A Complex-Valued Neuro-Fuzzy Inference System and its Learning Mechanism

K. Subramanian, R. Savitha *, S. Suresh

Center for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore.
**Corresponding Author: savi0001@ntu.edu.sg*

Abstract

In this paper, we present a Complex-valued Neuro-Fuzzy Inference System (CNFIS) and develop its meta-cognitive learning algorithm. CNFIS has four layers-an input layer with m rules, a Gaussian layer with K rules, a normalization layer with K rules and an output layer with n rules. The rules in the Gaussian layer map the m-dimensional complex-valued input features to a K-dimensional real-valued space. Hence, we use the Wirtinger calculus to obtain the complex-valued gradients of the real-valued function in deriving the learning algorithm of CNFIS. Next, we also develop the meta-cognitive learning algorithm for CNFIS, referred to as, "Meta-cognitive Complex-valued Neuro-Fuzzy Inference System (MCNFIS)". CNFIS is the cognitive component of MCNFIS and a self-regulatory learning mechanism that decides *what-to-learn*, *how-to-learn*, and *when-to-learn* in a meta-cognitive framework is its meta-cognitive component. Thus, for every epoch of the learning process, the meta-cognitive component decides if each sample in the training set must be deleted or used to update the parameters of CNFIS or to be reserved for future use.

The performances of CNFIS and MCNFIS are studied on a set of approximation and real-valued classification problems, in comparison to existing complex-valued learning algorithms in the literature. First, we evaluate the approximation performances of CNFIS and MCNFIS on a synthetic complex-valued

Email addresses: kartick1@e.ntu.edu.sg (K. Subramanian), ssundaram@ntu.edu.sg (S. Suresh)

function approximation problem, an adaptive beam forming problem and a wind prediction problem. Finally, we study the decision making performance of CNFIS and MCNFIS on a set of benchmark real-valued classification problems from the UCI machine learning repository. Performance study results on approximation and real-valued classification problems show that CNFIS and MCNFIS outperform existing algorithms in the literature.

Keywords: Complex-valued neural network, fuzzy inference system, Wirtinger calculus, meta-cognition, self-regulation.

---

1. Introduction

Soft computing techniques such as artificial neural networks, fuzzy logic, etc. are gaining popularity in solving many real-world problems, due to their exceptional ability in fitting non-linear models and representing inexact information. Some of these applications involve power system analysis [1], medicine [2], etc.

In some of the applications such as adaptive signal processing [3], telecommunication [4] and image processing [5, 6], the signals are inherently complex-valued. In certain applications, such as prediction of wind speed [7] and tree representation of hand in hand gesture recognition system [8], which involves a pair of independent signals, it is advantageous to represent the pair of signals as single complex-valued feature. Above mentioned applications require soft computing techniques which could efficiently represent and retain the physical properties of complex-valued signals and exploit the computational advantages of the complex-valued rules [9], efficiently.

There are a number of complex-valued neural networks in literature and a short review of the same is presented in Section 2. One of the issues in developing complex-valued neural networks is the 'non-holomorphic' behavior of commonly employed activation functions. In complex domain, boundedness and analyticity cannot be achieved together; and boundedness of the activation is preferred over analyticity [10]. Although holomorphic activation functions with singularities have been employed in literature [11, 12], an alternative approach

is to employ an activation function that projects the complex-valued input into a real-valued feature space and then maps it back to the complex-domain ($\mathbb{C} \rightarrow \mathbb{R} \rightarrow \mathbb{C}$) [13]. However, such an approach requires computing the gradients of the real-valued error function and real-valued activation function with respect to the complex-valued network parameters. Commonly, this issue is resolved by splitting the complex-valued parameters into a pair of real-valued parameters and calculating the real-valued gradients with respect to the real and imaginary parts of the parameter, independently. However, this technique results in loss of cross-correlation between the real and the imaginary parts of the complex-valued feature [12].

Wirtinger calculus [14] provides a platform to derive the complex-valued gradient of real-valued signals and hence, they can be used to avoid this issue. Wirtinger calculus has been employed to derive the gradient descent based learning algorithm of a feedforward complex-valued neural network in [15]. It has also been used to derive the gradients of a holomorphic activation function and a non-holomorphic error function of a fully complex-valued radial basis function network in [16]. Although, the complex-valued neural networks developed in [5, 12, 16–18] perform better than existing networks, they lack the ability to represent fuzzy information and interpret the rules from the resulting model. It has been shown in the literature of real-valued neuro-fuzzy inference system [19, 20] that neuro-fuzzy inference systems are capable of combining the representability of neural networks and interpretability of fuzzy inference system. Hence in this paper, we propose a Complex-valued Neuro-Fuzzy Inference System (CNFIS) which realizes Takagi-Sugeno-Kang type-0 fuzzy inference mechanism.

The proposed CNFIS maps the complex-valued inputs to real-valued feature space, employing a Gaussian membership function, and back to complex-valued output space, by virtue of complex-valued weights. For training the network, we develop a gradient-descent based learning algorithm which optimizes the network parameters with respect to real-valued error function. In order to avoid the loss of correlation between the real and imaginary parts of inputs

and the real and imaginary parts of outputs, we propose a fully complex-valued gradient-descent based learning algorithm derived using Wirtinger calculus.

One of the disadvantages of available learning algorithms including gradient-descent based learning algorithm is their lack of ability to judge their knowledge with respect to current sample and their own knowledge. As a result, these algorithms assume uniform distribution of knowledge among all the training samples and learn them, in the sequence in which they are presented. This will result in over-training. In human beings, it has been reported that the meta-cognitive aspect of brain helps them to self-regulate the knowledge acquisition by providing a mean to accurately assess ones current knowledge, identify new information and provide strategies to acquire that new knowledge [21]. The term meta-cognition was first coined by Flavell [22] as "one's knowledge concerning one's own cognitive processes and products or anything related to them". Following it, various other views and models of meta-cognition has been proposed such as Wellman's model [23], Perlovsky's knowledge instinct theory [24], Nelson and Narens model [25], Flavell's model [26].

In Wellman's model, which together with Flavell's early work, inspired recent research in this field, meta-cognition is viewed as involving separate but related cognitive processes with a self-monitoring component of level of comprehension. Perlovsky's knowledge instinct theory proposes that each individual has craving to be creative which stems from their knowledge of themselves and the environment [24]. On the contrary to [26] which focusses on knowledge about memory aspect of meta-cognition, Nelson and Narens model [25] presents a general framework for integrating meta-cognition and meta-memory, by the interactive process of monitoring and control. A detailed review on the models of meta-cognition is available in [27]. It could be noticed that one of the most comprehensive model of meta-cognition is the one proposed by Nelson and Narens. Many works in the literature of artificial neural networks [16, 28–31] and neuro-fuzzy inference systems [32] have employed this model of meta-cognition and it have been shown that a learning algorithm with meta-cognition performs significantly better than existing algorithms. Hence in this work, we also pro-

pose a meta-cognitive learning scheme for CNFIS referred to as Meta-Cognitive Complex-Valued Neuro-Fuzzy Inference System (MCNFIS).

The performances of CNFIS and MCNFIS are evaluated on a set of benchmark and practical approximation and classification problems. First, a synthetic complex-valued function approximation problem and an adaptive beamforming problem are considered to show the performance of CNFIS and MCNFIS in solving complex-valued function approximation. Then, to study its performance in dealing with a pair of independent real-valued signals, we compare the performance of the proposed algorithms on a wind prediction problem. Next, the classification performance of CNFIS and MCNFIS is studied on a set of benchmark real-valued classification problem from the UCI machine learning repository [33]. In all these problems, performance of the proposed algorithms was found to be better than other algorithms used in comparison.

To summarize, the major contribution of this paper are as follows:

- Development of a Complex-Valued Neuro-Fuzzy Inference System (CNFIS), for the first time in literature. It realizes Takagi-Sugeno-Kang type fuzzy inference mechanism in complex-valued neural network architecture. The rule antecedents and consequents of the proposed network are fully complex-valued.

- Development of gradient descent based learning algorithm (first time for the proposed CNFIS) which is derived using Wirtinger calculus. The use of Wirtinger calculus helps preserve the amplitude-phase correlation of the complex-valued quantities during learning.

- The use of the proposed CNFIS to solve practical wind speed and direction prediction problem.

- In addition, it has been shown in literature that incorporating the principles of meta-cognition helps the network learn and generalize the data more efficiently. Hence, we have incorporated such a meta-cognitive learning scheme in the training of CNFIS.

- The performance comparison of CNFIS with other complex-valued algorithms available in literature shows its improved performance, for both approximation as well as classification problems.

