Sequential Learning Algorithm for Complex-Valued Self-Regulating Resource Allocation Network-CSRAN

Sundaram Suresh, *Senior Member, IEEE*, Ramasamy Savitha, *Student Member, IEEE*, and Narasimhan Sundararajan, *Fellow, IEEE*

Abstract*— This paper presents a sequential learning algorithm for a complex-valued resource allocation network with a self-regulating scheme, referred to as complex-valued self-regulating resource allocation network (CSRAN). The self-regulating scheme in CSRAN decides *what to learn*, *when to learn*, and *how to learn* based on the information present in the training samples. CSRAN is a complex-valued radial basis function network with a sech activation function in the hidden layer. The network parameters are updated using a complex-valued extended Kalman filter algorithm. CSRAN starts with no hidden neuron and builds up an appropriate number of hidden neurons, resulting in a compact structure. Performance of the CSRAN is evaluated using a synthetic complex-valued function approximation problem, two real-world applications consisting of a complex quadrature amplitude modulation channel equalization, and an adaptive beam-forming problem. Since complex-valued neural networks are good decision makers, the decision-making ability of the CSRAN is compared with other complex-valued classifiers and the best performing real-valued classifier using two benchmark unbalanced classification problems from UCI machine learning repository. The approximation and classification results show that the CSRAN outperforms other existing complex-valued learning algorithms available in the literature.

***Index Terms*— Adaptive beam-forming, classification, complex-valued extended Kalman filter, fully complex-valued resource allocation network, quadrature amplitude modulation channel equalization, self-regulating/sequential learning.**

I. Introduction

COMPLEX-VALUED signals naturally arise in many applications like communication [1], signal processing [2]–[5], image processing [6]–[9], and radar applications [10]. Representing these signals and their nonlinear transformations in the complex domain is a natural way to preserve their physical characteristics. But, the analysis of complex-valued signals and their transformations in an efficient way pose many new challenges. Since the complex-valued neural networks (CVNNs) provide a convenient mechanism for the above, they have created renewed interest in researchers for developing new complex-valued neural architectures and their learning algorithms.

In earlier stages, split CVNNs [9], [11] that split the m-dimensional complex-valued signals into two real-valued signals comprising of m real and m imaginary components and treating them as two sets of inputs ($2m$) to a real-valued neural network were used. It has been observed that the process of splitting the complex-valued signals into real and imaginary components and handling them separately introduces phase distortions in complex-valued approximations due to the loss of correlation between the two components [12]–[14]. As accurate phase estimation is very important in many applications [9], [15], a fully CVNN becomes essential. Recently, it was shown that a CVNN has better computational power to handle complex-valued signals than real-valued neural networks [16]. Moreover, the presence of orthogonal decision boundaries in CVNNs helps them in solving real-valued classification problems more efficiently in the complex domain [17], [18]. Hence, research in CVNNs is gaining much attention.

A critical issue in a fully CVNN is the selection of an appropriate complex-valued activation function, which has to be *entire and bounded*. But, from the theory of complex variables, namely Liouville's theorem, an *entire and bounded function* is a constant function in the complex domain [14]. A constant function does not satisfy the desirable characteristics of a complex-valued activation function [19]. In order to overcome this problem, Kim and Adali [14], [20] have suggested the use of elementary transcendental functions (ETFs) that are analytic and bounded *almost everywhere* as activation functions for a fully complex-valued multilayer perceptron (FC-MLP). The FC-MLP parameters are updated using a complex-valued error backpropagation algorithm. Further, it was shown that the *asinh* and *atan* activation functions show better convergence characteristics than the other ETFs [21]. A complex-valued extreme learning machine (C-ELM) has been proposed in the ELM framework with any of the ETFs as an activation function [22]. Recently, in [23], an incremental learning version of the C-ELM called I-ELM has been developed to select the appropriate number of hidden neurons.

In the context of a radial basis function (RBF) network, a complex-valued RBF (CRBF) network and its batch learning algorithm were presented in [24] and [25].

Manuscript received February 21, 2010; revised ~~XXXX XX, XXXX~~; accepted ~~XXXX XX, XXXX~~. The work of S. Suresh and R. Savitha was supported in part by the Ministry of Education, Singapore, through a TIER I Grant.

EQ:1

The authors are with the School of Computer Engineering, Nanyang Technological University, 639798, Singapore (e-mail: ssundaram@ntu.edu.sg; SAVI0001@ntu.edu.sg; ensundara@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2011.2144618

Similarly, sequential versions of the learning algorithm in the CRBF framework, called the complex-valued minimal resource allocation network (CMRAN) [5] and the complex-valued growing and pruning RBF network (CGAP-RBF) [1], have been developed. While the CMRAN and CGAP algorithms determine the number of hidden neurons automatically, the CRBF algorithm requires that the number of neurons be fixed *a priori*. One of the main drawbacks in the aforementioned algorithms in the RBF framework is the use of a Gaussian activation function, which maps the signals in the complex domain to real domain ($\mathbb{C} \rightarrow \mathbb{R}$). This results in loss of cross-correlation information between the real and imaginary components of the complex-valued signals and a poor phase approximation [26]. Recently, a fully complex-valued RBF (FC-RBF) using the fully complex-valued "sech" activation function was presented in [26]. The activation function maps $\mathbb{C}^m \rightarrow \mathbb{C}$ and the gradients used are true complex-valued gradients. Hence, it has been shown in [26] that the FC-RBF approximates both the magnitude and phase more accurately compared to other CRBF networks.

The approximation ability of the CVNN is influenced by the following two major factors.

1) In general, the learning algorithms are based on the assumption that the training data is uniformly distributed in the input space, which is not always true, especially in practical problems. Learning similar samples repeatedly may result in overtraining, thereby affecting the generalization capability with an increased computational effort. Hence, there is a need to develop a learning algorithm that is capable of deciding *when to learn*, *what to learn*, and *how to learn* the samples in the training dataset.
2) In all the above complex-valued networks, the mean square magnitude error has been used as the minimization criterion for parameter updates in the learning algorithms. In the context of complex-valued signals, though the minimization of magnitude error implicitly minimizes the phase error to some extent, it will be advantageous to use the phase error explicitly in the minimization criterion for a better phase approximation [21].

When the complete training dataset is not available for batch learning and the data arrives sequentially, the above-mentioned issues complicate the problem further. To address these issues, in this paper, we present a sequential learning algorithm for CVNN with a self-regulating mechanism. The self-regulating scheme controls the learning process and uses the magnitude/phase errors explicitly for better complex-valued function approximation.

In this paper, we propose a "complex-valued self-regulating resource allocation network (CSRAN)" and its sequential learning algorithm. The CSRAN algorithm handles the training samples one by one and discards them after learning. The basic building block of the CSRAN is a FC-RBF network with a "sech" activation function. The network parameters are updated using a fully complex-valued extended Kalman filter (C-EKF) scheme presented in [27]. CSRAN starts with no hidden neuron and adds/prunes neurons based on criteria involving both the magnitude and phase errors. CSRAN employs a self-regulating scheme to control the learning process. When the training samples arrive one by one, the self-regulating scheme performs one of the following three actions: 1) *sample deletion* without learning; 2) *sample learning* (either add/prune the hidden neuron or update the network parameters); and 3) *sample reserve* (shift the sample for future use). Thus, CSRAN algorithm efficiently addresses three key issues, namely, *when to learn*, *what to learn*, and *how to learn*.

The performance of the CSRAN algorithm is evaluated using a synthetic complex-valued function approximation problem (SCFAP) and two real-world problems. First, we highlight the key features of the CSRAN algorithm in detail using a SCFAP. Next, we present the performance comparison based on a complex-valued quadrature amplitude modulation (QAM) nonminimum nonlinear phase equalization problem [28] and an adaptive beam-forming problem [29]. In these problems, the performance of the CSRAN is compared with existing CVNNs. The results indicate that the CSRAN algorithm provides better magnitude and (especially) phase approximation and generalization with fewer samples for training and a compact network structure.

As mentioned earlier, the better computational ability and the orthogonal decision boundaries of the CVNNs endow them with an exceptional ability to perform classification tasks. Recently, blur identification for image restoration has been solved as a classification problem using CVNNs [6]. Hence, to evaluate the classification ability of the CSRAN, we study the performance of the CSRAN in solving two real-valued multicategory benchmark classification problems with unbalanced datasets. To solve the real-valued classification problem, the complex-valued input features are obtained by phase encoding the real-valued input features as explained in [18]. Here, we compare the performance of the CSRAN with existing complex-valued and best performing real-valued neural classifiers.

This paper is organized as follows. Section II presents the architecture of the CSRAN algorithm along with its self-regulating learning scheme. Guidelines on the selection of the self-regulating thresholds are given and the functioning of the CSRAN is discussed in detail using a SCFAP. In Section III, the performance of the CSRAN is evaluated on the SCFAP presented in Section II, a complex QAM channel equalization problem, an adaptive beam-forming problem, and two real-valued benchmark classification problems in the complex domain. Section IV summarizes the work and presents the conclusions from this paper.

