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

IJCNN

A Projection Based Learning in Meta-cognitive Radial Basis Function Network for Classification Problems

G. Sateesh Babu, R. Savitha, and S. Suresh

***Abstract*—In this paper, we propose a 'Meta-cognitive Radial Basis Function Network (McRBFN)' and its 'Projection Based Learning (PBL)' algorithm for classification problems. McRBFN emulates human-like meta-cognitive learning principles. As each sample is presented to the network, McRBFN uses the estimated class label, the maximum hinge error and class-wise significance to address the self-regulating principles of *what-to-learn*, *when-to-learn* and *how-to-learn* in a meta-cognitive framework. McRBFN addresses the *what-to-learn* by choosing samples to participate in the learning process, also deleting samples with information similar to that already learnt by the network. A few samples that satisfy neither of these criteria are pushed to the rear end of the training data stack to be used in future, thereby satisfying the *when-to-learn*. The *how-to-learn* component of meta-cognition is addressed by using the participating samples to either add a neuron or update the output weights. Initially, McRBFN begins with zero hidden neurons and adds required number of neurons to approximate the decision surface. When a neuron is added, its parameters are initialized based on the sample overlapping conditions. The output weights are updated using a PBL algorithm such that the network finds the minimum point of an energy function defined by the hinge-loss error. The use of human meta-cognitive principles ensures efficient learning. Moreover, as samples with similar information are deleted, overtraining is avoided. The PBL algorithm helps to reduce the computational effort used in training.**

The performance of the PBL-McRBFN classifier is evaluated using a set of benchmark classification problems from the UCI machine learning repository. The performance evaluation study on these problems clearly indicates the superior performance of PBL-McRBFN classifier over results reported in the literature.

I. Introduction

Neural networks are powerful tools that can be trained to model complex input-output relationships. The learning algorithms to train a neural networks can be broadly classified as: (1) algorithms based on gradient-descent method with the network architecture fixed a priori (e.g. Classical back propagation algorithm [1] for multi-layer feed forward neural networks), (2) algorithms based on linear least-squares methods with the network architecture and the input parameters fixed a priori (e.g. Extreme Learning Machine (ELM) [2] for the generalized single-hidden layer feed forward networks), (3) sequential learning algorithms with an evolving architecture and network parameters (e.g. minimal resource allocation network [3], growing and pruning radial basis function network [4] etc). Neural networks using such algorithms are used to emulate several tasks of human activity, including classification tasks. In a classification task, an object is assigned to a predefined group or class based on a set of object attributes. A brief survey of the different neural network classifiers has been presented in [5]. Some of the recently developed classifiers include the Sequential Multi Category Radial Basis Function network (SMC-RBF) [6], and the Optimized Extreme Learning Machine Classifier (O-ELM) [7]. SMC-RBF uses similarity measures within class, misclassification rate and prediction error are used in neuron growing and parameter update criterion. Whereas, O-ELM chooses input weights randomly with fixed number of hidden neurons and analytically determines the output weights using minimum norm least squares.

Aforementioned neural network algorithms use all the samples in the training data set to gain knowledge about the information contained in the samples. In other words, they possess information-processing abilities of humans, including perception, learning, remembering, judging, and problem-solving, and these abilities are cognitive in nature. However, recent studies [8], [9], [10] on human learning has revealed that the learning process is effective when the learners adopt self-regulation in learning process using meta-cognition. Meta-cognition means cognition about cognition. In a meta-cognitive framework, human-beings think about their cognitive processes, develop new strategies to improve their cognitive skills and evaluate the information contained in their memory. Hence, there is a need to develop a meta-cognitive radial basis function network that analyzes its cognitive processes and chooses suitable strategies to improve its cognitive skills. Such a neural network must be capable of deciding *what-to-learn*, *when-to-learn* and *how-to-learn* the decision function from the training data.

