WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IJCNN

Complex-valued Neuro-Fuzzy Inference System for Wind Prediction

K. Subramanian
School of Computer Engineering
Nanyang Technological University
Singapore, 639798
Email: kartick1@e.ntu.edu.sg

R. Savitha
School of Computer Engineering
Nanyang Technological University
Singapore, 639798
Email: savi0001@e.ntu.edu.sg

S. Suresh
School of Computer Engineering
Nanyang Technological University
Singapore, 639798
Email: ssundaram@ntu.edu.sg

***Abstract*—In this paper, we present a complex-valued neuro-fuzzy inference system (CNFIS) and its gradient descent based learning algorithm developed employing Wirtinger calculus. The proposed CNFIS is a four layered network which realizes zero-order Takagi-Sugeno-Kang based fuzzy inference mechanism. CNFIS is used to predict the speed and direction of wind. Here, the speed and direction are considered as statistically independent variables and are represented as a complex-valued signal (with speed as magnitude and direction as phase). Performance of CNFIS is compared with other algorithms available in the literature and results indicate improved performance of CNFIS.**

The major contribution of this paper is as follows: (1) Propose a complex-valued neuro-fuzzy inference system (2) Employ Wirtinger calculus for complex-valued gradient descent algorithm (3) Solve wind speed and direction prediction problem in complex domain.

***Index Terms*—complex-valued neuro fuzzy inference system, Wirtinger calculus, wind speed prediction**

I. Introduction

In recent times, a large number of problems such as adaptive signal processing [1], communication [2], image processing [3], image reconstruction [4], etc employ complex-valued signals due to the intrinsic nature of the signals involved. In addition, it has been shown that some real-valued problems like wind speed and direction prediction [5], tree representation of hand in hand gesture recognition system [6], etc are better represented in the complex domain. Artificial neural networks, which have been well established as efficient function approximator for real-valued problems, are being extended to the complex domain in order to utilize their non-linear processing abilities, for solving these complex-valued problems. Although desirable, such extensions are not trivial as neural networks require a non-linear activation function which are entire and bounded everywhere [7]. Whereas, in the domain of complex calculus, Lioville's theorem [8] which states that an entire and bounded function is a constant function in complex domain forbids the use of these functions as activation functions. In the literature, this problem has been circumvented using three approaches: split complex-valued neural networks, fully complex-valued neural networks with real-valued activation function and fully complex-valued neural networks with fully complex-valued activation function. Works available in the literature using these three approaches are briefly reviewed in section II.

In literature, it has been shown that neuro-fuzzy inference systems [9], [10] combine the approximation ability of artificial neural networks and the data representability of fuzzy inference systems. Hence, for the first time in the literature, we develop a complex-valued zero-order Takagi-Sugeno-Kang type neuro-fuzzy inference system (CNFIS) in this paper. CNFIS uses a Gaussian activation function and a fully complex-valued gradient descent algorithm derived using Wirtinger calculus [11]. Although neural networks using Gaussian activation function have been reported in literature [12], these networks do not use fully complex-valued gradients to update the network parameters. Instead, they use the real and imaginary part of error to update the real and imaginary part of the network parameters, respectively. As the gradients of CNFIS are derived using Wirtinger calculus, the gradients are fully complex-valued and phase is preserved.

Next, CNFIS is used to solve a real-world wind speed and direction prediction problem. Prediction of wind speed and direction is of significance, especially for efficient wind power generation. It has been shown in [5] that the speed and direction of wind are statistically dependent quantities and hence, their characteristics are better represented using a complex-valued signal. Therefore, in this paper, CNFIS is used to solve the wind prediction problem using the complex-valued features extracted from real-world data obtained from Iowa dept. of transportation[1].

The paper is organized as follows: Section II presents a brief review of the literature on complex-valued neural networks. In section III, we introduce a complex-valued neuro-fuzzy inference system and derive its complex-valued gradient descent algorithm. The developed algorithm is used to predict the speed and direction of wind in section IVand the paper is concluded in section V.

II. Background and related works

In this section, we present a brief review on the works available in the literature on complex-valued neural networks. We broadly classify the literature on complex-valued neural

Corresponding Author: ssundaram@ntu.edu.sg

[1]http://weatherview.iowadot.gov/

networks based on the nature of the complex-valued neural networks and the mode of learning.

A. Nature of Complex-Valued Neural Networks

Based on the nature of complex-valued neural networks, the literature could be broadly split into two: split complex-valued and fully complex-valued neural networks.

Split Complex-valued Neural Networks: One of the earliest approaches to circumvent the restrictions of Lioville's theorem was to replace complex-valued inputs and outputs with pairs of independent real-valued signals and employ real-valued neural networks to approximate the function represented by the given data [13], [14], [15], [16], [17]. These networks suffer from poor approximation abilities due to loss of significant complex-valued information during both the forward and backward computation, resulting in poor performance in terms of convergence and generalization ability [18]. Moreover, the convergence of split-complex valued neural networks with complex weights depends on proper initialization and choice of learning rate [19].

