Performance Evaluation of Complex Valued Neural Networks Using Various Error Functions

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

***Abstract*—**The backpropagation algorithm in general employs quadratic error function. In fact, most of the problems that involve minimization employ the Quadratic error function. With alternative error functions the performance of the optimization scheme can be improved. The new error functions help in suppressing the ill-effects of the outliers and have shown good performance to noise. In this paper we have tried to evaluate and compare the relative performance of complex valued neural network using different error functions. During first simulation for complex XOR gate it is observed that some error functions like Absolute error, Cauchy error function can replace Quadratic error function. In the second simulation it is observed that for some error functions the performance of the complex valued neural network depends on the architecture of the network whereas with few other error functions convergence speed of the network is independent of architecture of the neural network.

***Keywords*—**Complex backpropagation algorithm, complex error functions, complex valued neural network, split activation function.

I. Introduction

In recent years, complex-valued neural networks have widened the scope of application in optoelectronics, imaging, remote sensing, quantum neural devices and systems, spatiotemporal analysis of physiological neural systems, and artificial neural information processing.

The generalization of real valued algorithms cannot be simply done as complex valued algorithm. The complex backpropagation algorithm can be applied to multilayered neural networks whose weights, threshold values, inputs and outputs all are complex numbers. Complex version of backpropagation (CVBP) algorithm made its first appearance when Widrow, Mc Cool and Ball [1] announced their complex least mean squares (LMS) algorithm. Kim and Guest [2] published a complex valued learning algorithm for signal processing application. Georgiou and Koutsougeras [3] 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. Hirose [4] studied the dynamics of CVNN which was later applied to the problem of reconstructing vectors lying on the unit circle. Benvenuoto and Piazza [5] published a different version of CVBP by using different activation function. Wang [6] proposed a complex valued recurrent neural network to solve the complex valued linear equations. A complex activation function for digital VLSI neural networks was implemented by Deville [7] that required lesser hardware than the conventional real valued neural network. A complex valued neural network was used by Smith and Hui[8] to implement a data extrapolation algorithm. Leung and Haykin[9] published the CVBP in which the activation function used was an extended version of sigmoid function and the error function was Quadratic error function. An extensive study of CVBP was reported by Nitta [10]. Decision boundary of a single complex valued neuron consists of two hypersurfaces that 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. Werbos and Titus [11] and then Gill and Wright [12] discussed the different consequences of changing error functions in an optimization scheme. Rey [13] has shown that the results could be substantially improved by varying error function in an optimization scheme. He applied absolute error function based optimization to solve a curve fitting problem more efficiently than the standard quadratic error function based optimization. Fernandeze [14] implemented some new error functions for the training of real valued neural network as tools to counter the ill effects of local minima by weighting error functions according to their magnitudes. Matsuoka and Yi[15] used logarithmic error function to eliminate the local minima. Ooyen and Nienhaus [16] have used entropy type error function and have concluded that it's performance is better than the quadratic error function based backpropagation algorithm for function approximation problems.

II. Complex Valued Neural Network

A three layered complex valued neural network is shown in (1). In this network all the inputs, outputs, weights, and biases are complex values. According to the Liouville's theorem, a

Manuscript received June 15, 2007.

Anita S. Gangal is with Uttar Pradesh Technical University, Lucknow, India (phone:+91-9450141454; e-mail: 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 Lovely Professional University, Jalandhar, India (e-mail: pdschauhan@gmail.com).

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 Prashant[17] 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 sigmoidal activation function is used for training the network.

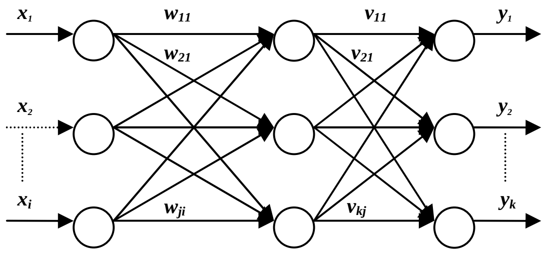

Fig. 1 complex valued neural network

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. We have used complex BP learning rule which has been obtained by using a steepest descent method for multilayered complex valued neural network given by Nitta [10]. The weights are initiated to some random values. The outputs are obtained for these random input values. The error between actual output and the desired output is calculated. This error is back propagated and the weights are updated. Then for these new values of weights, outputs are once again calculated. These actual calculated outputs are once again compared with the target outputs and the error is calculate, which is again back propagated and the weights are once again updated. This iterative process is continued till the error becomes less then the minimum defined.