Self-adaptive Resource Allocation Network (SRAN) [11] and Complex-valued Self-regulating Resource Allocation Network (CSRAN) [12] address the *what-to-learn* component of meta-cognition by selecting significant samples using misclassification error and hinge loss function. It has been shown in SRAN and CSRAN, that the selecting appropriate samples for learning and removing repetitive samples helps in improving the generalization performance. In literature, Meta-cognitive Neural Network (McNN) [13], Meta-cognitive Fully Complex-valued Radial Basis Function (Mc-FCRBF) network [14] and Meta-cognitive neuro-Fuzzy Inference System (McFIS) [15] address the three components of meta-cognition. However, Mc-FCRBF updates the network parameters using the gradient descent based algorithm and McNN, McFIS update the network parameters using extended kalman filter algorithm which increases computational burden for large networks. Therefore, in this paper, we

Giduthuri Sateesh Babu, R. Savitha and Sundaram Suresh are with the School of Computer Engineering, Nanyang Technological University, Singapore. Email: (sateesh1, ravi0001, ssundaram)@ntu.edu.sg

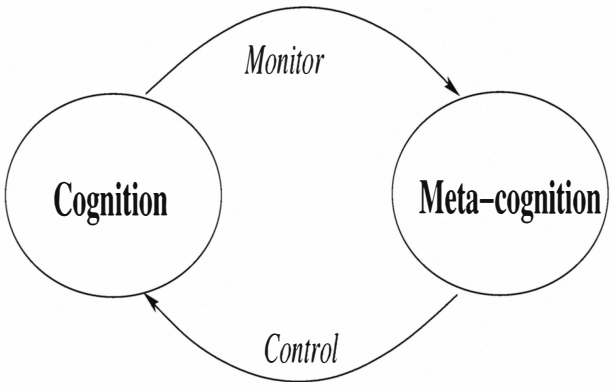

Fig. 1. Nelson and Narens Model of Meta-cognition

introduce a Meta-cognitive Radial Basis Function Network (McRBFN) that addresses the three components of meta-cognition with least computational effort simultaneously.

There are several models of meta-cognition and a brief survey of the various meta-cognition models has presented in [16]. McRBFN is developed based on the model of meta-cognition proposed by Nelson & Narens [17] as shown in Fig. 1. The model is analogous to the meta-cognition in human-beings and has two components, a cognitive component and a meta-cognitive component. The information flow from the cognitive component to meta-cognitive component is considered monitoring, while the information flow in the reverse direction is considered control. Monitoring informs the meta-cognitive component about the state of cognitive component, thus continuously updating the meta-cognitive component's model of cognitive component, including, 'no change in state'. The information flowing from the meta-cognitive component to the cognitive component either changes the state of the cognitive component or changes the cognitive component itself.

Analogous to the Nelson & Narens model of meta-cognition [17], McRBFN has two components namely the cognitive component and the meta-cognitive component. The cognitive component of McRBFN is a single hidden layer radial basis function network. The meta-cognitive component of McRBFN has a dynamic model of this cognitive component during the entire learning process and controls the learning process of the cognitive component by choosing one of the several strategies for each sample in the training data set. When a sample is presented to McRBFN, the meta-cognitive component of McRBFN measures the knowledge contained in the current training sample with respect to the cognitive component using estimated class label, maximum hinge error and class-wise significance. Using this information, McRBFN uses a 'Projection Based Learning (PBL)' algorithm to add a neuron or update the output parameters of the cognitive component (*how-to-learn*), or deletes the sample (*what to learn*), or reserves the sample for future use (*when-to-learn*). Thus, it addresses the three components of meta-cognition to ensure accurate approximation of the decision surface and achieves better generalization performance.

Unlike the existing batch learning algorithms that require the number of hidden neurons to be fixed a priori, the PBL begins with zero hidden neurons and adds neurons during the learning process to obtain an optimum network structure. When a neuron is added to the cognitive component, the input/hidden layer parameters are fixed based on the input of the sample and the output weights are estimated by minimizing an energy function given by the hinge-loss error function [18]. The problem of finding optimal weights is first formulated as a linear programming. The projection based learning algorithm then converts the linear programming problem into a system of linear equations and provides a solution for the optimal weights, corresponding to the minimum energy point of the energy function. The McRBFN using the PBL to obtain the network parameters is referred to as, 'Projection Based Learning algorithm for a Meta-cognitive Radial Basis Function Network (PBL-McRBFN)'.

