Inversion of Complex Valued Neural Networks Using Complex Back-propagation Algorithm

Anita S. Gangal, P.K. Kalra, and D.S.Chauhan

***Abstract*—**This paper presents the inversion of complex valued neural networks. Inversion means predicting the inputs for given output. We have tried inversion of complex valued neural network using complex back-propagation algorithm. We have used split sigmoid activation function both for training and inversion of neural network to overcome the problem of singularities. Since inversion is a one to many mapping, means for a given output there are number of possible combinations of inputs. So in order to get the inputs in the desired range conditional constraints are applied to inputs. Simulation on benchmark complex valued problems support the investigation.

***Keywords*—**activation function, back propagation, complex valued neural network, inversion

I. INTRODUCTION

The complex valued neural networks are those neural networks whose weights, threshold values, input and output signals all are complex numbers. The complex valued neural network is extending its field both in theories and applications. Typically signal processing, image processing, radar imaging, array antenna, and mapping inverse kinematics of robots are the areas where such requirements exist. Neural network inversion procedure seeks to find one or more input values that produce a desired output response. For inversion of real valued neural network researchers have worked with many approaches. These inversion algorithms can be placed into three broad classes:

1) Exhaustive Search
2) Multi-component Evolutionary Method
3) Single-element Search Method

In choosing among inversion techniques for real valued neural networks, Exhaustive Search should be considered when both the dimensionality of the input and allowable range of each input variable are low. The simplicity of the approach coupled with the swiftness in which a layered perceptron can be executed in the feedforward mode makes this approach even more attractive as computational speed increases. Multicomponent Evolutionary method proposed by Reed and Marks [1] on the other hand, seeks to minimize the objective function using numerous search points in turn resulting in numerous solutions. This method results in population of initial points in the search space at a time and new points are generated in the input space to replace existing points so as to explore all the solutions. Single element search method for inversion of real valued neural network was first introduced by Williams [2] and then Kinderman and Linden [3]. They used this to extract codebook vectors for digits. This method of inversion involves two main steps: first training the network and the second step is inversion. During the training neural network is trained to learn a mapping from input to output with the help of training data. The weights are the free parameters and by finding the proper set by minimizing some error criterion, neural network learns a functional relationship between the inputs and the outputs. All the weights are fixed after training of neural network. After training, the network is initialized with a random input vector. Output is calculated, compared with the given output. Error is calculated. This error is back propagated to minimize the error function and the input vector is updated. This iterative process continues till the error is less than the minimum set value. Eberhart and Dobbins [4] applied it to invert the trained real valued neural network for the diagnosis of appendicitis. Jordan and Rumelhart [5] have proposed a method to invert the feed forward real valued neural network. They tried to solve the inverse kinematics problems for redundant manipulators. There approach is a two-stage procedure. In the first stage, a network is trained to approximate the forward mapping. In the second stage, a particular inverse solution is obtained by connecting another network with the previously trained network in series and learning an identity mapping across the composite network. Behera, Gopal, Chaudhary [6] used real valued neural network inversion in the control of multilink robot manipulators. They have developed an inversion algorithm for inverting radial basis function (RBS) neural networks which is based on an extended Kalman filter. Bio-Liang Lu, Hajime, and Nishikawa [7] have formulated the inversion problem as a nonlinear programming problem and a separable programming problem or a linear programming problem according to the architectures of the real valued network to be inverted.

II. INVERSION OF REAL VALUED NEURAL NETWORK USING BACK-PROPAGATION ALGORITHM

Inversion is finding a set of input vectors for given output; which when applied to a system will produce the same output. The search is initialized with a random input vector x_i^0. The iterative inversion algorithm consists of two passes of computation first, the forward pass and second, the backward

Manuscript received November 29, 2008.

Anita S. Gangal is with Uttar Pradesh Technical University, Lucknow, India. (Phone:+91-9450141454; email: anita.sethia@yahoo.co.in)

P. K. Kalra is Professor and Head of Electrical Engineering Department, Indian Institute of Technology, Kanpur, India, (email: kalra@iitk.ac.in)

D.S.Chauhan is Vice Chancellor of J P University of Information Technology, Waknaghat, Solan, India, (email: pdschauhan@gmail.com)

pass. In the forward pass the output is calculated for the randomly initialized inputs using trained network. The error signal between the given output and the actual output is calculated. In the backward pass, the error signal is back propagated to the input layer through the network layer by layer, and the input is adjusted to decrease the output error. If x_i^t is the i^{th} component of the input vector after 't' iterations, then gradient descent suggests the recursion

$$x_i^{t+1} = x_i^t - \eta \frac{\partial E}{\partial x_i^t} \tag{1}$$

Where, η is the learning rate constant. Iteration for inversion can be solved as

$$\frac{\partial E}{\partial x_i} = \delta_i, \qquad i \in I \tag{2}$$

Where, for any neuron

$$\delta_j = \phi'(o_j)(o_j - d_j); \qquad j \in O \tag{3}$$

$$= \phi'(o_j) \sum_{m \in H,O} \delta_m w_{mj}; \qquad j \in I, H$$

I,H,O number of input, hidden and output neurons

w_{jm} synaptic weight from neuron m to neuron j

ϕ_j' derivative of the j^{th} neuron activation function

o_j actual output of the j^{th} neuron

d_j desired output of the j^{th} neuron

The neuron derivative δ_j in "(3)" is solved in a backward order from output to input similar to the standard back propagation algorithm.