To overcome the problem of phase distortion due to the splitting of complex-valued signals, complex-valued neural networks with complex-valued weights and real-valued activation functions have been proposed in [12], [2], [20], [21]. These networks use the Gaussian activation function that maps the complex-valued inputs to real-valued hyper-dimensional feature space ($\mathbb{C} \rightarrow \mathbb{R}$) at the hidden layer. Examples of this type of networks include the complex-valued radial basis function networks and its variants [12], [2], [20]. However, as the activation function maps $\mathbb{C} \rightarrow \mathbb{R}$ in hidden layer, the responses of the hidden neurons are real-valued during forward computation. Moreover, these algorithms rely on Cauchy Riemann conditions during backward computation. Hence, in these networks, the real part of the complex-valued error is used to update the real part of the network parameters and the imaginary part of the error is used to update the imaginary part of the network parameters. Thus, these networks are also inefficient in mapping the nonlinear relationship between the complex-valued signals.

Fully Complex-Valued Neural Networks: To overcome the challenges imposed by the Liouville's theorem, the desired properties of a fully complex-valued activation function was relaxed in [18] as: *In a bounded domain of complex plane $\mathbb{C}$, a fully complex non-linear activation function $f(z)$ needs to be analytic and bounded almost everywhere*. Since then, fully complex-valued neural networks using complex-valued weights and fully complex-valued activation functions that are analytic and bounded *almost everywhere* in the complex plane have been developed. Two types of activation functions have been used in the literature: Elementary transcendental activation functions and axial symmetric activation functions.

In [18], Kim and Adali have proposed a set of elementary transcendental functions for fully complex-valued multi-layer perceptron and derived its complex-valued back propagation algorithm. These functions help to overcome the issue of non-complex gradients employed in split complex-valued neural networks. On the other hand You *et al.* have developed a fully complex-valued neural network using an axial symmetric activation function in [22]. The advantage of axial symmetric activation function is that it is bounded everywhere. However, the real and imaginary parts of a complex-valued signals are considered independent.

The fully complex-valued back propagation algorithm in these works were derived based on the mean-squared error criterion that is only representative of the magnitude of the complex-valued error. To overcome this issue, Savitha *et al.* have developed an improved complex-valued multi-layer perceptron network in [23] using a logarithmic error function that is an explicit representation of both the magnitude and phase of the complex-valued error. Recently, fully complex-valued relaxation network have also been developed using this logarithmic error function in [24], [25].

On the other hand, radial basis function neural networks are extensively used due to their localization property. In the complex-valued neural network literature, Complex-valued Radial Basis Function (CRBF) network was first developed by Chen *et al.,* in [12]. CRBF uses the Gaussian activation function and gradient descent based learning algorithm to update the network parameters. As mentioned earlier, the learning algorithm in these networks updates the real part of the parameters using the real part of the error and the imaginary part of the parameters using the imaginary part of the error. Thus, the correlation between the real and imaginary part of the error is lost during both forward and backward computation, and hence, significant phase information is also lost. Recently, this problem was overcome with the development of a fully complex-valued radial basis function network using a Gaussian-like fully complex-valued activation function (*sech*) and fully complex-valued gradient descent based learning algorithm in [26], [27], [28]. More recently, a fully complex-valued relaxation network, that estimates the minimum energy point of the logarithmic error function from a given initial condition has been developed using this activation function in [24], [25].

Earlier, it was shown by Nitta that complex-valued neural networks have better computational power than real-valued neural networks [29]. Moreover, he also showed that complex-valued neural networks have two decision boundaries that are orthogonal to each other, that help them to solve real-valued classification tasks efficiently [30]. Since then, a few classifiers have been developed in the complex domain, including the phase encoded fully complex-valued neural network [31], the multi-layered multi-valued neural network [32], and the fully complex-valued radial basis function classifier [27]. Fast learning fully complex-valued classifiers include the phase encoded complex-valued extreme learning machine, the bilinear branch-cut complex-valued extreme learning machine [33], the fast learning fully complex-valued classifier [34], and the circular complex-valued extreme learning machine [35]. A fast learning fully complex-valued neural network has been used to solve the human action recognition problem in [36].

Next, we shall review literature based on the mode of learning for a complex-valued network.

B. Modes of Learning

Depending on the sequence in which samples are presented to the network a learning algorithm can be classified as : batch learning or sequential learning algorithm.

Complex-Valued Batch Learning Algorithms: In a batch learning scheme, all the samples in the training set are presented repeatedly to the network, until a specified accuracy is achieved. Some of the batch learning algorithms for complex-valued neural networks include the split complex-valued multi-layer perceptron neural networks , the fully complex-valued neural network by Kim and Adali [18], [37]. These algorithms suffer from the problem of local minima that is inherent to multi-layer perceptron networks, and this was overcome by Chen *et al.,* in [38]. On the other hand, in the framework of radial basis functions, CRBF [12], fully complex-valued radial basis function network [26], meta-cognitive fully complex-valued radial basis function network [25], fully complex-valued relaxation network [24] are some of the networks employing complex-valued batch learning algorithms using fully complex-valued gradients. All these algorithms require the entire training data apriori, and also requires that the network structure is fixed apriori. This is not always possible, especially, in real-world problems, like medical diagnosis. For such applications, sequential learning algorithms are preferred over batch learning algorithms.

