Chapter 6
Circular Complex-valued Extreme Learning Machine Classifier

Artificial Neural Networks (ANN) were originally inspired by the central nervous system and its components that constitute the biological neural network, as investigated by the Neuroscience community. Ever since then, several tasks of human activity have been emulated by the ANNs. Classification is one such decision making task that occurs frequently in human activity and one that has been emulated in the artificial neural network framework. A classification task in the ANN framework is defined as assigning an object to a predefined group or class based on a set of object attributes. As the ANNs are capable of constructing complex decision boundaries without any assumption on the statistics on the input data, they have been used to perform classification tasks in a large range of applications spanning from business to medical diagnosis to speech recognition. Over the past twenty years, supervised learning has become a standard tool of choice to analyze the data in many fields, particularly in classfication problems. The objective of the classification problem is to approximate the decision surface described by the training data and predict the class label of the unknown data as accurately as possible. Recently, it has been shown by Nitta that the complex-valued neural networks have better computational power than real-valued networks [1] and they outperform real-valued networks in their ability to approximate the decision boundaries to solve classification problems. Moreover, it has been shown by Nitta [2, 3] that a fully complex-valued neural network with a split type of activation function has two decision boundaries that are orthogonal to each other. These decision boundaries help the complex-valued neural network to perform classification tasks more efficiently than real-valued networks. These findings have inspired researchers to develop efficient classifiers in the Complex domain. The 'Multi Layered neural network based on Multi Valued Neurons (MLMVN)' [4] and the single-layered network with phase encoded transformation, referred to here as, 'Phase Encoded Complex Valued Neural Network (PE-CVNN)' [5] are the two such complex-valued classifiers available in the literature. In this chapter, we briefly discuss these classifiers and show that the complex-valued learning algorithms described in the previous chapters can be modified to solve real-valued classification problems. In addition, we present an efficient and fast learning

S. Suresh et al.: Supervised Learning with Complex-valued Neural Networks, SCI 421, pp. 109–123.
springerlink.com

complex-valued classifier, referred to as 'Circular Complex-valued Extreme Learning Machine (CC-ELM)' to solve real-valued classification problems. CC-ELM is a single hidden layer network with a non-linear input and hidden layer and a linear output layer. A circular transformation with a translational/rotational bias that performs an unique one-to-one transformation of the real-valued feature to the Complex plane is used as an activation function for the neurons in the input layer. Neurons in the hidden layer employ a fully complex-valued Gaussian-like ('*sech*') activation function. The input parameters of the CC-ELM are chosen randomly and the output weights are computed analytically. This chapter also presents an analytical proof to show that the decision boundaries of a single complex-valued neuron at the hidden and output layer of the CC-ELM consists of two hyper-surfaces that intersect orthogonally.

6.1 Complex-valued Classifiers in the Literature

In this section, we review the existing complex-valued classifiers. First, we define the real-valued classification problem in the Complex domain and describe the learning algorithms of the two complex-valued classifiers, viz., the Multi-Layer neural network based on Multi-Valued Neurons (MLMVN) and the Phase Encoded Complex-valued Neural Network (PE-CVNN).

6.1.1 Description of a Real-valued Classification Problem Done in the Complex Domain

Suppose we have N observations $\{(\mathbf{x}_1, c_1), \cdots, (\mathbf{x}_t, c_t), \cdots, (\mathbf{x}_N, c_N)\}$, where $\mathbf{x}_t \in \Re^m$ be the m-dimensional input features of t-th observation, $c_t \in [1, 2, \cdots, n]$ are its class labels, and n is the number of distinct classes. The observation data $(\mathbf{x}_t, c_t)$ are random in nature and the observation $\mathbf{x}_t$ provides some useful information on the probability distribution over the observation data to predict the corresponding class label (c_t) with a certain accuracy.

To solve the real-valued classification problems using complex-valued neural networks, the coded class label ($\mathbf{y}^t = [y_1^t \cdots y_k^t \cdots y_n^t]^T$) is defined in the Complex domain as:

$$y_k^t = \begin{cases} 1+i1 & \text{if} \quad c_t = k \\ -1-i1 & \text{otherwise} \end{cases} \quad k = 1, 2, \cdots, n \tag{6.1}$$

The real-valued classification problem using complex-valued neural networks can be viewed as finding the decision function F that maps the real-valued input features to the complex-valued coded class labels, i.e., $F : \Re^m \rightarrow \mathbb{C}^n$. For notational convenience, the superscript t will be dropped in the rest of the chapter.

6.1.2 Multi-Layer Neural Network Based on Multi-Valued Neurons (MLMVN)

The complex-valued MLMVN using multi-valued neurons was developed in [4]. A single multi-valued neuron maps m inputs and a single output. The mapping is described by a multi-valued function of m variables ($f_m(x_1,\cdots,x_m)$) with $m+1$ complex-valued weights as the parameters, and is given by

$$f_m(x_1,x_2,\cdots,x_m) = P(v_0+v_1x_1+\cdots+v_mx_m) \tag{6.2}$$

where $P(.)$ is the activation function of the neuron, given by

$$P(z) = exp(i(arg(z))) = \frac{z}{|z|} \tag{6.3}$$

where $z = v_0+v_1x_1+\cdots+v_mx_m$ is the weighted sum, and $|z|$ is the modulo of the complex number z. The function defined in Eq. (6.3) maps the Complex plane into the whole unit circle and is continuous. It must also be noted that the function is not differentiable as a function of a complex-valued variable.