III. COMPLEX VALUED NEURAL NETWORK

The complex plane is very much different from real line. Complex plane is two dimensional with respect to real numbers and is one dimensional with respect to complex number. The order that existed on the real numbers is absent in the set of complex numbers hence, no two numbers can be compared as being big or small with respect to each other but their magnitudes can be compared which are real values. The complex numbers have a magnitude associated with them and a phase that locates the complex number uniquely on the plane. The generalization of real valued algorithms cannot be simply done as complex valued algorithm. Complex version of back-propagation (CVBP) algorithm made its first appearance when Widrow, McCool and Ball [8] announced their complex least mean squares (LMS) algorithm. Kim and Guest [9] published a complex valued learning algorithm for signal processing application. Georgiou and Koutsougeras [10] published another version of CVBP incorporating a different activation function and have shown if real valued algorithms be simply done as complex valued algorithm then singularities and other such unpleasant phenomena may arise. In the complex back propagation algorithm suggested by Leung and Haykins [11], the nonlinear function maps the complex value without splitting it into the real and imaginary part

$$f(z) = \frac{1}{1+e^{-z}} \tag{4}$$

Where,

$$z = x + iy$$

The function *f(z)* is holomorphic complex function. But according to the Liouville's theorem, a bounded holomorphic function in the complex plane C is a constant. So the attempt to extend the sigmoidal function to complex plane is met with the difficulty of singularities in the output. To deal, with this difficulty A Prashanth [12] suggested that the input data should be scaled to some region in complex domain. Although the input data can be scaled but there is no limit over the values the complex weights can take hence it is difficult to implement it. To overcome this problem split activation function is used both for training and inversion of complex valued neural network (CVNN). An extensive study of CVBP was reported by Nitta [13]. Decision boundary of a single complex valued neuron consists of two hyper-surfaces which intersect orthogonally, and divide a decision region into four equal sections. If both the absolute values of real and imaginary parts of the net inputs to all hidden neurons are sufficiently large, then the decision boundaries for real and imaginary parts of an output neuron in three layered complex valued neural network intersect orthogonally. The average learning speed of complex BP algorithm is faster than that of real BP algorithm. The standard deviation of the learning speed of complex BP is smaller than that of the real BP. Hence the complex valued neural network and the related algorithm are natural for learning of complex valued patterns. The complex BP algorithm can be applied to multilayered neural networks whose weights, threshold values, inputs and outputs all are complex numbers. In split activation function, nonlinear function is applied separately to real and imaginary parts of the aggregation at the input of the neuron

$$\phi_c(z) = \phi_R(x) + i\phi_R(y) \tag{5}$$

Where,

$$\phi_R(a) = \frac{1}{1+e^{-a}} \tag{6}$$

Here sigmoid activation function is used separately for real and imaginary part. This arrangement ensures that the magnitude of real and imaginary part of *f(z)* is bounded between 0 and 1. But now the function *f(z)* is no longer holomorphic, because the Cauchy-Riemann equation does not hold i.e.

$$\frac{\partial f(z)}{\partial x}+i\frac{\partial f(z)}{\partial y}=(1-f_R(x))f_R(x)+i(1-f_R(y))f_R(y)\neq 0 \quad (7)$$

So, effectively the holomorphy is compromised for boundedness of the activation function.

We have tried the inversion of a three layered complex valued neural network shown in Fig. 1

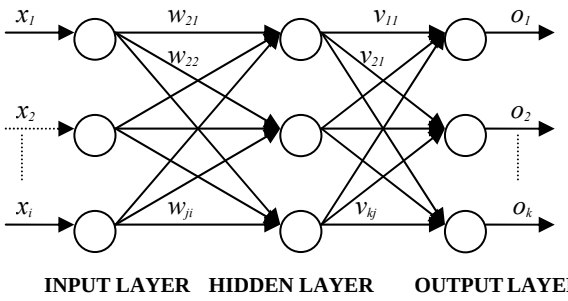

Fig. (1) complex valued neural network

In this complex valued neural network:

L number of input layer neurons
M number of hidden layer neurons
N number of output layer neurons
x_i output value of input neuron i (input)
z_j output of hidden layer neuron j
o_k output of the output neuron k
w_{ji} weight between input layer neuron i and hidden layer neuron j
v_{kj} weight between hidden layer neuron j and output layer neuron k
θ_j threshold / bias of hidden layer neurons
γ_k threshold / bias of output layer neurons

Training is done with a given set of input and output data to learn a functional relationship between input and output.

Internal potential of hidden neuron j :

$$u_j=\sum_{i=1}^{L}(w_{ji}x_i)+\theta_j=\mathrm{Re}[u_j]+i\,\mathrm{Im}[u_j] \quad (8)$$

Output of hidden neuron j:

$$z_j=\phi(u_j)=\frac{1}{1+e^{-\mathrm{Re}[u_j]}}+i\frac{1}{1+e^{-\mathrm{Im}[u_j]}}=\mathrm{Re}[z_j]+i\,\mathrm{Im}[z_j] \quad (9)$$

Internal potential of output neuron k:

$$s_k=\sum_{j=1}^{M}(v_{kj}z_j)+\gamma_k=\mathrm{Re}[s_k]+i\,\mathrm{Im}[s_k] \quad (10)$$

Output of output neuron k:

$$o_k=\phi(s_k)=\frac{1}{1+e^{-\mathrm{Re}[S_k]}}+i\frac{1}{1+e^{-\mathrm{Im}[S_k]}}=\mathrm{Re}[y_k]+i\,\mathrm{Im}[y_k] \quad (11)$$

Error $e_k=o_k-d_k$ (12)

Sum squared error for the outputs

$$E=\frac{1}{2}\sum_{k=1}^{N}\left|o_k-d_k\right|^2 \quad (13)$$

For real time application the cost function of the network is given by

$$E=\frac{1}{2}\sum_{k=1}^{N}\left|e_k\right|^2=\frac{1}{2}\sum e_k e_k^*=\frac{1}{2}\sum_{k=1}^{N}(\mathrm{Re}[e_k]^2+\mathrm{Im}[e_k]^2) \quad (14)$$

$(.)^*$ denotes the complex conjugate.