Internal potential of hidden neuron j:

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

Output of hidden neuron j:

$$z_j = \phi(u_j) = \frac{1}{1+e^{-\mathrm{Re}[u_j]}} + i\frac{1}{1+e^{-\mathrm{Im}[u_j]}} = \mathrm{Re}[z_j] + i\,\mathrm{Im}[z_j] \tag{2}$$

Internal potential of output neuron k:

$$s_k = \sum_{j=1}^{m} (v_{kj} z_j) + \gamma_k = \mathrm{Re}[s_k] + i\,\mathrm{Im}[s_k] \tag{3}$$

Output of output neuron k:

$$o_k = \phi(s_k) = \frac{1}{1+e^{-\mathrm{Re}[S_k]}} + i\frac{1}{1+e^{-\mathrm{Im}[S_k]}} = \mathrm{Re}[y_k] + i\,\mathrm{Im}[y_k] \tag{4}$$

Error

$$e_k = o_k - d_k \tag{5}$$

With the help of this error e_k using different complex error functions the error E is obtained. Then we derive the gradient of E w.r.t. both the real and imaginary part of the complex weights.

$$\nabla_{w_{ji}} E = \frac{\partial E}{\partial \mathrm{Re}[w_{ji}]} + i\frac{\partial E}{\partial \mathrm{Im}[w_{ji}]} \tag{6}$$

During training the network cost function E is minimized by recursively altering the weight coefficient based on gradient descent algorithm, given by

$$w_{ji}(p+1) = w_{ji}(p) + \Delta w_{ji}(p) = w_{ji}(p) - \eta \nabla_{w_{ji}} E_P\big|_{w_{ji}=w_{ji(p)}}\big| \tag{7}$$

Where 'p' is the number of iterations and 'η' is the learning rate constant.

III. Various Error Functions

It must be noted that in the description of Error functions, the function's form has been retained to that of real error functions forms while extending to complex domain. This was done to make sure that the error computed kept the same formula even while operating in the complex domain. This also makes sure that the surface plot of the function is close to the plane plot of the same. We have studied the performance of the complex valued neural networks by using following error functions:

A. The Absolute Error Function

Absolute error is one of several robust functions that display less skewing of error due to the outliers. A small numbers of outliers are less likely to affect the total error and so they do not affect the learning algorithm as severely as mean squared error.

Complex absolute error function is defined to be

$$E = \sum_n abs(e_i) = \sum_n \sqrt{(e_i e_i^*)} \tag{8}$$

B. Andrew Error Function

Complex Andrew error function is defined as

$$E = \sum_n 1/\pi^2 * \cos(\pi * abs(e_i)); \text{ if } abs(e_i \le 1) \tag{9}$$

$= 0;$ else

The Andrew error function has a discontinuity at the origin. The point beyond which the function can be suppressed can be chosen as desired while the dynamics of update operate according to a sinusoid by definition to take complex values. The surface plot of the complex definition reveals that the function is rotationally symmetrical about the z-axis.

C. Bipolar Hyperbolic Squared Error Function

Complex Bipolar Hyperbolic Squared error function is defined as

$$E = \sum_n \ln\left(\frac{2 - e_i e_i^*}{2 + e_i e_i^*}\right) \tag{10}$$

The surface is characterized by a unique maximum and rotational symmetry. The surface is differentiable with respect to real and imaginary parts of the complex variables that now appear as argument of the function.

D. Cauchy Error Function

Complex Cauchy error function is given as

$$E = \sum_n \frac{c^2}{2} \ln\left(1 + \left(e_i e_i^* / c\right)^2\right) \tag{11}$$

The surface plot shows a unique point of global minimum. This surface is differentiable through the real plane and is rotationally symmetrical about the z-axis. The surface is characterized by changing convexity as the radius vector increase.

E. Fair Error Function

The complex Fair error function is rotationally symmetrical and has one global minimum. The convexity with respect to the xy- plane is maintained everywhere. Complex Fair error function is defined as

$$E = \sum_n c^2 \left[\left(abs(e_i)/c\right) - \ln\left(1 + abs(e_i)/c\right)\right] \tag{12}$$

Where, c is the tuning constant. This function is defined continuous derivatives except at the origin. The tuning constant, c is usually set as 1.3998.