The paper is organized as follows. In Section 2, we present a literature review on complex-valued neural networks. In Section 3, we present a complex-valued neuro-fuzzy inference system and derive its fully complex-valued gradient-descent based learning algorithm. We then present a meta-cognitive learning scheme to improve the learning ability of CNFIS. We evaluate the performance of the developed network and its meta-cognitive learning scheme for regression problems in Section 4. In Section 5, the performance on classification problems is evaluated. Finally, Section 6 presents our conclusions from this study.

2. Literature Review on Complex-Valued Neural Network

Complex-valued neural networks deal with information in the complex domain. According to the nature of input, output and weight parameters, complex-valued neural networks could be divided into split complex-valued neural networks and fully complex-valued neural networks.

Split complex-valued networks could further be divided into two types based on the network parameters, namely, split complex-valued network with real-valued weights and real-valued activation functions and split complex-valued networks with complex-valued weights and real-valued activation functions. Initial approaches in processing complex-valued signals involved splitting the signals into a pair of real-valued signals and employing a real-valued network to deal with them. As each complex-valued signal is split into two real-valued signals, the number of neurons in the input and output layers are twice the number of inputs and outputs. This results in increased network size and hence, more parameters to be estimated. Moreover, this kind of formulation introduces phase distortion in the complex-valued function that is approximated [11].

To overcome the problem of phase distortion due to splitting of complex-valued signals, complex-valued weights with real-valued activation functions

have been presented in [34, 35]. As the Gaussian function that maps the complex-valued input features to real-valued feature space is used as the basis of the activation function here, the response of the hidden layer is real-valued. In [13], a Complex-valued Radial Basis Function (CRBF) network has been developed using the Gaussian activation function and its gradient-descent based batch learning algorithm has been developed based on Cauchy Riemann equations. Moreover, a few sequential learning algorithms like the complex-valued growing and pruning radial basis function network [4] and complex-valued minimal resource allocation network [34] were also developed using real-valued Gaussian activation function. In [15, 16], authors have employed Wirtinger calculus to find the gradients of a real-valued membership with respect to complex-valued network parameters, thus avoiding loss of correlation between the real and imaginary parts of the complex-valued inputs and outputs.

On the other hand, fully complex-valued neural networks with fully complex-valued activation functions and weights are capable of dealing with complex-valued signals as a single entity. Development of an efficient fully complex-valued learning algorithm requires an appropriate fully complex-valued activation function that are entire and bounded [36]. This is refuted by the Liouville's theorem [37] that states that an entire and bounded function in the Complex domain is a constant. Therefore, Kim and Adali [11] reduced the desired properties for a fully complex-valued activation functions to be *analytic (differentiable in a restricted region of the Complex domain) and bounded almost everywhere*. Further, they also suggested a set of elementary transcendental functions that satisfy the relaxed desired properties of a fully complex-valued activation function as a choice of activation functions for the fully complex-valued multi-layer perceptron network. However, since mean squared error function does not consider the phase of the error explicitly [38], authors in [31, 39] have proposed a logarithmic error function for a fully complex-valued multi layer perceptron network. Later, a Fully Complex-valued Radial Basis Function (FC-RBF) network has also been developed using a fully complex-valued activation function in [12]. As these networks use a gradient descent based approach to estimate

the parameters of the network, a Complex-valued Extreme Learning Machine (C-ELM) that chooses the input parameters randomly, and estimates the output weights analytically, has been presented in [40].

Recently, in [41] Nitta has shown that complex-valued neural networks are better decision makers than the real-valued networks due to their orthogonal decision boundaries. Following this many research works have been reported on developing complex-valued classifiers for real-valued classification problems [42–44]. In [42], authors extended FC-RBF to solve real-valued classification problems and in [16], proposed a meta-cognitive learning scheme for FC-RBF. Other complex-valued classifiers available in literature include the Multi Layer Multi Valued Network (MLMVN) [43] and Phase Encoded Complex-Valued Neural Network (PE-CVNN) [44]. In MLMVN, authors employ a multi-valued neuron and multi-valued threshold logic to map the complex-valued input to discrete output. However, in PE-CVNN, the input feature is phase encoded in [0,π] using the transformation $\exp(i\pi x)$, to obtain the complex-valued input feature which retains the relational property and spatial relationship. The activation function employed in [44] is also similar to split complex-valued activation function. Therefore, PE-CVNN uses a gradient descent based learning algorithm, and MLMVN uses a derivative-free global error correcting rule for parameter update, which are both computationally intensive.

Therefore, in [45], authors present a Bilinear Branch-cut Complex-valued ELM (BB-CELM) and a Phase Encoded Complex-valued ELM (PE-CELM) using the transformations used in MLMVN and PE-CVNN, respectively. In BB-CELM, authors introduce a branch cut around 2π as the bilinear transformation [43] to avoid aliasing at 0 and 2π. In PE-CELM, the network with the phase encoded transformation given in [44] is employed. However, the transformation used in MLMVN is not unique and the transformation used in PE-CVNN does not exploit the advantages of orthogonal decision boundaries completely. Therefore, a Circular Complex-valued Extreme Learning Machine (CC-ELM) has been presented for real-valued classification problems in [46]. In [46], the authors employ a circular transformation with a translational/rotational bias

that performs a unique transformation of the real-valued feature to the complex-plane. However, although all these algorithms ensure accurate approximation of the input to the targets, they lack the ability to represent fuzzy information and interpret the rules from the resulting model. Moreover, the fully complex-valued activation functions used in the above-mentioned algorithms and their derivatives have their singularities in the finite region of the Complex plane. These singularities might affect the convergence of these network, if the network hits these points during the training process.

To overcome these issues, we propose a complex-valued neuro-fuzzy inference system which employs a real-valued Gaussian activation function and derive its learning algorithm using fully complex-valued gradients obtained through Wirtinger calculus. Next, we will present the structure and learning algorithm for complex-valued neuro-fuzzy inference system.

3. Complex-valued Neuro-Fuzzy Inference System and its Learning Mechanism

Let the training data set be given by $\{\mathbf{x}^t, \mathbf{y}^t\}_{t=1}^N$, where N represents the total number of input-output pairs, $\mathbf{x}^t = [x_1^t, \cdots, x_m^t] \in \mathbb{C}^m$ is the m-dimensional complex-valued input features and $\mathbf{y}^t = [y_1^t, \cdots, y_n^t] \in \mathbb{C}^n$ is the n-dimensional complex-valued target vectors. Then, the objective of CNFIS is to estimate the functional relationship $\mathbb{F}$ such that, the predicted output

$$\hat{\mathbf{y}}^t = \hat{\mathbb{F}}[\mathbf{x}^t, \mathbf{w}] \tag{1}$$

is as close as possible to the desired target $\mathbf{y}^t$. It should be noted that $\mathbf{w}$ is the complex-valued parameter vector of CNFIS. The difference between the actual and predicted output (error $\mathbf{e}^t = [e_1^t, \cdots, e_n^t]^T$) is defined as,

$$e_b^t = y_b^t - \hat{y}_b^t, \quad b = 1, 2, \cdots, n \tag{2}$$

3.1. Complex-valued Neuro-Fuzzy Inference System Architecture

In this paper, we present a complex-valued neuro-fuzzy inference system which is a four layered network that realizes Takagi-Sugeno-Kang type-0 fuzzy

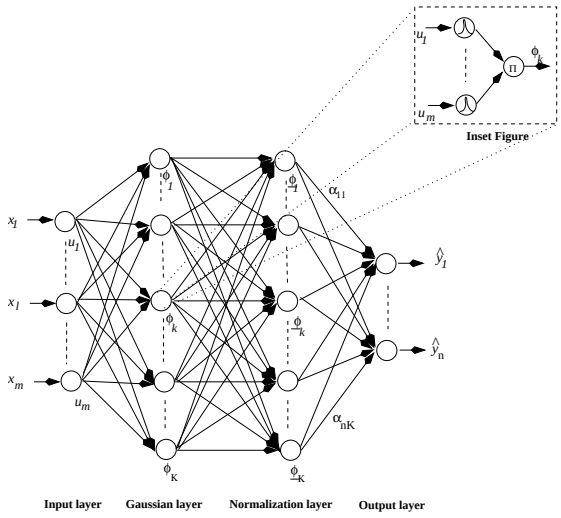

Figure 1: Architecture of CNFIS. The inset figure describes the internal structure of k^{th} Gaussian rule antecedent.

inference mechanism [47]. It consists of an input layer with m nodes, a Gaussian layer with K nodes which forms the rule antecedent and aggregation layer of a neuro-fuzzy inference system, a normalization layer with K nodes and an output layer with n nodes. The complex-valued weight vector connecting the normalization layer and output layer forms the rule consequent parameters. The architecture of CNFIS is presented in Fig.1 and a detailed description of each layer is given below. For the sake of convenience, the superscript t is dropped in future discussions.