The performance of PBL-McRBFN classifier is evaluated on a set of benchmark data sets from the University of California, Irvine machine learning repository [19]. In all these problems, the performance of McRBFN is compared against the best performing classifiers available in the literature for these problems. The performance study results show the superior classification ability of PBL-McRBFN.

The paper is structured as follows: In section 2, PBL-McRBFN for classification problems is described in detail, including a detailed description of the various strategies of the meta-cognitive component. Section 3 presents the performance evaluation of PBL-McRBFN classifier on a set of benchmark classification problems, in comparison with the best performing classifiers available in the literature. Section 4 summarizes the conclusions from this study.

II. Projection Based Learning in Meta-cognitive Radial Basis Function Network for Classification Problems

Given a training data set with N samples, $(\mathbf{x}^1, c^1), \cdots, (\mathbf{x}^t, c^t), \cdots, (\mathbf{x}^N, c^N)$, where $\mathbf{x}^t = [x_1^t, \cdots, x_m^t]^T \in \Re^m$ is the m-dimensional input of the t^{th} sample, and c^t is its class label. The coded class labels $\left(\mathbf{y}^t = \left[y_1^t, \cdots, y_j^t, \cdots, y_n^t\right]^T\right) \in \Re^n$ is given by:

$$y_j^t = \begin{cases} 1 & \text{if} \quad c^t = j \\ -1 & \text{otherwise} \end{cases} \quad j = 1, \cdots n; \; t = 1, \cdots N \quad (1)$$

where n is the total number of classes. The objective of McRBFN classifier is to approximate the underlying decision function that maps $\mathbf{x}^t \in \Re^m \rightarrow \mathbf{y}^t \in \Re^n$. McRBFN begins with zero hidden neurons and selects suitable strategy for each sample (adds neurons or update output parameters or deletes sample or reserves sample) to achieve this objective.

A. McRBFN Architecture

McRBFN has two components, namely the cognitive component and the meta-cognitive component, as shown in Fig. 2. The cognitive component of McRBFN consists of a three-layered feed forward radial basis function network. The

meta-cognitive component contains a dynamic model of the cognitive component. When a new training sample arrives, the meta-cognitive component of McRBFN predicts the class label and estimates the knowledge present in the new training sample with respect to the cognitive component. Based on this information, the meta-cognitive component controls the learning process of the cognitive component by selection suitable strategy for the current training sample to address *what-to-learn*, *when-to-learn* and *how-to-learn* properly.

We present a detailed description of the cognitive and the meta-cognitive components of McRBFN in the following sections:

1) Cognitive component of McRBFN: The cognitive component of McRBFN is a three layered feed forward radial basis function network with a linear input and output layers. The neurons in the hidden layer of the cognitive component of McRBFN employ the Gaussian activation function.

Without loss of generality, we assume that the McRBFN builds K Gaussian neurons from $t-1$ training samples. For a given input $\mathbf{x}^t$, the predicted output $\widehat{y}_j^t$ of McRBFN is

$$\widehat{y}_j^t = \sum_{k=1}^{K} w_{kj} h_k^t, \qquad j = 1, 2, \cdots, n \tag{2}$$

where w_{kj} is the weight connecting the k^{th} hidden neuron to the j^{th} output neuron and h_k^t is the response of the k^{th} hidden neuron to the input $\mathbf{x}^t$ is given by

$$h_k^t = exp\left(-\frac{\|\mathbf{x}^t - \boldsymbol{\mu}_k^l\|^2}{(\sigma_k^l)^2}\right) \tag{3}$$

where $\boldsymbol{\mu}_k^l \in \Re^m$ is the center and $\sigma_k^l \in \Re^+$ is the width of the k^{th} hidden neuron. Here, the superscript l represents the class of that hidden neuron to which it belongs.

The cognitive component uses Projection Based Learning (PBL) algorithm for learning process. The PBL algorithm is described as follows.

Projection Based Learning Algorithm: The projection based learning algorithm works on the principle of minimization of energy function and finds the network output parameters for which the energy function is minimum, i.e, the network achieves the minimum energy point of the energy function.