F. Fourth Power Error Function

This function is useful when dealing with the data known to be free from outliers, or in cases where it is important to minimize the worst-case error, rather than the average error. This error function increases more rapidly for errors more than unity as compared to Quadratic error function. The surface is smooth for the derivatives of all orders that exist and is rotationally symmetrical about the z- axis. The complex Fourth Power error function is defied as

$$E = \sum_n \frac{1}{2}\left(e_i e_i^*\right)^2 \tag{13}$$

G. German McClure Error Function

The German-McClure error function is defined to suppress large errors near the origin. The asymptotes of the function suppress the outliers. This function approximates to Quadratic function for smaller values of error as the denominator can be approximated as unity. The complex German McClure error function is given by

$$E = \sum_n \frac{e_i e_i^* / 2}{1 + e_i e_i^*} \tag{14}$$

H. Huber Error Function

The complex Huber error function is given by

$$E = \sum_n e_i e_i^* / 2 ; \qquad \text{if } |e_i| < c \tag{15}$$
$$= \sum_n c\left(abs(e_i) - c/2\right); \qquad \text{if } |e_i| \geq c$$

Where n is the number of outputs and c is the tuning constant. A typical value for c is 1.345. When dealing with noisy data, the training values may contain outliers with unusual deviation from the true underlying function. Huber function can be used to ignore these outliers, or at least reduce the ill effect they have on learning. The function has good effects of Quadratic and Absolute error function.

I. Hyperbolic Squared Error Function

Hyperbolic Squared error needs normalization while running training with one of the backpropagation algorithm or its variants.The complex Hyperbolic Squared error function is given by

$$E = \sum_n \ln\frac{1 - e_i e_i^*}{1 + e_i e_i^*} \tag{16}$$

Hyperbolic Squared error is similar in shape to the Bipolar Hyperbolic Squared error function.

J. Log Cosh Error Function

The complex Log-Cosh error function is defined as

$$E = \sum_n \ln\left(\cosh\left(e_i e_i^*\right)\right) \tag{17}$$

K. Mean Median Error Function

This has the advantage of both the Mean error function and Median error function. Hence reduces the influence of large errors but at the same time retains its convexity.
Complex Mean meridian function is defined as

$$E = \sum_n 2 * \left(\sqrt{\left(1 + e_i e_i^* / 2\right)} - 1\right) \tag{18}$$

L. Minkowski Error Function

The complex Minkowski error function is given as

$$E = \sum_n abs(e_i)^r \tag{19}$$

Where n is the number of outputs and typical value of r is chosen as 0.4.

M. Quadratic Error Function

This is the standard error function. Complex Quadratic error function is defined as

$$E = \sum_{n} \frac{1}{2} e_i e_i^* \qquad (20)$$

N. Sinh Error Function

This function is steeper than Quadratic error function and is symmetrical about the origin and hence the update involves two parts, the first is the gradient in the first quadrant and the second is the gradient in the third quadrant. In both cases, the gradient is directed towards the origin.

The complex Sine-Hyperbolic function is given by:

$$E = \sum_{n} \left(Sinh\left(abs\left(e_i \right) \right) \right) \qquad (21)$$

O. Turkey Biweight Error Function

Turkey Biweight error function reduces the effect of large errors and suppresses the outliers. Hence the contribution of an outlier to this error function is smaller. The complex Turkey Biweight error function is defined as

$$E = \sum_{n} c^2 / 6\left(1 - \left[1 - \left(e_i e_i^* / c^2\right)\right]^3\right); \quad \text{if} \left|e_i\right| \le c \qquad (22)$$
$$= \sum_{n} c^2 / 6\,; \quad \text{if} \left|e_i\right| > c$$

Where c is the tuning constant and its typical value is 4.6851.

P. Welsch Error Function

The complex Welsch error function is defined as

$$E = \sum_{n} \frac{c^2}{2}\left[1 - \exp\left(-i\left(e_i e_i^* / c^2\right)\right)\right] \qquad (23)$$

This function reduces the effect of large errors. The typical value of the tuning constant c=2.9846.

IV. Simulation

A. Simulation 1

For the first experiment we have taken three layers complex valued neural network with architecture 2-5-1. Nitta [9] has defined the complex XOR gate in complex domain as:

1. The real part of the output is unity if the first input is equal to the second input else it is zero
2. The imaginary part of the output is unity if the second input is equal to unity else it is zero

We trained the complex valued neural network with the first eight patterns of Table I. Then for the testing we have done simulation on all the sixteen patterns. The experiments were run twice and average values of the epochs with various error functions are shown in Table II. The learning rate parameter is chosen 1.3 for all the simulations and the target error is set to 0.001.