Complex-Valued Sequential Learning Algorithms: In a sequential learning algorithm, samples are presented one-by-one and only once to the network. They are discarded after being learnt by the network. The Complex-valued Minimal Resource Allocation Network (CMRAN) [20] and the Complex-valued Growing and Pruning Radial Basis Function (CGAP-RBF) algorithm [2] are the sequential learning algorithms that were first developed in the Complex domain. These networks begin with zero hidden neurons and add and prune neurons during the training process, until a parsimonious structure is achieved. However, these networks employ the Gaussian activation and use the real-valued extended Kalman filter to learn the function relationship represented by the training data. Hence, they also suffer from inaccurate phase approximation. Recently, two fully complex-valued sequential learning algorithms using the fully complex-valued $sech$ activation function, namely, the complex-valued self-regulatory resource allocation network [39] and the meta-cognitive learning algorithm for a fully complex-valued relaxation network [25] have been developed to overcome the issues of inaccurate phase approximations.

Although the networks discussed in these sections can approximate both the magnitude and phase of the functional relationship between the inputs and outputs efficiently, these networks do not interpret the relationship between each output and each input efficiently. Therefore, in the next section, we present a complex-valued neuro-fuzzy inference mechanism that combines the advantages of data representability of neural networks and the interpretability of a fuzzy inference mechanism.

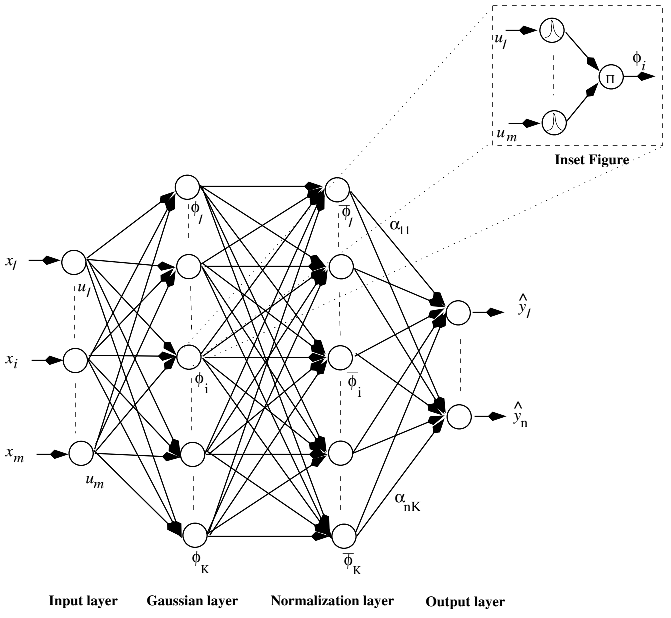

Fig. 1. Architecture of TSK type-0 neuro-fuzzy inference system. The inset figure describes the internal structure of ith Gaussian rule antecedent

III. Complex-valued Neuro Fuzzy Inference System

In this section, we present the architecture of a complex-valued neuro-fuzzy inference system architecture and describe its complex gradient descent based learning algorithm. Let us assume we have a set of training data, $\{(\mathbf{x}^1,\mathbf{y}^1),\cdots,(\mathbf{x}^t,\mathbf{y}^t),\cdots,(\mathbf{x}^N,\mathbf{y}^N)\}$, where $\mathbf{x}^t = [x_1^t,\cdots, x_m^t] \in \mathbb{C}^m$ is the input to the network and $\mathbf{y}^t = [y_1^t,\cdots, y_n^t] \in \mathbb{C}^n$ is the output of the network. The aim of complex-valued neuro-fuzzy inference system is to find the functional relationship, f, that maps the input to the output: $f(\mathbf{x} \rightarrow \mathbf{y})$. For notational convenience, the superscript t is dropped in rest of the discussion.

A. Architecture

Complex-valued neuro-fuzzy inference system is a four layered network which realizes zero-order Takagi-Sugeno-Kang (TSK) based fuzzy inference system. The architecture of CNFIS is presented in fig. 1. A detailed description of each layer is presented herein.