Input Layer: The input layer is a linear layer with m nodes. It transmits the complex-valued input features directly to the Gaussian layer. The output of the l^{th} input node is

$$u_l = x_l, \quad l = 1, 2, \cdots, m \tag{3}$$

Gaussian Layer: The nodes in the Gaussian layer employ the Gaussian membership function to compute the membership value of each input node. The nodes in this layer form the antecedent and does rule aggregation operations of a fuzzy inference system. The Gaussian membership function projects the

complex-valued input features to a real-valued hyperplane ($\mathbb{C} \rightarrow \mathbb{R}$), resulting in a real-valued hidden layer response. The firing strength of k^{th} rule is given by

$$\phi_k(\mathbf{u}) = exp(-(\mathbf{u} - \boldsymbol{\mu}_k)^H (\Sigma_{kk}^H \Sigma_{kk})^{-1} (\mathbf{u} - \boldsymbol{\mu}_k)) \tag{4}$$

where, $\boldsymbol{\mu} \in \mathbb{C}^m$ is the complex-valued center of the k^{th} Gaussian rule antecedent and $\Sigma_{kk} \in \mathbb{C}$ is its corresponding Gaussian spread. Here H represents the Hermitian of a complex number.

Normalization Layer: This layer consists of as many nodes as 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

$$\underline{\phi}_k = \frac{\phi_k}{\sum_{p=1}^{K} \phi_p}, \quad k = 1, 2, \cdots, K \tag{5}$$

Output Layer: The output layer is a linear layer with n nodes to represent the n outputs of the function. The predicted output of the b^{th} output node is given by

$$\hat{y}_b = \sum_{k=1}^{K} \alpha_{bk} \underline{\phi}_k, \quad b = 1, 2, \cdots, n \tag{6}$$

It should be noted that α_{bk} is the complex-valued output weight connecting k^{th} normalization layer node and b^{th} output node.

3.2. Complex-valued Gradient-Descent Algorithm

We will now develop a gradient-descent based batch learning algorithm, using the fully complex-valued gradients obtained from Wirtinger calculus, to estimate the free parameters of CNFIS, namely, center of Gaussian rule ($\boldsymbol{\mu}_k$), width of Gaussian rule (Σ_{kk}), and output weight ($\boldsymbol{\alpha}_k$).

Let the sum of squares error be given as

$$E = \frac{1}{2} \sum_{t=1}^{N} (\mathbf{e}^{t^H} \mathbf{e}^t). \tag{7}$$

Henceforth, the superscript t is dropped for the sake of convenience.

Gaussian Rule Center Update: Let the gradient of sum of squares error with respect to center of k^{th} Gaussian rule be given by $\nabla_{\boldsymbol{\mu}_k} E$. Based on

Wirtinger calculus, the derivative of real-valued error with respect to complex-valued rule center is given by

$$\nabla_{\boldsymbol{\mu}_k} E = \left(\frac{\partial E}{\partial \boldsymbol{\mu}_k} \right)^H \tag{8}$$

which could be further expanded based on eqn. (7) and Wirtinger calculus as,

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_k} E &= \frac{1}{2} \left(\frac{\partial \mathbf{e}^H \mathbf{e}}{\partial \boldsymbol{\mu}_k} \right)^H & (9) \\ &= \frac{1}{2} \left(\mathbf{e}^H \frac{\partial \mathbf{e}}{\partial \boldsymbol{\mu}_k} + \mathbf{e} \frac{\partial \mathbf{e}^H}{\partial \boldsymbol{\mu}_k} \right)^H & (10) \end{aligned}$$

Employing the chain rule, we find that

$$\begin{aligned} \frac{\partial \mathbf{e}}{\partial \boldsymbol{\mu}_k} &= \frac{\partial \mathbf{e}}{\partial \underline{\phi}_k} \frac{\partial \underline{\phi}_k}{\partial \phi_k} \frac{\partial \phi_k}{\partial \boldsymbol{\mu}_k} & (11) \\ \frac{\partial \mathbf{e}^H}{\partial \boldsymbol{\mu}_k} &= \frac{\partial \mathbf{e}^H}{\partial \underline{\phi}_k} \frac{\partial \underline{\phi}_k}{\partial \phi_k} \frac{\partial \phi_k}{\partial \boldsymbol{\mu}_k} & (12) \end{aligned}$$

From eqs. (2) and (6), it could be found that

$$\begin{aligned} \frac{\partial \mathbf{e}}{\partial \underline{\phi}_k} &= -\boldsymbol{\alpha}_k^T & (13) \\ \frac{\partial \mathbf{e}^H}{\partial \underline{\phi}_k} &= -\boldsymbol{\alpha}_k^H & (14) \end{aligned}$$

It should be noted that $\underline{\phi}_k$ is a real-valued quantity.

Similarly, based on quotient rule and product rule in differential calculus

$$\frac{\partial \underline{\phi}_k}{\partial \phi_k} \frac{\partial \phi_k}{\partial \boldsymbol{\mu}_k} = -\underline{\phi}_k (1 - \underline{\phi}_k)(\mathbf{u} - \boldsymbol{\mu}_k)^H (\Sigma_k^H \Sigma_k)^{-1} \tag{15}$$

From eqs. (11)-(15), $\nabla_{\boldsymbol{\mu}_k} E$ is obtained as,

$$\nabla_{\boldsymbol{\mu}_k} E = -\frac{1}{2} [\underline{\phi}_k (1 - \underline{\phi}_k)(\mathbf{u} - \boldsymbol{\mu}_k)^H (\Sigma_{kk}^H \Sigma_{kk})^{-1} (\boldsymbol{\alpha}_k^T \mathbf{e}^H + \boldsymbol{\alpha}_k^H \mathbf{e}^T)]^H \tag{16}$$

The center of the Gaussian rule is updated as,

$$\boldsymbol{\mu}_k^* = \boldsymbol{\mu}_k - \eta_\mu \cdot \nabla_{\boldsymbol{\mu}_k} E \tag{17}$$

$$\begin{aligned} \boldsymbol{\mu}_k^* &= \boldsymbol{\mu}_k - \eta_\mu \cdot \nabla_{\boldsymbol{\mu}_k} E \\ &= \boldsymbol{\mu}_k + \eta_\mu [\underline{\phi}_k (1 - \underline{\phi}_k)(\mathbf{u} - \boldsymbol{\mu}_k)^H (\Sigma_{kk}^H \Sigma_{kk})^{-1} (\boldsymbol{\alpha}_k^T \mathbf{e}^H + \boldsymbol{\alpha}_k^H \mathbf{e}^T)]^H & (18) \end{aligned}$$

where, η_μ is the corresponding learning rate and $\boldsymbol{\mu}^*$ denotes rule center at the succeeding epoch.

Gaussian Rule Width Update: Similar to rule center update, let the gradient with respect to width Σ_{kk} be given by $\nabla_{\Sigma_k} E$

$$\begin{aligned} \nabla_{\Sigma_k} E &= \left(\frac{\partial E}{\partial \Sigma_{kk}}\right)^H = \frac{1}{2}\left(\frac{\partial \mathbf{e}^H \mathbf{e}}{\partial \Sigma_{kk}}\right)^H \quad (19) \\ &= \frac{1}{2}\left(\mathbf{e}^H \frac{\partial \mathbf{e}}{\partial \Sigma_{kk}} + \mathbf{e}\frac{\partial \mathbf{e}^H}{\partial \Sigma_{kk}}\right)^H \quad (20) \end{aligned}$$

where,

$$\begin{aligned} \frac{\partial \mathbf{e}}{\partial \Sigma_{kk}} &= \frac{\partial \mathbf{e}}{\partial \underline{\phi}_k} \frac{\partial \underline{\phi}_k}{\partial \phi_k} \frac{\partial \phi_k}{\partial \Sigma_{kk}} \quad \text{and} \quad (21) \\ \frac{\partial \mathbf{e}^H}{\partial \Sigma_{kk}} &= \frac{\partial \mathbf{e}^H}{\partial \underline{\phi}_k} \frac{\partial \underline{\phi}_k}{\partial \phi_k} \frac{\partial \phi_k}{\partial \Sigma_{kk}} \quad (22) \end{aligned}$$