In case of absolute error function the absence of index (the power, unlike the quadratic error function) is a distinguishing feature of this error function as this enables smoothing out the ill-effects of the outlier points that would otherwise have offset the best-fit of the optimization scheme. It can also be noted that the function form in fact is the quadric cone. The update rule for the complex valued neural network steers the real part and the imaginary part of the weights to the minima separately. The problem of local minima exists in general with this error function based algorithm. The initial weights and the learning parameter decide how the training should progress. The dynamics of real part depends not only on the real part of the weights but also on the imaginary parts as the updates (6),(7) of the real and imaginary parts are dependent on each other. This complex error function is not differentiable at the origin.

TABLE I
COMPLEX XOR GATE FOR SIMULATION

Input 1	Input 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	0+i	i
1	1+i	0
1+i	0	0
1+i	1	i

The complex Bipolar hyperbolic has unique maxima, and hence the training process should steer the network so as to attain the maxima of the function. While implementing the sign of the function is reversed so the update runs in accordance with accepted conventions. While implementing the Complex Andrew error function, the training was directed towards the minima. In case of Cauchy error function, the training steers the weights so as to reach the minimum of the function.

For fourth power error function, the weight update is more rapid for error values greater than unity, and the rate of training is diminished for fractional errors, lying in the interval [0,1]. The cube term that results from the form of the error function enhances the update if the error is greater than unity and suppresses it if error is fractional. Complex German-McClure error function is just the Quadratic function for smaller values of the errors, while for the large values the denominator comes into play, and the function deviates from being quadratic. The Huber error function is defined piece-wise. The characteristic feature of the function is that it involves both the Quadratic error function and the Absolute error function. The parameter *c* is the point of demarcation to assign a domain of operation for each error function. The function enables one to optimally choose error functions. If the data is prone to outliers and if their scatter is biased to one side, then an obvious choice would be to suppress the influence of these spurious points by assigning an Absolute error function to this side, and set Quadratic function to operate on the other side. It was found that in statistical analysis such choice improved the results as a judicial assignment was proven to be effective. The function is a paraboloid of revolution for the part of definition that is quadratic and for the part that is Absolute function it is a cone.

TABLE II
AVERAGE EPOCHS FOR COMPLEX XOR PROBLEM SOLVED USING COMPLEX VALUED NEURAL NETWORK

Error function	Average Epochs
Absolute error	12100
Andrew error	28000
Cauchy error	17600
Fourth order error	26300
Fair error	21000
German McClure error	27700
Huber error	24900
Hyperbolic Squared	27600
Bipolar Hyperbolic	27000
Log Cosh error	14500
Mean Median error	15000
Minkowski error	17600
Quadratic error	31300
Sinh error	16900
Tukey error	22100
Welsch error	24300

For complex Hyperbolic Squared error function the surface is characterized by a unique maximum at the origin to which the training process should steer the network error. During implementation of the function a negative sign was prefixed to the error function, and the usual gradient was developed for the function. The complex Mean-Median error function behaves like the quadratic error function for smaller complex errors, and as absolute error function for large errors. This function finds best application for data that are prone to an outlier scatter that should be treated by assigning a function to nullify the ill effects due to them. The Minkowski error function is characterized by the parameter that appears as the index in the definition. The other standard error functions discussed so far e.g. Absolute error function; Fourth Power error function etc. can be obtained as particular cases of this function by setting the index according to the requirement. For even indices the Minkowski function behaves like the Quadratic function typically. Odd indices generate functions that need a piecewise defined update rule for the function would be symmetric about the origin in this case. The complex Sinh error function is the extended version of Hyperbolic Sine function in complex domain. A complex conjugate is employed in the argument of the function to make it an even function. This complex function is rotationally symmetric about z-axis and the surface maintains convexity with respect to xy-plane. The steepness of the slope increase as the index increases. For complex Turkey error function, the surface is convex with respect to the xy-plane for small errors but changes convexity once before restoring back to convex. Hence the weight update depends on the part of the surface at which the error vector lies. Rest of the surface is plane. The Welsch error function suppresses the large errors and gives Quadratic error function like performance for small errors.

B. Simulation 2

In this experiment we have shown that how the architectural size of the complex valued neural network affects the training process. We have trained and tested the CVNN to map CXOR with varying number of hidden layer neurons. The learning rate is taken as 1.3. Three different architectures: 2-2-1; 2-5-1; 2-7-1; respectively are taken and trained. The number of iterations required with all the complex error functions for all the three network architectures are given in Table III.