However, as the learning in the MLMVN is reduced to the movement along the unit circle, the derivative of the activation function is not required because it is impossible to move in an incorrect direction. Any direction of movement along the circle will lead to the target and the shortest way of this movement is determined by the error which is the difference between the desired output and the actual output. The weight update rule for a multi-valued neuron used in an MLMVN is given by:

$$V_{m+1} = V_m + \frac{\eta_v}{m+1}\tilde{\delta}\overline{X} \tag{6.4}$$

$$\text{where } \tilde{\delta} = \frac{1}{|z|}\left(y-\frac{z}{|z|}\right) \tag{6.5}$$

From Eq. (6.4), it can be observed that using this approach, the function is a smooth function of the weights. Moreover, a small change in the inputs or weights does not result in a significant change of z. However, as the input features are mapped on to a full unit circle, this mapping results in the same complex-valued features for real-valued features with a value of both 0 and 1. Hence, this transformation is not unique. In addition, the multi-valued neurons map the complex-valued inputs to C discrete outputs on the unit circle. Thus, as the number of classes (C) increases, the number of sectors with the unit circle increases. As a result, the region of each sector (representing each class) within the unit circle decreases, increasing the chances of misclassification. Furthermore, the output neurons of the MLMVN have multiple discrete values and hence, the MLMVN classifier lacks the orthogonal decision boundaries that are the main characteristics of complex-valued neural classifiers.

As the transformation used in the MLMVN does not perform an one-to-one mapping of the real-valued features onto the Complex plane, Amin et. al. [5] proposed a phase encoded transformation to convert the real-valued features to the complex

domain. They also developed a single layered complex-valued classifier using this phase encoded transformation. We describe the transformation and the learning algorithm of the network developed in the next section.

6.1.3 Phase Encoded Complex-Valued Neural Network (PE-CVNN)

In PE-CVNN [5], the following phase encoded transformation is used to transform the real-valued input features to the complex domain:

$$\text{Let } x_j \in [a,b], \text{where} a,b \in \Re, \text{ then } \phi = \frac{\pi(x_j - a)}{b-a} \tag{6.6}$$

$$\text{and } z_j = e^{i\phi} = cos(\phi) + isin\phi \tag{6.7}$$

It can be observed from Eq. (6.6) that the the transformation linearly maps the input features from $x \in [a,b] \rightarrow \phi \in [0,\pi]$.

The weighted inputs at the output layer of the PE-CVNN are given by:

$$z_{ok} = z_{ok}^R + iz_{ok}^I = \sum_{j=1}^{m} v_{kj} z_j + \theta_k \tag{6.8}$$

and the output of the network is

$$\widehat{y}_k = f_{\mathbb{C}\rightarrow\Re} \tag{6.9}$$

$$\text{and } f_{\mathbb{C}\rightarrow\Re} = \sqrt{(f_R(z_{ok}^R))^2 + (f_R(z_{ok}^I))^2} \tag{6.10}$$

$$\text{OR } f_{\mathbb{C}\rightarrow\Re} = (f_R(z_{ok}^R) - f_R(z_{ok}^R))^2$$

$$\text{where, } f_R(z_{ok}^R) = 1/1 + exp(-(z_{ok}^R)) \text{ and}$$

$$f_R'(z_{ok}^R) = f_R(z_{ok}^R)(1 - f_R(z_{ok}^R)) \tag{6.11}$$

The residual error of the network output is given as:

$$e_k = y_k - \widehat{y}_k \tag{6.12}$$

The PE-CVNN uses a gradient descent based learning rule with the mean squared error criterion to update the weights of the network. The weight update rule is given by:

$$\Delta v_{kj} = \overline{z_j} \Delta\theta_k$$

$$\text{where} \Delta\theta_k = \Delta\theta_k^R + i\Delta\theta_k^I \tag{6.13}$$

If the activation function used in the PE-CVNN is given by Eq. (6.11), then $\Delta\theta_k^R$ is

$$\Delta\theta_k^R = 2\eta e_k \left(f_R(z_{ok}^R) - f_R(z_{ok}^I)\right) f_R'(z_{ok}^R) \tag{6.14}$$

$$\text{and } \Delta\theta_k^I = 2\eta e_k \left(f_R(z_{ok}^I) - f_R(z_{ok}^R)\right) f_R'(z_{ok}^I) \tag{6.15}$$

The transformation used in the PE-CVNN considers only the first two quadrants of the Complex plane and does not fully exploit the advantages of the orthogonal decision boundaries offered by the complex-valued neural networks. In addition, the activation functions used in the PE-CVNN are similar to those used in the split-type complex-valued neural networks, as shown in Eqs. (6.10) and (6.11). Also, the gradients used in the PE-CVNN are not fully complex-valued [6] thereby resulting in a significant loss of complex-valued information. PE-CVNN uses a gradient-descent based batch learning algorithm that requires a significant computational effort to approximate the decision surface.

It is noteworthy that the transformations used in MLMVN and PE-CVNN can be used to transform the real-valued input features to the Complex domain and any of the complex-valued learning algorithms discussed in Chapters 2-4 can be used to solve real-valued classification problems using the transformed features. Two factors play a vital role in deciding the classification ability of the complex-valued neural networks:

1. The transformation used to convert the real-valued input features to the Complex domain
2. The activation function and learning algorithm used in the complex-valued neural network.

In the next section, we discuss the modifications required to enhance the decision making ability of FC-MLP, FC-RBF and Mc-FCRBF such that these algorithms can be used to solve real-valued classification problems.