- Input layer: This layer is a linear layer with m nodes. There are as many nodes as the number of features for the given problem. It transmits the complex-valued input features directly to the Gaussian layer. The output of the l^{th} input node is given as

$$u_l = x_l, \quad l = 1,\ 2,\ \cdots,\ m \qquad (1)$$

- Gaussian layer: This layer consists of K nodes where, each node represents each hidden rule antecedent in the network. The nodes in this layer compute the membership value of each input node by employing a Gaussian activation function and aggregates them. The Gaussian membership function projects the complex-valued input features to a real hyperplane $\mathbb{C} \to \mathbb{R}$, resulting in a real-valued hidden layer response. The firing strength of k^{th} rule is given by

$$\phi_k(\mathbf{u}) = \exp\left(-(\mathbf{u}-\boldsymbol{\mu}_j)^H(\Sigma_{jj}^H\Sigma_{jj})^{-1}(\mathbf{u}-\boldsymbol{\mu}_j)\right) \quad (2)$$

where, $\mathbf{u}$ is the complex-valued input feature, $\boldsymbol{\mu}_k$ is the center of the Gaussian rule antecedent and Σ_{jj} is the width of the Gaussian rule.

- Normalization layer: This layer consists of as many nodes as the Gaussian layer. The function of this layer is to normalize the outputs of the Gaussian layer. The output of the k^{th} normalization node is

$$\bar{\phi}_k = \frac{\phi_k}{\sum_{l=1}^{K}\phi_l}, \quad k = 1,\ 2,\ \cdots,\ K \quad (3)$$

- Output layer: The output layer is a linear layer with n nodes representing output features. The predicted output is given by

$$\hat{y}_j = \sum_{k=1}^{K}\alpha_{jk}\bar{\phi}_k \quad j = 1, 2, \cdots, n \quad (4)$$

Here, α_{jk} is the complex-valued rule consequent between the k^{th} nomalization layer node and j^{th} output layer node.

Next, we shall describe the complex gradient descent based learning algorithm for CNFIS.

B. Gradient Descent Algorithm

Let the predicted complex-valued output for t^{th} sample be given by $\hat{\mathbf{y}}$ where $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_n] \in \mathbb{R}^n$ and the actual output be given by $\mathbf{y}$. The sum of squares error can be given as

$$E = \frac{1}{2}\sum_t\left(\mathbf{e}^H\mathbf{e}\right);\ where\ \mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} \quad (5)$$

where H represents hermitian of a complex number.

The error based gradient descent algorithm proceeds by updating the parameter depending on gradient of error with respect to the parameter.

Output weight update: Let the gradient of error with respect to j^{th} output weight α_j be given by $\nabla_{\boldsymbol{\alpha}_j}E$. Since the cost function to be minimized E is a real-valued function, its gradient with respect to a complex variable is given as

$$\nabla_{\boldsymbol{\alpha}_j}E = \left(\frac{\partial E}{\partial \boldsymbol{\alpha}_j}\right)^H \quad (6)$$

which in turn could be written as (ref. Eq.(5))

$$\nabla_{\boldsymbol{\alpha}_j}E = \frac{1}{2}\left(\frac{\partial \mathbf{e}^H\mathbf{e}}{\partial \boldsymbol{\alpha}_j}\right)^H \quad (7)$$

$$= \frac{1}{2}\left(\mathbf{e}^H\frac{\partial \mathbf{e}}{\partial \boldsymbol{\alpha}_j} + \mathbf{e}\frac{\partial \mathbf{e}^H}{\partial \boldsymbol{\alpha}_j}\right)^H \quad (8)$$

From Eqs. (4) and (5)

$$\frac{\partial \mathbf{e}}{\partial \boldsymbol{\alpha}_j} = \frac{\partial(\mathbf{y} - \boldsymbol{\alpha}\bar{\phi}^T)}{\partial \boldsymbol{\alpha}_j} = -\bar{\phi}_j^T \quad (9)$$

$$\frac{\partial \mathbf{e}^H}{\partial \boldsymbol{\alpha}_j} = 0 \quad (10)$$

Therefore,

$$\nabla_{\boldsymbol{\alpha}_j}E = -\frac{1}{2}\mathbf{e}\cdot\bar{\phi}_j^T \quad (11)$$

The corresponding output weight update equation is given as

$$\boldsymbol{\alpha}_j^* = \boldsymbol{\alpha}_j + 0.5\cdot\eta_\alpha\cdot\mathbf{e}\cdot\bar{\phi}_j^T \quad (12)$$

where $\boldsymbol{\alpha}^*$ denotes output weight vector at the succeeding epoch and η_α is the pre-defined learning rate.