II. CSRAN

In this section, we first describe the network architecture and then the principles behind the self-regulating learning scheme.

A. CSRAN Architecture

CSRAN is a single-hidden-layer network. The basic building block of the CSRAN is an FC-RBF network as shown in Fig. 1. CSRAN is a sequential learning algorithm and starts with no hidden neuron and builds the necessary number of hidden neurons based on the information contained in the current sample. CSRAN has a self-regulating scheme which

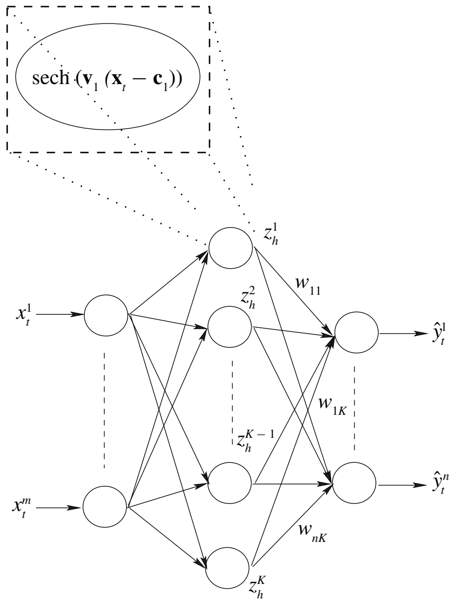

Fig. 1. Architecture of CSRAN.

controls the learning process by proper selection of the training samples.

Let the sequence of observation data drawn randomly be $\{(\mathbf{x}_1,\mathbf{y}_1), (\mathbf{x}_2,\mathbf{y}_2),\ldots, (\mathbf{x}_t,\mathbf{y}_t),\ldots\}$, where $\mathbf{x}_t = [x_t^1, \ldots, x_t^m]^T \in \mathbb{C}^m$ is an m-dimensional input vector and $\mathbf{y}_t = [y_t^1, \ldots, y_t^n]^T \in \mathbb{C}^n$ is its n-dimensional desired output.

Without loss of generality, we assume that after sequentially learning $t-1$ observations, the CSRAN has built a network with K hidden neurons. For a given input, the predicted output ($\widehat{\mathbf{y}}_t = [\widehat{y}_t^1, \ldots, \widehat{y}_t^n]^T \in \mathbb{C}^n$) of the CSRAN with K hidden neurons is given by

$$\widehat{y}_t^j = \sum_{i=1}^{K} w_{ji} z_h^i; \qquad j = 1, \ldots, n$$
$$z_h^i = \text{sech}\left[\mathbf{v}_i^T(\mathbf{x}_t - \mathbf{c}_i)\right]; \qquad i = 1, 2, \ldots K \tag{1}$$

where $w_{ji} \in \mathbb{C}$ is the complex-valued weight connecting the ith hidden neuron to the jth output neuron, $\mathbf{v}_i = [v_{i1}, \ldots, v_{im}]^T \in \mathbb{C}^m$ is the complex-valued scaling factor of the ith hidden neuron, $\mathbf{c}_i = [c_{i1}, \ldots, c_{im}]^T \in \mathbb{C}^m$ is the center of the ith hidden neuron, the superscript T denotes the transpose operator, and $\text{sech}(z) = 2/(e^z + e^{-z})$.

The magnitude response of the "sech($\mathbf{v}_i^T(\mathbf{x}_t - \mathbf{c}_i)$)" activation function is similar to that of the Gaussian activation function. Also, the phase response of this function is close to zero when both the real and imaginary parts of the complex-valued signal ($\mathbf{v}_i^T(\mathbf{x}_t - \mathbf{c}_i)$) are zero. Thus the activation function response is the highest when the inputs are located closer to the center, making the function a good choice for an RBF.

The residual error ($\mathbf{e}_t$) of the CSRAN for the current observation ($\mathbf{x}_t$, $\mathbf{y}_t$) is defined as

$$\mathbf{e}_t = \mathbf{y}_t - \widehat{\mathbf{y}}_t. \tag{2}$$

Using the residual error, we estimate the instantaneous magnitude error (M_t^e) and the normalized absolute phase error (ϕ_t^e) as

$$M_t^e = \frac{1}{n}\sqrt{\mathbf{e}_t^H.\mathbf{e}_t} \tag{3}$$

$$\phi_t^e = \frac{1}{n\pi}\sum_{l=1}^{n}\left|\arg\left(y_t^l\right) - \arg\left(\widehat{y}_t^{\,l}\right)\right| \tag{4}$$

where the superscript H denotes the complex Hermitian operator and arg(.) is a function that returns the phase of a complex-valued number in $[-\pi,\ \pi]$, given by

$$\arg(z) = \text{atan}\left(\frac{imag(z)}{real(z)}\right). \tag{5}$$

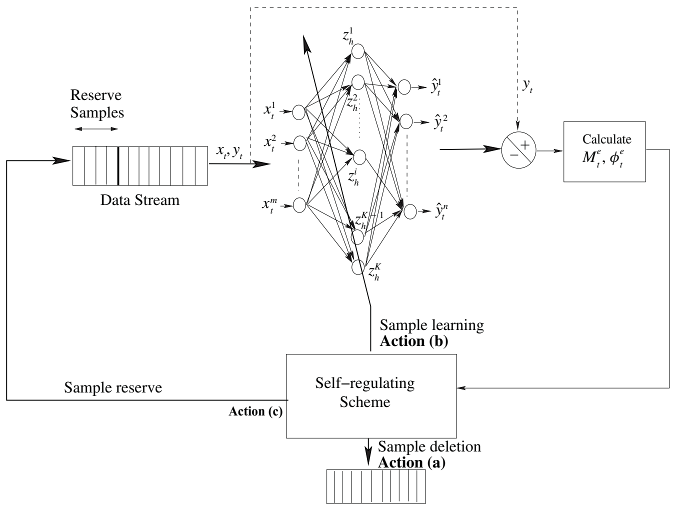

Fig. 2. Schematic diagram of CSRAN and its self-regulating learning scheme.

B. Sequential Self-Regulating Learning Scheme of CSRAN

In a sequential learning framework, the observation data/samples arrive one by one and one at a time. Most of the sequential learning algorithms (both in real-valued/complex-valued) learn all the training samples in the sequence as they are presented, whereas the proposed CSRAN algorithm regulates the sequential learning process by selecting appropriate samples for learning. The schematic diagram of the self-regulating scheme is shown in Fig. 2. The basic working principles of the self-regulating scheme are explained in the following paragraph.

Based on the instantaneous magnitude (M_t^e) and absolute phase error (ϕ_t^e) of each sample in the training sequence, the self-regulating scheme performs one of the following three actions.

Action (a) *Sample Deletion:* Samples are deleted without being used in the learning process.

Action (b) *Sample Learning:* Learning includes growing/pruning the hidden neuron or updating the network parameters.

Action (c) *Sample Reserve:* The samples are pushed to the rear end of the training sequence and can be used at a later stage.

The concept behind these actions representing each block of Fig. 2 are described in detail in the following text.

Action (a) *Sample Deletion:* When both the instantaneous magnitude error (M_t^e) [given in (3)] and the absolute phase error (ϕ_t^e) [given in (4)] of a sample are less than their fixed delete thresholds, the self-regulating scheme deletes the sample without using it in the learning process. The sample deletion criterion is given by

$$M_t^e \leq E_d^M \quad \text{and} \quad \phi_t^e \leq E_d^\phi \tag{6}$$

where E_d^M is the sample delete magnitude threshold and E_d^ϕ is the sample delete phase threshold. The "sample deletion" criterion removes similar samples from the training sequence. Hence, it avoids overtraining and reduces the computational effort.

Action (b) *Sample Learning:* In a self-regulating scheme, the learning process involves the allocation of new hidden neurons ("growing"), updating of network parameters ("update") and removing redundant neurons ("pruning").

Neuron Growing Criterion: As the training samples arrive sequentially, some of the selected samples will be used to "add" new hidden neurons based on the following criterion:

$$M_t^e \geq E_a^M \quad \text{or} \quad \phi_t^e \geq E_a^\phi \tag{7}$$

where E_a^M is the neuron growing magnitude threshold and E_a^ϕ is the neuron growing phase threshold.

It should be pointed out here that the neuron growing thresholds (E_a^M, E_a^ϕ) are not kept constant. They are adaptively varied based on the current residual error as given below

$$\begin{aligned} &\text{if } M_t^e \geq E_a^M \text{ then } E_a^M := \delta E_a^M - (1-\delta) M_t^e \\ &\text{if } \phi_t^e \geq E_a^\phi \text{ then } E_a^\phi := \delta E_a^\phi - (1-\delta) \phi_t^e \end{aligned} \tag{8}$$

where the slope parameter (δ) controls the rate at which the neuron growing thresholds (E_a^M, E_a^ϕ) are regulated and hence influence the neuron growth. In general, the slope parameter is initialized close to 1.