The considered energy function is the sum of squared errors at McRBFN output neurons

$$J(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{n}\left(y_j^i - \sum_{k=1}^{K} w_{kj} h_k^i\right)^2 \tag{4}$$

where h_k^i is the response of the k^{th} hidden neuron for i^{th} training sample.
The optimal output weights ($\mathbf{W}^* \in \Re^{K\times n}$) are estimated such that the total energy reaches its minimum.

$$\mathbf{W}^* = arg \min_{\mathbf{W}\in\Re^{K\times n}} J(\mathbf{W}) \tag{5}$$

The optimal $\mathbf{W}^*$ corresponding to the minimum energy point of the energy function ($J(\mathbf{W}^*)$) is obtained by equating the first order partial derivative of $J(\mathbf{W})$ with respect to the output weight to zero, i.e.,

$$\frac{\partial J(\mathbf{W})}{\partial w_{pj}} = 0, \quad p = 1, \cdots, K;\ j = 1, \cdots, n \tag{6}$$

After solving Eq. (6), we can obtain

$$\sum_{k=1}^{K}\sum_{i=1}^{t} h_k^i h_p^i w_{kj} = \sum_{i=1}^{t} h_p^i y_j^i \tag{7}$$

Eq. (7) can be written as

$$\sum_{k=1}^{K} a_{kp} w_{kj} = b_{pj},\ p = 1, \cdots, K;\ j = 1, \cdots, n \tag{8}$$

which can be represented in matrix form as

$$\mathbf{AW} = \mathbf{B} \tag{9}$$

where the projection matrix $\mathbf{A} \in \Re^{K\times K}$ is given by

$$a_{kp} = \sum_{i=1}^{t} h_k^i h_p^i, \quad k = 1, \cdots, K;\ p = 1, \cdots, K \tag{10}$$

and the output matrix $\mathbf{B} \in \Re^{K\times n}$ is

$$b_{pj} = \sum_{i=1}^{t} h_p^i y_j^i, \quad p = 1, \cdots, K;\ j = 1, \cdots, n \tag{11}$$

Eq. (8) gives the set of $K \times n$ linear equations with $K \times n$ unknown output weights $\mathbf{W}$. Note that the projection matrix is always a square matrix of order $K \times K$.

The solution for the system of equations in Eq. (9) can be determined as follows:

$$\mathbf{W}^* = \mathbf{A}^{-1}\mathbf{B} \tag{12}$$

2) Meta-cognitive component of McRBFN: The meta-cognitive component uses estimated class label ($\widehat{c}^t$), maximum hinge error (E^t) and spherical potential based class-wise significance as the measures of knowledge in the new training sample. Using these measures, the meta-cognitive component controls the learning process of the cognitive component.

Estimated Class label ($\widehat{c}^t$): Using the predicted output ($\widehat{\mathbf{y}}^t$), the estimated class label ($\widehat{c}^t$) can be obtained as

$$\widehat{c}^t = arg \max_{j\in 1,2,\cdots,n} \widehat{y}_j^t \tag{13}$$

Maximum Hinge Error (E^t): The objective of the classifier is to minimize the error between the predicted output ($\widehat{\mathbf{y}}^t$) and actual output ($\mathbf{y}^t$). In classification problems, it has been shown in [18], [20] that the classifier developed using hinge loss function estimates the posterior probability more accurately than the classifier developed using mean square error function. Hence, in McRBFN, we use the hinge loss error $\left(\mathbf{e^t} = \left[e_1^t, \cdots, e_j^t, \cdots, e_n^t\right]^T\right) \in \Re^n$ defined as

$$e_j^t = \begin{cases} 0 & \text{if } y_j^t \widehat{y}_j^t > 1 \\ y_j^t - \widehat{y}_j^t & \text{otherwise} \end{cases} \qquad j = 1, 2, \cdots, n \tag{14}$$

The maximum absolute hinge error (E^t) is given by

$$E^t = \max_{j \in 1,2,\cdots,n} \left| e_j^t \right| \tag{15}$$

Class-wise Significance (ψ_c): In general, the input feature ($\mathbf{x}^t$) is mapped on to a hyper-dimensional spherical feature space $\mathbb{S}$ using K Gaussian neurons, i.e., $\mathbf{x}^t \rightarrow \mathbf{H}$. Therefore, all $\mathbf{H}(\mathbf{x}^t)$ lie on a hyper-dimensional sphere as shown in [21]. The knowledge or spherical potential of any sample in original space is expressed as a squared distance from the hyper-dimensional mapping $\mathbb{S}$ centered at h_0 as shown in [22].

