Inversion ofcomplex Neural Network

Manmohan Shukla, B. K. Tripathi

Abstract*: This paper presents a novel application of complex neural network which has been modeled by the implementation of gradient descent inversion algorithm in complex domain. The methods reported prior to this work were limited to real domain only. By the learning of function mapping in complex domain, the performance of neural network has been analyzed. An improved performance of complex neural network has resulted in the development of a Novel Complex Neuron Model.

***Key words:* inversion, complex-valued neural network, gradient descent search and activation function.**

I. INTRODUCTION

Inversion is finding a set of input vectors which when applied to a system will result in an efficient and desired outcome. In inversion of neural network, an attempt is made to achieve the desired response for a fixed set of weights by the judicious selection of one or more input values [1].

The neural network learns from the training phase. As a consequence, the neural network is able to produce an output which has a specific correlation with input provided [3]. The weights are free parameter and by finding the proper set based upon minimization of some error criteria, neural network learns the functional relationship [5]. Since specific input results in specific output, a functional relationship is established.

Consider a trained neural network to map input vector ‘x’, to output vector y where w is the set of weights obtained after training.

$$y = f(x, w)$$

In inversion, the output vector y is given; w is fixed after training and procedure seeks to find out the inputs which will produce the desired output. In general, the functional mapping between input and output will be many to one [16]. So while inverting the function, there is a probability of generating the identical output for different inputs. In case, input has more number of dimensions than that of output, inversion is not unique [9].

The algorithms of search methods for inversion can be broadly classified into three categories as appended below [14]:

- Exhaustive search algorithm
- Single-element search algorithm
- Evolutionary algorithm

Out of the above search methods, exhaustive search algorithm is aptly suited [8] only when the dimensionality and allowable range of input variables is low. Gradient descent search [13] is one of the single-element search methods which search for one input vector at a time. The convergence at a solution depends upon the initial search position [4]. Finding out all the input combinations is difficult and laborious. On the other hand, evolutionary search algorithm [15] generates numerous initial points in the search space at a time and results in many solutions simultaneously.

II. CVNN Inversion

The gradient descent algorithm used in complex back-propagation algorithm can be used in the inversion of CVNN. As described in [2], the CVNN can learn the geometric transformations like rotation, translation etc. For example, a CVNN trained for rotation of 45o, gives an output vector in complex-inversion algorithm that can find out the input vector which is displaced by -45o with respect to output. This property can be used in applications like the response of robot arm based on inverse kinematics [7]. Once trained to learn the forward mapping, the same CVNN can be used to find out the configuration of robot arm to reach the output position co-ordinates.

2.1 Algorithm

A random value is assigned initially to the input vector x0. Later, successive iterations update its value according to the following equation.

$$x_k^{t+1} = x_k^t + \eta \frac{\partial E}{\partial x_k^t}$$

$$\frac{\partial E}{\partial x_k^t} = \delta_k, \ldots\ldots\ldots\ldots k \in I \qquad (1.1)$$

Where, the step size and iteration index are represented by ‘η’ and ‘t’ respectively.

The rate of error change with respect to input can be found out by using following equations.

$$\mathrm{Re}[\delta^k] = \begin{cases} \mathrm{Re}[\phi'(y_k)] \times \mathrm{Re}[(o_k - y_k)] \ldots\ldots\ldots\ldots k \in O \\ \mathrm{Re}[\phi'(y_k)] \times \mathrm{Re}[\sum \delta^k \times w_{jk}] \ldots\ldots\ldots\ldots\ldots\ldots k \in I, H \end{cases} \qquad (1.2)$$

$$\mathrm{Im}[\delta^k] = \begin{cases} \mathrm{Im}[\phi'(y_k)] \times \mathrm{Im}[(o_k - y_k)] \ldots\ldots\ldots\ldots k \in O \\ \mathrm{Im}[\phi'(y_k)] \times \mathrm{Im}[\sum \delta^k \times w_{jk}] \ldots\ldots\ldots\ldots\ldots\ldots k \in I, H \end{cases}$$

where, I represents the set of inputs neurons; H represents set of hidden neurons and O represents the set of output neurons.

wjk the weight between neuron j to k;

ϕ' the derivative of activation function;

o_k the desired outcome of kth neuron;

Manuscript published on 28 February 2019.
***Correspondence Author(s)**

Manmohan Shukla, Computer Science and Engineering Dept., PSIT Kanpur, India
Marion Sinclair, Science and Engineering Dept. HBTU Kanpur, India

y_k the outcome of kth neuron.