When a new hidden neuron ($K+1$) is added to the network, the parameters associated with it are initialized as

$$\mathbf{w}_{K+1} = \mathbf{e}_t; \quad \mathbf{c}_{K+1} = \mathbf{x}_t; \quad \mathbf{v}_{K+1} = \kappa(\mathbf{x}_t - \mathbf{c}_{nr}) \tag{9}$$

where nr is the nearest neuron, defined as that neuron with the smallest Euclidean distance from the current sample. The scaling factor κ determines the overlap between the samples in the input space. As κ increases, the overlap between the responses of the hidden neurons also increases.

Network Parameter Update Criterion: If a new observation $(\mathbf{x}_t, \mathbf{y}_t)$ arrives and the parameter update criterion is satisfied, then the parameters of the network are updated using a C-EKF [27]. The parameter update criterion is given by

$$M_t^e \geq E_l^M \quad \text{or} \quad \phi_t^e \geq E_l^\phi \tag{10}$$

where E_l^M is the parameter update magnitude threshold and E_l^ϕ is the parameter update phase threshold. The parameter update thresholds (E_l^M, E_l^ϕ) are also adapted based on the residual error of the current sample as

$$\begin{aligned} &\text{if } M_t^e \geq E_l^M \text{ then } E_l^M := \delta E_l^M - (1-\delta) M_t^e \\ &\text{if } \phi_t^e \geq E_l^\phi \text{ then } E_l^\phi := \delta E_l^\phi - (1-\delta) \phi_t^e \end{aligned} \tag{11}$$

where δ is a slope that controls the rate of self-adaptation of the parameter update magnitude and phase thresholds. Usually, δ is set close to 1.

The main advantage of the self-regulating thresholds is that it helps in selecting appropriate samples to add neuron or update the network parameters, i.e., the CSRAN algorithm uses sample with higher error to either add a new hidden neuron or update the network parameters first and the remaining samples for fine-tuning the network parameters.

The network parameters ($\boldsymbol{\alpha}_t = [\mathbf{c}_1, \ldots, \mathbf{c}_K, \mathbf{v}_1, \ldots, \mathbf{v}_K, \mathbf{w}_1, \ldots, \mathbf{w}_k]^T$) are updated for the current sample (t) as

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + G_t \ \mathbf{e}_t \tag{12}$$

where $\mathbf{e}_t$ is the residual error and G_t is complex-valued Kalman gain matrix given by

$$G_t = P_{t-1}\mathbf{a}_t \left[R + \mathbf{a}_t^H \ P_{t-1} \ \mathbf{a}_t \right]^{-1} \tag{13}$$

where $\mathbf{a}_t$ is the complex-valued gradient vector, $R = r_0 I_{n \times n}$ is the variance of the measurement noise, and P_t is the error covariance matrix.

The gradient vector ($\mathbf{a}_t$) (set of partial derivatives of output with respect to $\boldsymbol{\alpha}_t$) is defined as

$$\mathbf{a}_t = \begin{bmatrix} \frac{\partial \widehat{y}_t^1}{\partial c_{11}} & \cdots & \frac{\partial \widehat{y}_t^1}{\partial c_{Km}} & \frac{\partial \widehat{y}_t^1}{\partial v_{11}} & \cdots & \frac{\partial \widehat{y}_t^1}{\partial v_{Km}} & \frac{\partial \widehat{y}_t^1}{\partial w_{11}} & \cdots & \frac{\partial \widehat{y}_t^1}{\partial w_{nK}} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial \widehat{y}_t^n}{\partial c_{11}} & \cdots & \frac{\partial \widehat{y}_t^n}{\partial c_{Km}} & \frac{\partial \widehat{y}_t^n}{\partial v_{11}} & \cdots & \frac{\partial \widehat{y}_t^n}{\partial v_{Km}} & \frac{\partial \widehat{y}_t^n}{\partial w_{11}} & \cdots & \frac{\partial \widehat{y}_t^n}{\partial w_{nK}} \end{bmatrix}^T$$

where

$$\begin{aligned} \frac{\partial \widehat{y}_t^l}{\partial c_{ji}} &= \overline{w}_{lj} \ \overline{u}_h^j \ \overline{v}_{ji}; \ u_h^j = \ -z_h^j . \tanh\left(\mathbf{v}_j^T \left(\mathbf{x}_t - \mathbf{c}_j \right) \right) \\ \frac{\partial \widehat{y}_t^l}{\partial v_{ji}} &= \overline{w}_{lj} \ \overline{u}_h^j \ \overline{(\mathbf{x}_t^i - c_{ji})} \\ \frac{\partial \widehat{y}_t^l}{\partial w_{lj}} &= \overline{z}_h^j; \ i = 1, \ldots, m; \ j = 1, \ldots, K; \ l = 1, \ldots, n \end{aligned} \tag{14}$$

where $\overline{u}_h^j$ is the conjugate of the derivative of the jth hidden neuron output and $\overline{z}_h^j$ is the conjugate of the jth hidden neuron output. The error covariance matrix is updated as

$$P_t = \left[I - G_t \mathbf{a}_t^H \right] P_{t-1} + qI \tag{15}$$

where q is a process noise covariance usually set close to 0 and I is an identity matrix of dimension $K(2m+n) \times K(2m+n)$.

Neuron Pruning Criterion: Similar to CMRAN, the proposed CSRAN algorithm uses the contribution of the hidden neuron to delete the superfluous neuron. The contribution of the ith hidden neuron is defined as

$$r_i = \frac{z_h^i}{\max_j z_h^j}. \tag{16}$$

If $\|r_i\| < E_p$ and $\arg(r_i) < E_p$ for N_w consecutive samples, then the ith neuron is superfluous and is removed from the network. Here, E_p is the neuron pruning threshold. If the neuron pruning threshold (E_p) is set at a lower value, then pruning seldom occurs and all the added neurons will remain in the network irrespective of their contribution to the network output. On the other hand, higher value of E_p results in frequent pruning, resulting in oscillations and insufficient neurons to approximate the function.

When a neuron is added to the network, the error covariance matrix (P_t) is updated as

$$P_t = \begin{bmatrix} P_{t-1} & 0 \\ 0 & p_0 I_{(2m+n)\times(2m+n)} \end{bmatrix} \quad (17)$$

where p_0 is the estimated uncertainty of initial parameters. On the other hand, when a neuron (say, ith neuron) is removed from the network, the dimensionality of the error covariance matrix is reduced by removing the respective rows and columns of the P_t matrix, i.e., remove $(i-1)(2m+n)+1$ to $i(2m+n)$ rows and columns of the P_t matrix.

For initialization of the C-EKF parameters, p_0, q, and r_0, one should refer to [27].

Action (c) *Sample Reserve:* If the current observation $(\mathbf{x}_t, \mathbf{y}_t)$ does not satisfy the sample deletion criterion or the neuron growing criterion or the parameter update criterion, then the sample is pushed to the rear end of the data stream. Due to the self-adaptive nature of the thresholds, these reserve samples may also contain some useful information which will be used later in the learning process.

These three actions of self-regulating learning are repeated for all the samples in the training sequence.

C. CSRAN Algorithm

The CSRAN algorithm can be summarized as follows.

1) For each input $(\mathbf{x}_t)$ **compute** the network output $(\widehat{\mathbf{y}}_t)$

$$\widehat{y}_t^l = \sum_{i=1}^{K} w_{li} \ \text{sech}\left[\mathbf{v}_i^T(\mathbf{x}_t - \mathbf{c}_i)\right]; \ l = 1, \ldots n. \quad (18)$$

2) Using residual error $(\mathbf{e}_t = \mathbf{y}_t - \widehat{\mathbf{y}}_t)$, **estimate** M_t^e and ϕ_t^e using (3) and (4).
3) **Sample deletion**: If $M_t^e \leq E_d^M$ **AND** $\phi_t^e \leq E_d^\phi$, then delete the sample from the sequence without learning.
4) **Sample learning:** During learning, either a hidden neuron is added or pruned or the network parameters are updated.
    a) **Neuron growing criterion**: If $M_t^e \geq E_a^M$ **OR** $\phi_t^e \geq E_a^\phi$, then allocate new hidden neuron with $\mathbf{w}_{K+1} = \mathbf{e}_t$, $\mathbf{c}_{K+1} = \mathbf{x}_t$, and $\mathbf{v}_{K+1} = \kappa(\mathbf{x}_t - \mathbf{c}_{nr})$. Also, update the neuron growing thresholds using (8) and increase the dimensionality of P_t.
    b) **Parameter update criterion**: If $M_t^e \geq E_l^M$ **OR** $\phi_t^e \geq E_l^\phi$, then update the CSRAN network parameters using the C-EKF. Also, update the learning thresholds using (11).
    c) **Neuron pruning criterion**: Identify the noncontributing neuron for N_w consecutive samples and remove it. The dimensionality of P_t is reduced by removing the rows/columns corresponding to the pruned neuron.
5) **Sample reserve**: When a sample does not satisfy deletion, growing and update criteria, it is pushed to the rear end of data stream.