'*E*' is a real-valued function, and we are required to derive the gradient of Ep w.r.t. both the real and imaginary part of the complex weights

.

$$\nabla_{w_{ji}}E=\frac{\partial E}{\partial\,\mathrm{Re}[w_{ji}]}+i\frac{\partial E}{\partial\,\mathrm{Im}[w_{ji}]} \quad (15)$$

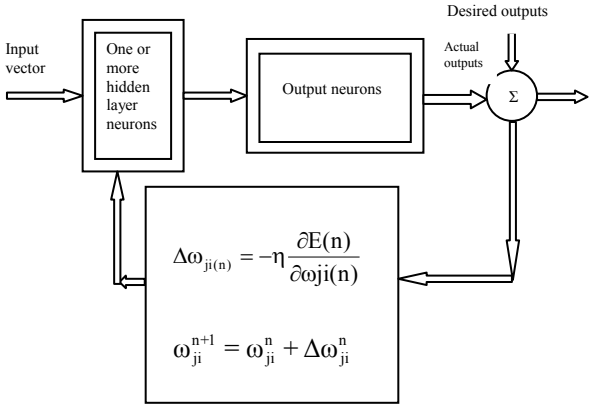

Fig 2 weight update during training

The training process of neural network is shown in Fig. 2. During training the network cost function E is minimized by recursively altering the weight coefficient based on gradient descent algorithm, given by

$$w_{ji}(t+1)=w_{ji}(t)+\Delta w_{ji}(t)=w_{ji}(t)-\eta\nabla_{w_{ji}}E \quad (16)$$

Where 't' is the number of iterations and 'η' is the learning rate constant. Once the network is trained for the given training data, all the weights are fixed.

IV. INVERSION OF COMPLEX VALUED NEURAL NETWORK

Once the network is trained, the weights are fixed. Inversion is the procedure that seeks to find out the inputs which will produce the desired output. We have used complex back-propagation algorithm for inversion. The input vector x^0 is initialized to some random value. The output of this trained network is calculate with this initialized input vector and is compared with the desired output. The error between actual output and the desired output is calculated. This error is back propagated to minimize the error function and the input vector is updated as shown in Fig. 3.

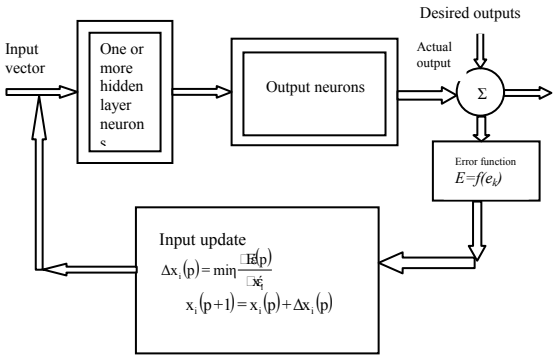

Fig 3 input update for inversion

This iterative process is continued till the error becomes less then the minimum defined error according to the following equation

$$x_i^{t+1} = x_i^t + \eta \frac{\partial E}{\partial x_i^t} \tag{17}$$

Cost function E is a scalar quantity which is minimized by modifying input.

$$\Delta x_i = -\eta\left(\frac{\partial E}{\partial \mathrm{Re}[x_i]} + i\frac{\partial E}{\partial \mathrm{Im}[x_i]}\right) = -\eta\sum_{j=1}^{m}\left\{\begin{array}{l}\left(\dfrac{\partial \mathrm{E}}{\partial \mathrm{Re}[\mathrm{u_i}]}\dfrac{\partial \mathrm{Re}[\mathrm{u_i}]}{\partial \mathrm{Re}[\mathrm{u_i}]} + \dfrac{\partial \mathrm{E}}{\partial \mathrm{Im}[\mathrm{u_i}]}\dfrac{\partial \mathrm{Im}[\mathrm{u_i}]}{\partial \mathrm{Re}[\mathrm{u_i}]}\right) \\ + i\left(\dfrac{\partial \mathrm{E}}{\partial \mathrm{Re}[\mathrm{u_i}]}\dfrac{\partial \mathrm{Re}[\mathrm{u_i}]}{\partial \mathrm{Im}[\mathrm{x_i}]} + \dfrac{\partial \mathrm{E}}{\partial \mathrm{Im}[\mathrm{u_i}]}\dfrac{\partial \mathrm{Im}[\mathrm{u_i}]}{\partial \mathrm{Im}[\mathrm{x_i}]}\right)\end{array}\right\} \tag{18}$$

From "(8)" internal potential of hidden neuron j:

$$u_j = \sum_{i=1}^{L}\left\{\begin{array}{l}\left(\mathrm{Re}[w_{ji}]\mathrm{Re}[x_i] - \mathrm{Im}[w_{ji}]\mathrm{Im}[x_i]\right) \\ + i\left(\mathrm{Re}[w_{ji}]\mathrm{Im}[x_i] + \mathrm{Im}[w_{ji}]\mathrm{Re}[x_i]\right)\end{array}\right\} \tag{19}$$