Here,

$$\frac{\partial \underline{\phi}_k}{\partial \phi_k} \frac{\partial \phi_k}{\partial \Sigma_{kk}} = -\underline{\phi}_k(1-\underline{\phi}_k)(\mathbf{u}-\boldsymbol{\mu}_k)^H(\Sigma_{kk}^H\Sigma_{kk})^{-2}(\mathbf{u}-\boldsymbol{\mu}_k)\Sigma_{kk}^H \quad (23)$$

Hence, $\nabla_{\Sigma_k} E$ is given as,

$$\nabla_{\Sigma_k} E = -\frac{1}{2}[\underline{\phi}_k(1-\underline{\phi}_k)(\mathbf{u}-\boldsymbol{\mu}_k)^H(\Sigma_{kk}^H\Sigma_{kk})^{-2}(\mathbf{u}-\boldsymbol{\mu}_k)\Sigma_{kk}^H(\boldsymbol{\alpha}_k^T\mathbf{e}^H + \boldsymbol{\alpha}_k^H\mathbf{e}^T)]^H \quad (24)$$

Combining the above equations, the width update equation is given as,

$$\begin{aligned} \Sigma_{kk}^* &= \Sigma_{kk} - \eta_\Sigma \cdot \nabla_{\Sigma_k} E \\ &= \Sigma_{kk} + \eta_\Sigma[\underline{\phi}_k(1-\underline{\phi}_k)(\mathbf{u}-\boldsymbol{\mu}_k)^H(\Sigma_{kk}^H\Sigma_{kk})^{-2}(\mathbf{u}-\boldsymbol{\mu}_k)\ldots \\ &\ldots\Sigma_{kk}^H(\boldsymbol{\alpha}_k^T\mathbf{e}^H + \boldsymbol{\alpha}_k^H\mathbf{e}^T)]^H \quad (25) \end{aligned}$$

where η_Σ corresponds to the learning rate of the Gaussian rule width.

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

$$\nabla_{\boldsymbol{\alpha}_l} E = \left(\frac{\partial E}{\partial \boldsymbol{\alpha}_l}\right)^H \quad (26)$$

which could be written as

$$\nabla_{\boldsymbol{\alpha}_l} E = \frac{1}{2}\left(\frac{\partial \mathbf{e}^H \mathbf{e}}{\partial \boldsymbol{\alpha}_l}\right)^H \tag{27}$$

$$= \frac{1}{2}\left(\mathbf{e}^H \frac{\partial \mathbf{e}}{\partial \boldsymbol{\alpha}_l} + \mathbf{e}\frac{\partial \mathbf{e}^H}{\partial \boldsymbol{\alpha}_l}\right)^H \tag{28}$$

From Eqs. (6) and (7)

$$\frac{\partial \mathbf{e}}{\partial \boldsymbol{\alpha}_l} = \frac{\partial(\mathbf{y} - \boldsymbol{\alpha}_l \underline{\phi}^T)}{\partial \boldsymbol{\alpha}_l} = -\underline{\phi}_k^T \tag{29}$$

$$\frac{\partial \mathbf{e}^H}{\partial \boldsymbol{\alpha}_k} = 0 \tag{30}$$

Therefore,

$$\nabla_{\boldsymbol{\alpha}_l} E = -\frac{1}{2}\mathbf{e} \cdot \underline{\phi}_k^T \tag{31}$$

The corresponding output weight update equation is given as

$$\boldsymbol{\alpha}_l^* = \boldsymbol{\alpha}_l - \eta_\alpha \cdot \nabla_{\boldsymbol{\alpha}_l} E$$

$$= \boldsymbol{\alpha}_l + \eta_\alpha \mathbf{e} \cdot \underline{\phi}_k^T \tag{32}$$

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

In this paper, we employ three learning rate parameters, viz. center learning rate η_μ, width learning rate η_σ and weight learning rate η_α. The learning rate affects the convergence of CNFIS significantly. A large learning rate results in failure to converge, and a very small learning rate results in slow convergence. Here, we have performed a systematic study to choose the optimal learning rate for each problem. The performance of the network for different learning rates such as 10^{-1}, 10^{-2}, 10^{-3} and 10^{-6} were observed and the learning rate corresponding to the best performance is chosen as the best learning rate.

In this section, we have presented a complex-valued neuro-fuzzy inference system, which employs a real-valued Gaussian membership function and uses the Wirtinger calculus to obtain the complex-valued gradients of the real-valued function in deriving the learning algorithm of CNFIS. The learning mechanism of CNFIS is summarized in Algorithm 1.

Next, we will describe the meta-cognitive learning scheme for CNFIS.

Input: The data set $\{\mathbf{x}^t, \mathbf{y}^t\}_{t=1}^{N}$ to be learnt

Output: Parameters of the network,

centers ($\boldsymbol{\mu}$), spread of Gaussian (Σ) and output weight ($\boldsymbol{\alpha}$)

begin

Initialize the number of rules K

Initialize $\boldsymbol{\mu}_k$, Σ_k and $\boldsymbol{\alpha}_k$ $k = 1,\ 2,\ \cdots,\ K$

Select the number of epochs

while *Number of epochs reached* **do**

for $t\ =\ 1,\ 2,\ \cdots,\ N$ **do**

Compute the network output using Eqn. (6)

Compute the error using Eqn. (2)

Update the center $\boldsymbol{\mu}$ as given in Eqn. (18)

Update the width Σ as given in Eqn. (25)

Update the output weight $\boldsymbol{\alpha}$ as given in Eqn. (32)

end

end

end

Algorithm 1: Gradient Descent based Learning Algorithm for Complex-Valued Neuro-Fuzzy Inference System

3.3. Meta-Cognitive Learning Scheme for Complex-Valued Neuro-Fuzzy Inference System

Meta-cognition refers to the ability of human brain to determine *what-to-learn*, *when-to-learn* and *how-to-learn*, i.e., ability to identify a specific piece of knowledge, judge when to start/stop learning by employing a best learning strategy. In literature of human educational psychology, many models of meta-cognition have been proposed. In this work, we employ a simple and efficient model of meta-cognition proposed by Nelson and Narens [25]. According to Nelson and Narens model, meta-cognition consists of two interrelated components, viz., a cognitive component, which represents knowledge and inference mechanism and a meta-cognitive component which consists of a dynamic model of the

cognitive component. The meta-cognitive component monitors the knowledge in the cognitive component and controls the learning, efficiently [25].

We develop a meta-cognitive learning mechanism for CNFIS referred to as Meta-Cognitive Complex-Valued Neuro-Fuzzy Inference System (MCNFIS) based on Nelson and Narens model of meta-cognition. Similar to this model, MCNFIS consists of two components, viz., a cognitive component which is CNFIS and a meta-cognitive component, which is a self-regulatory learning mechanism. The self-regulatory learning mechanism monitors the instantaneous magnitude (M_e^t) error and instantaneous phase error (P_e^t) of the current sample, and controls the learning by deciding on *what-to-learn*, *when-to-learn* and *how-to-learn*, efficiently.

The instantaneous magnitude error for t^{th} sample is given by

$$M_e^t = \frac{1}{n}\sqrt{\mathbf{e}^{t^H} \cdot \mathbf{e}^t} \tag{33}$$

The instantaneous phase error is given by

$$P_e^t = \frac{1}{n}\sum_{b=1}^{n} |\arg\left(y_b^t \bar{\hat{y}}_b^t\right) \tag{34}$$

where, the function $\arg(\cdot)$ returns the phase of a complex-valued number in $[-\pi, \pi]$, and is given by:

$$\arg(y_b^t) = atan\left(\frac{imag(y_b^t)}{real(y_b^t)}\right) \tag{35}$$

In MCNFIS, the three components of *what-to-learn*, *when-to-learn* and *how-to-learn* are realized by 'sample deletion strategy', 'sample reserve strategy' and 'sample learning strategy', respectively. These three strategies are implemented as follows:

- **Sample deletion strategy**: When prediction error of a sample is very low, it is deleted from the training data set without being learnt. This helps avoid over-training and saves computational time. This strategy is formulated as,

$$\text{IF } M_e^t < \epsilon_m^D \text{ AND } P_e^t < \epsilon_p^D \text{ THEN } \textit{delete the sample} \tag{36}$$

where, ϵ_m^D and ϵ_p^D are the delete magnitude threshold and phase threshold, respectively. Higher value of threshold will result in no samples being deleted, leading to over-fitting, while lower delete threshold will result in frequent pruning, resulting in unstable system. In this work, the thresholds are set in the range [1e^{-5},1e^{-4}].