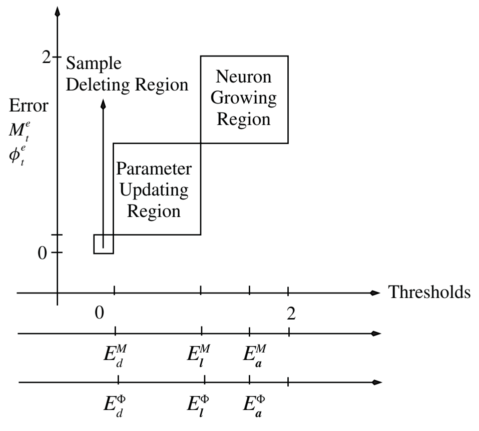

Fig. 3. Error regions for the various self-regulating thresholds of the CSRAN.

6) Continue steps 1 to 5 until there are no more samples in the training data stream.

D. Guidelines for Initialization of the Self-Regulating Thresholds in CSRAN

In this section, we explain the influence of the sample deletion, parameter update, and neuron growing self-regulating thresholds on the performance of the CSRAN and provide some guidelines for their initialization. The thresholds are based on either the instantaneous sample magnitude error (M_t^e) or the normalized absolute sample phase error (ϕ_t^e). CSRAN works by dividing the error region into three different subregions, namely, the sample deleting region, the parameter updating region, and the neuron growing region, as shown in Fig. 3.

When the sample magnitude and phase errors fall within the sample deleting region, the corresponding sample is deleted from the training sequence without being used in the learning process. The area of this region depends on the expected approximation accuracy and is controlled by the fixed sample deleting magnitude (E_d^M) and phase (E_d^ϕ) thresholds. Increasing E_d^M and E_d^ϕ above the desired accuracy results in deletion of many samples from the training sequence. But, the resultant network may not satisfy the desired accuracy. Hence, they are fixed at the expected accuracy level.

The sample with a higher residual error contains significant information and must be used first to add a new hidden neuron. The significant information is identified using the self-regulating neuron growing magnitude (E_a^M) and phase thresholds (E_a^ϕ). If these thresholds are chosen closer to the maximum residual error, then very few neurons will be added to the network. Such a network will not approximate the function properly. If a lower value is chosen, then the resultant network may contain many neurons with poor generalization ability. Hence, the range for the growing thresholds can be selected in the interval of 50%–95% of the maximum residual error. Since the growing thresholds are self-regulating, they will be decreasing based on the current sample error. Hence, the lower limit on the growing thresholds can be taken as the initial value of the parameter update thresholds.

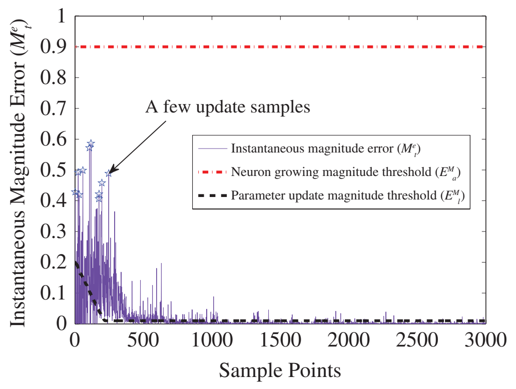

Fig. 4. Sample magnitude error and self-regulating magnitude growing and update thresholds history for SCFAP-I.

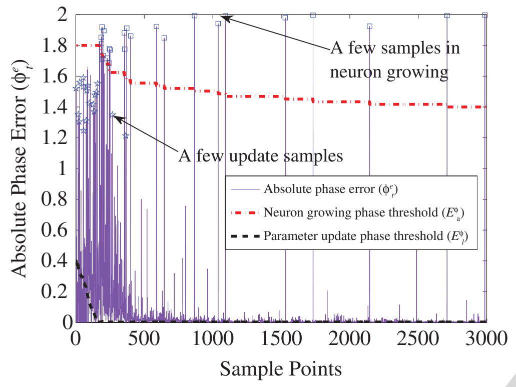

Fig. 5. Sample absolute phase error and self-regulating phase growing and update thresholds history for SCFAP-I.

Intuitively, one can see that the parameter update magnitude and phase thresholds (E_l^M and E_l^ϕ) can be chosen lower than the neuron growing thresholds (E_a^M and E_a^ϕ). The parameter update thresholds are initialized at a value that is at most equal to the lowest value of the growing thresholds after self-regulation, i.e., between 10% and 40% of the maximum error values. If these thresholds are chosen closer to 50% of maximum residual error, then very few samples will be used for adapting the network parameters and most of the samples will be pushed to the end of the training sequence. The resultant network will not approximate the function accurately. If a lower value is chosen, then all samples will be used in updating the network parameters without altering the training sequence. Similar to the neuron growing thresholds, the parameter update thresholds are also self-regulating in nature. The lowest values for update thresholds are same as the sample delete thresholds.

E. CSRAN Functioning Using an Illustrative Example

We use the SCFAP-I given in [21] to illustrate the key features of the CSRAN. The complex-valued function to be approximated is given by

$$f_1(\mathbf{x}) = \frac{1}{6}\left(x_1^2 + x_2^2\right) \tag{19}$$

where $x_1, x_2 \in \mathbb{C}$. A random sampling in the circle of radius 2.5 is used to obtain 3000 input/output data for the training set and 1000 input/output data for the testing set. The thresholds in

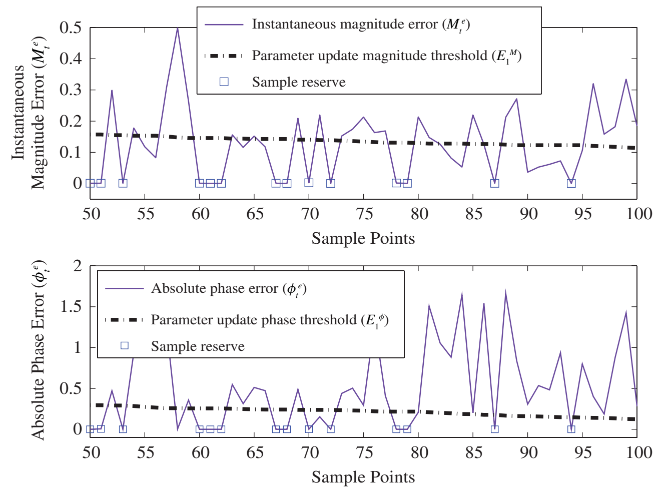

Fig. 6. Snapshot of magnitude error, absolute phase error, and self-regulating magnitude/phase update thresholds history between 50 and 100 samples.

the CSRAN are initialized as follows: magnitude thresholds, $\{E_d^M = 0.002,\ E_a^M = 0.9,\ E_l^M = 0.2\}$; phase thresholds, $\{E_d^\phi = 0.002, E_a^\phi = 1.8, E_l^\phi = 0.4\}$; and the other parameters, $\{\delta = 0.9995, E_p = 0.08, N_w = 50, \kappa = 0.83, p_0 = 1.05$, and $q = 0.005\}$.

Fig. 4 shows the sample magnitude error (M_t^e), the neuron growing magnitude threshold (E_a^M), and the parameter update magnitude threshold (E_l^M) for the 3000 uniformly sampled training data. From the figure, we can see that sample magnitude error is always less than 0.6 ($<E_a^M$) and hence no neurons are added based on the magnitude error (i.e., no regulation in the neuron growing magnitude threshold). The influence of M_t^e in the adaptation of the network parameters can be seen from the history of the E_l^M (shown as dashed line in Fig. 4). Few sample points used for the network parameter update are highlighted (using the symbol "$\star$" in Fig. 4). Since M_t^e is greater than E_l^M at these points, the samples are used for parameter updates and the E_l^M is adapted. After approximately 300 points, E_l^M reaches a minimum threshold value (E_d^M).

The absolute phase error (ϕ_t^e), the neuron growing phase (E_a^ϕ), and the parameter update phase (E_l^ϕ) thresholds are shown in Fig. 5. From the figure, we can observe that the ϕ_t^e at some sample instants (highlighted using symbol "□") is greater than E_a^ϕ and hence new hidden neurons are added at these sample instants. Also, the threshold (E_a^ϕ) is adapted based on the error as shown using the dash-dot line. From Figs. 4 and 5, we can observe that the magnitude and absolute phase errors for the first 179 sample instants are lower than the neuron growing thresholds (E_a^M and E_a^ϕ) and are either used in network parameter update or shifted to the rear end of the training data stream.