From "(18)" the input update is given by,

$$\Delta x_i = -\eta\sum_{j=1}^{M}\left\{\begin{array}{l}\left(\dfrac{\partial E}{\partial \mathrm{Re}[u_i]}\mathrm{Re}[w_{ji}] + \dfrac{\partial E}{\partial \mathrm{Im}[u_i]}\mathrm{Im}[w_{ji}]\right) \\ + i\left(\dfrac{\partial E}{\partial \mathrm{Re}[u_i]}\left(-\mathrm{Im}[w_{ji}]\right) + \dfrac{\partial E}{\partial \mathrm{Im}[u_i]}\mathrm{Re}[w_{ji}]\right)\end{array}\right\}$$

$$= -\eta\sum_{\mathrm{j=1}}^{\mathrm{M}}\left\{\begin{array}{l}\left(\dfrac{\partial \mathrm{E}}{\partial \mathrm{Re}[\mathrm{u_i}]}\left(\mathrm{Re}[\mathrm{w_{ji}}] - \mathrm{i}\,\mathrm{Im}[\mathrm{w_{ji}}]\right)\right) \\ + \left(\dfrac{\partial \mathrm{E}}{\partial \mathrm{Im}[\mathrm{u_i}]}\left(\mathrm{Im}[\mathrm{w_{ji}}] + \mathrm{i}\,\mathrm{Re}[\mathrm{w_{ji}}]\right)\right)\end{array}\right\}$$

$$= -\eta\sum_{\mathrm{j=1}}^{\mathrm{M}}\left\{\begin{array}{l}\left(\dfrac{\partial \mathrm{E}}{\partial \mathrm{Re}[\mathrm{u_i}]}\left(\mathrm{Re}[\mathrm{w_{ji}}] - \mathrm{i}\,\mathrm{Im}[\mathrm{w_{ji}}]\right)\right) \\ + \mathrm{i}\left(\dfrac{\partial \mathrm{E}}{\partial \mathrm{Im}[\mathrm{u_i}]}\left(\mathrm{Re}[\mathrm{w_{ji}}] - \mathrm{i}\,\mathrm{Im}[\mathrm{w_{ji}}]\right)\right)\end{array}\right\} \tag{20}$$

$$= -\eta\sum_{\mathrm{j=1}}^{\mathrm{M}}\left\{\mathrm{w_{ji}^{*}}\left(\frac{\partial \mathrm{E}}{\partial \mathrm{Re}[\mathrm{u_i}]} + \mathrm{i}\frac{\partial \mathrm{E}}{\partial \mathrm{Im}[\mathrm{u_i}]}\right)\right\}$$

The partial derivative of the cost function w.r.t. Re [u_i] is:

$$\frac{\partial E}{\partial \mathrm{Re}[u_i]} = \frac{\partial E}{\partial \mathrm{Re}[z_j]}\frac{\partial \mathrm{Re}[z_j]}{\partial \mathrm{Re}[u_i]} + \frac{\partial E}{\partial \mathrm{Im}[z_j]}\frac{\partial \mathrm{Im}[z_j]}{\partial \mathrm{Re}[u_i]} \tag{21}$$

From "(9)" we get

$$\frac{\partial \mathrm{Im}[z_j]}{\partial \mathrm{Re}[u_i]} = 0$$

$$\frac{\partial \mathrm{Re}[\mathrm{z_j}]}{\partial \mathrm{Re}[\mathrm{u_j}]} = \left\{\mathrm{Re}[\mathrm{z_j}]\left(1 - \mathrm{Re}[\mathrm{z_j}]\right)\right\}$$

$$\frac{\partial E}{\partial \mathrm{Re}[z_j]} = \sum_{k=1}^{N}\left(\frac{\partial E}{\partial \mathrm{Re}[e_k]}\frac{\partial \mathrm{Re}[e_k]}{\partial \mathrm{Re}[z_j]}\right) + \sum_{k=1}^{N}\left(\frac{\partial E}{\partial \mathrm{Im}[e_k]}\frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Re}[z_j]}\right)$$

$$= \sum_{k=1}^{N}\mathrm{Re}[e_k]\left(\frac{\partial \mathrm{Re}[e_k]}{\partial \mathrm{Re}[z_j]}\right) + \sum_{k=1}^{N}\mathrm{Im}[e_k]\left(\frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Re}[z_j]}\right) \tag{22}$$

$$\frac{\partial \mathrm{Re}[\mathrm{e_k}]}{\partial \mathrm{Re}[\mathrm{z_j}]} = \frac{\partial \mathrm{Re}[\mathrm{e_k}]}{\partial \mathrm{Re}[\mathrm{s_k}]}\frac{\partial \mathrm{Re}[\mathrm{s_k}]}{\partial \mathrm{Re}[\mathrm{z_j}]} + \frac{\partial \mathrm{Re}[\mathrm{e_k}]}{\partial \mathrm{Im}[\mathrm{s_k}]}\frac{\partial \mathrm{Im}[\mathrm{s_k}]}{\partial \mathrm{Re}[\mathrm{z_j}]}$$

From "(11)"and "(12)" we get

$$\frac{\partial \mathrm{Re}[\mathrm{e_k}]}{\partial \mathrm{Im}[\mathrm{s_k}]} = 0$$

$$\frac{\partial \mathrm{Re}[\mathrm{e_k}]}{\partial \mathrm{Re}[\mathrm{s_k}]} = -\mathrm{Re}[\mathrm{y_k}]\left(1 - \mathrm{Re}[\mathrm{y_k}]\right)$$