- **Sample learning strategy**: When prediction error for the current sample is higher than the learning threshold, adjust the network parameters as in Eqs. (18), (25), and (32). This strategy is formulated as,

  IF $M_e^t > \epsilon_m^L$ AND $P_e^t > \epsilon_p^L$ THEN *update the parameters of the network.* (37)

  Here, ϵ_m^L and ϵ_p^L are the learning magnitude and phase thresholds, respectively. These parameters are set in the range [1e^{-4}, 1e^{-3}]. A higher value of these thresholds will result in fewer samples being used in learning, while, a lower value will lead to network over-training due to repeated learning of same information.

  Moreover, the learning threshold are designed to be self-adaptive, such that the network gains coarse knowledge (learn samples with higher error) first, then fine tunes with samples of lower error. These thresholds self-adapt as,

  $$\epsilon_m^L = \delta\epsilon_m^L - (1-\delta) * M_e^t \tag{38}$$

  $$\epsilon_p^L = \delta\epsilon_p^L - (1-\delta) * P_e^t \tag{39}$$

  where, δ is the rate of adaptation set close to 1.

- **Sample reserve strategy**: When a sample is neither deleted nor learnt, it is reserved in the training data set. It might be used in the next epoch for learning.

Thus the sample deletion, the sample learning and the sample reserve strategy address the self-regulatory principles of *what-to-learn*, *when-to-learn* and *how-to-learn* in a meta-cognitive framework. The learning algorithm of MCNFIS is summarized in algorithm 2.

Input: The data set $\{\mathbf{x}^t, \mathbf{y}^t\}_{t=1}^{N}$ to be learnt

Output: Parameters of the network,

centers ($\boldsymbol{\mu}$), spread of Gaussian (Σ) and output weight ($\boldsymbol{\alpha}$)

begin

Initialize the number of rules K

Initialize $\boldsymbol{\mu}_k$, Σ_k and $\boldsymbol{\alpha}_k$ $k = 1,\ 2,\ \cdots,\ K$

Initialize delete threshold ϵ_m^D and ϵ_p^D and learning threshold ϵ_m^L and ϵ_p^L

Select the number of epochs

while *Number of epochs reached* **do**

for $t\ =\ 1,\ 2,\ \cdots,\ N$ **do**

Compute the network output using Eqn. (6)

Compute the error using Eqn. (2)

if $M_e^t < \epsilon_m^D$ AND $P_e^t < \epsilon_p^D$ **then**

Delete the samples from the training data set.

else if $M_e^t > \epsilon_m^L$ AND $P_e^t > \epsilon_p^L$ **then**

Update the center $\boldsymbol{\mu}$ as given in Eqn. (18)

Update the width Σ as given in Eqn. (25)

Update the output weight $\boldsymbol{\alpha}$ as given in Eqn. (32)

else

Reserve the sample in the training data set.

end

end

end

end

Algorithm 2: Meta-Cognitive Learning Scheme for Complex-Valued Neuro-Fuzzy Inference System

Next, we will evaluate the performance of the proposed network on regression problems.

4. Performance Evaluation on Regression Problems

In this section, we will evaluate the performance of the developed CNFIS and its learning scheme on three regression problems. First, we will study the working of the network on a benchmark synthetic complex-valued function approximation problem [12]. Next, we will evaluate the performance of the network on an adaptive beam-forming problem [48] with non-circular signals. Finally, the performance on complex-valued wind-speed prediction problem is evaluated.

4.1. Synthetic Complex-Valued Function Approximation Problem

The synthetic complex-valued function approximation problem [12] is given as

$$\mathbb{F}(\mathbf{x}) = \frac{1}{1.5}\left(x_3 + 10x_1x_4 + \frac{x_2^2}{x_1}\right) \tag{40}$$

where $x_l;\ l = 1, \cdots, 4 \in \mathbb{C}$ are initialized within a ball of radius 2.5. A training data set of 3000 randomly selected samples and a testing data set of 1000 randomly selected samples are used for the study.

The performances of CNFIS and MCNFIS are evaluated in comparison with those of C-ELM [40], IC-MLP [49], CRBF [13] and FC-RBF [12]. Table 1 provides the training time, number of rules/ hidden neurons employed, training and testing root mean squared error (J_{Me}) and phase error (Φ_e). These measures are defined as:

$$J_{Me} = \sqrt{\frac{1}{N \times n}\sum_{t=1}^{N}\sum_{b=1}^{n}(e_b^t \cdot \bar{e_b}^t)} \tag{41}$$

$$\Phi_e = \frac{1}{N \times n}\sum_{t=1}^{N}\sum_{b=1}^{n} |\arg\left(y_b^t \bar{\hat{y}}_b^t\right)| \times \frac{180}{\pi} \tag{42}$$

The results for C-ELM, FC-RBF and CRBF has been reproduced from [17]. It could be seen that CNFIS attains better training as well as generalization

Table 1: Performance comparison for synthetic approximation problem.

Algorithm	Time (sec)	K	Training		Training	
			J_{Me}	ϕ_e (deg.)	J_{Me}	Φ_e (deg.)
C-ELM	0.2	15	0.19	90	0.23	88
IC-MLP	-	10	0.02	0.46	0.04	1.14
CRBF	9233	15	0.15	51	0.18	52
FC-RBF	6013	20	0.019	16	0.048	16
CNFIS	8962	15	0.0054	2.75	0.034	2.78
MCNFIS	5708	15	0.0053	2.39	0.030	2.5

performance as compared to other algorithms used in comparison. The use of TSK type-0 fuzzy inference mechanism and Wirtinger calculus, have contributed to this performance enhancement. Although MCNFIS attains slightly better training as well as testing performance than CNFIS, it does so in significantly lesser time than CNFIS. The meta-cognitive component in the learning algorithm has helped it remove redundant samples thereby avoid over-training and saving computational time.

4.2. Adaptive Beam-Forming Problem

Adaptive beam-forming is an antenna array signal processing problem, where the beams are directed to desired signal directions (beam pointing) and the nulls are directed to the interference directions (null pointing) [16, 48]. A typical beam forming mechanism is constituted by a set of M single-transmit antennas operating at the same carrier frequency and L receiver elements whose spacing (d) is usually set at half the wavelength(λ) of the received signal. Assuming θ to be the angle of incidence that an incoming signal makes with the receiver array broadside, the signal received at the k^{th} receiver antenna element is given as

$$\bar{x}_k = \exp^{\frac{j2\pi k d sin\theta}{\lambda}} \tag{43}$$

The total signal induced at the k^{th} receiver element at a given instant is the

input to the beam former and is given by

$$\mathbf{z} = [\bar{x}_0 + \eta_0\bar{x}_1 + \eta_1\bar{x}_2 + \cdots + \eta_{L-1}\bar{x}_L + \eta_L]^T \tag{44}$$

where, η_k is the noise at the k^{th} element and T represents transpose of the vector.

Let $\mathbf{a} = [a_1, a_2, \cdots, a_k]^T$ be the weight vector for the sensor array. The actual signal transmitted at a given instant (y) is given by

$$y = \mathbf{a}^H\mathbf{z} \tag{45}$$

This transmitted signal (y) forms the target of the beam former. Thus, the objective of an adaptive beam former is to estimate the weight ($\mathbf{a}$ of eqn. (45)), given the transmitted signal and the signal received by the beam former antenna array ($\mathbf{z}$).

In this paper, a five-sensor uniform linear array is considered [48]. The desired signal directions are set as -30 degrees and +30 degrees and the directions of interferences considered are -15 degrees, 0 degrees, and +15 degrees, as shown in [48]. The received signal at array elements ($\mathbf{z}$) is corrupted with an additive Gaussian noise at 50 dB SNR. A training data set of 250 random samples with 50 for each signal or interference angle is used to train the various beam former. A network with ten hidden rules is used for the batch learning of normal complex-valued neuro-fuzzy inference system (CNFIS), meta-cognitive learning based complex-valued neuro-fuzzy inference system (MCNFIS) and IC-MLP [49]. In C-ELM [4], CRBF [13], and FC-RBF [12], 5 hidden neurons are used. The gains for the signals and interference nulls for different beam formers are summarized in Table 2. From the table, it can be observed that CNFIS beam former outperforms all other beam-formers in beam-forming performance except optimal matrix method [50]. MCNFIS on the other hand outperforms even optimal matrix method based beam former in the case of beam-null. Optimal matrix method based beam former out-performs MCNFIS at signal direction of 30° by 0.02 dB.

Table 2: Performance Comparison of Various Complex-Valued Network based Beam Formers.