In McRBFN, the center ($\boldsymbol{\mu}$) and width (σ) of the Gaussian neurons describe the feature space $\mathbb{S}$. Let the center of the K-dimensional feature space be $h_0 = \frac{1}{K}\sum_{k=1}^{K} h(\boldsymbol{\mu}_k)$. The knowledge present in the new data $\mathbf{x}^t$ can be expressed as the potential of the data in the original space, which is squared distance from the K-dimensional feature space to the center h_0. The potential (ψ) is given as

$$\psi = ||h(\mathbf{x}^t) - h_0||^2 \tag{16}$$

Using the steps shown in [22], the above equation can be expressed as

$$\psi = h\left(\mathbf{x}^t, \mathbf{x}^t\right) - \frac{2}{K}\sum_{k=1}^{K} h\left(\mathbf{x}^t, \boldsymbol{\mu}_k^l\right) + \frac{1}{K^2}\sum_{k,r=1}^{K} h\left(\boldsymbol{\mu}_k^l, \boldsymbol{\mu}_r^l\right) \tag{17}$$

From the above equation, we can see that for Gaussian function the first term ($h\left(\mathbf{x}^t, \mathbf{x}^t\right)$) and last term ($\frac{1}{K^2}\sum_{k,r=1}^{K} h\left(\boldsymbol{\mu}_k^l, \boldsymbol{\mu}_r^l\right)$) are constants. Since potential is a measure of novelty, these constants may be discarded and the potential can be reduced to

$$\psi \approx -\frac{2}{K}\sum_{k=1}^{K} h\left(\mathbf{x}^t, \boldsymbol{\mu}_k^l\right) \tag{18}$$

Since we are addressing classification problems, the class-wise distribution plays a vital role and it will influence the performance the classifier significantly as shown in [6]. Hence, we use the measure of the spherical potential of the new training sample $\mathbf{x}^t$ belonging to class c with respect to the neurons associated to same class (i.e., $l = c$). Let K^c be the number of neurons associated with the class c, then class-wise spherical potential or class-wise significance (ψ_c) is defined as

$$\psi_c = \frac{1}{K^c}\sum_{k=1}^{K^c} h\left(\mathbf{x}^t, \boldsymbol{\mu}_k^c\right) \tag{19}$$

The spherical potential explicitly indicates the knowledge contained in the sample, a higher value of spherical potential (close to one) indicates that the sample is similar to the existing knowledge in the cognitive component and a smaller value of spherical potential (close to zero) indicates that the sample is novel.

B. Learning Strategies

Based on the meta-cognitive measures, the meta-cognitive component devices various learning strategies which directly addresses the basic principles of self-regulated human learning (i.e., *what-to-learn*, *when-to-learn* and *how-to-learn*). The meta-cognitive part controls the learning process in cognitive component by selecting one of the following four learning strategies.

