
- $I_l = [I_{lx}, I_{ly}, I_{lz}]$ :  $l$ th input to network
- $\alpha_m = [\alpha_{mx}, \alpha_{my}, \alpha_{mz}]$ : threshold of  $m$ th hidden neuron
- $\beta_n = [\beta_{nx}, \beta_{ny}, \beta_{nz}]$ : threshold of  $n$ th output neuron
- $V_m$ : net potential of hidden layer neuron
- $V_n$ : net potential of output layer neuron
- $O_m$ : output of the  $m$ th hidden neuron
- $Y_n$ : actual output of the  $n$ th output neuron
- $Y_n^D$ : desired output of the  $n$ th output neuron
- $f$ : activation function
- $f'$ : derivative of activation function
- $\eta$ : learning rate
- $e_n$ : difference between actual and desired value at  $n$ th output
- $E$ : mean square error

The weights defined previously are three-dimensional orthogonal matrices, defined as:

$$W_{lm} = \begin{vmatrix} w_{lm}^x & -w_{lm}^y & 0 \\ w_{lm}^y & -w_{lm}^x & 0 \\ 0 & 0 & w_{lm}^z \end{vmatrix} \quad S_{mn} = \begin{vmatrix} s_{mn}^x & 0 & 0 \\ 0 & s_{mn}^y & -s_{mn}^z \\ 0 & w_{mn}^z & s_{mn}^y \end{vmatrix}$$

where,  $w_{lm}^z = \sqrt{(w_{lm}^x)^2 + (w_{lm}^y)^2}$  and  $s_{mn}^x = \sqrt{(s_{mn}^y)^2 + (s_{mn}^z)^2}$   
 The network can be defined as follows:

$$V_m = \sum_l w_{lm} I_l + \alpha_m \tag{3}$$

$$O_m = f(V_m) = [f(V_m^x), f(V_m^y), f(V_m^z)]^T \tag{4}$$

$$V_n = \sum_m s_{mn} O_m + \beta_n \tag{5}$$

$$Y_n = f(V_n) = [f(V_n^x), f(V_n^y), f(V_n^z)]^T \tag{6}$$

The weight update equations can be obtained using gradient descent on following error function (from Eq. 2):

$$E = \frac{1}{2} \sum_n |e_n|^2 \tag{7}$$

where  $|e_n| = \sqrt{e_n^{x2} + e_n^{y2} + e_n^{z2}}$  and  $e_n = [e_n^x, e_n^y, e_n^z]^T = Y_n - Y_n^D$

In 3D vector version of back-propagation algorithm, the weight update for any weight may be defined as:

$$\Delta w = \eta \begin{vmatrix} -\frac{\partial E}{\partial w^x} \\ -\frac{\partial E}{\partial w^y} \\ -\frac{\partial E}{\partial w^z} \end{vmatrix}$$

Hence, the learning rules for respective weight are as follows:

#### weight updation in output layer:

$$\Delta \beta_n = \eta \begin{vmatrix} e_n^x \cdot f'(V_n^x) \\ e_n^y \cdot f'(V_n^y) \\ e_n^z \cdot f'(V_n^z) \end{vmatrix}$$

$$\begin{vmatrix} \Delta s_{mn}^y \\ \Delta s_{mn}^z \end{vmatrix} = \eta \begin{vmatrix} \frac{s_{mn}^y}{s_{mn}^x} O_m^x & O_m^y & O_m^z \\ \frac{s_{mn}^z}{s_{mn}^x} O_m^x & -O_m^z & O_m^y \end{vmatrix} \begin{vmatrix} e_n^x \cdot f'(V_n^x) \\ e_n^y \cdot f'(V_n^y) \\ e_n^z \cdot f'(V_n^z) \end{vmatrix}$$

#### weight updation in hidden layer:

$$\begin{vmatrix} \Delta \alpha_m^x \\ \Delta \alpha_m^y \\ \Delta \alpha_m^z \end{vmatrix} = \eta \begin{vmatrix} f'(V_m^x) & 0 & 0 \\ 0 & f'(V_m^y) & 0 \\ 0 & 0 & f'(V_m^z) \end{vmatrix}$$

$$\times \sum_n \begin{vmatrix} s_{mn}^x & 0 & 0 \\ 0 & s_{mn}^y & s_{mn}^z \\ 0 & -s_{mn}^z & s_{mn}^y \end{vmatrix} \begin{vmatrix} e_n^x \cdot f'(V_n^x) \\ e_n^y \cdot f'(V_n^y) \\ e_n^z \cdot f'(V_n^z) \end{vmatrix}$$

$$\begin{vmatrix} \Delta w_{mn}^x \\ \Delta w_{mn}^y \end{vmatrix} = \begin{vmatrix} I_l^x & I_l^y & \frac{w_{lm}^x}{w_{lm}^z} I_l^z \\ -I_l^y & I_l^x & \frac{w_{lm}^y}{w_{lm}^z} I_l^z \end{vmatrix} \begin{vmatrix} \Delta \alpha_m^x \\ \Delta \alpha_m^y \\ \Delta \alpha_m^z \end{vmatrix}$$

### References

1. Nitta T (1997) An extension of the back-propagation to complex numbers. *Neural Netw* 10(8):1391–1415
2. Nitta T (1992) A 3D vector version of the back-propagation algorithm. In: *Proceedings of IJCNN’92 Beijing*, vol. 2, pp 511–516
3. Nitta T (2006) Three-dimensional vector valued neural network and its generalization ability. *Neural Inform Process* 10(10):237–242

4. Nitta T (2000) An analysis of the fundamental structure of complex-valued neurons. *Neural Process Lett* 12:239–246
5. Piazza F, Benvenuto N (1992) On the complex back-propagation algorithm. *IEEE Trans Signal Process* 40(4):967–969
6. Phillips PJ, Flynn PJ, Scruggs T, Bowyer KW, Chang J, Hoffman K, Marques J, Min J, Work W (2005) Overview of the face recognition grand challenge. In: *Proceedings of IEEE conference on computer vision and pattern recognition*
7. Tripathi BK, Kalra PK (2009) The novel aggregation function based neuron models in complex domain. *Soft Comput.* doi: [10.1007/s00500-009-0502-5](https://doi.org/10.1007/s00500-009-0502-5)
8. Tripathi BK, Chandra B, Kalra PK (2008) The generalized product neuron model in complex domain. In: *ICONIP 2008*, November 25–28 Auckland
9. Watanabe A, Miyauchi A, Miyauchi M (1994) A method to interpret 3D motion using neural networks. *IIEICE Trans Fundamental Elec Comput Sci* E77-A(8):1363–1370
10. Zhao W, Chellappa R, Rosenfeld A, Phillips PJ (2003) Face recognition: a literature survey. *ACM Comput Surv*, 35(4):339–458