Here, δ is a complex value. It is computed in the reverse sequence from output layer to input layer.

III. COMPLEX NEW NEURON MODEL INVERSION:

The inversion algorithm can be applied on complex new neuron model also. The algorithm is same as the gradient descent for CVNN feed-forward model. Only the weight adjustment in the input layer involving the product neuron weights are altered [11].

The key task is to find out the rate of change of error with respect to the inputs. The equations are given as below:

$$\mathrm{Re}[\delta^k]=\begin{cases}\mathrm{Re}[\phi'(y_k)]\times\mathrm{Re}[(o_k-y_k)]...........k\in O\\ \mathrm{Re}[\phi'(y_k)]\times\mathrm{Re}\left[\sum[(\delta^k\times w_{jk})+w'_{jk}(\delta^k\times\frac{\sum(\prod_{k\neq l}x_k w_{jk}x_l w_{jl})}{x_k(n)})]\right]................k\in I,H\end{cases}$$

$$\mathrm{Im}[\delta^k]=\begin{cases}\mathrm{Im}[\phi'(y_k)]\times\mathrm{Im}[(o_k-y_k)]...........k\in O\\ \mathrm{Im}[\phi'(y_k)]\times\mathrm{Im}\left[\sum[(\delta^k\times w_{jk})+w'_{jk}(\delta^k\times\frac{\sum(\prod_{k\neq l}x_k w_{jk}x_l w_{jl})}{x_k(n)})]\right]................k\in I,H\end{cases}$$

where, represents the weight calculated from the input ‘i’ to the product neuron of jth block. Remaining subscripts are same as CVNN inversion.

IV. INVERSION OF COMPLEX NEURAL NETWORK

4.1 Inversion of Similarity transformation:

The trained model of CVNN for similarity transformation is used for inversion. The network is trained for the transformation to reduce the magnitude of each point by a factor of 0.5 as presented in figure 1:

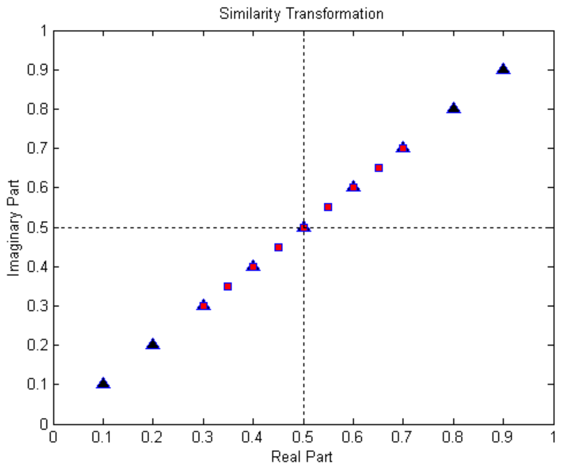

Figure 1 Training inputs and outputs

The network is presented with the output points (triangles) as depicted in figure 2. The actual inputs are represented as squares, and circles denote the inverted input. As evident from the figure, the inverted inputs match the actual inputs very closely.