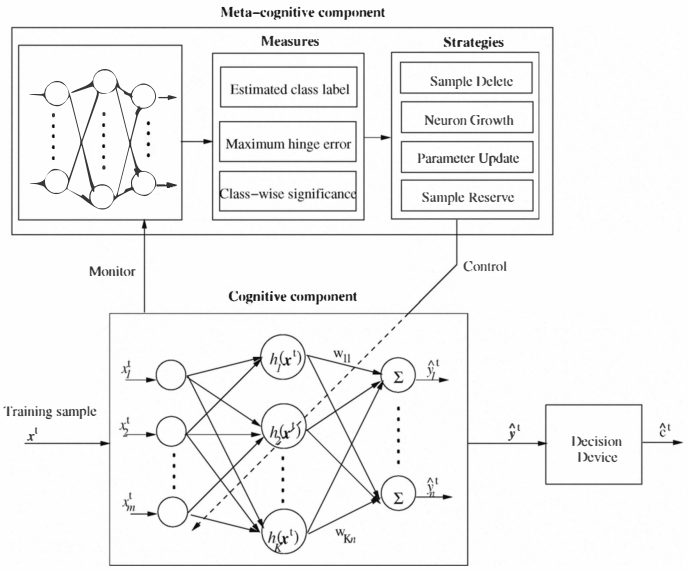

Fig. 2. Schematic diagram of McRBFN Classifier

- *Sample Delete Strategy*: If the new training sample contains information similar to the knowledge present in the cognitive component, then delete the new training sample from the training data set without using it in the learning process.
- *Neuron Growth Strategy*: Use the new training sample to add a new hidden neuron in the cognitive component.
- *Parameter Update Strategy*: The new training sample is used to update the parameters of the cognitive component.
- *Sample Reserve Strategy*: The new training sample contains some information but not significant, they can be used at later stage of the learning process for fine tuning the parameters of the cognitive component. These sample may be discarded without learning or used for fine tuning the cognitive component parameters in a later stage.

The principle behind these four learning strategies are described in detail below:

1) Sample Delete Strategy: When the predicted class label of the new training sample is same as the actual class label and the maximum hinge error is very small then the new training sample does not provide additional information to the classifier and can be deleted from training sequence without being used in learning process. The sample deletion criterion is given by

$$c^t == \widehat{c}^t \ \ \textbf{AND} \ \ E^t \leq \beta_d \tag{20}$$

The meta-cognitive deletion threshold (β_d) commands the number of samples participating in the learning process.

If one selects β_d close to 0 then all the training samples participates in the learning process which results in over-training with similar samples. Increasing β_d beyond the desired accuracy results in deletion of too many samples from the training sequence. But, the resultant network may not satisfy the desired accuracy. Hence, it is fixed at the expected accuracy level. β_d is selected in the interval [0.1 - 0.2]. The sample deletion strategy prevents learning of samples with similar information, and thereby, avoids over-training and reduces the computational effort.

2) Neuron Growth Strategy: When a new training sample contains significant information and the estimated class label is different from the actual class label then one need to add new hidden neuron to represent the knowledge contained in the sample. The neuron growth criterion is given by

$$\left(\widehat{c}^t \neq c^t \ \textbf{OR} \ E^t \geq \beta_a\right) \ \textbf{AND} \ \psi_c(\mathbf{x}^t) \leq \beta_c \quad (21)$$

where β_c is the meta-cognitive knowledge measurement threshold and β_a is the self-adaptive meta-cognitive addition threshold. The terms β_c and β_a selects samples with significant knowledge for learning first then uses the other samples for fine tuning. If β_c is chosen closer to zero and the initial value of β_a is chosen closer to the maximum value of hinge error, then very few neurons will be added to the network. Such a network will not approximate the function properly. If β_c is chosen closer to one and the initial value of β_a is chosen closer to the minimum value of hinge error, then the resultant network may contain many neurons with poor generalization ability. Hence, the β_c can be selected in the interval [0.3 - 0.7] and the initial value of β_a can be selected in the interval [1.3 - 1.7]. The β_a is adapted based on the prediction error as:

$$\beta_a \ := \ \delta\beta_a + (1-\delta)E^t \quad (22)$$

where δ is the slope that controls rate of self-adaptation and is set close to 1.

When a new hidden neuron $K+1$ is added, then its parameters are initialized using the overlapping and distinct cluster criterion. The new training sample may have overlap with other classes or will be from a distinct cluster far away from the nearest neuron in the same class. Therefore, the overlapping and condition affects the classification performance of a classifier significantly. However, the existing classifiers do not address this condition. Hence, McRBFN measures inter/intra class nearest neuron distances from the current sample in assigning the new neuron parameters.