Direction of Arrival	Gain (db)						
	C-ELM	IC-MLP	CRBF	FC-RBF	CNFIS	MCNFIS	Matrix
Beam -1: -30°	-18.05	-13.87	-17.94	-16.99	-13.97	-13.96	-13.98
Null -1: -15°	-48.28	-55.4	-27.53	-58.45	-56.37	-57.21	-57.02
Null -2: 0°	-41.64	-56.99	-27	-57.23	-56.7	-57.09	-57
Null -3: 15°	-47.5	-56	-28.33	-56.32	-56.66	-57.61	-57.02
Beam -2: 30°	-16.68	-13.86	-17.92	-17	-14.01	-14.00	-13.98

Next, we will evaluate the performance of the developed network and its learning scheme on wind speed and direction prediction problem.

4.3. Wind Prediction

In recent times, the importance of wind as a perennial source of energy has increased. Having said that, the power generated by wind turbines has a non-schedulable nature due to the stochastic nature of weather conditions. Hence the prediction of wind speed and direction is indispensable for efficient wind power generation. Originally, wind speed and direction forecasting were done by considering them as independent variables. However, recent studies have shown a better modeling and forecasting of wind is possible by considering the direction and speed as a complex variable, noting the statistical dependence between the speed and direction [7]. Since the prediction problem can be casted as a function approximation problem, we employ CNFIS for one step forecasting of wind.

We obtained real-world data from Iowa (USA) Department of Transport at the location, Washington (AWG). The data sampled for every 10 minutes was downloaded from the website[1] between a period of February 1, 2011 and February 28, 2011. Then the data were averaged over every 1 hour interval. Among these averaged data points, the first 500 samples were used for training

[1] http://mesonet.agron.iastate.edu/request/awos/1min.php

Table 3: Performance comparison for wind prediction problem.

Algorithm	No. of rules/ neurons	R_p	J_{Me}
ACLMS [7]	10	9.26	0.4701
FLANN [51]	30	8.17	0.5329
CFLN [18]	8	11.29	0.1179
IC-MLP [49]	10	11.23	.1114
FC-RBF [12]	10	11.77	0.1126
CNFIS	10	11.25	0.1053
MCNFIS	10	11.75	0.0836

the CNFIS with and without meta-cognitive learning scheme and the next 100 samples were used as a testing set. The aim of the this problem is to predict the speed and direction of wind at a particular time instant based on speed and direction of wind at previous four time instants.

The prediction gain [7, 36] is used as the performance measure to evaluate the performance of the algorithms.It is defined as

$$R_p \triangleq 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right) \text{ (dB)} \tag{46}$$

where, σ_x^2 and σ_e^2 denote the variance of input signal and the prediction error signal, respectively. A smaller prediction error would result in a larger value of prediction gain. In addition to prediction gain we also employ root mean squared error, as given in Eqn. (41) as a performance measure.

The performance of the proposed algorithms are compared with ACLMS [7], FLANN [51], CFLN [18], IC-MLP [49] and FC-RBF [12]. The number of rules employed, prediction gain and mean squared error for these algorithms are given in Table 3. The results for ACLMS, FLANN, RFLN and CFLN are reproduced from [18]. From the Table 3, it could be seen that the CNFIS and MCNFIS attains lower MSE than other algorithms compared against, with MCNFIS attaining the lowest MSE. Upon analyzing the prediction gain, it could be observed that MCNFIS outperforms ACLMS, FLANN, RLFN, CFLN and IC-MLP by about 0.5 to 2.03 dB. However, FC-RBF attains slightly better

prediction gain (by 0.02 dB) that MCNFIS. This could be attributed to the fully complex-valued activation function employed in the algorithm.

The use of fully complex-valued derivative found using Wirtinger calculus and the neuro-fuzzy inference mechanism has helped the CNFIS achieve better performance than the existing algorithms. In addition, the meta-cognitive components of *what-to-learn*, *when-to-learn* and *how-to-learn* have helped the MCNFIS learning algorithm to learn the functional relationship between the input and the output, efficiently.

Next, we will compare the performance of CNFIS and MCNFIS on real-valued classification problems.

5. Performance Evaluation on Real-Valued Classification Problems

In classification problems, we consider the data set to be given by $\{\mathbf{x}^t, c^t\}_{t=1}^{N}$, where, $\mathbf{x}^t \in \mathbb{R}^m$ and $c^t \in \{1,\ 2, \cdots,\ n\}$ are the real-valued input and corresponding class labels, respectively and n is the number of distinct classes.

Real-valued features are mapped to all the four quadrants of complex domain using a circular transformation as described in [42]. The real-valued class label c^t is converted to coded class label in complex domain ($\mathbf{y}^t \in \mathbb{C}^n$) as

$$y_b^t = \begin{cases} 1+1i, & \text{if } c^t == b \\ -1-1i, & otherwise \end{cases} \qquad b = 1,\ 2, \cdots, n \tag{47}$$

Hence, the aim of the learning algorithm is to find the functional relationship between the inputs and the coded class labels, so as to approximate the decision surface accurately.

For classification problems, it has been proved that a classifier developed using a hinge-loss function can estimate the posterior probability more accurately than a mean square error function [52, 53]. The hinge-loss error ($\mathbf{e}^t = [e_1^t, \cdots, e_n^t] \in \mathbb{R}^n$) is defined as,

$$e_b^t = \begin{cases} 0, & \text{if } y_b^t \hat{y}_b^t > 1 \\ y_b^t - \hat{y}_b^t, & otherwise \end{cases} \qquad b = 1,\ 2, \cdots, n \tag{48}$$

Table 4: Real-valued classification data set

Data set	No. of Features	No. of Classes	No. of samples		I.F.	
			Training	Testing	Training	Testing
Liver	6	2	200	145	0.17	0.14
PIMA	8	2	400	368	0.22	0.39
BC	9	2	300	383	0.26	0.33
ION	34	2	100	251	0.28	0.28
Iris	4	3	45	105	0	0
IS	19	7	210	2100	0	0
VC	18	4	424	422	0.1	0.12
GI	9	6	336	105	0.72	0.77

Hence in this paper, we employ hinge-loss function as error loss function for solving real-valued classification problems.

Now we will evaluate the performance of CNFIS on real-valued classification problems. In this paper we consider five multi-category classification problems (Iris Classification, Image Segmentation (IS), Acoustic Emission (AE) Vehicle Classification (VC) and Glass Identification (GI)) and four binary classification problems (Liver Disorder, PIMA Indian Diabetes, Breast Cancer (BC) and Ionosphere (ION)) chosen from UCI machine learning repository [33]. Details regarding the problems considered, including the number of features, number of classes, number of training and testing samples are provided in Table 4. The table also provides details on imbalance factor (I.F.), which is a measure of class-wise sample density. It is defined as,

$$\text{I.F.} = 1 - \frac{n}{N} \min_{b=1,\cdots,n} n_b \tag{49}$$

From the Table 4, it could be seen that the data sets chosen vary from perfectly balanced problems such as iris and IS to problems with high degree of overlap and imbalance such as GI.

Since CNFIS and MCNFIS are batch learning algorithms, the performance of the developed CNFIS and MCNFIS are compared with three complex-valued

batch learning algorithms: IC-MLP [49], BB-ELM [45] and the Fully Complex-valued Radial Basis Function (FC-RBF) [12]. In order to emphasize the performance improvement in comparison to other real-world algorithms, we compare the proposed network and algorithm on two real-world algorithms including Support Vector Machine (SVM) [54], Extreme Learning Machine (ELM) [55].

In order to evaluate the performance of the network, two measures are employed:

- **Overall classification efficiency** η_o: $\eta_o = \frac{\sum_{b=1}^{n} q_{bb}}{N} \times 100\%$, where, q_{bb} is the total number of correctly classified samples in the b^{th} class.
- **Average classification efficiency** η_a: $\eta_a = \frac{1}{n} \sum_{b=1}^{n} \frac{q_{bb}}{N_b} \times 100\%$, where N_b is the number of samples in class b.

First, we will study the performance of the proposed networks on multi-category classification problems, followed by binary classification problems.

5.1. Multi-Category Classification Problems

The number of rules/ hidden neurons / support vectors, training and testing results for multi-category benchmark classification problems are presented in Tables 5, and 6.

From the table, it could be observed that complex-valued networks perform at least as good as real-valued networks employing fewer number of rules.

In the case of balanced multi-category classification problems, it could be noticed that CNFIS and MCNFIS attains better performance than other complex-valued networks, by employing fewer rules. Moreover, MCNFIS attains atleast better training performance than CNFIS. However, for vehicle classification and glass identification problems, the performance of BB-ELM and FC-RBF is better than CNFIS and MCNFIS. This could be due to the use of fully complex-valued activation function used in BB-ELM as well as FC-RBF. Next, we consider the performance evaluation of binary classification problems.