6.1.4 Modifications in FC-MLP, FC-RBF and Mc-FCRBF Learning Algorithm to Solve Real-valued Classification Problems

The learning algorithm of FC-MLP, FC-RBF and Mc-FCRBF described in Chapters 2 and 3 have been used to solve complex-valued function approximation problems in Chapter 5. Although they can also be used to approximate the decision surface to solve real-valued classification problems, we modify these algorithms to improve their classification performance. In this respect, the mean squared error defined in Eq. (3.14) is replaced with the hinge loss function and the criteria for parameter update in Mc-FCRBF is modified to incorporate a classification measure also.

Hinge loss error function: Recently, it was shown in [7] and [8] that in real-valued classifiers, the hinge loss function helps the classifier to estimate the posterior probability more accurately than the mean squared error function. Hence, while using FC-MLP, FC-RBF and Mc-FCRBF to solve real-valued classification problems, we use the hinge loss error function defined as:

$$e_l^t = \begin{cases} 0 \text{ if } (y_l^{tR})(\widehat{y}_l^{tR}) > 0 \\ y_l^t - \widehat{y}_l^t \quad \text{otherwise} \end{cases} \quad l = 1, 2, \cdots n \tag{6.16}$$

where the superscript R refers to the real-part of the Complex signal.

Criteria for parameter update: In addition to the hinge loss error function, we modify the parameter update criteria of Mc-FCRBF to enhance its decision making ability. While solving real-valued classification problems in the Complex domain, it has to be ensured that the predicted class label of Mc-FCRBF ($\widehat{c}^t$) is the same as that of the target class label (c^t). Therefore, we have modified the parameter update conditions to accommodate this class label information also. Accordingly, the sample learning condition of Mc-FCRBF (Eq. (3.38)) to solve real-valued classification problems is:

$$\text{If } \widehat{c}^t \neq c^t \text{ OR } \left(M_t^e \geq E_l^M \text{ AND } \phi_t^e \geq E_l^\phi\right) \tag{6.17}$$

Then, update the network parameters according to Eqs. (3.31), (3.32), and (3.33). Here, $\widehat{c}^t$ is the predicted class label for the sample t defined as:

$$\widehat{c}^t = \max_{l=1,2,\cdots,C} \text{ real } \left(\widehat{y}_l^t\right) \tag{6.18}$$

6.2 Circular Complex-valued Extreme Learning Machine Classifier

In this section, a fast learning fully-complex valued classifier called 'Circular Complex-valued Extreme Learning Machine' is developed to solve real-valued classification tasks.

6.2.1 Architecture of the Classifier

The circular complex-valued extreme learning machine classifier is a single hidden layer network with m input neurons, h hidden neurons and n output neurons, as shown in Fig. 6.1. The neurons in the input layer employ a non-linear circular transformation as the activation function, as shown in the inset of Fig. 6.1.

The circular transformation that transforms the real-valued input features into the Complex domain ($\Re \rightarrow \mathbb{C}$) is given by

$$z_l = sin\left(ax_l + ibx_l + \alpha_l\right), \;\; l = 1,2,\cdots,m \tag{6.19}$$

where $a,\ b,\ \alpha_l \in \Re^+$ are non-zero constants and x_l is the input feature normalized in $[0,1]$. The scaling factors a, and b, and the translational, rotational bias term α_l are randomly chosen such that $0 < a,b \leq 1$, and $0 < \alpha_l < 2\pi$. The translational, rotational bias term α_l shifts the origin of the resultant complex-valued feature (z_l) and rotates it to any quadrant of the Complex domain.

The effect of the circular transformation for different values of the translational, rotational bias term (α_l) is shown in Fig. 6.2. It can be observed from this figure that the randomly chosen bias terms perform the translation/rotation of the feature vector in different quadrants of the complex-valued feature space. Therefore, the bias term ($\alpha_l,\ l = 1,\ \cdots,\ m$) associated with each input feature ensures that all the input

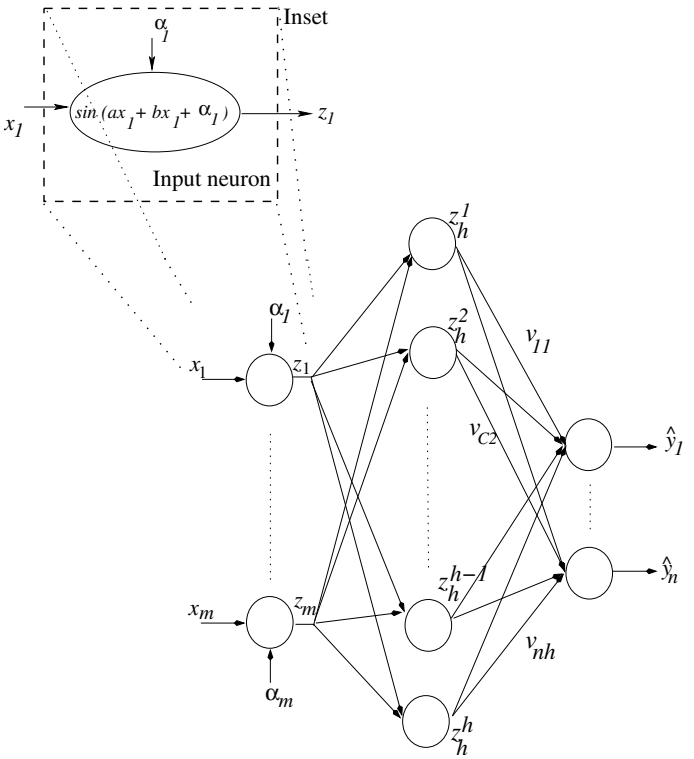

Fig. 6.1 The architecture of a circular complex-valued extreme learning machine. The expanded box (inset figure) shows the actual circular transformation function which maps real-valued input feature to complex-valued feature.

features are not mapped onto the same quadrant of the Complex plane. Thus, as the input features are well distributed in the Complex plane, the CC-ELM exploits the orthogonal decision boundaries of the fully complex-valued neural networks more efficiently.