Let nrS be the nearest hidden neuron in the intra-class and nrI be the nearest hidden neuron in the inter-class. They are defined as

$$nrS = arg \min_{l==c;\forall k} \|\mathbf{x}^t - \boldsymbol{\mu}_k^l\|; \quad nrI = arg \min_{l\neq c;\forall k} \|\mathbf{x}^t - \boldsymbol{\mu}_k^l\| \quad (23)$$

Let the Euclidian distances between the current sample and nearest neurons are given as follows

$$d_S = ||\mathbf{x}^t - \boldsymbol{\mu}_{nrS}^c||; \quad d_I = ||\mathbf{x}^t - \boldsymbol{\mu}_{nrI}^l|| \quad (24)$$

Using the nearest neuron distances, we can determine the overlapping/no-overlapping conditions as follows:

- *no-overlapping with any class*: when a new training sample is far away from both intra/inter class nearest neurons ($d_S >> \sigma_{nrS}^c$ **AND** $d_I >> \sigma_{nrI}^l$) then the new training sample does not overlap with any class cluster, and is from a distinct cluster. In this case, the new hidden neuron center ($\boldsymbol{\mu}_{K+1}^c$) and width (σ_{K+1}^c) parameters are determined as

$$\boldsymbol{\mu}_{K+1}^c = \mathbf{x}^t; \quad \sigma_{K+1}^c = \kappa\sqrt{\mathbf{x}^{t^T}\mathbf{x}^t} \quad (25)$$

  where κ is a positive constant which controls the overlap of the responses of the hidden units in the input space, which lies in the range $0.5 \leq \kappa \leq 1$.
- *no-overlapping with the inter-class*: When a new training sample is close to the intra-class nearest neuron then the sample does not overlap with the other classes, i.e., the intra/inter class distance ratio is less than 1, then the sample does not overlap with the other classes. In this case, the new hidden neuron center ($\boldsymbol{\mu}_{K+1}^c$) and width (σ_{K+1}^c) are determined as

$$\boldsymbol{\mu}_{K+1}^c = \mathbf{x}^t; \quad \sigma_{K+1}^c = \kappa\|\mathbf{x}^t - \boldsymbol{\mu}_{nrS}^c\| \quad (26)$$

- *Minimum Overlapping with the inter-class*: when a new training sample is close to the inter-class nearest neuron compared to the intra-class nearest neuron, i.e., the intra/inter class distance ratio is in range 1 to 1.5, then the sample has minimum overlapping with the other class. In this case, the center of the new hidden neuron is shifted away from the inter-class nearest neuron and shifted towards the intra-class nearest neuron, and is initialized as

$$\begin{aligned} \boldsymbol{\mu}_{K+1}^c &= \mathbf{x}^t + \zeta(\boldsymbol{\mu}_{nrS}^c - \boldsymbol{\mu}_{nrI}^l) \\ \sigma_{K+1}^c &= \kappa\|\boldsymbol{\mu}_{K+1}^c - \boldsymbol{\mu}_{nrS}^c\| \end{aligned} \quad (27)$$

  where ζ is center shift factor which determines how much center has to be shifted from the new training sample location. It lies in range [0.01-0.1].

The above mentioned center and width determination conditions helps in minimizing the misclassification in McRBFN classifier.

When a neuron is added to McRBFN, the output weights are estimated using the PBL as follows:

The size of the matrix $\mathbf{A}$ is increased from $K \times K$ to $(K+1) \times (K+1)$

$$\mathbf{A}_{(K+1)\times(K+1)} = \left[\begin{array}{c|c} \mathbf{A}_{K\times K} + \left(\mathbf{h}^t\right)^T \mathbf{h}^t & \mathbf{a}_{K+1}^T \\ \hline \mathbf{a}_{K+1} & a_{K+1,K+1} \end{array}\right] \quad (28)$$

where $\mathbf{h}^t = [h_1^t, h_2^t, \cdots, h_K^t]$ is a vector of the existing K hidden neurons response for current (t^{th}) training sample. $\mathbf{a}_{K+1} \in \Re^{1\times K}$ is assigned as