In order to clearly highlight the "sample learning" and the "sample reserve" mechanisms, we present a snapshot of the sample instants 50–100 in Fig. 6. From the snapshot, we can see that the parameter update magnitude and phase thresholds are self-regulating if and only if M_t^e is greater than E_l^M or ϕ_t^e is greater than E_l^ϕ. Otherwise, the current sample is skipped

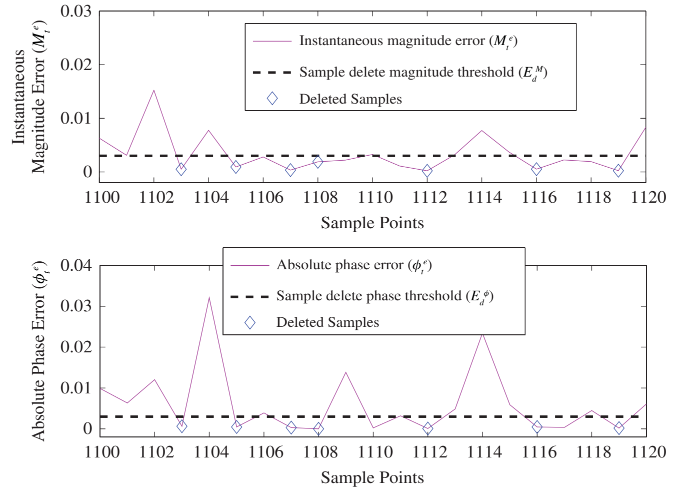

Fig. 7. Snapshot of magnitude error, phase error, and CSRAN magnitude/phase delete thresholds between 1100 and 1120 samples.

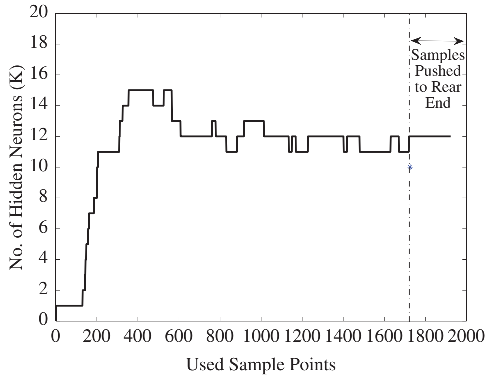

Fig. 8. Neuron history for SCFAP-I.

from learning and is pushed to the rear end of the training sequence. For example, at sample instants 57–59, M_t^e is greater than E_l^M and hence these samples are used to update the network parameters. Also, E_l^M is self-adapted using (11) at these sample instants. At the sample point 90, even though M_t^e is less than E_l^M, the sample is used for a network parameter update as $\phi_t^e > E_l^\phi$. At samples 78–79 (marked with "□" symbol in Fig. 6), both M_t^e and ϕ_t^e are lower than their respective thresholds and are therefore shifted to the rear end of the training data sequence.

Next, we present another snapshot between sample instants 1100–1120 in Fig. 7 to illustrate the "sample deletion" mechanism. From Fig. 7, we can see that M_t^e and ϕ_t^e are lower than their deletion thresholds at sample points marked with the "◊" symbol. These samples are deleted without being used for updating the network parameters. Even though M_t^e in sample points 1106 and 1118 are smaller than E_d^M, they are used in the parameter update, as ϕ_t^e at these sample instants are greater than E_d^ϕ.

The neuron history for samples that are used for learning is shown in Fig. 8. Out of the 3000 training samples, only 1726 samples are used for the learning process, 230 samples are deleted from the training set, and 1044 samples are shifted to the rear end of the training data stream. Out of these 1044 samples, only 197 samples are used for fine-tuning the network parameters and the remaining samples are deleted from the sequence. At the end of the training stage, only 1923 samples are used in the learning process and 1077 samples are deleted from the training sequence.

1) Effects of the Self-Regulating Scheme in CSRAN: To study the effect of the self-regulating behavior of the CSRAN, we define the following sequences.

a) *Training Sequence A:* 3000 training samples are randomly presented one by one.
b) *Training Sequence B:* 1726 training samples (without samples being pushed to the rear end of the data stream) are presented one by one in the same order/sequence in which they were used for learning in the CSRAN algorithm.
c) *Training Sequence C:* (1726 + 197) training samples are presented one by one. Here, the samples from "Sequence B" are followed by the 197 reserve samples.
d) *Training Sequence D:* (1726 + 197 + 1077) training samples are presented. Here, samples from "Sequence C" are followed by the 1077 deleted samples in the CSRAN.

These sequences are used in the CMRAN algorithm to show the benefits of self-regulating scheme of the CSRAN. In addition, we also stopped the CSRAN algorithm without using the pushed samples from the rear end of the data stream ("Sequence B") and evaluated the training/testing performances. The root mean square magnitude error (J_{Me}) and average absolute phase error (Φ_e) are used as the performance measures

$$J_{Me} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left[\frac{1}{n}\sum_{l=1}^{n}(e_t^{l}.\bar{e}_t^{l})\right]} \qquad (20)$$

$$\Phi_e = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{n}\left[\sum_{l=1}^{n}\mid \arg(y_t^l) - \arg(\widehat{y}_t^{\,l})\mid\right] \times \frac{180}{\pi} \qquad (21)$$

where $\bar{e}_t^{\,l}$ is the conjugate of the error e_t^l at the lth output neuron for the tth sample and N is the number of training/testing samples.

The thresholds for CMRAN are initialized as follows. The maximum distance threshold $\epsilon_{\max} = 0.9$, the minimum distance threshold $\epsilon_{\min} = 0.1$, the slope of the distance threshold $\gamma = 0.001$, the minimum magnitude error $e_{\min} = 0.1$, the root mean square error for a window of 50 samples $e'_{\min} = 0.1$, the overlap factor $\kappa = 0.3$, the real-valued EKF parameters, $p_0 = 1$, $R_n = 1$, $q = 0.005$, the pruning threshold $E_p = 0.01$, and the pruning window size $N_w = 100$. For a detailed explanation on CMRAN and the guidelines for selection of these parameters, refer [5].

Performance results of the CMRAN and the CSRAN using the above defined training sequences are presented in Table I. From the table, we can see that the performance of the CSRAN has improved considerably from "Sequence B" to "Sequence C," while the number of hidden neurons remains the same, i.e., the samples pushed to the rear end of the data

TABLE I
PERFORMANCE COMPARISON OF THE CSRAN AND CMRAN FOR DIFFERENT TRAINING SEQUENCE (SCFAP-I)

Perf.		Seq. for CSRAN		Seq. for CMRAN			
Meas.		B	C	A	B	C	D
Training	J_{Me}	0.02	0.003	0.07	0.06	0.05	0.04
	Φ_e (deg.)	1.86	0.66	12.6	15.8	12.8	16.6
Training	J_{Me}	0.004	0.003	0.07	0.06	0.05	0.04
	Φ_e (deg.)	1.52	0.31	15.8	14.3	11.4	20.1
K		12	12	27	21	22	17

TABLE II
PERFORMANCE COMPARISON FOR SCFAP-I

Algo.	Time (s)	K	Training		Testing	
			J_{Me}	Φ_e (deg.)	J_{Me}	Φ_e (deg.)
CSRAN	31	12	0.003	0.7	0.003	0.31
CMRAN	84	27	0.068	12.6	0.07	15.81
CRBF	5020	20	0.59	45.3	0.62	47.15
C-ELM	0.22	20	0.69	34.9	0.704	36.15
I-ELM	109	10^3	0.00002	0.3	0.001	0.41
FC-RBF	1341	15	0.0019	0.3	0.003	0.34
FC-MLP	1423	15	0.005	0.4	0.006	0.61

stream are mostly used to fine-tune the network parameters. From the table, we can also see that the magnitude/phase approximation performances of CMRAN for "Sequence B" and "Sequence C" are better than the "Sequence A." Comparing the performances of "Sequence C" and "Sequence D," we can see that the magnitude error decreases slightly and phase error increases significantly with the use of deleted samples. Similar to the CSRAN, CMRAN also shows an improvement in performance with the samples pushed to the rear end of the data stream. From this paper, we can say that the self-regulating scheme identifies the appropriate samples and uses them in a correct sequence. This helps in better approximation ability of the CSRAN network.

III. PERFORMANCE EVALUATION OF CSRAN

In this section, we present the performance analysis of the CSRAN for the synthetic complex-valued approximation problem and two real-world applications. Performance results of the CSRAN are compared with the well-known CMRAN, CRBF, FC-RBF, C-ELM (Gaussian activation function), I-ELM (Gaussian activation function), and FC-MLP (with $asinh$ activation function at hidden layer).

It has been shown in the literature that the computational/classification capability of complex-valued networks excel over the real-valued networks. Hence, the classification ability of the CSRAN is evaluated on two real-valued multicategory benchmark classification problems with unbalanced datasets from the UCI machine learning repository [30], namely, the vehicle classification (VC) problem and the glass identification (GI) problem. For the real-valued classification problems, the CSRAN results are compared with the multilayer multivalued network (MLMVN) [17], phase encoded CVNN (PE-CVNN) [18], and the real-valued sequential multicategory classifier for RBF networks (SMC-RBF) [31].

All the studies were conducted in MATLAB 7.10 environment on a desktop PC with Pentium duo processor (2.6 GHz) and 4 GB memory.

A. SCFAP

First, we present the performance of the CSRAN using the SCFAP-I defined in the previous section and compare it with existing complex-valued learning algorithm in the literature. The results for CMRAN, FC-MLP, CRBF, and FC-RBF are reproduced from [21] and [26].