The essential properties of a fully complex-valued activation function given in [6] states that for a complex-valued non-linear function to be used as an activation function, the function has to be analytic and bounded *almost everywhere*. As the circular transformation is used as an activation function in the input layer, we need to ensure that the transformation satisfies the essential properties of a fully complex-valued activation function. The '*sine*' activation function is an analytic function with an essential singularity at ∞. The net input to the '*sine*' function

$$ax_l + ibx_l + \alpha_l = \infty \text{ if } ax_l + \alpha_l = \infty \text{ or } bx_l = \infty \tag{6.20}$$

Since the transformation constants (a,b) and the translational/rotational bias (α_l) are restricted between $(0,1]$ and $(0,2\pi)$ respectively, the transformation becomes unbounded only when the input feature is ∞. Since, the real-valued features are

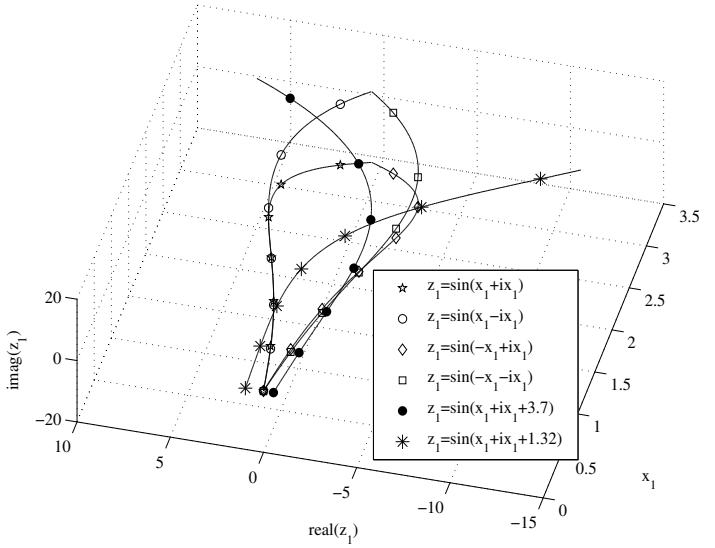

Fig. 6.2 Significance of translational/rotational bias term (α) in the circular transformation

normalized between $[0,1]$, the circular transformation is analytic and bounded *almost everywhere*. Hence, the circular transformation used in the input layer of the CC-ELM is a valid activation function.

The neurons in the hidden layer of the CC-ELM employ a fully-complex valued '*sech*' activation function (Gaussian like) as developed in [9]. The hidden layer response (z_h^j) of the CC-ELM is given by

$$z_h^j = sech\left[\boldsymbol{\sigma}_\mathbf{j}^\mathbf{T}\left(\mathbf{z}-\mathbf{c_j}\right)\right], \quad j = 1,2,\cdots,h \tag{6.21}$$

where $\boldsymbol{\sigma}_\mathbf{j} \in \mathbb{C}^\mathbf{m}$ is the complex-valued scaling factor of the j-th hidden neuron, $\mathbf{c}_j \in \mathbb{C}^m$ is the complex-valued center of the j-th hidden neuron and the superscript T represents the matrix transpose operator.

The neurons in the output layer employ a linear activation function. The output ($\widehat{\mathbf{y}} = [\widehat{y}_1 \cdots \widehat{y}_n \cdots \widehat{y}_n]^T$) of the CC-ELM network with h hidden neurons is

$$\widehat{y}_n = \sum_{j=1}^{h} v_{kj} z_h^j, \quad k = 1,2,\cdots,n \tag{6.22}$$

where w_{kj} is the output weight connecting the j-th hidden neuron and the k-th output neuron.

The estimated class label ($\widehat{c}$) is then obtained using

$$\widehat{c} = arg \max_{n=1,2,\cdots,n} \text{ real-part of } \widehat{y}_k \tag{6.23}$$

6.2.2 Learning Algorithm of CC-ELM

The output of CC-ELM given in Eq. (8.17) can be written in matrix form as,

$$\widehat{Y} = VH \tag{6.24}$$

where V is the matrix of all output weights connecting hidden and output neurons. The H is the response of hidden neurons for all training samples and is given as

$$H(V,U,Z) = \begin{bmatrix} sech\left(\boldsymbol{\sigma}_1^T \|\mathbf{z}_1 - \mathbf{c}_1\|\right) & \cdots & sech\left(\boldsymbol{\sigma}_1^T \|\mathbf{z}_N - \mathbf{c}_1\|\right) \\ \vdots & \vdots & \vdots \\ sech\left(\boldsymbol{\sigma}_j^T \|\mathbf{z}_1 - \mathbf{c}_j\|\right) & \cdots & sech\left(\boldsymbol{\sigma}_j^T \|\mathbf{z}_N - \mathbf{c}_j\|\right) \\ \vdots & \vdots & \vdots \\ sech\left(\boldsymbol{\sigma}_h^T \|\mathbf{z}_1 - \mathbf{c}_h\|\right) & \cdots & sech\left(\boldsymbol{\sigma}_h^T \|\mathbf{z}_N - \mathbf{c}_h\|\right) \end{bmatrix} \tag{6.25}$$

Note that H is $h \times N$ hidden layer output matrix. The j-th row of the H matrix represents the hidden neuron response (z_h^j) with respect to the inputs $\mathbf{z}_1, \cdots, \mathbf{z}_N$.