From "(10)", we get

$$s_k = \sum_{j=1}^{M} \left\{ \begin{array}{l} \left(\mathrm{Re}[v_{kj}]\mathrm{Re}[z_j] - \mathrm{Im}[v_{kj}]\mathrm{Im}[z_j] \right) \\ + i\left(\mathrm{Re}[v_{kj}]\mathrm{Im}[z_j] + \mathrm{Im}[v_{kj}]\mathrm{Re}[z_j] \right) \end{array} \right\}$$

From "(10)" "(11)"and "(12)" we get

$$\frac{\partial \mathrm{Re}[e_k]}{\partial \mathrm{Re}[z_j]} = -\mathrm{Re}[y_k](1-\mathrm{Re}[y_k])\mathrm{Re}[v_{kj}]$$

$$\frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Re}[z_j]} = \frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Re}[s_k]}\frac{\partial \mathrm{Re}[s_k]}{\partial \mathrm{Re}[z_j]} + \frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Im}[s_k]}\frac{\partial \mathrm{Im}[s_k]}{\partial \mathrm{Re}[z_j]}$$

$$\frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Re}[s_k]} = 0$$

$$\frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Re}[z_j]} = -\mathrm{Im}[y_k](1-\mathrm{Im}[y_k])\mathrm{Im}[v_{kj}]$$

Substituting these values in "(22)" we get

$$\frac{\partial E}{\partial \mathrm{Re}[z_j]} = \sum_{k=1}^{N} \mathrm{Re}[e_k]\mathrm{Re}[y_k](1-\mathrm{Re}[y_k])\mathrm{Re}[v_{kj}] - \sum_{k=1}^{N} \mathrm{Im}[e_k]\mathrm{Im}[y_k](1-\mathrm{Im}[y_k])\mathrm{Im}[v_{kj}]$$

Hence from "(21)"

$$\frac{\partial E}{\partial \mathrm{Re}[u_i]} = \frac{\partial E}{\partial \mathrm{Re}[z_j]}\{\mathrm{Re}[z_j](1-\mathrm{Re}[z_j])\}$$

$$= \mathrm{Re}[z_j](1-\mathrm{Re}[z_j]) \left\{ \begin{array}{l} \sum_{k=1}^{N} \mathrm{Re}[e_k]\mathrm{Re}[y_k](1-\mathrm{Re}[y_k])\mathrm{Re}[v_{kj}] \\ -\sum_{k=1}^{N} \mathrm{Im}[e_k]\mathrm{Im}[y_k](1-\mathrm{Im}[y_k])\mathrm{Im}[v_{kj}] \end{array} \right\} \quad (23)$$

Similarly, the partial derivative of the cost function w.r.t. Im[u_i] is

$$\frac{\partial E}{\partial \mathrm{Im}[u_i]} = \frac{\partial E}{\partial \mathrm{Re}[z_j]}\frac{\partial \mathrm{Re}[z_j]}{\partial \mathrm{Im}[u_i]} + \frac{\partial E}{\partial \mathrm{Im}[z_j]}\frac{\partial \mathrm{Im}[z_j]}{\partial \mathrm{Im}[u_i]} \quad (24)$$

Once again from "(9)" we get

$$\frac{\partial \mathrm{Re}[z_j]}{\partial \mathrm{Im}[u_i]} = 0$$

$$\frac{\partial \mathrm{Im}[z_j]}{\partial \mathrm{Im}[u_i]} = \mathrm{Im}[z_j](1-\mathrm{Im}[z_j])$$

$$\frac{\partial E}{\partial \mathrm{Im}[z_j]} = \sum_{k=1}^{N}\left(\frac{\partial E}{\partial \mathrm{Re}[e_k]}\frac{\partial \mathrm{Re}[e_k]}{\partial \mathrm{Im}[z_j]} \right) + \sum_{k=1}^{N}\left(\frac{\partial E}{\partial \mathrm{Im}[e_k]}\frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Im}[z_j]} \right)$$

$$= \sum_{k=1}^{N} \mathrm{Re}[e_k]\left(\frac{\partial \mathrm{Re}[e_k]}{\partial \mathrm{Im}[z_j]} \right) + \sum_{k=1}^{N} \mathrm{Im}[e_k]\left(\frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Im}[z_j]} \right) \quad (25)$$

$$\frac{\partial \mathrm{Re}[e_k]}{\partial \mathrm{Im}[z_j]} = \frac{\partial \mathrm{Re}[e_k]}{\partial \mathrm{Re}[s_k]}\frac{\partial \mathrm{Re}[s_k]}{\partial \mathrm{Im}[z_j]} + \frac{\partial \mathrm{Re}[e_k]}{\partial \mathrm{Im}[s_k]}\frac{\partial \mathrm{Im}[s_k]}{\partial \mathrm{Im}[z_j]}$$

From "(11)" and "(12)", we get

$$\frac{\partial \mathrm{Re}[e_k]}{\partial \mathrm{Im}[s_k]} = 0$$

$$\frac{\partial \mathrm{Re}[e_k]}{\partial \mathrm{Im}[z_j]} = -\mathrm{Re}[y_k](1-\mathrm{Re}[y_k])\{-\mathrm{Im}[v_{kj}]\} \quad (26)$$

$$= \mathrm{Re}[y_k](1-\mathrm{Re}[y_k])\mathrm{Im}[v_{kj}]$$

$$\frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Im}[z_j]} = \frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Re}[s_k]}\frac{\partial \mathrm{Re}[s_k]}{\partial \mathrm{Im}[z_j]} + \frac{\partial \mathrm{Im}[e_k]}{\partial \mathrm{Im}[s_k]}\frac{\partial \mathrm{Im}[s_k]}{\partial \mathrm{Im}[z_j]}$$

$$= -\mathrm{Im}[y_k](1-\mathrm{Im}[y_k])\mathrm{Re}[v_{kj}] \quad (27)$$

Substituting these values from "(26)", "(27)" in "(25)" we get

$$\frac{\partial E}{\partial \mathrm{Im}[z_j]} = \sum_{k=1}^{N} \mathrm{Re}[e_k]\mathrm{Re}[y_k](1-\mathrm{Re}[y_k])\mathrm{Im}[v_{kj}] - \sum_{k=1}^{N} \mathrm{Im}[e_k]\mathrm{Im}[y_k](1-\mathrm{Im}[y_k])\mathrm{Re}[v_{kj}]$$