Table 5: Performance Comparison for Balanced Multi-category Classification Problems

Data Set	Classifier Domain	Classifier	No. of Rules	Training		Testing	
				η_o	η_a	η_o	η_a
Iris Classification	Real Valued	SVM	25	100	100	96.19	96.19
		ELM	10	100	100	96.66	96.66
	Complex	IC-MLP	5	100	100	97.14	97.14
		BB-ELM	10	93.33	93.33	91.43	91.43
		FC-RBF	5	100	100	98.10	98.10
		CNFIS	4	100	100	97.14	97.14
	Valued	MCNFIS	5	100	100	98.10	98.10
Image Segmentation	Real Valued	SVM	127	94.28	94.28	91.38	91.38
		ELM	132	92.86	92.86	90.23	90.23
	Complex	IC-MLP	80	94.29	94.29	91.67	91.67
		BB-ELM	60	95.238	95.238	92.1	92.1
		FC-RBF	38	96.19	97.19	92.33	92.33
		CNFIS	35	94.28	94.28	92.9	92.9
	Valued	MCNFIS	35	96.19	96.19	92.9	92.9

5.2. Binary Classification Problems

Table 7 gives the number of rules and the training overall accuracy for the binary classification problems considered. It could be observed that the CNFIS and MCNFIS performs better than other algorithms considered, for all problems, except PIMA. The use of circular complex-valued signals along with neuro-fuzzy formulation has helped the network attain this better performance. Further, the meta-cognitive components of *what-to-learn*, *when-to-learn* and *how-to-learn* has helped MCNFIS achieve better performance.

Next, we statistically compare the performance of the classifiers used in this study on all the data sets. Since the performance of CNFIS and MCNFIS are compared with multiple algorithms on multiple data sets, we have chosen the well-known Friedman test and two-tailed Bonferroni Dunn test for the statistical

Table 6: Performance Comparison for Unbalanced Multi-category Classification Problems

Data Set	Classifier Domain	Classifier	No. of Rules	Training		Testing	
				η_o	η_a	η_o	η_a
Vehicle Classification	Real Valued	SVM	340	79.48	79.82	70.62	68.51
		ELM	150	85.14	85.09	77.01	77.59
	Complex	IC-MLP	80	87.74	87.74	76.78	76.76
		BB-ELM	80	86.08	86.32	78.19	78.52
		FC-RBF	70	88.67	88.88	77.01	77.46
		CNFIS	70	79.49	79.89	73.73	73.79
	Valued	MCNFIS	80	81.37	81.51	74.11	74.27
Glass Identification	Real Valued	SVM	183	86.24	93.23	70.47	75.61
		ELM	80	92.66	96.34	81.31	87.43
	Complex	IC-MLP	70	87.80	87.80	82.86	80.55
		BB-ELM	80	88.99	94.47	86.69	80.89
		FC-RBF	70	88.67	88.88	83.76	80.95
		CNFIS	70	82.44	82.44	79.05	81.31
	Valued	MCNFIS	70	82.44	82.44	80	81.63

Table 7: Performance Comparison for Binary Classification Problems

Classifier	Liver		ION		BC		PIMA	
	Rules	η_o	Rules	η_o	Rules	η_o	Rules	η_o
SVM	158	68.24	30	90.18	190	94.20	209	76.43
ELM	132	71.79	25	88.78	65	96.28	218	76.54
IC-MLP	25	71.03	15	73.01	10	96.28	25	77.17
BB-ELM	20	75.17	15	79.28	15	97.13	25	77.17
FC-RBF	20	74.6	10	89.48	10	97.12	20	78.53
CNFIS	20	76.2	8	92.03	8	97.12	25	80
MCNFIS	20	76.2	11	94.2	10	97.39	25	80

Table 8: Ranking of each algorithm for every classification data set

Data Set	SVM	ELM	IC-MLP	BB-ELM	FC-RBF	CNFIS	MCNFIS
Iris	6	5	3.5	7	1.5	3.5	1.5
IS	6	7	5	4	3	1.5	1.5
VC	7	2	4	1	3	6	5
GI	7	1	6	5	4	3	2
Liver	7	5	6	3	4	2	1
Ion	3	5	7	6	4	2	1
BC	7	5.5	5.5	3	3	3	1
IS	7	6	5	4	3	1.5	1.5

analysis [56].

Friedman test is a non-parametric equivalent of the repeated-measures ANOVA. It compares the average ranks of algorithms and determines whether the mean rank of individual experimental condition differs significantly from the aggregate mean rank across all conditions. If the Friedman statistics is found to be greater than the critical value for χ^2 distribution, then the null hypothesis (which states that all algorithms are equivalent) can be rejected. If the Friedman test rejects the null hypothesis, a post-hoc Bonferroni-Dunn test is conducted to check whether the difference between two classifiers is greater than the critical difference [56].

In our study, we have employed six different algorithms and 8 different data sets. The relative ranking of each algorithm for every data set employed is given in Table 8. Based on this ranking, the F-score is obtained using modified Friedman statistics as 9.71, which is greater than the critical value of 2.25 at 95% significance. Hence, the null hypothesis can be rejected. Next we conduct a pair-wise comparison using Bonferroni-Dunn test to highlight the performance significance of CNFIS and MCNFIS classifiers. Two algorithms are said to be different if difference between average rank of the two algorithms are greater than critical difference calculated (2.48). The difference in average ranks be-

tween MCNFIS and CNFIS, FC-RBF, BB-ELM, IC-MLP, ELM and SVM are 0.87, 1.31, 2.56, 3.32, 2.68 and 4.13, respectively. It can be observed that difference in average ranks of MCNFIS in comparison to BB-ELM, IC-MLP, ELM and SVM are greater than critical difference. However, the difference between MCNFIS and FC-RBF is slightly lesser than the critical difference. This could be due to the use of fully complex-valued activation function in FC-RBF. Similarly, the difference in ranks between MCNFIS and ELM below critical difference, for real-valued classification problems. However, MCNFIS employs significantly lesser number of rules compared to ELM and other real-valued networks.

6. Conclusion

In this paper, we presented a TSK type-0 complex-valued neuro-fuzzy inference system. The learning algorithm is based on fully complex-valued gradient descent algorithm derived using Wirtinger calculus. We also propose a meta-cognitive learning scheme for CNFIS referred to as, Meta-Cognitive Complex-Valued Neuro-Fuzzy Inference System (MCNFIS). The use of meta-cognitive principle helps the network select proper samples for learning by deciding on *what-to-learn*, *when-to-learn* and *how-to-learn* based on knowledge contained the network and current sample. In each epoch, the proposed learning scheme proceeds by deleting samples with lower error (*what-to-learn*), learning samples with higher error (*how-to-learn*) and reserving the remaining samples until the next epoch (*when-to-learn*). Performance evaluation on regression and real-valued classification problems indicate improved performance of the network when compared to other real-valued and complex-valued networks.

In future we would investigate the performance of different radial basis membership functions that would uniquely map complex-valued inputs to complex-valued feature space, thereby preserving the phase approximation abilities of the network.

References

[1] N. R. Prabha, N. S. Marimuthu, C. K. Babulal, Adaptive neuro-fuzzy inference system based total demand distortion factor for power quality evaluation, Neurocomputing 73 (1 – 3) (2009) 315 – 323.

[2] J. S. Lim, D. Wang, Y.-S. Kim, S. Gupta, A neuro-fuzzy approach for diagnosis of antibody deficiency syndrome, Neurocomputing 69 (7 – 9) (2006) 969 – 974.

[3] J. Bregains, F. Ares, Analysis, synthesis and diagnostics of antenna arrays through complex-valued neural networks, Microwave and Optical Technology Letters 48 (8) (2006) 1512 – 1515.

[4] M. Li, G. Huang, P. Saratchandran, N. Sundararajan, Complex valued growing and pruning RBF neural networks for communication channel equalization, IEEE Proc. Vision, Image and Signal Processing 153 (4) (2006) 411 – 418.

[5] I. Aizenberg, D. Paliy, J. Zurada, J. Astola, Blur identification by multilayer neural network based on multivalued neurons, IEEE Trans. on Neural Networks 19 (5) (2008) 883 – 898.

[6] M. Muezzinoglu, C. Guzelis, J. Zuruda, A new design method for the complex valued multistate hopfield associative memory, IEEE Trans. on Neural Networks 14 (4) (2003) 891 – 899.

[7] D. Mandic, S. Javidi, S. Goh, A. Kuh, K. Aihara, Complex valued prediction of wind profile using augmented complex statistics, renewable Energy 34 (1) (2009) 196 – 201.