In CC-ELM, the transformation constants (a, b), the translational/rotation bias term ($\boldsymbol{\alpha}$), center (C) and scaling factor ($\boldsymbol{\sigma}$) are chosen randomly and the output weights V are estimated by the least squares method according to:

$$V = YH^{\dagger} \tag{6.26}$$

where $H^{\dagger}$ is the generalized Moore-Penrose pseudo-inverse [10] of the hidden layer output matrix and Y is the complex-valued coded class label.

In short, the CC-ELM algorithm can be summarized as:

- For a given training set (X, Y), select the appropriate number of hidden neurons h.
- Randomly select $0 < a,\ b,\ 0 < \alpha_l < 2\pi$, the neuron scaling factor ($\boldsymbol{\sigma}$) and the neuron centers (C).
- Then calculate the output weights V analytically: $V = YH^{\dagger}$.

Performance of the CC-ELM is influenced by the selection of appropriate number of hidden neurons. Recently, an incremental constructive method to determine the appropriate number of hidden neurons for the C-ELM has been presented in [11]. In this paper, we use a simple neuron incremental-decremental strategy, similar to the one presented in [12] for real-valued networks. The following steps are followed to select the appropriate number of hidden neurons for the CC-ELM network:

Step 1. Select a network with a minimum configuration ($h = m + n$).
Step 2. Select the input weights randomly and compute the output weights analytically.
Step 3. Use leave-one cross-validation to determine training/validation accuracy from the training data.

Step 4. Increase h until the validation accuracy improves and return to Step 2.
Step 5. If the validation accuracy decreases as h increases, then stop.

From this section, it can be observed that the two important properties of the CC-ELM classifier are that:

- The CC-ELM classifier uses a unique circular transformation that makes an one-to-one mapping while transforming the real-valued input features to the Complex domain ($\Re \rightarrow \mathbb{C}$).
- The CC-ELM classifier requires lesser computational effort than other complex-valued classifiers as the weights are computed analytically.

6.2.3 Orthogonal Decision Boundaries in CC-ELM

In this section, we show that the three layered CC-ELM with the fully complex-valued *sech* activation function at the hidden layer exhibits orthogonal decision boundaries. Since CC-ELM maps the complex-valued input features to the higher dimensional Complex plane in the hidden layer randomly, we prove the existence of orthogonal decision boundaries in the output neuron with respect to the hidden layer output. Next, we also show that the decision boundaries formed by the real and imaginary parts of the hidden layer response with respect to the input are orthogonal to each other.

6.2.4 Case (i): Orthogonality of Decision Boundaries in the Output Layer

The responses of the k^{th} output neuron can be written as:

$$\widehat{y}_k = \sum_{j=1}^{h} v_{kj} z_h^j;\ k = 1, \cdots, n \tag{6.27}$$

$$= \sum_{j=1}^{h} \left(v_{kj}^R + i v_{kj}^I\right) \left(z_h^{j^R} + i z_h^{j^I}\right) \tag{6.28}$$

where the superscripts R and I represents the real and imaginary parts of the complex-valued signals, respectively. Therefore,

$$\widehat{y}_k = \widehat{y}_k^R + i\widehat{y}_k^I = \sum_{j=1}^{h} \left(v_{kj}^R z_h^{j^R} - v_{kj}^I z_h^{j^I}\right) + i\left(v_{kj}^R z_h^{j^I} + v_{kj}^I z_h^{j^R}\right) \tag{6.29}$$

From the above equation, we can see that the real-part and the imaginary-part of the kth output neuron forms two decision boundaries with respect to the hidden layer output ($\mathbf{z_h}$). The two decision boundaries are

$$\widehat{y}_k^R = \sum_{j=1}^{h} \left(v_{kj}^R z_h^{j^R} - v_{kj}^I z_h^{j^I} \right) \rightarrow \mathbf{S}^R \tag{6.30}$$

$$\text{and } \widehat{y}_k^I = \sum_{j=1}^{h} \left(v_{kj}^R z_h^{j^I} + v_{nj}^I z_h^{j^R} \right) \rightarrow \mathbf{S}^I \tag{6.31}$$

The real-part decision boundary $\mathbf{S}^R$ classifies the hidden layer output $\mathbf{z}_h \in \mathbb{C}^h$ into two decision regions

$$\left\{ \mathbf{z}_h \in \mathbb{C}^h, | \, \widehat{y}_k^R \geq \mathbf{S}^R \right\} \text{ and } \left\{ \mathbf{z}_h \in \mathbb{C}^h, | \, \widehat{y}_k^R < \mathbf{S}^R \right\} \tag{6.32}$$

Similarly, the imaginary-part decision boundary $\mathbf{S}^I$ classifies the hidden layer output $\mathbf{z}_h \in \mathbb{C}^h$ into two decision regions

$$\left\{ \mathbf{z}_h \in \mathbb{C}^h, | \, \widehat{y}_k^I \geq \mathbf{S}^I \right\} \text{ and } \left\{ \mathbf{z}_h \in \mathbb{C}^h, | \, \widehat{y}_k^I < \mathbf{S}^I \right\} \tag{6.33}$$