Table II presents the training and testing performance measures (J_{Me} and Φ_e), number of neurons (K), and training time used for sequential and batch learning algorithms considered in this paper. From the table, it can be seen that the CSRAN algorithm outperforms the existing sequential learning CMRAN. With only 12 neurons, the magnitude and phase errors are at least 20 times lower than that of CMRAN results. Due to the self-regulating scheme, the CSRAN uses only 1923 samples of the total 3000 samples, and hence the computational effort is much lesser than that of the CMRAN algorithm.

From the table, one can also note that the CSRAN outperforms the CRBF and the C-ELM, whereas the performance of FC-RBF is similar to that of the CSRAN. Since CSRAN is a sequential learning algorithm with a self-regulating scheme, it requires fewer neurons and less computational effort. Also, it can be inferred from the table that the C-ELM, being analytical, requires the lowest computation time to approximate the function. However, its magnitude and phase errors are significantly higher than those of the CSRAN algorithm. The magnitude and phase approximation ability of the I-ELM algorithm is closer to that of CSRAN. However, it requires 1000 neurons to approximate the function, which is impractical and also increases the computational cost.

Effect of Noise: To study the effect of the noise on the performance of the CSRAN, an additive white Gaussian noise of mean 0 and variance 0.01 was added to the input data. The noise power is chosen such that the ratio of the signal power to the noise power is 30 dB. Results for the CSRAN and CMRAN with noisy data are presented in Table III.

From the table, it can be seen that the performance of the CSRAN is less affected by noise even at 30 dB signal-to-noise ratio (SNR). It can also be noted that the performance of CSRAN with noise is far better than the performance of CMRAN without noise, although the CSRAN algorithm requires four additional neurons to approximate the noisy data. During the study, it was observed that only 56 samples have not been used and the remaining 2944 samples have participated in the learning process.

B. Complex QAM Equalization Problem

QAM is an analog/digital modulation scheme that conveys two message signals by modulating the amplitudes of two carrier waves (quadrature carriers) that are 90° out of phase with each other. Thus, the signals in the QAM schemas are

TABLE III
EFFECT OF ADDITIVE WHITE GAUSSIAN NOISE

Algo.	K	Training		Testing	
		J_{Me}	Φ_e (deg.)	J_{Me}	Φ_e (deg.)
CSRAN	16	0.0072	5.8	0.007	5.4
CMRAN	27	0.0773	18.1	0.082	21.3

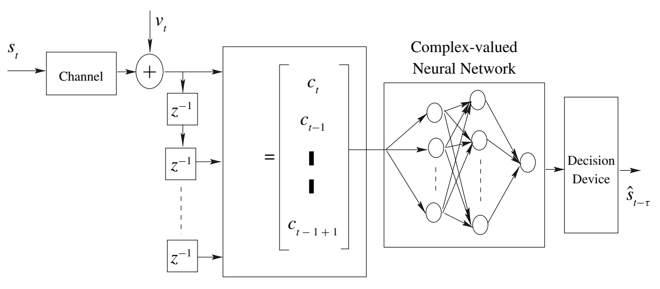

Fig. 9. CVNN equalization scheme.

complex-valued. When the QAM signals are transmitted over a channel, the nonlinear characteristics of the channel cause spectral spreading, inter-symbol interference, and constellation warping. Hence, an equalizer is essential at the receiver of the communication channel to reduce the precursor inter-symbol interference without any substantial degradation in the SNR.

Among the many channel models available in the literature for the complex QAM, the Cha and Kassam [28] channel model is a well-known model. The Cha and Kassam channel model is a complex-valued nonlinear nonminimum phase channel model and is therefore considered in this paper. It is a third-order channel model with nonlinear distortion for 4-QAM signaling. The different symbols transmitted in the channel are $\{1+i, 1-i, -1+i, -1-i\}$.

The channel output (c_t) at time t defined in [28] is given by

$$c_t = o_t + 0.1o_t^2 + 0.05o_t^3 + \nu_t \tag{22}$$

where $o_t = (0.34 - 0.27i)s_t + (0.87 + 0.43i)s_{t-1} + (0.34 - 0.21i)s_{t-2}$, ν_t is a white Gaussian noise with zero mean and 0.01 variance, and s_t is the transmitted symbol at time t. Since the channel model is of order three, the neural network equalizer input vector at time t is a 3-D complex-valued channel observation $\mathbf{x}_t = [c_t\ c_{t-1}\ c_{t-2}]^T$ and the target output ($\mathbf{y}_t$) is the transmitted symbol $s_{t-\tau}$. In this paper, the equalizer decision delay (τ) is set as 1 (as used in [22]).

An architecture of the complex-valued neural equalizer is shown in Fig. 9. The output of the channel forms the input to the equalizer. The objective of the equalizer is to estimate the phase of the transmitted symbol ($s_{t-\tau}$) from the channel output ($\mathbf{x}_t$). Finally, a decision device predicts the transmitted symbol depending on the phase of the equalizer output.

First, we study the effect of the number of samples and the SNR in the training data on the symbol classification performance of the CSRAN equalizer. For this purpose, a simulation study is conducted to observe the performance of the CSRAN equalizer using four training datasets. These four datasets consist of 5000 samples generated at 16 dB/20 dB SNR and 1000 samples generated at 16 dB/20 dB SNR. CSRAN network trained with the above four datasets are tested using 100 000 samples each at noise levels between 4 and 20 dB, in increments of 2 dB, and the symbol classification performance of the CSRAN equalizer is observed in terms of their symbol error rates (SERs). SER is defined as the percentage of symbols that have errors relative to the total number of symbols received in a transmission system. The SER of the CSRAN equalizer for the testing datasets at various noise levels are shown in Fig. 10.

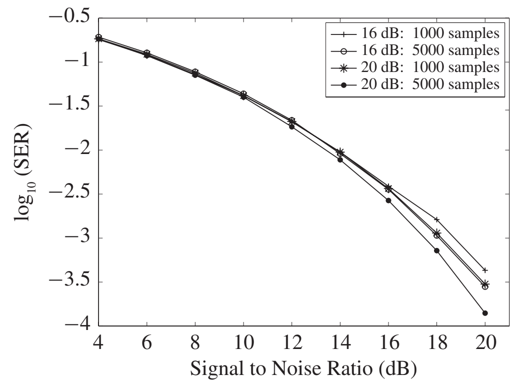

Fig. 10. Effect of number of training samples and SNR on the equalization performance.

From the figure, it can be observed that the CSRAN equalizer trained using a dataset with 5000 samples at 20 dB SNR performs slightly better than that trained with 1000 samples at 16 dB SNR. Hence, in this paper, we use the training sample set with 5000 samples at 20 dB SNR for all the complex-valued neural equalizers.

The performances of the various complex-valued neural equalizers are evaluated using the magnitude error (J_{Me}), the average absolute phase error (Φ_e), and the SER. The thresholds of the CSRAN are initialized as $\{E_a^M = 1.8, E_a^\phi = 1.8\}$, $\{E_l^M = 0.2$, and $E_l^\phi = 0.2\}$, the other parameters are same as in SCFAP-I. For the batch learning networks used in this paper, the training sample set is presented 1000 times repeatedly. Finally, the results of all these algorithms are also compared with the Bayesian equalizer, which is optimal (the best result one can achieve) [1].

From the results, we find that there are no reserve samples. Out of 5000 samples, 4710 samples are used for learning and the remaining 290 samples are deleted. For the QAM problem, CSRAN used seven neurons to capture the nonlinear relationship between the channel output and the transmitted symbol.

Table IV summarizes the training/testing performances of the different complex-valued neural equalizers, the number of hidden neurons (K) used, and the training time. From the table, it can be seen that J_{Me} and Φ_e of the CSRAN are lower than all the other equalizers chosen for the comparison, except that the I-ELM has lower magnitude error than the CSRAN. However, I-ELM requires 100 neurons to achieve this magnitude error, whereas the CSRAN requires 7 neurons to obtain a comparable magnitude error and a minimum phase error. Fig. 11 gives the SER plot for the different equalizers.

TABLE IV
PERFORMANCE COMPARISON FOR THE NONLINEAR NONMINIMUM COMPLEX PHASE EQUALIZATION PROBLEM AT 20 dB

Algorithm	Time (s)	K	Training		Testing	
			J_{Me}	Φ_e (deg.)	J_{Me}	Φ_e (deg.)
CSRAN	45	7	0.2	6.4	0.38	7.09
CMRAN	555	40	0.4	17.47	0.43	21.17
CRBF	8107	15	0.6	35.19	0.6	39.86
C-ELM	0.36	15	0.6	34.14	0.58	35.11
I-ELM	17.5	100	0.2	12.05	0.21	12.11
FC-RBF	3840	15	0.4	31.17	0.41	14.69
FC-MLP	3862	15	0.2	6.47	0.72	31.1

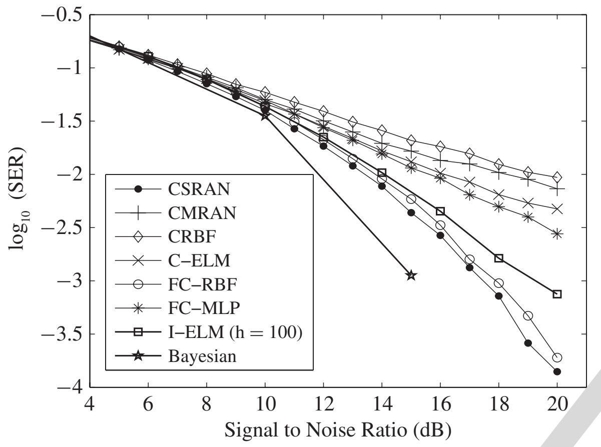

Fig. 11. Error probability curve for various complex-valued neural equalizers.