Therefore from "(24)"

$$\frac{\partial E}{\partial \mathrm{Im}[u_i]} = \frac{\partial E}{\partial \mathrm{Im}[z_j]}\{\mathrm{Im}[z_j](1-\mathrm{Im}[z_j])\}$$

$$= \mathrm{Im}[z_j](1-\mathrm{Im}[z_j]) \left\{ \begin{array}{l} \sum_{k=1}^{N} \mathrm{Re}[e_k]\mathrm{Re}[y_k](1-\mathrm{Re}[y_k])\mathrm{Im}[v_{kj}] \\ -\sum_{k=1}^{N} \mathrm{Im}[e_k]\mathrm{Im}[y_k](1-\mathrm{Im}[y_k])\mathrm{Re}[v_{kj}] \end{array} \right\} \quad (28)$$

Substituting the values of $\frac{\partial E}{\partial \mathrm{Re}[u_i]}$ from "(23)" and $\frac{\partial E}{\partial \mathrm{Im}[u_i]}$ from "(28)" in "(20)" we get

$$\Delta x_i = -\eta \sum_{j=1}^{M} \left\{ w_{ji}^{*} \left(\frac{\partial E}{\partial \mathrm{Re}[u_i]} + i \frac{\partial E}{\partial \mathrm{Im}[u_i]} \right) \right\}$$

$$= -\eta \sum_{j=1}^{M} w_{ji}^{*} \left\{ \begin{array}{l} \left\{ \begin{array}{l} \mathrm{Re}[z_j](1-\mathrm{Re}[z_j]) \\ \sum_{k=1}^{N} \mathrm{Re}[e_k]\mathrm{Re}[y_k](1-\mathrm{Re}[y_k])\mathrm{Re}[v_{kj}] \\ + \mathrm{Im}[e_k]\mathrm{Im}[y_k](1-\mathrm{Im}[y_k])\mathrm{Im}[v_{kj}] \end{array} \right\} \\ -i \left\{ \begin{array}{l} \mathrm{Im}[z_j](1-\mathrm{Im}[z_j]) \\ \sum_{k=1}^{N} \mathrm{Re}[e_k]\mathrm{Re}[y_k](1-\mathrm{Re}[y_k])\mathrm{Im}[v_{kj}] \\ - \mathrm{Im}[e_k]\mathrm{Im}[y_k](1-\mathrm{Im}[y_k])\mathrm{Re}[v_{kj}] \end{array} \right\} \end{array} \right\} \quad (29)$$

Δx_i is the input update. Hence new inputs are calculated at each iteration by the following relation

$$x_{inew} = x_{iold} + \Delta x_i \quad (30)$$

With these new values of inputs the outputs are calculated. This output is compared with desired output and error is calculated. When this error is less than the minimum set error value, iterative process is stopped and the inversion is completed. This final value of the input vector '*x*' is the actual value of input by inversion of complex valued neural network.

EXPERIMENT 1

We have a taken 3 layered neural network with 2 inputs, 5 hidden layer neurons and one output neuron. First we trained the network for the input and output data of complex valued XOR gate given in table I. Once the network is created by training on the given data, the functional relationship between inputs and outputs is set. The complex valued target outputs are given in table II for which we have done inversion. We predicted the inputs by inversion of complex valued neural network. For this trained network the inputs are initiated to some random values. The outputs are obtained for these random input values. These actual outputs are compared to the target outputs and the error is calculated. This error is back-propagated and the new values of inputs are calculated by updating the inputs using "(29)" and "(30)". With these new input values once again the outputs are calculated, compared with the target outputs, and then the error is calculated and back-propagated to correct the inputs to further new values. This process is repeated till the error is minimized and becomes less then the assumed minimum value of the error. Finally with these predicted inputs we found the actual outputs as given in table III. The actual outputs obtained from the predicted inputs are nearly the same to the target outputs.

Table I
Training data for experiment 1 (Complex XOR gate)

Input x_1: (a_1+ib_1)	Input x_2:(a_2+ib_2)	output
0	0	1
0	i	i
i	0	0
i	i	1+i
i	1	i
1	1	1+i
1+i	i	i
1+i	1+i	1
0	1	i
0	1+i	0
i	1+i	0
1	0	0
1	i	i
1	1+i	0
1+i	0	0
1+i	1	i

Table II
Target outputs, desired inputs and corresponding actual inputs from inversion

Desired inputs		Actual Inputs by inversion		Target output
X_1	X_2	X_1	X_2	
i	0	0.9731i	0.056	0
0	1	0.2345	0.8834	i
1+i	1+i	0.9834+0.8765i	0.8976+0.9821i	1
i	i	0.8976i	0.9231i	1+i

Table III
Target outputs and actual outputs calculated from inputs obtained by inversion

Target outputs	Actual outputs
0	0.1381+0.0671i
i	0.0057+0.8405i
1	0.8692+0.1094i
1+i	0.8979+0.9014i

The main problem in inversion using complex back propagation algorithm is to find the inverse solution lying nearest to a specified point. For this we have used nearest inversion approach which is a single element search method. Given a function *f(i)*, a target output level *t*, and an initial base point i_0. We try to find the point i^* that satisfies $f(i^*)=t$ and is closest to i_0 in some sense. Nearest inversion is a constrained optimization problem. This constrained problem is solved by minimizing $E=i-i_0$ subject to $f(i)=t$.