The normal vector ($Q^R(\mathbf{h}^R, \mathbf{h}^I)$ to the real-part of the decision boundary ($\mathbf{S}^R$) and the normal vector ($Q^I(\mathbf{h}^R, \mathbf{h}^I)$ to the imaginary-part of the decision boundary ($\mathbf{S}^I$) are:

$$\begin{aligned} Q^R(\mathbf{z}_h^R, \mathbf{z}_h^I) &= \left(\frac{\partial \widehat{y}_k^R}{\partial z_h^{1^R}} \cdots \frac{\partial \widehat{y}_k^R}{\partial z_h^{h^R}} \frac{\partial \widehat{y}_k^R}{\partial z_h^{1^I}} \cdots \frac{\partial \widehat{y}_k^R}{\partial z_h^{h^I}} \right) \\ &= \left(v_{k1}^R \ \cdots \ v_{kh}^R \ - v_{k1}^I \ \cdots \ - v_{kh}^I \right) \end{aligned} \tag{6.34}$$

$$\begin{aligned} Q^I(\mathbf{z}_h^R, \mathbf{z}_h^I) &= \left(\frac{\partial \widehat{y}_k^I}{\partial z_h^{1^R}} \cdots \frac{\partial \widehat{y}_k^I}{\partial z_h^{h^R}} \frac{\partial \widehat{y}_k^I}{\partial z_h^{1^I}} \cdots \frac{\partial \widehat{y}_k^I}{\partial z_h^{h^I}} \right) \\ &= \left(v_{k1}^I \ \cdots \ v_{kh}^I \ v_{k1}^R \ \cdots \ v_{kh}^R \right) \end{aligned} \tag{6.35}$$

The decision boundaries ($\mathbf{S}^R$ and $\mathbf{S}^I$) of the k^{th} output neuron are orthogonal iff the dot product of their normal vectors is zero.

$$\begin{aligned} Q^R(\mathbf{z}_h^R, \mathbf{z}_h^I).Q^I(\mathbf{z}_h^R, \mathbf{z}_h^I) &= \begin{pmatrix} c_{k1}^R.v_{k1}^I + \cdots + v_{kh}^R.v_{kh}^I \\ -v_{k1}^I.v_{k1}^R + \cdots - v_{kh}^I.v_{kh}^R \end{pmatrix} && (6.36) \\ &= 0. && (6.37) \end{aligned}$$

From the above results, we can say that any output neuron in the CC-ELM has two decision boundaries with respect to the hidden neuron output and they are orthogonal to each other.

6.2.5 Case (ii): Orthogonality of Decision Boundaries in the Hidden Layer

The response of the j^{th} neuron in the hidden layer can be written as

$$z_{h_j} = z^R_{h_j} + iz^I_{h_j} = sech(O_j),$$

$$\text{where } O_j = \boldsymbol{\sigma}^T_j (\mathbf{z} - \mathbf{c}_j), \quad j = 1,2,\cdots,h \tag{6.38}$$

$$O_j = O^R_j + iO^I_j = \left(\boldsymbol{\sigma}^R_j + i\boldsymbol{\sigma}^I_j\right)^T \left[\left(\mathbf{z}^R - \mathbf{c}^R_j\right) + i\left(\mathbf{z}^I - \mathbf{c}^I_j\right)\right] \tag{6.39}$$

$$= \left[\boldsymbol{\sigma}^R_j \left(\mathbf{z}^R - \mathbf{c}^R_j\right) - \boldsymbol{\sigma}^I_j \left(\mathbf{z}^I - \mathbf{c}^I_j\right)\right] + i\left[\boldsymbol{\sigma}^R_j \left(\mathbf{z}^I - \mathbf{c}^I_j\right) + \boldsymbol{\sigma}^I_j \left(\mathbf{z}^R - \mathbf{c}^R_j\right)\right] \tag{6.40}$$

Using trigonometric and hyperbolic trigonometric definitions, the *sech* function can be written as:

$$sech(O^R_j + iO^I_j) = \frac{2\left(\cos(O^I_j \cosh(O^R_j) - i\sin(O^I_j)\sinh(O^R_j)\right)}{\cos(2O^I_j) + \cosh(2O^I_j)} \tag{6.41}$$

The two decision boundaries formed by the real and imaginary parts of the j^{th} hidden neuron response with respect to the inputs are:

$$z^R_{h_j} = \frac{2\left(\cos(O^I_j \cosh(O^R_j)\right)}{\cos(2O^I_j) + \cosh(2O^I_j)} \rightarrow \mathbf{S}^R \tag{6.42}$$

$$z^I_{h_j} = \frac{2\left(-i\sin(O^I_j)\sinh(O^R_j)\right)}{\cos(2O^I_j) + \cosh(2O^I_j)} \rightarrow \mathbf{S}^I \tag{6.43}$$

Note that here the decision boundaries are shown with respect to the net input to the hidden neuron O_j, which is a linear function of actual input $\mathbf{z}$.