Gaussian rule center update: Let the gradient of error with respect to the k^{th} rule center be represented as $\nabla_{\boldsymbol{\mu}_k}E$. Similar to output weight update equations,

$$\nabla_{\boldsymbol{\mu}_k}E = \left(\frac{\partial E}{\partial \boldsymbol{\mu}_k}\right)^H = \frac{1}{2}\left(\frac{\partial \mathbf{e}^H\mathbf{e}}{\partial \boldsymbol{\mu}_k}\right)^H \quad (13)$$

$$= \frac{1}{2}\left(\mathbf{e}^H\frac{\partial \mathbf{e}}{\partial \boldsymbol{\mu}_k} + \mathbf{e}\frac{\partial \mathbf{e}^H}{\partial \boldsymbol{\mu}_k}\right)^H \quad (14)$$

Here,

$$\frac{\partial \mathbf{e}}{\partial \boldsymbol{\mu}_k} = \frac{\partial \mathbf{e}}{\partial \bar{\phi}_k}\frac{\partial \bar{\phi}_k}{\partial \phi_k}\frac{\partial \phi_k}{\partial \boldsymbol{\mu}_k} \text{ and} \quad (15)$$

$$\frac{\partial \mathbf{e}^H}{\partial \boldsymbol{\mu}_k} = \frac{\partial \mathbf{e}^H}{\partial \bar{\phi}_k}\frac{\partial \bar{\phi}_k}{\partial \phi_k}\frac{\partial \phi_k}{\partial \boldsymbol{\mu}_k} \quad (16)$$

Since $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$, substituting Eq.(4)

$$\frac{\partial \mathbf{e}}{\partial \bar{\phi}_k} = -\boldsymbol{\alpha}_k^T \quad (17)$$

$$\frac{\partial \mathbf{e}^H}{\partial \bar{\phi}_k} = -\boldsymbol{\alpha}_k^H \quad (18)$$

$$(19)$$

and

$$\frac{\partial \bar{\phi}_k}{\partial \phi_k}\frac{\partial \phi_k}{\partial \boldsymbol{\mu}_k} = \bar{\phi}_k(1-\bar{\phi}_k)(\mathbf{u}-\boldsymbol{\mu}_k)^H(\Sigma_k^H\Sigma_k)^{-1} \quad (20)$$

Therefore the gradient $\nabla_{\boldsymbol{\mu}_k}E$ can be obtained as given in Eq. (21) and the update equation is given by Eq. (22). In the equation, η_μ is the learning rate corresponding to the Gaussian rule center.

$$\nabla_{\boldsymbol{\mu}_k}E = -\frac{1}{2}\left[\kappa(\Sigma_k^H\Sigma_k)^{-1}(\boldsymbol{\alpha}_k^T e^H + \boldsymbol{\alpha}_k^H e^T)\right]^H \quad (21)$$

$$\boldsymbol{\mu}_k^* = \boldsymbol{\mu}_k - \eta_\mu\cdot\left[\kappa(\Sigma_k^H\Sigma_k)^{-1}(\boldsymbol{\alpha}_k^T e^H + \boldsymbol{\alpha}_k^H e^T)\right]^H \quad (22)$$

where, for notational convenience, we assign

$$\kappa = \bar{\phi}_k(1-\bar{\phi}_k)(\mathbf{u}-\boldsymbol{\mu}_k)^H \quad (23)$$

Gaussian rule width update: Assuming that the gradient of error with respect to the k^{th} rule width be represented as $\nabla_{\Sigma_k}E$,

$$\nabla_{\Sigma_k} E = \left(\frac{\partial E}{\partial \Sigma_k}\right)^H = \frac{1}{2}\left(\frac{\partial \mathbf{e}^H \mathbf{e}}{\partial \Sigma_k}\right)^H \tag{24}$$

$$= \frac{1}{2}\left(\mathbf{e}^H \frac{\partial \mathbf{e}}{\partial \Sigma_k} + \mathbf{e}\frac{\partial \mathbf{e}^H}{\partial \Sigma_k}\right)^H \tag{25}$$

where,

$$\frac{\partial \mathbf{e}}{\partial \Sigma_k} = \frac{\partial \mathbf{e}}{\partial \bar{\phi}_k}\frac{\partial \bar{\phi}_k}{\partial \phi_k}\frac{\partial \phi_k}{\partial \Sigma_k} \quad \text{and} \tag{26}$$

$$\frac{\partial \mathbf{e}^H}{\partial \Sigma_k} = \frac{\partial \mathbf{e}^H}{\partial \bar{\phi}_k}\frac{\partial \bar{\phi}_k}{\partial \phi_k}\frac{\partial \phi_k}{\partial \Sigma_k} \tag{27}$$

Therefore,

$$\frac{\partial \bar{\phi}_k}{\partial \phi_k}\frac{\partial \phi_k}{\partial \Sigma_k} = \bar{\phi}_k(1-\bar{\phi}_k)(\mathbf{u}-\boldsymbol{\mu}_k)^H(\Sigma_k^H\Sigma_k)^{-2}(\mathbf{u}-\boldsymbol{\mu}_k)\Sigma^H \tag{28}$$

Combining the above equations, the width update equation is given in Eq.(29), where η_Σ corresponds to the learning rate of the Gaussian rule width.

$$\Sigma_k^* = \Sigma_k - \eta_\Sigma \cdot [\kappa(\Sigma_k^H\Sigma_k)^{-2}(\mathbf{u}-\boldsymbol{\mu}_k)\Sigma_k^H(\boldsymbol{\alpha}_k^T e^H + \boldsymbol{\alpha}_k^H e^T)]^H \tag{29}$$

The learning rate affects the convergence of CNFIS significantly. A large learning rate results in failure to converge whereas a very small learning rate results in slow convergence. In the succeeding sections the effect of learning rate and initial parameters will be elucidated. In this paper, the number of rules were arbitrarily chosen and the initial centers were chosen randomly from the training data set.