It can be observed that the CSRAN has a lower SER at a SNR of 20 dB. However, at lower SNR, the effects of all the equalizers are almost similar. The CSRAN equalizer achieves an SER performance of −3 (approximately) at 17 dB SNR, whereas the optimal Bayesian equalizer achieves the same SER performance at 15 dB. Other complex-valued neural equalizers attain the same performance at a higher SNR.

C. Adaptive Beam Forming Problem

Adaptive beam-forming is an antenna-array signal processing problem, where the beams are directed to the desired signal directions (beam-pointing) and the nulls are directed to the interference directions (null-pointing) [29]. The antenna array of a typical beam former consists of M single-transmit antennae operating at the same carrier frequency and L uniformly spaced receiver elements. The spacing between the receiver elements (d) is half the wavelength (λ) of the received signal [29]. Assuming θ to be the angle of incidence that an incoming signal makes with the receiver array broadside, it can be derived from the basic trigonometric identities and the geometry of the sensor array [29] that the signal received at the kth receiver antenna element is

$$\bar{u}_k = e^{\frac{j2\pi k d sin\theta}{\lambda}}. \tag{23}$$

At any given instant t, the input to the beam former is the total signal induced at the receiver antenna elements and is given by

$$\mathbf{x}_t = [\bar{u}_0 + \eta_0 \;\; \bar{u}_1 + \eta_1 \;\; \bar{u}_2 + \eta_2 \;\; \ldots \;\; \bar{u}_L + \eta_L]^T \tag{24}$$

where η_k is the noise at the kth receiver element.

Let $\mathbf{b} = [b_1 \ldots b_k]^T$ be weight vector for the sensor array. The actual signal transmitted at a given instant (y_t) is given by

$$y_t = \mathbf{b}^H \mathbf{x}. \tag{25}$$

The actual signal transmitted (y_t) is the target of the beam former. Thus, the objective of an adaptive beam former is to estimate the weights [$\mathbf{b}$ of (25)], given the signal transmitted (y_t) and the signal received by the beam former antenna array ($\mathbf{x}_t$).

In this paper, we consider a uniform linear array of five sensors. The adaptive beam former should be trained to amplify the signals at −30° and +30° and also null the interferences at −15, 0, and +15° [29]. The received signal at array elements ($\mathbf{x}_t$) is corrupted with an additive Gaussian noise at 50 dB SNR. A training dataset of 250 randomly chosen samples (50 for each signal/interference angles) is used to train the various beam formers. The results for the CRBF, C-ELM, FC-RBF, and FC-MLP beam formers are reproduced from [26]. The self-regulating thresholds of the CSRAN for the adaptive beam-forming problem are initialized as $\{E_a^M = 1.0, E_a^\phi = 1.5\}$, and $\{E_l^M = 0.1, E_l^\phi = 0.1\}$. The other parameters are same as in SCFAP-I.

CMRAN [5] uses 32 neurons to perform beam forming while the CSRAN requires only 6 neurons. Moreover, CSRAN beam former uses only 234 samples in the learning process. The gains for the signals and interference nulls for the various beam-formers are summarized in Table V. From the table, it can be observed that the CSRAN beam former outperforms all the other complex-valued beam formers and its performance is also slightly better than the conventional optimal matrix method [32].

As mentioned earlier in Section I, the orthogonal decision boundaries of a complex-valued network makes them well suited for solving real-valued classification problems. Hence, we also study the classification performance of CSRAN using real-valued benchmark problems from the UCI machine learning repository [30].

D. Real-Valued Benchmark Classification Problems

In this section, we evaluate the classification performance of the CSRAN in the complex domain using two real-valued benchmark classification problems. Two unbalanced datasets from the UCI machine learning repository [30], namely, the VC and the GI, are chosen for the study. The datasets for the VC and the GI problems are unbalanced datasets with 18 and 9 input features, respectively. The total number of classes in the VC is 4 while the GI has 7. The number of samples used for training/testing in VC and GI problems are 424/422 and 109/105, respectively.

Given N training samples, $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots (\mathbf{x}_t, \mathbf{y}_t), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$, the classification problem in the complex domain is defined as estimating the decision function ($F : \mathbb{C}^m \rightarrow \mathbb{C}^n$)

TABLE V
PERFORMANCE COMPARISON OF VARIOUS CVNN-BASED BEAM FORMERS

DOA [a]	Gain (dB)						
	CSRAN	CMRAN	CRBF	C-ELM	FC-RBF	FC-MLP	MM[b]
Beam-1: $-30°$	-13.83	-16.84	-17.94	-18.05	-16.99	-13.98	-13.98
Null-1: $-15°$	-59.56	-49.77	-27.53	-48.28	-58.45	-53.99	-57.02
Null-2: $0°$	-57.37	-49.6	-27	-41.64	-57.23	-53.99	-57
Null-3: $15°$	-57.93	-48.18	-28.33	-47.5	-56.32	-53.99	-57.02
Beam-2: $30°$	-13.93	-17.52	-17.92	-16.68	-17.00	-13.98	-13.98

[a] Direction of Arrival (DoA)
[b] Matrix Method (MM)

TABLE VI
PERFORMANCE COMPARISON ON THE BENCHMARK CLASSIFICATION PROBLEMS

Problem	Domain	Algorithm	K	Time (s)	Testing	
					η_o	η_a
VC	Real	SMC-RBF	75	120	74.18	73.52
	Comp.	PE-CVNN	-	-	78.7[b]	-
		MLMVN	90	1396	78	77.25
		CSRAN	80	352	79.15	79.16
GI	Real	SMC-RBF	58	97	78.09	77.16
	Comp.	PE-CVNN	-	-	65.5[b]	-
		MLMVN	85	1421	73.24	66.83
		CSRAN	80	452	83.5	78.09

[b] In [18], a 10-fold validation has been done using 90% of the total samples in training and the remaining 10% for testing in each validation. In this paper, only 50% of the total samples are used in training as per the guidelines in UCI machine learning database.

to enable accurate prediction of the class labels of new unseen samples. Here, the complex-valued input features ($\mathbf{x}_t$) are obtained by transforming the real-valued input features to the complex domain using the PE transformation presented in [18] and the complex-valued coded class labels are given as

$$y_t^k = \begin{cases} 1 + 1i & \text{if} \quad c_t = k \\ -1 - 1i & \text{otherwise} \end{cases} \quad k = 1, \ldots n \tag{26}$$

where c_t is the actual class label.

The performance of CSRAN is compared with complex-valued PE-CVNN [18] and MLMVN [17] classifiers. Also, we compare the results with the best performing real-valued SMC-RBF classifier [31]. The overall (η_o) and the average (η_a) classification accuracies defined in [31] are used as the performance measures for comparison. For the classification problems, the various self-regulating thresholds of the CSRAN are initialized as $\{E_a^M = 1.6,\ E_a^\phi = 1.0\}$, $\{E_l^M = 0.6$, and $E_l^\phi = 0.6\}$.

The results of the classifiers on the two problems are presented in Table VI. It can be observed from the Table that the performance of the CSRAN classifier is better than the other complex-valued classifiers in both the problems. The CSRAN presents at least 1% improvement in the classification of vehicles and at least 10% improvement in the identification of glass. Moreover, the classification performance of the CSRAN is at least 5% better than the best performing real-valued SMC-RBF classifier in the VC and GI problems.

Hence, it can be inferred from the study that the CSRAN outperforms other complex-valued learning algorithms and is also better than the other complex-valued and best performing real-valued classifiers available in the literature.

IV. CONCLUSION

A sequential learning algorithm with a self-regulating scheme has been developed for a complex-valued resource allocation network. The self-regulating scheme uses explicit magnitude and phase thresholds to control the "sample deletion," "sample learning," and "sample reserve" mechanisms. The sample learning includes growing/pruning the hidden neurons and updating the network parameters using a C-EKF algorithm. The self-regulating scheme in the CSRAN helps in achieving better generalization performance with a compact network structure. A complex-valued synthetic function approximation has been used to clearly describe the various key features of the CSRAN algorithm. The performance of the CSRAN has been evaluated using a complex-valued QAM channel equalization problem and an adaptive beam-forming problem. The results indicate the superior performance of the CSRAN algorithm. From a set of real-valued benchmark classification problems, we highlighted the better decision-making ability of the proposed CSRAN classifier over other existing complex-valued/real-valued classifiers.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable suggestions and insightful comments that helped in improving the quality of this manuscript.