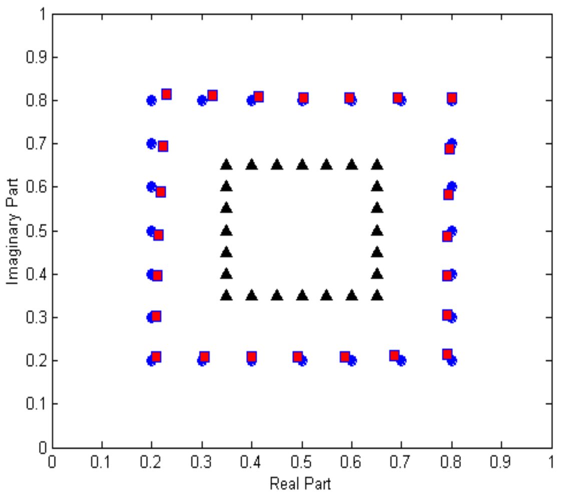

Figure 2 Inversion Results for similarity transformation: Outputs (triangles), actual inputs (squares) and inverted inputs (circles).

4.2 Inversion of Rotation transformation:

The trained network for rotation by 45o counterclockwise is used for inversion. The network is presented with the output (triangles), as presented in figure 3. The aim of inversion is to find out the input combination which when applied to the network will produce the given output [10] . The important thing is that even though the network is trained for anticlockwise rotation from imaginary axis, the network inversion produces anticipated results when the output is centered on real axis [6].

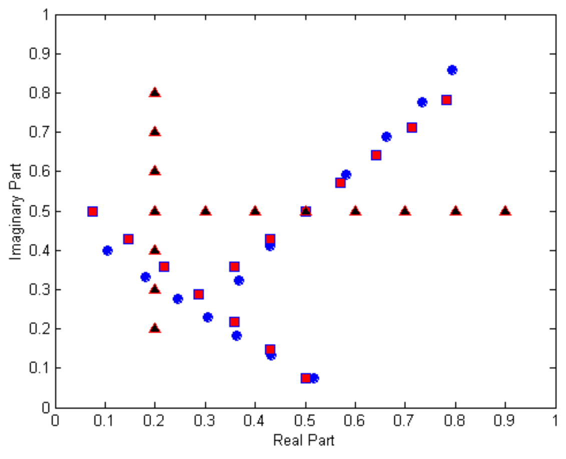

Figure 3 Inversion Results for rotation transformation: Outputs (triangles), actual inputs (squares) and inverted inputs (circles).

V. INVERSION OF COMPLEX NEW NEURON MODEL

The network is trained for the patterns presented in table 4.

Table 4 Training patterns for CNNM inversion

Input 1	Input 2	Input 3	Output
0 + i1	1 + i1	1 + i0	0 + i0
1 + i1	1 + i0	0 + i0	0 + i1
0 + i0	0 + i1	1 + i1	1 + i0
1 + i0	0 + i0	0 + i1	1 + i1

The four distinct complex values in Table 4 represent a square in the complex plane. If the three corners of this square are the inputs, then the forth corner is given as the output as represents in the figure 5. Architecture of 3-10-1 is used and the network is trained for the input patterns.

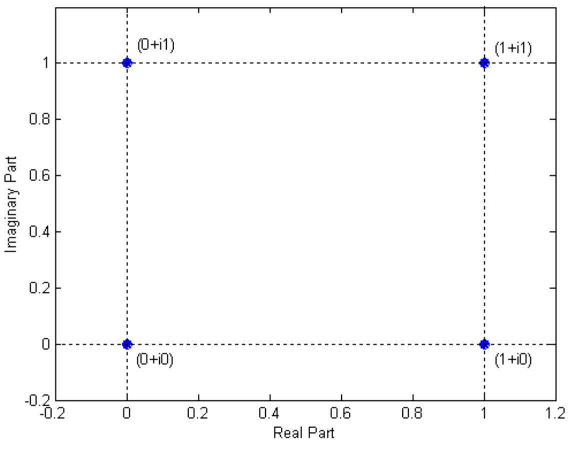

Figure 5, Complex input and output data points

In inversion, the output is presented to the network and the inverted inputs produced are given in table 6. Input update is continued till the RMS error is reduced till 0.1. The inputs obtained by the inversion are not exactly the training inputs for respective outputs [12]. This is on account of the fact that, although the training points are corners of the square in complex plane, the mapping learned by the network is continuous [10]. However, if the inversion results are interpreted with the help of following rules:

$$if\ \mathrm{Re}(i) < 0.5 then\ \mathrm{Re}(i) = 0$$

$$if\ \mathrm{Re}(i) \geq 0.5 then\ \mathrm{Re}(i) = 1$$

$$if\ \mathrm{Im}(i) < 0.5 then\ \mathrm{Im}(i) = 0$$

$$if\ \mathrm{Im}(i) \geq 0.5 then\ \mathrm{Im}(i) = 1$$

then, the inverted inputs are equivalent to the inputs for which the network is trained.