In the next section, we present the performance comparison of CNFIS on wind speed and direction prediction problem.

IV. Performance Evaluation

In this section, we evaluate the performance of CNFIS on a real world problem of wind speed prediction. For this study, wind speed and direction data obtained from Iowa department of transportation[2] (USA) is employed. For evaluation, data which is sampled for every ten minutes was downloaded for the period February 1, 2011 to February 8, 2011. These sampled data were averaged over one hour interval from which five features are extracted for training and testing the network. In this study, the network is trained using 500 samples and testing using remaining 100 samples. The CNFIS code was executed in Matlab R2011a in Windows environment with memory of 4GB.

The history of magnitude error and phase error with respect to number of epochs is given in Fig. 2. It can be noticed from the figure that the magnitude and phase error decreases significantly within the first 1000 epochs. This is due to the fact that the gradient descent algorithm employed optimizes the parameters based on magnitude error alone.

Next, the performance analysis with respect to prediction accuracy is shown. The actual speed vs predicted speed and actual wind direction vs prediction wind direction in training phase is shown in fig. 3. It could be seen that CNFIS is able to learn the trend in the wind speed and direction, efficiently.

[2]http://weatherview.iowadot.gov/

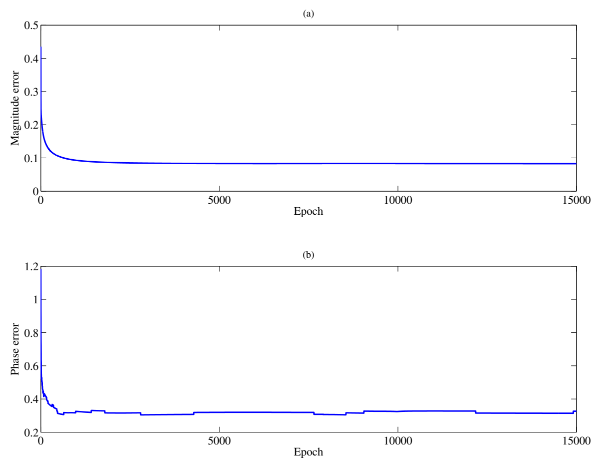

Fig. 2. (a) Magnitude error vs Epoch (b) Phase error vs Epoch

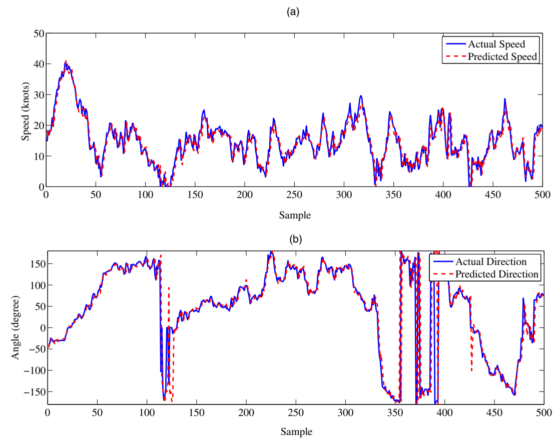

Fig. 3. (a) Speed (knots) vs Training sample (b) Direction vs Training sample

Similarly, the performance analysis for test samples is shown in fig. 4. The figure shows actual speed vs predicted speed and actual wind direction vs predicted wind direction for each test sample. It could be noticed that CNFIS predicts the wind speed and direction with a little error for almost all the instances.

Now, we shall compare the performance of the developed algorithm with augmented complex LMS (ACLMS) [5] algorithm based on the prediction gain, R_p (dB). The prediction gain is given by

$$R_p \triangleq 10\log_{10}\left(\frac{\sigma_{\mathrm{x}}^2}{\sigma_{\mathrm{e}}^2}\right) \tag{30}$$

where, σ_x^2 represents the variance of actual output signals $(\mathbf{y})$ and σ_e^2 denotes the variance of error signal $(\mathbf{e})$.

The table I gives the number of hidden nodes employed and prediction gain of ACLMS and CNFIS. It can be seen from the table that by employing the same number of hidden rules,

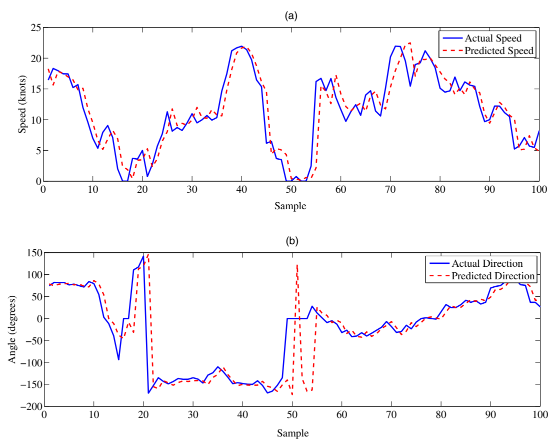

Fig. 4. (a) Speed (knots) vs Testing sample (b) Direction vs Testing sample

TABLE I
PERFORMANCE COMPARISON FOR WIND PREDICTION PROBLEM

Algorithm	No. hidden nodes	R_p
ACLMS	10	8.069
CNFIS	**10**	**11.281**

CNFIS is able to achieve better prediction gain as compared to ACLMS. The TSK based inference mechanism is able to better represent the information in the sample. It should be noted that both ACLMS and CNFIS were repeatedly trained until convergence.