REFERENCES

[1] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Complex-valued growing and pruning RBF neural networks for communication channel equalisation," *IEE Proc.-Vis., Image Signal Process.*, vol. 153, no. 4, pp. 411–418, Aug. 2006.

[2] J. C. Brégains and F. Ares, "Analysis, synthesis, and diagnostics of antenna arrays through complex-valued neural networks," *Microw. Opt. Technol. Lett.*, vol. 48, no. 8, pp. 1512–1515, Aug. 2006.

[3] R. Savitha, S. Vigneshwaran, S. Suresh, and N. Sundararajan, "Adaptive beamforming using complex-valued radial basis function neural networks," in *Proc. IEEE Reg. 10 Conf.*, Nov. 2009, pp. 1–6.

[4] C. Shen, H. Lajos, and S. Tan, "Symmetric complex-valued RBF receiver for multiple-antenna-aided wireless systems," *IEEE Trans. Neural Netw.*, vol. 19, no. 9, pp. 1659–1665, Sep. 2008.

[5] D. Jianping, N. Sundararajan, and P. Saratchandran, "Complex-valued minimal resource allocation network for nonlinear signal processing," *Int. J. Neural Syst.*, vol. 10, no. 2, pp. 95–106, 2000.

[6] I. Aizenberg, D. V. Paliy, J. M. Zurada, and J. T. Astola, "Blur identification by multilayer neural network based on multivalued neurons," *IEEE Trans. Neural Netw.*, vol. 19, no. 5, pp. 883–898, May 2008.

[7] M. K. Muezzinoglu, C. Guzelis, and J. M. Zurada, "A new design method for the complex-valued multistate Hopfield associative memory," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 891–899, Jul. 2003.

[8] G. Tanaka and K. Aihara, "Complex-valued multistate associative memory with nonlinear multilevel functions for gray-level image reconstruction," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1463–1473, Sep. 2009.

[9] N. Sinha, M. Saranathan, K. R. Ramakrishna, and S. Suresh, "Parallel magnetic resonance imaging using neural networks," in *Proc. IEEE Int. Conf. Image Process.*, vol. 3. San Antonio, TX, Sep.–Oct. 2007, pp. 149–152.

[10] C.-C. Yang and N. K. Bose, "Landmine detection and classification with complex-valued hybrid neural network using scattering parameters dataset," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 743–753, May 2005.

[11] H. Leung and S. Haykin, "The complex backpropagation algorithm," *IEEE Trans. Signal Process.*, vol. 39, no. 9, pp. 2101–2104, Sep. 1991.

[12] S.-S. Yang, C.-L. 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 Trans. Neural Netw.*, vol. 18, no. 5, pp. 1280–1293, Sep. 2007.

[13] S.-S. Yang, S. Siu, and C.-L. Ho, "Analysis of the initial values in split-complex backpropagation algorithm," *IEEE Trans. Neural Netw.*, vol. 19, no. 9, pp. 1564–1573, Sep. 2008.

[14] T. Kim and T. Adali, "Fully complex multilayer perceptron network for nonlinear signal processing," *J. VLSI, Signal Process.*, vol. 32, nos. 1–2, pp. 29–43, Aug.–Sep. 2002.

[15] D. V. Loss, M. C. F. DeCastro, P. R. G. Franco, and F. C. C. DeCastro, "Phase transmittance RBF neural networks," *Electron. Lett.*, vol. 43, no. 16, pp. 882–884, Aug. 2007.

[16] T. Nitta, "Solving the XOR problem and the detection of symmetry using a single complex-valued neuron," *Neural Netw.*, vol. 16, no. 8, pp. 1101–1105, Oct. 2003.

[17] I. Aizenberg and C. Moraga, "Multilayer feedforward neural network based on multi-valued neurons (MLMVN) and a backpropagation learning algorithm," *Soft Comput.*, vol. 11, no. 2, pp. 169–193, Sep. 2007.

[18] M. F. Amin, M. M. Islam, and K. Murase, "Ensemble of single-layered complex-valued neural networks for classification tasks," *Neurocomputing*, vol. 72, nos. 10–12, pp. 2227–2234, Jun. 2009.

[19] G. M. Georgiou and C. Koutsougeras, "Complex domain backpropagation," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Process.*, vol. 39, no. 5, pp. 330–334, May 1992.

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

[21] 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, nos. 16–18, pp. 3771–3781, Oct. 2009.

[22] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, nos. 1–4, pp. 306–314, Oct. 2005.

[23] G.-B. Huang, M. Li, L. Chen, and C. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing*, vol. 71, nos. 4–6, pp. 576–583, Jan. 2008.

[24] S. Chen, S. McLaughlin, and N. Mulgrew, "Complex-valued radial basis function network, part I: Network architecture and learning algorithms," *EURASIP Signal Process. J.*, vol. 35, no. 1, pp. 19–31, 1994.

[25] S. Chen, X. Hong, C. J. Harris, and L. Hanzo, "Fully complex-valued radial basis function networks: Orthogonal least squares regression and classification," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3421–3433, 2008.

[26] R. Savitha, S. Suresh, and N. Sundararajan, "A fully complex-valued radial basis function network and its learning algorithm," *Int. J. Neural Syst.*, vol. 19, no. 4, pp. 253–267, 2009.

[27] S. L. Goh and D. P. Mandic, "An augmented extended Kalman filter algorithm for complex-valued recurrent neural networks," *Neural Comput.*, vol. 19, no. 4, pp. 1039–1055, Apr. 2007.

[28] I. Cha and S. A. Kassam, "Channel equalization using adaptive complex radial basis function networks," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 1, pp. 122–131, Jan. 1995.

[29] A. B. Suksmono and A. Hirose, "Intelligent beamforming by using a complex-valued neural network," *J. Intell. Fuzzy Syst.*, vol. 15, nos. 3–4, pp. 139–147, Dec. 2004.

[30] C. Blake and C. Merz. (1998). *UCI Repository of Machine Learning Databases*. Dept. Inf. Comput. Sci., Univ. California, Irvine [Online]. Available: http://archive.ics.uci.edu/ml/

[31] S. Suresh, N. Sundararajan, and P. Saratchandran, "A sequential multi-category classifier using radial basis function networks," *Neurocomputing*, vol. 71, nos. 7–9, pp. 1345–1358, Mar. 2008.

[32] R. A. Monzingo and T. W. Miller, *Introduction to Adaptive Arrays*. Raleigh, NC: SciTech, 2003.

Sundaram Suresh (SM'–) received the B.E. degree in electrical and electronics engineering from Bharathiyar University, Coimbatore, India, in 1999, and the M.E. and Ph.D. degrees in aerospace engineering from the Indian Institute of Science, Bangalore, India, in 2001 and 2005, respectively. AQ:1

He was a Post-Doctoral Researcher in the School of Electrical Engineering, Nanyang Technological University, Singapore, from 2005 to 2007. From 2007 to 2008, he was with the National Institute for Research in Computer Science and Control-Sophia Antipolis, Nice, France, as a Research Fellow of the European Research Consortium for Informatics and Mathematics. He was with Korea University, Seoul, Korea, for a short period as a Visiting Faculty in industrial engineering. From January 2009 to December 2009, he was with the Department of Electrical Engineering, Indian Institute of Technology, Delhi, India, as an Assistant Professor. Since 2010, he has been an Assistant Professor with the School of Computer Engineering, Nanyang Technological University. His current research interests include flight control, unmanned aerial vehicle design, machine learning, and optimization and computer vision.

Ramasamy Savitha (S'–) received the B.E. degree in electrical and electronics engineering from Manonmaniam Sundaranar University, Thirunelveli, India, in 2000, and the M.E. degree in control and instrumentation engineering from the College of Engineering Guindy, Anna University, Chennai, India, in 2002. She received the Ph.D. degree from Nanyang Technological University, Singapore, in 2006. AQ:2

Her current research interests include neural networks and control.

Narasimhan Sundararajan (F'–) received the B.E. degree in electrical engineering with first class honors from the University of Madras, Chennai, India, in 1966, the M.Tech. degree from the Indian Institute of Technology, Chennai, in 1968, and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign, Urbana, in 1971. AQ:3

He worked in various capacities with the Indian Space Research Organization, Bangalore, India, from 1971. Since 1991, he has been working as a Professor in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He also worked as a National Research Council Research Associate at NASA Ames Research Center, Ames, CA, in 1974, and a Senior Research Associate at NASA Langley, Hampton, VA, in 1981 and 1986. He has published more than 200 papers and written four books. His current research interests include aerospace control and neural networks.

Prof. Sundararajan has served as an Associate Editor for a number of journals and as a program committee member in a number of international conferences. He is an Associate Fellow of the American Institute of Aeronautics and Astronautics and a Fellow of the Institution of Engineers Singapore.

EDITOR QUERY

EQ:1 = Please provide the revised and accepted date for this article.

AUTHOR QUERIES

AQ:1, 2 & 3	=	Please provide the membership year for the authors “Sundaram Suresh, Ramasamy Savitha, and Narasimhan Sundararajan.”