EXPERIMENT 2

In this experiment we have tried the inversion for similarity transformation. We have taken a three layered neural network with architecture (1-5-1). The complex input pattern is scaled down by 0.5. The scaling is in terms of magnitude only the angle is preserved. The training input pattern consists of a set of complex values represented by *star* signs and corresponding output pattern data points are represented by *diamond* sign as shown in Fig.5. Once the network is created by training on the given data, the functional relationship between inputs and outputs is set. This trained model of CVNN for similarity transformation is used for inversion. The network is presented with the target output points shown by diamond symbols arranged in the shape of a rectangle as shown in Fig. 6. For this trained network the inputs are initiated to some random values. The outputs are obtained for these random input values. These actual outputs are compared to the target outputs and the error is calculated. This error is back-propagated and the new values of inputs are calculated by updating the inputs using (29) and (30). This iterative process is continued till the error is minimized and becomes less then the assumed minimum value of the error.

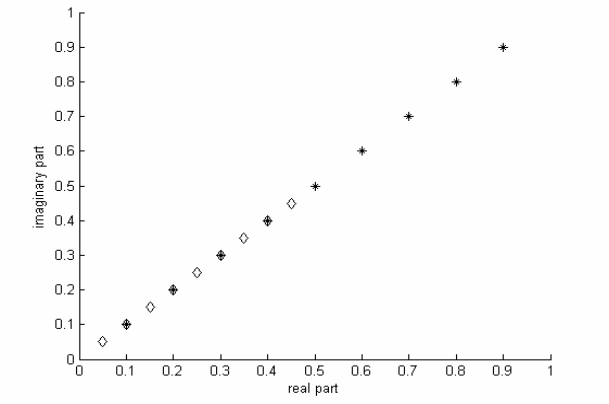

Fig. 5: similarity transformation: training input points (*star* signs) and training output points (*diamond* signs)

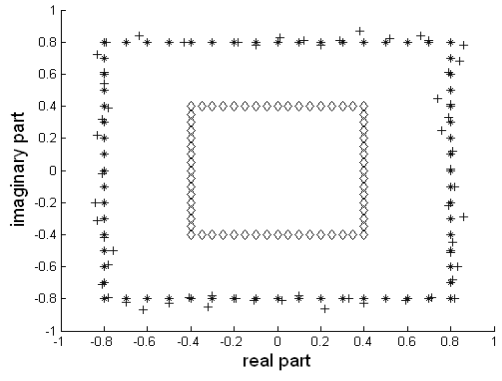

Fig. 6 inversion results for similarity transformation showing target outputs by *diamonds* expected inputs by *stars* actual inputs obtained from inversion by plus signs

In Fig. 6 desired inputs are indicated by *stars* and the *plus* signs denote the actual inputs obtained from the inversion of the network. As seen in the figure the inputs from inversion are very close to the expected inputs. Thus inversion of complex valued neural network is successfully done.

EXPERIMENT 3

In this experiment we have taken (1-7-1) neural network. The network is trained for rotational transformation data in counter clockwise direction. The training input data points are represented by stars and the corresponding output data points are represented by diamonds in Fig. 7. After training the weights of the neural network are fixed. We have tried the inversion on some different values of outputs in the same range.

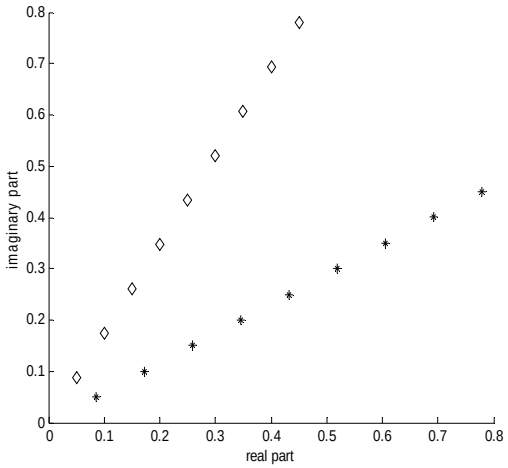

Fig. 7 training data for rotational transform in complex plane: *stars* showing inputs and *diamond* symbols showing corresponding outputs

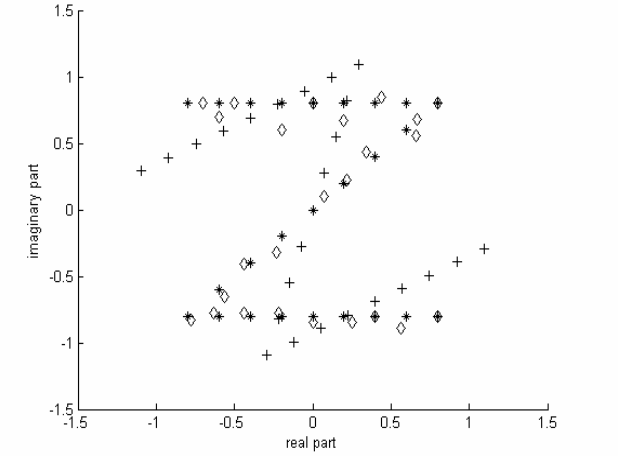

Fig. 8 showing target outputs by *plus* sign, desired inputs by *star* symbols and the inputs predicted by inversion by *diamond* symbols for rotational transform in complex plane

For inversion the target output points are shown in Fig. 8 by plus signs. These target data points are arranged in the shape of English letter 'z'. Inputs are initiated with some random values. Then the inversion of this neural network is done by using complex back-propagation algorithm. Inputs obtained by the inversion of the trained neural network are represented by diamond signs and the expected inputs are represented by the star signs as shown in Fig. 8. As clear from the figure that the inputs obtained from inversion are nearly the same as to the expected inputs. Hence inversion is done successfully for rotational transformation.