Table 6 Test results

Output	Inverted Input 1	Inverted input 2	Inverted input 3
1 + i0	-0.4227 + i0.1415	-0.3207 + i0.7315	0.5675 + i0.7355
0 + i0	-0.2138 + i0.8731	0.6172 + i0.8398	0.6047 - i0.3355
0 + i1	0.689 + i0.8069	0.8091 - i0.2726	0.0085 - i0.1335
1 + i1	0.5756 - i0.1504	-0.01695 - i0.4157	-0.0352 + i0.7234

VI. CONCLUSION

The inversion of the complex valued neural network gives good results for geometric transformation problems. This ability can be used in mapping inverse kinematics of link robots.

REFERENCES

1. H. Leung and S. Haykin, “The Complex Backpropagation Algorithm”, IEEE Trans.On Signal Processing, Vol. 39, No. 9, September (1991).
2. G. M. Georgiou and C. Koutsougeras, “Complex Domain Backpropagation”, IEEE Trans. on Circuits and Systems-II : Analog and Digital Signal Processing, Vol 39, No. 5, May (1992).
3. N. Benvenuto and F. Piazza, “On the Complex Backpropagation Algorithm”, IEEE Trans. On Signal Processing, Vol. 40, No. 4, April (1992).
4. T. Nitta, “A Back-propagation Algorithm for Neural Networks Based on 3D Vector Product”, Proc. of 1993 International Joint Conference on Neural Networks.
5. S. Lee and R. M. Kil, “Inverse Mapping of Continuous Functions Using Local and Global Information”, IEEE Tran. on Neural Networks, Vol. 5, No. 3, May(1994).

6. T. Nitta, “A Quaternary Version of the Back-propagation Algorithm”, ICNN 1995.
7. S. Jankowski, A. Lozowski and J. Zurada, “Complex-Valued Multistate Neural Associative Memory”, IEEE Trans. on Neural Networks, Vol. 7, No. 6, November 1996.
8. C.You and D. Hong, “Adaptive Equalization Using the Complex Backpropagation Algorithm”, IEEE International Conference on Neural Networks, Vol. 4, Jun 1996
9. T. Nitta, “An Extension of the Back-Propagation Algorithm to Complex Numbers”, Neural Networks, Vol. 10, No. 8, 1997.
10. M. H. Hassoun, Fundamentals of Artificial Neural Networks, New Delhi, Prentice Hall of India, 1998.
11. C. A. Jensen, R. D. Reed, R. J. Marks, M. A. El-sharkawi, J. Jung, R. T. Miyamoto, G. M. Anderson and C. J. Eggen, “Inversion of Feedforward Neural Networks: Algorithms and Applications”, Proceedings of the IEEE, Vol. 87, No. 9, September(1999).
12. C. Lin and C. Li, “A sum-of-product neural network (SOPNN)”, Neurocomputing, 2000.
13. T. Nitta, “An Analysis of the Fundamental Structure of Complex-Valued Neurons”, Neural Processing Letters12, pp.239-246, 2000.
14. T. Nitta, “Generalization of the Complex-valued Neural Networks with the Orthogonal Decision Boundary”, KES 2002.
15. C. Igel and M. Husken, “Improving the Rprop Learning Algorithm”, Proc. of the Second International Symposium on Neural Computation, pp. 115-121, 2000.
16. M. Sinha, P. K. Kalra and K. Kumar, “Parameter estimation using compensatory neural networks”, Sadhana, Vol. 25, April 2000.