Hence, the complex-valued input $\mathbf{z} \in \mathbb{C}^m$ are classified into two decision regions with respect to the real and imaginary parts ($\mathbf{S}^R$ and $\mathbf{S}^I$)

$$\left\{\mathbf{z} \in \mathbb{C}^m, |\ z^R_{h_j} \geq \mathbf{S}^R\right\} \text{ and } \left\{\mathbf{z} \in \mathbb{C}^m, |\ z^R_{h_j} < \mathbf{S}^R\right\} \tag{6.44}$$

$$\left\{\mathbf{z} \in \mathbb{C}^m, |\ z^I_{h_j} \geq \mathbf{S}^I\right\} \text{ and } \left\{\mathbf{z} \in \mathbb{C}^m, |\ z^I_{h_j} < \mathbf{S}^I\right\} \tag{6.45}$$

The normal vectors to the decision boundaries formed by the hidden neuron (given by Eqs. (6.42) and (6.43)) with respect to the input ($\mathbf{z}$) are given by:

$$Q^R_h(\mathbf{z}^R, \mathbf{z}^I) = \left(\frac{\partial z^R_{h_j}}{\partial z^R_1} \cdots \frac{\partial z^R_{h_j}}{\partial z^R_k} \cdots \frac{\partial z^R_{h_j}}{\partial z^R_m} \frac{\partial z^R_{h_j}}{\partial z^I_1} \cdots \frac{\partial z^R_{h_j}}{\partial z^I_k} \cdots \frac{\partial z^R_{h_j}}{\partial z^I_m}\right) \tag{6.46}$$

$$Q^I_h(\mathbf{z}^R, \mathbf{z}^I) = \left(\frac{\partial z^I_{h_j}}{\partial z^R_1} \cdots \frac{\partial z^I_{h_j}}{\partial z^R_k} \cdots \frac{\partial z^I_{h_j}}{\partial z^R_m} \frac{\partial z^I_{h_j}}{\partial z^I_1} \cdots \frac{\partial z^I_{h_j}}{\partial z^I_k} \cdots \frac{\partial z^I_{h_j}}{\partial z^I_m}\right) \tag{6.47}$$

The decision boundaries are orthogonal to each other iff the dot product of these normal vectors is zero. The dot product of the normal vectors is given by:

$$Q_h^R(\mathbf{z}^R,\mathbf{z}^I).Q_h^I(\mathbf{z}^R,\mathbf{z}^I) = \sum_{k=1}^{m}\left(\frac{\partial z_{h_j}^R}{\partial z_k^R}.\frac{\partial z_{h_j}^I}{\partial z_k^R}\right)+\left(\frac{\partial z_{h_j}^R}{\partial z_k^I}.\frac{\partial z_{h_j}^I}{\partial z_k^I}\right) \tag{6.48}$$

$$\frac{\partial z_h^{j^R}}{\partial z_k^R} = \frac{\partial z_h^{j^R}}{\partial O_j^R}\frac{\partial O_j^R}{\partial z_k^R}+\frac{\partial z_h^{j^R}}{\partial O_j^I}\frac{\partial O_j^I}{\partial z_k^R} = \frac{\partial z_h^{j^R}}{\partial O_j^R}\sigma_{jk}^R+\frac{\partial z_h^{j^R}}{\partial O_j^I}\sigma_{jk}^I \tag{6.49}$$

$$\frac{\partial z_h^{j^I}}{\partial z_k^R} = \frac{\partial z_h^{j^I}}{\partial O_j^R}\frac{\partial O_j^R}{\partial z_k^R}+\frac{\partial z_h^{j^I}}{\partial O_j^I}\frac{\partial O_j^I}{\partial z_k^R} = \frac{\partial z_h^{j^I}}{\partial O_j^R}\sigma_{jk}^R+\frac{\partial z_h^{j^I}}{\partial O_j^I}\sigma_{jk}^I \tag{6.50}$$

$$\frac{\partial z_h^{j^R}}{\partial z_k^I} = \frac{\partial z_h^{j^R}}{\partial O_j^R}\frac{\partial O_j^R}{\partial z_k^I}+\frac{\partial z_h^{j^R}}{\partial O_j^I}\frac{\partial O_j^I}{\partial z_k^I} = \frac{\partial z_h^{j^R}}{\partial O_j^R}(-\sigma_{jk}^I)+\frac{\partial z_h^{j^R}}{\partial O_j^I}\sigma_{jk}^R \tag{6.51}$$

$$\frac{\partial z_h^{j^I}}{\partial z_k^I} = \frac{\partial z_h^{j^I}}{\partial O_j^R}\frac{\partial O_j^R}{\partial z_k^I}+\frac{\partial z_h^{j^I}}{\partial O_j^I}\frac{\partial O_j^I}{\partial z_k^I} = \frac{\partial z_h^{j^I}}{\partial O_j^R}(-\sigma_{jk}^I)+\frac{\partial z_h^{j^I}}{\partial O_j^I}\sigma_{jk}^R \tag{6.52}$$

Using the laws of differentiation, the derivative of the j^{th} hidden layer response with respect to its net input

$$\frac{\partial h_j^R}{\partial O_j^R} = \frac{sinh(O_j^R)cos(3O_j^I)-sinh(3O_j^R)cos(O_j^I)}{\left(cos(2O_j^I)+cosh(2O_j^R)\right)^2}; \tag{6.53}$$

$$\frac{\partial h_j^I}{\partial O_j^I} = \frac{sinh(O_j^R)cos(3O_j^I)-sinh(3O_j^R)cos(O_j^I)}{\left(cos(2O_j^I)+cosh(2O_j^R)\right)^2} \tag{6.54}$$

$$\frac{\partial h_j^R}{\partial O_j^I} = \frac{cosh(O_j^R)sin(3O_j^I)-cosh(3O_j^R)sin(O_j^I)}{\left(cos(2O_j^I)+cosh(2O_j^R)\right)^2} \tag{6.55}$$

$$\frac{\partial h_j^I}{\partial O_j^R} = \frac{cosh(3O_j^R)sin(O_j^I)-cosh(O_j^R)sin(3O_j^I)}{\left(cos(2O_j^I)+cosh(2O_j^R)\right)^2} \tag{6.56}$$

From Eqs. (6.53)-(6.56), it can be observed that

$$\frac{\partial z_h^{j^R}}{\partial O_j^R} = \frac{\partial z_h^{j^I}}{\partial O_j^I} \tag{6.57}$$

$$\text{and } \frac{\partial z_h^{j^R}}{\partial O_j^I} = -\frac{\partial z_h^{j^I}}{\partial O_j^R} \tag{6.58}$$

Hence, it is evident that the *sech* activation function satisfies the Cauchy Riemann equations.