V. CONCLUSION

In this paper, a complex-valued neuro-fuzzy inference system (CNFIS) has been developed and its complex gradient descent algorithm employing based on Wirtinger calculus has been derived. The CNFIS is a four layer TSK type neuro-fuzzy inference system, which employs a real-valued Gaussian activation function. In order to evaluate the performance of CNFIS, a real world wind speed and direction prediction problem was considered. Training and testing data are obtained by representing speed and direction of wind as complex-valued features and the performance comparison with other algorithm available in literature indicate motivating results.

REFERENCES

[1] J. Bregains and F. Ares, “Analysis, synthesis and diagnostics of antenna arrays through complex-valued neural networks,” *Microwave and Optical Technology Letters*, vol. 48, no. 8, pp. 1512 – 1515, 2006.

[2] M. Li, G. Huang, P. Saratchandran, and N. Sundararajan, “Complex valued growing and pruning rbf neural networks for communication channel equalization,” *IEEE Proc. Vision, Image and Signal Processing*, vol. 153, no. 4, pp. 411 – 418, 2006.

[3] I. Aizenberg, D. Paliy, J. Zurada, and J. Astola, “Blur identification by multilayer neural network based on multivalued neurons,” *IEEE Trans. on Neural Networks*, vol. 19, no. 5, pp. 883 – 898, 2008.

[4] M. Muezzinoglu, C. Guzelis, and J. Zuruda, “A new design method for the complex values multistate hopfield associative memory,” *IEEE Trans. on Neural Networks*, vol. 14, no. 4, pp. 891 – 899, 2003.

[5] D. Mandic, S. Javidi, S. Goh, A. Kuh, and K. Aihara, “Complex valued prediction of wind profile using augmented complex statistics,” *renewable Energy*, vol. 34, no. 1, pp. 196 – 201, 2009.

[6] A. Ghani, M. Amin, and K. Murase, “Real time hand gesture recognition using complex valued neural network (cvnn),” in *Intl. Conf. on Neural Information Processing*, 2011, p. accepted.

[7] S. Haykins, *Neural networks: A comprehensive foundation*. Upper Saddle River, NJ: Prentice Hall, 1999.

[8] R. Remmert, *Theory of complex functions*. Derlin: Springer Verlag, 1991.

[9] H. Rong, N. Sundararajan, G. Huang, and P. Saratchandran, “Sequential adaptive fuzzy inference system SAFIS for nonlinear system identification and prediction,” *Fuzzy Sets and Syst.*, vol. 157, no. 9, pp. 1260 – 1275, 2006.

[10] P. Angelov and P. Filev, “An approach to online identification of Takagi-Sugeno fuzzy models,” *IEEE Trans. Syst., Man and Cybern., Part B: Cybern.*, vol. 34, no. 1, pp. 484 – 498, 2004.

[11] R. Fischer, *Precoding and signal shaping for digital transmission*. John Wiley and Sons, 2005.

[12] S. Chen, S. McLaughlin, and N. Mulgrew, “Complex valued radial basis function network, part I: Network architecture and learning algorithms,” *EURASIP Singal Processing Hournal*, vol. 35, no. 1, pp. 19 – 31, 1994.

[13] H. Leung and S. Haykin, “Complex backpropagation algorithm,” *IEEE T. Signal Processing*, vol. 39, no. 9, pp. 2101 – 2104, 1991.

[14] A. Hirose, “Continuous complex-valued back-bropagation learning,” *Electronic Letters*, vol. 28, no. 20, pp. 1854 – 1855, 1992.

[15] N. Benvenuto and F. Piazza, “On complex backpropagation algorithm,” *IEEE T. Signal Processing*, vol. 40, no. 4, pp. 967 – 969, 1992.

[16] T. Nitta, “An extension of the bac-propagation algorithm to complex numbers,” *Neural Networks*, vol. 10, no. 8, pp. 1391 – 1415, 1997.

[17] J. Patra, R. Pal, R. Baliarsingh, and G. Panda, “Nonlinear channel equlaization for qam signal constellation using artificial neural networks,” *IEEE Trans. Syst., Man and Cybern., Part B: Cybern.*, vol. 29, no. 2, pp. 262 – 271, 1999.