TABLE III
AVERAGE RESULTS FOR DIFFERENT ARCHITECTURES OF COMPLEX VALUED NEURAL NETWORKS FOR VARIOUS COMPLEX ERROR FUNCTIONS

Error function	2-2-1	2-5-1	2-7-1
Absolute error	17800	13400	11000
Andrew error	23000	20300	16000
Cauchy error	13700	13400	13000
Fourth order error	21300	17400	19000
Fair error	15000	14300	14100
German McClure error	18700	17900	17000
Huber error	11000	10000	10000
Hyperbolic Squared	21000	20000	19500
Bipolar Hyperbolic	22400	21000	21100
Log Cosh error	12100	12000	11700
Mean Median error	15000	13000	14000
Minkowski error	16200	15700	16000
Quadratic error	11000	12000	12000
Sinh error	16700	15000	12300
Tukey error	12200	13000	16400
Welsch error	15100	13000	18000

As clear from the results shown in Table III complex valued neural networks with Absolute error function, Andrew error function, Fourth power error function, Logarithmic error function, Turkey error function, and Welsch error function are sensitive to the architecture. The number of iterations for convergence during training varies widely with the architecture. On the other hand, the Cauchy error function, Huber error function, Log-Cosh error function, Mean-Median error function, Quadratic error function, and Sinh error function have been robust and have shown little dependence on the architecture. Hence, the extra neurons and extra weights are not necessary as smaller architecture shows the similar performance and could solve the problem.

V. Conclusion

The above simulations reveal that the error functions can indeed be treated as a parameter for training complex valued neural network. Absolute error, Cauchy error, Fair error, Mean-Median error, Fourth error functions can replace Quadratic error function depending on the applications and requirements. With a proper choice of the parameter *c* the Huber function has the features of both Quadratic error function and absolute error function. The Huber error function consistently works well. Hence, Quadratic error function can be completely replaced by Huber error function.

References

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

[2] M. S. Kim and C. C. Guest, "Modification of backpropagation for complex- valued signal processing in frequency domain," Int. Joint Conf. on Neural Networks, San Diego, vol. 3, pp. 27-31, June 1990.

[3] 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.

[4] A. Hirose, "Dynamics of fully complex-valued neural networks," Electronics letters, vol. 28, no. 13, pp.1492-1494.

[5] N. Benvenuoto and F. Piazza, "On the complex backpropagation algorithm," IEEE Trans. On Signal Processing, vol. 40, no. 4, April 1992.

[6] J. Wang, "Recurrent neural networks for solving Systems of complex valued linear equations," Electronics letters, vol. 28, no. 18, pp. 1751-1753.

[7] Y. Deville, "A neural network implementation of complex activation function for digital VLSI neural networks," Microelectronics Jjournal, vol. 24, pp. 259-262.

[8] M. R. Smith and Y. Hui, "A data exploration algorithm using a complex domain neural network," IEEE Trans.on Circuits and systems-II: Analog and digital signal processing, vol. 22,no.2.

[9] H. Leung and S. Haykin," The complex backpropagation algorithm", IEEE Trans. On signal Processing, Vol. 39,No.9, September(1991).

[10] T. Nitta, "An extension of the back-propagation algorithm to complex numbers," Neural Networks, vol. 10, no. 8, pp 1391-1415, 1997.

[11] P. J. Werbos, andJ Titus, "An empirical test of new forecasting methods derived from a theory of intelligence: The prediction of conflict in Latin America," IEEE Trans.on Systems, Man and Cybernetics, September, 1978.

[12] P. E. Gill, and M. H. Wright, "Practical optimization," Academic Press, 1981.

[13] W. J. J. Rey, "Introduction to robust and quasi-robust statistical methods," Springer- Verlag, Berlin.

[14] B. Fernandez, "Tools for artificial neural networks learning," Intelligent engineering systems through Artificial Neural Networks, ASME Press, New York, pp. 69-76, 1991.

[15] K. Matsuoka, and J. Yi, "Backpropagation based on the Logarithmic error function and elimination of local minima," in Proc. Of the International Joint Conference on Neural Networks, Singapore, vol. 2, pp. 1117-1122, 1991.

[16] V. Ooyen, Nienhaus, "Improving the convergence of backpropagetion algorithm," Neural Networks, vol. 5, pp. 465-571.

[17] A. Prashanth, "Investigation on complex variable based backpropagation algorithm and applications," Ph.D. Thesis, IIT, Kanpur, India, March 2003.