Substituting the Cauchy Riemann Equations in Eqs. (6.49)-(6.52), and obtaining the dot product of the normal vectors, we have,

$$Q_h^R(\mathbf{z}^R,\mathbf{z}^I).Q_h^I(\mathbf{z}^R,\mathbf{z}^I) = \sum_{k=1}^{m} \begin{bmatrix} -\frac{\partial z_h^{j^R}}{\partial O_j^R}\frac{\partial z_h^{j^R}}{\partial O_j^I}\left(\sigma_{jk}^R\right)^2 - \left(\frac{\partial z_h^{j^R}}{\partial O_j^I}\right)^2 \sigma_{jk}^R\sigma_{jk}^I \\ +\left(\frac{\partial z_h^{j^R}}{\partial O_j^R}\right)^2 \sigma_{jk}^R\sigma_{jk}^I + \frac{\partial z_h^{j^R}}{\partial O_j^R}\frac{\partial z_h^{j^R}}{\partial O_j^I}\left(\sigma_{jk}^I\right)^2 \\ -\frac{\partial z_h^{j^R}}{\partial O_j^R}\frac{\partial z_h^{j^R}}{\partial O_j^I}\left(\sigma_{jk}^I\right)^2 + \left(\frac{\partial z_h^{j^R}}{\partial O_j^I}\right)^2 \sigma_{jk}^R\sigma_{jk}^I \\ -\left(\frac{\partial z_h^{j^R}}{\partial O_j^R}\right)^2 \sigma_{jk}^R\sigma_{jk}^I + \frac{\partial z_h^{j^R}}{\partial O_j^R}\frac{\partial z_h^{j^R}}{\partial O_j^I}\left(\sigma_{jk}^R\right)^2 \end{bmatrix} \tag{6.59}$$

$$= 0. \tag{6.60}$$

Thus, the decision boundaries formed by the real and imaginary parts of the hidden layer output are orthogonal. It is also clear that this fact is also valid for any fully complex-valued activation function that satisfies the Cauchy Riemann equations.

Based on the above results, we state the following lemma:

Lemma 6.1. *The decision boundaries formed by the real and imaginary parts of an output/hidden neuron in a fully complex-valued network with any fully complex-valued activation function that satisfies the Cauchy Riemann conditions are orthogonal to each other.*

6.3 Summary

In this chapter, the various complex-valued classifiers available in the literature have been discussed and a fast learning circular complex-valued extreme learning machine classifier was described in detail. CC-ELM classifier uses a single hidden layer network with a non-linear input/hidden layer and a linear output layer. At the input layer, a unique nonlinear circular transformation is used as the activation function to make an one-to-one mapping of the real-valued input features to the Complex domain. At the hidden layer, the complex-valued input features are mapped on to a higher dimensional Complex plane using the '*sech*' activation function. In CC-ELM, the input parameters and the parameters of the hidden layer are chosen

randomly and the output weights are calculated analytically, requiring lesser computational effort to perform classification tasks. The presence of orthogonal decision boundaries in CC-ELM are proved.

References

1. Nitta, T.: The Computational Power of Complex-Valued Neuron. In: Kaynak, O., Alpaydın, E., Oja, E., Xu, L. (eds.) ICANN 2003 and ICONIP 2003. LNCS, vol. 2714, pp. 993–1000. Springer, Heidelberg (2003)
2. Nitta, T.: Solving the xor problem and the detection of symmetry using a single complex-valued neuron. Neural Networks 16(8), 1101–1105 (2003)
3. Nitta, T.: Orthogonality of decision boundaries of complex-valued neural networks. Neural Computation 16(1), 73–97 (2004)
4. Aizenberg, I., Moraga, C.: Multilayer feedforward neural network based on multi-valued neurons (MLMVN) and a backpropagation learning algorithm. Soft Computing 11(2), 169–183 (2007)
5. Amin, M.F., Murase, K.: Single-layered complex-valued neural network for real-valued classification problems. Neurocomputing 72(4-6), 945–955 (2009)
6. Kim, T., Adali, T.: Fully complex multi-layer perceptron network for nonlinear signal processing. Journal of VLSI Signal Processing 32(1/2), 29–43 (2002)
7. Zhang, T.: Statistical behavior and consistency of classification methods based on convex risk minimization. Ann. Stat. 32(1), 56–85 (2003)
8. Suresh, S., Sundararajan, N., Saratchandran, P.: Risk-sensitive loss functions for sparse multi-category classification problems. Information Sciences 178(12), 2621–2638 (2008)
9. Savitha, R., Suresh, S., Sundararajan, N.: A fully complex-valued radial basis function network and its learning algorithm. International Journal of Neural Systems 19(4), 253–267 (2009)
10. Ortega, J.M.: Matrix Theory. Plenum Press, New York (1986)
11. Huang, G.B., Li, M.B., Chen, L., Siew, C.K.: Incremental extreme learning machine with fully complex hidden nodes. Neurocomputing 71(4-6), 576–583 (2008)
12. Suresh, S., Omkar, S.N., Mani, V., Prakash, T.N.G.: Lift coefficient prediction at high angle of attack using recurrent neural network. Aerospace Science and Technology 7(8), 595–602 (2003)