[18] T. Kim and T. Adali, “Fully complex multilayer perceptron network for non-linear signal processing,” *Journal of VLSI, Signal Processing*, vol. 32, no. 1 – 2, pp. 29 – 43, 2002.

[19] S. Yang, C. Ho, and S. Siu, “Sensitivity analysis of the split-complex valued multilayer perceptron due to the errors of the i.i.d. inputs and weights,” *IEEE T. Neural Networks*, vol. 18, no. 5, pp. 1280 – 1293, 2007.

[20] D. Jianping, N. Sundararajan, and P. Saratchandran, “Complex valued minimal resource allocation network for nonlinear signal processing,” *Intl. Journal of Neural Systems*, vol. 10, no. 2, pp. 95 – 106, 2000.

[21] A. Hirose, *Complex valued neural networks*, ser. Studies in Computational Intelligence. Springer-Verlag New York, Inc, 2006.

[22] C. You and D. Hong, “Nonlinear blind equalization schemes using complex valued multilayer feedforward neural networks,” *IEEE T. Neural Networks*, vol. 9, no. 6, pp. 1442 – 1455, 1998.

[23] R. Savitha, S. Suresh, N. Sundararajan, and P. Saratchandran, “A new learning algorithm with logarithmic performance index for complex-valued neural networks,” *Neurocomputing*, vol. 72, no. 16-18, pp. 3771–3781, 2009.

[24] S. Suresh, R. Savitha, and N. Sundararajan, “A fast learning fully complex-valued relaxation network (fcrn),” in *IEEE Intl. Joint Conf. on Neural Networks*, 2011, pp. 1372 – 1377.

[25] R. Savitha, S. Suresh, and N. Sundararajan, “A meta-cognitive learning algorithm for a fully complex-valued relaxation network,” *Neural Networks*, 2012, http://dx.doi.org/10.1016/j.neunet.2012.02.015.

[26] ——, “A fully complex-valued radial basis function network and its learning algorithm,” *International Journal of Neural Systems*, vol. 19, no. 4, pp. 253 – 267, 2009.

[27] R. Savitha, S. Suresh, N. Sundararajan, and H. Kim, “A fully complex valued radial basis function classifier for real valued classification problems,” *Neurocomputing*, vol. 78, no. 1, pp. 104 – 110, 2012.

[28] R. Savitha, S. Suresh, and N. Sundararajan, “Metacognitive learning in a fully complex-valued radial basis function neural network,” *Neural Computation*, vol. 24, no. 5, pp. 1297–1328, 2012.

[29] T. Nitta, “The computational power of complex-valued neuron,” *Artificial Neural Networks and Neural Information Processing ICANN/ICONIP. Lecture Notes in Computer Science*, vol. 2714, pp. 993 – 1000, 2003.

[30] ——, “Orthogonality of decision boundaries of complex-valued neural networks,” *Neural Computation*, vol. 16, no. 1, pp. 73 – 97, 2004.

[31] M. F. Amin and K. Murase, “Single-layered complex-valued neural network for real-valued classification problems,” *Neurocomputing*, vol. 72, no. 4-6, pp. 945 – 955, 2009.

[32] I. Aizenberg and C. Moraga, "Multilayer feedforward neural network based on multi-valued neurons (MLMVN) and a backpropagation learning algorithm," *Soft Computing*, vol. 11, no. 2, pp. 169 – 183, 2007.

[33] R. Savitha, S. Suresh, N. Sundararajan, and H. J. Kim, "Fast learning fully complex-valued classifiers for real-valued classification problems," *D. Liu et al. (Eds.): ISNN 2011, Part I, Lecture Notes in Computer Science (LNCS)*, vol. 6675, pp. 602–609, 2011.

[34] R. Savitha, S. Suresh, and N. Sundararajan, "A fast learning complex-valued neural classifier for real-valued classification problems," in *Intl. Joint. Conf. Neural Networks*, 2011, pp. 2243 – 2249.

[35] ——, "Fast learning circular complex-valued extreme learning machine (CC-ELM) for real-valued classification problems," *Information Sciences*, vol. 187, no. 1, pp. 277 – 290, 2012.

[36] R. V. Babu, S. Suresh, and R. Savitha, "Human action recognition using a fast learning fully complex-valued classifier," *Neurocomputing (accepted)*, 2012.

[37] T. Kim and T. Adali, "Approximation by fully complex multilayer perceptron," *Neural Computation*, vol. 15, no. 7, pp. 1641 – 1666, 2003.

[38] X. Chen, Z. Tang, C. Variappan, S. Li, and T. Okada, "A modified error backpropagation algorithm for complex-valued neural networks," *Intl. J. Neural Systems*, vol. 15, no. 6, pp. 435 – 443, 2005.

[39] S. Suresh, R. Savitha, and N. Sundararajan, "A sequential learning algorithm for complex-valued self-regulating resource allocation network-csran," *IEEE Transactions on Neural Networks*, vol. 22, no. 7, pp. 1061 – 1072, 2011.