V. CONCLUSIONS

Inversion of complex valued neural network is still a relatively low explored field and there are many aspects which can be further studied and explored. Some other inversion algorithms of real domain can be expanded to complex domain. In most researches conducted on the complex valued neural networks, the learning constant used is real valued. In principle a complex learning constant could be employed. In this approach, we have used complex Quadratic error function for optimization. The other real domain error functions extended to complex domain can be applied for optimization during inversion.

REFERENCES

[1] R. D. Reed and R. J. Marks, II, "An evolutionary algorithm for function inversion and boundary marking," in Proc. IEEE Int. Conf. Evolutionary Computation (ICEC'95), Perth, Western Australia, pp. 794-797, 1995 .

[2] T. J. Williams, "Inverting a connectionist network mapping by backpropagation of error," in Proc. 8th Annu. Conf. Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum, pp. 859-865, 1986

[3] J. Kinderman and A. Linden, "Inversion of neural networks by gradient descent," Parallel Comput., pp.277-286, 1990.

[4] R.C. Eberhart and R.W. Dobbins, "Designing neural network explanation facilities using genetic algorithms," in Proc.Int. Joint Conf. Neural networks, vol.II, Singapore, pp.1758-1763, 1991.

[5] M.I. Jordan and D.E. Rumelhart, "Forward models: supervised learning with a distal teacher," Cognitive Sci., vol. 16 , pp.307-354, 1992.

[6] L. Behera, M. Gopal, and S. Chaudhary, "On adaptive trajectory tracking of a robot manipulator using inversion of its neural emulator," IEEE Ttans. Neural Networks, vol. 7, no. 6, Nov. 1996.

[7] Bio-Liang Lu, Hajime Kita and Y. Nishikawa, "Inversion of feedforward neural networks by separable programming," in Proc. World Congr. Neural networks (Portland), vol. 4, 1993, pp 415-420.

[8] B.Widrow, J. McCool, and M. Ball, "The Complex LMS algorithm," Proc. of the IEEE, April, 1975.

[9] M.S. Kim, and C.C. Guest, 1990, "Modification of back-propagation for complex- valued signal processing in frequency domain," IJCNN Int.Joint Conf. Neural Networks, pp. III-27-III-31,June.

[10] G. M. Georgiou and C..Koutsougeras, "Complex domain back-propagation," IEEETrans. On Circuits and Systems – II: Analog and Digital Signal Processing, Vol.39, No. 5., May1992.

[11] H. Leung and S. Haykins, "The complex back-propagation algorithm," IEEE Trans. On signal Processing, Vol. 39,No.9, September1991.

[12] A. Prashanth, "Investigation on complex variable based back-propagation algorithm and applications," Ph.D. thesis, IIT, Kanpur, India, 2003.

[13] Nitta, "An extension of the back-propagation algorithm to complex numbers," neural networks, Vol. 10, No. 8, 1997.

Anita S. Gangal She received B.Tech Degree in Electronics Engineering from HBTI, Kanpur in 1992. She is pursuing her Ph. D. in Electronics Engineering from Uttar Pradesh Technical University, India. She had worked as lecturer in Electronics Engineering Department at HBTI, Kanpur, India and C.S.J.M. University, Kanpur, India. She is member of IETE, India. Her major fields of interests are Neural Networks, Computational Neuroscience and Power Electronics.

P. K. Kalra He received his BSc (Engg) degree from DEI Agra, India in 1978, M. Tech degree from Indian Institute of Technology, Kanpur, India in 1982, and Ph.D. degree from Manitoba University , Canada in 1987. He worked as assistant professor in the Department of Electrical Engineering, Montana State University Bozeman, MT, USA from January 1987 to June 1988. In July-August 1988 he was visiting assistant professor in the Department of Electrical Engineering, University of Washington Seattle, WA, USA. Since September 1988 he is with the Department of Electrical Engineering, Indian Institute of Technology Kanpur, India where he is Professor and Head of Department. Dr. Kalra is a member of IEEE, fellow of IETE and Life member of IE(I), India. He has published over 150 papers in reputed National and International journals and conferences. His research interests are Expert Systems applications, Fuzzy Logic, Neural Networks and Power Systems.

D. S. Chauhan He received his BSc (Engg) degree from BHU Varanasi, India in 1972, M.E. degree from Madras University, India in 1978, and Ph.D. degree from Indian Institute of Technology Delhi, India in 1986. He is Former Vice Chancellor of Uttar Pradesh Technical University, India. He is vice chancellor of L.P. University, Jalandhar, India. Dr. Chauhan is Fellow of IE(I), member of IEEE, USA, and member of National Power Working Group, India. He has published over 70 papers in reputed National and International journals and conferences. His research interests are Linear Controls, Power Systems Analysis, Artificial Intelligence, Fuzzy Systems HVDC Transmission, and Neural Networks.