[8] A. Ghani, M. Amin, K. Murase, Real time hand gesture recognition using complex valued neural network (CVNN), in: Neural Information Processing, Lecture Notes in Computer Science, 2011, pp. 541 – 549.

[9] T. Nitta, The computational power of complex-valued neuron, Artificial Neural Networks and Neural Information Processing ICANN/ICONIP. Lecture Notes in Computer Science 2714 (2003) 993–1000.

[10] T. Nitta, An extension of back-propagation algorithm to complex numbers, Neural Networks 48 (1) (1997) 180 – 192.

[11] T. Kim, T. Adali, Approximation by fully complex multilayer perceptron, Neural Computation 15 (7) (2003) 1641 – 1666.

[12] R. Savitha, S. Suresh, N. Sundararajan, A fully complex-valued radial basis function network and its learning algorithm, International Journal of Neural Systems 19 (4) (2009) 253–267.

[13] S. Chen, S. McLaughlin, N. Mulgrew, Complex valued radial basis function network, part I: Network architecture and learning algorithms, EURASIP Singal Processing Hournal 35 (1) (1994) 19 – 31.

[14] W. Wirtinger, Zur formalen theorie der funktionen von mehr komplexen vernderlichen, Annals of Mathematics 97 (1927) 357–375.

[15] H. Li, T. Adali, Complex-valued adaptive signal processing using nonlinear functions, EURASIP Journal on Advances in Signal Processing 2008:765615 (2008) 1 – 9.

[16] R. Savitha, S. Suresh, N. Sundararajan, Metacognitive learning in a fully complex-valued radial basis function neural network, Neural Computation 24 (5) (2012) 1297–1328.

[17] R. Savitha, S. Suresh, N. Sundararajan, Complex-valued function approximation using a fully complex-valued RBF (FC-RBF) learning algorithm, in: Proceedings of Intl. Joint Conf. on Neural Networks, 2009, pp. 2819 – 2825.

[18] M. F. Amin, R. Savitha, M. I. Amin, K. Murase, Orthogonal least squares based comlpex-valued functional link network, Neural Networks 32 (Special Issue) (2012) 257 – 266.

[19] K. Subramanian, S. Suresh, A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system, Applied Soft Computing 12 (11) (2012) 3603 – 3614.

[20] H.-J. Rong, N. Sundararajan, G.-B. Huang, P. Saratchandran, Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction, Fuzzy Sets and Syst. 157 (9) (2006) 1260 – 1275.

[21] D. P. Joysula, H. Vadali, J. Donahue, F. Hughes, Modeling meta-cognition for learning in artificial systems, in: World Congress on Nature and Biologically Inspired Computing, 2009, pp. 1419 – 1424.

[22] J. Flavell, Meta-cognition and cognitive monitoring: A new area of cognitive-developmental inquiry, American Psychologist 34 (10) (1979) 906 – 911.

[23] H. Wellman, Metamemory revisited, in: M.T.H Chi ed. Contributions to Human Development, Vol. 9: Trends in memory development research, 1983, pp. 31 – 51.

[24] L. Perlovsky, Neural mechanisms of the mind, aristotle, zadeh and fmri, IEEE Trans. Neual Networks 21 (5) (2010) 718 – 733.

[25] T.-O. Nelson, L. Narens, Metamemory: A theoretical framework and new findings, Psychology of Learning and Motication 26 (C) (1990) 125 – 173.

[26] J. Flavell, Cognitive Development, Prentice Hall: Great Britain, 1988.

[27] K. Subramanian, S. Sundaram, N. Sundararajan, A meta-cognitive neuro-fuzzy inference system (McFIS) for sequential classification problems, IEEE Trans. Fuzzy Syst.DOI: 10.1109/TFUZZ.2013.2242894.

[28] S. Suresh, K. Dong, H. Kim, A sequential learning algorithm for self adaptive resource allocation network classifier, Neurocomputing 73 (16 – 18) (2010) 3012 – 3019.

[29] S. Suresh, R. Savitha, N. Sundararajan, A sequential learning algorithm for complex valued self regulating resource allocation network- CSRAN, IEEE Trans. Neural Networks 22 (7) (2011) 1061 – 1072.

[30] G. Sateesh Babu, S. Suresh, Metacognitive neural network for classification problems in a sequential learning framework, Neurocomputing 81 (1) (2011) 86 – 96.

[31] R. Savitha, S. Suresh, N. Sundararajan, A meta-cognitive learning algorithm for a fully complex-valued relaxation network, Neural Networks 32 (Special Issue) (2012) 209 – 218.

[32] S. Suresh, K. Subramanian, A sequential learning algorithm for meta-cognitive neuro-fuzzy inference system for classification problems, in: Proc. of Intl. Joint Conf. Neural Networks, 2011, pp. 2507 – 2512.

[33] C. Blake, C. Merz, UCI repository of machine learning databases, department of Information and Computer Sciences, University of California, Irvine (1998).
URL `http://archive.ics.uci.edu/ml/`

[34] D. Jianping, N. Sundararajan, P. Saratchandran, Complex valued minimal resource allocation network for nonlinear signal processing, Intl. Journal of Neural Systems 10 (2) (2000) 95 – 106.

[35] N. Benvenuto, F. Piazza, On the complex backpropagation algorithm, IEEE Trans. Signal Processing 40 (4) (1992) 967 – 969.

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

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

[38] R. Savitha, S. Suresh, N. Sundararajan, A new learning algorithm with logarithmic performance index for complex-valued neural networks, Neurocomputing 72 (16 – 18) (2009) 3771 – 3781.

[39] R. Savitha, S. Suresh, N. Sundararajan, A projection based fast learning fully complex-valued relaxation neural network, in: IEEE Trans. Neural Networks, Vol. 24, 2013, pp. 529–541.

[40] M. Li, G. Huang, P. Saratchandran, N. Sundararajan, Fully complex extreme learning machine, Neurocomputing 68 (2005) 306 – 314.

[41] T. Nitta, Orthogonality of decision boundaries of complex-valued neural networks, Neural Computation 16 (1) (2004) 73 – 97.

[42] R. Savitha, S. Suresh, N. Sundararajan, H. Kim, A fully complex valued radial basis function classifier for real valued classification problems, Neurocomputing 78 (1) (2012) 104 – 110.

[43] I. Aizenberg, C. Moraga, Multilayer feedforward neural network based on multi-valued neurons MLMVN and a backpropagation learning algorithm, Soft Computing 11 (2) (2007) 169 – 193.

[44] M. F. Amin, K. Murase, Single-layer complex-valued neural network for real-valued classification problems, Neurocomuting 72 (4 – 6) (2009) 945 – 955.

[45] R. Savitha, S. Suresh, N. Sundararajan, H. J. Kim, Fast learning fully complex-valued classifiers for real-valued classification problems, Lecture Notes in Computer Science: Advances in Neural Networks 6675 (2011) 602 – 609.

[46] R. Savitha, S. Suresh, N. Sundararajan, Fast learning circular complex-valued extreme learning machine (CC-ELM) for real-valued classification problems, Information Sciences 187 (2012) 277 – 290.

[47] T. Takagi, M. Sugeno, Fuzzy identification of systems and its application to modeling and control, IEEE Trans. Syst., man and Cybern. 15 (1) (1985) 116 – 132.

[48] A. Suksmono, A. Hirose, Intelligent beam-forming by using a complex-valued neural network, J. Intelligent and Fuzzy Syst. 15 (3 – 4) (2004) 139 – 147.

[49] R. Savitha, S. Suresh, N. Sundararajan, A new learning algorithm with logarithmic performance index for complex-valued neural networks, Neurocomputing 72 (16 – 18) (2009) 3771 – 3781.

[50] R. A. Monzingo, T. W. Miller, Introduction to adaptive arrays, Raleigh, NC: SciTech Publishing, 2003.

[51] J. Patra, R. Ral, R. Baliarsingh, G. Panda, Nonlinear channel equalization for qam signal constellation using artificial neural networks, IEEE Trans. Syst., Man and Cybern., Part B: Cybern. 29 (2) (1999) 262 – 271.

[52] S. Suresh, N. Sundararajan, P. Saratchandran, Risk-sensitive loss functions for sparse multi-category classification problems, Information Sciences 178 (2008) 2621 – 2638.

[53] T. Zhang, Statistical behavior and consistency of classification methods based on convex risk minimization, Annals of Statistics 32 (1) (2003) 56–85.

[54] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 272 – 297.

[55] S. Suresh, R. Venkatesh Babu, H. J. Kim, No-reference image quality assessment using modified extreme learning machine classifier, Applied Soft Computing 9 (2) (2009) 541 – 552.

[56] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (1) (2006) 1 – 30.