$$a_{K+1,p} = \sum_{i=1}^{t} h_{K+1}^i h_p^i, \ p = 1, \cdots, K \quad (29)$$

and $a_{K+1,K+1} \in Re^{+}$ value assigned as

$$a_{K+1,K+1} = \sum_{i=1}^{t} h_{K+1}^{i} h_{K+1}^{i} \tag{30}$$

The size of matrix $\mathbf{B}$ is increased from $K \times n$ to $(K+1) \times n$

$$\mathbf{B}_{(K+1)\times n} = \begin{bmatrix} \mathbf{B}_{K\times n} + \left(\mathbf{h}^t\right)^T \left(\mathbf{y}^t\right)^T \\ \mathbf{b}_{K+1} \end{bmatrix} \tag{31}$$

where $\mathbf{b}_{K+1} \in \Re^{1\times n}$ is a row vector assigned as

$$b_{K+1,j} = \sum_{i=1}^{t} h_{K+1}^{i} y_j^i, \;\; j = 1, \cdots, n \tag{32}$$

Finally the output weights are estimated as

$$\begin{bmatrix} \mathbf{W}_K \\ \mathbf{w}_{K+1} \end{bmatrix} = \left(\mathbf{A}_{(K+1)\times(K+1)}\right)^{-1} \mathbf{B}_{(K+1)\times n} \tag{33}$$

After calculating inverse of the matrix $\mathbf{A}_{(K+1)\times(K+1)}$ recursively using matrix identities, the resultant equations are

$$\mathbf{W}_K = \left[\mathbf{I}_{K\times K} + \frac{\left(\mathbf{A}_{K\times K}\right)^{-1} \mathbf{a}_{K+1}^T \mathbf{a}_{K+1}}{\Delta}\right]$$

$$\left[\mathbf{W}_K + \left(\mathbf{A}_{K\times K}\right)^{-1} \left(\mathbf{h}^t\right)^T \left(\mathbf{y}^t\right)^T\right] - \frac{\left(\mathbf{A}_{K\times K}\right)^{-1} \mathbf{a}_{K+1}^T \mathbf{b}_{K+1}}{\Delta} \tag{34}$$

$$\mathbf{w}_{K+1} = -\frac{\mathbf{a}_{K+1}\left(\mathbf{W}_K + \left(\mathbf{A}_{K\times K}\right)^{-1} \left(\mathbf{h}^t\right)^T \left(\mathbf{y}^t\right)^T\right)}{\Delta} + \frac{\mathbf{b}_{K+1}}{\Delta} \tag{35}$$

$$\Delta = a_{K+1,K+1} - \mathbf{a}_{K+1}\left(\mathbf{A}_{K\times K} + \left(\mathbf{h}^t\right)^T \mathbf{h}^t\right)^{-1} \mathbf{a}_{K+1}^T$$

3) Parameters Update Strategy: The current (t^{th}) training sample is used to update the output weights of the cognitive component ($\mathbf{W}_K = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K]^T$) if the following criterion is satisfied.

$$c^t == \widehat{c}^t \;\; \textbf{AND} \;\; E^t \geq \beta_u \tag{36}$$

where β_u is the self-adaptive meta-cognitive parameter update threshold. If β_u is chosen closer to 50% of maximum hinge 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. Hence, the initial value of β_u can be selected in the interval [0.4 - 0.7]. The β_u is adapted based on the prediction error as:

$$\beta_u := \delta\beta_u + (1-\delta)E^t \tag{37}$$

When a sample is used to update the output weight parameters, the PBL algorithm updates the output weight parameters as follows:

The matrices $\mathbf{A} \in \Re^{K\times K}$ and $\mathbf{B} \in \Re^{K\times n}$ are updated as

$$\mathbf{A} = \mathbf{A} + \left(\mathbf{h}^t\right)^T \mathbf{h}^t \tag{38}$$

$$\mathbf{B} = \mathbf{B} + \left(\mathbf{h}^t\right)^T \left(\mathbf{y}^t\right)^T \tag{39}$$

and the output weights are updated as

$$\mathbf{W}_K = \mathbf{W}_K + \mathbf{A}^{-1} \left(\mathbf{h}^t\right)^T \left(\mathbf{e}^t\right)^T \tag{40}$$

4) Sample Reserve Strategy: If the new training sample does not satisfy either the deletion or the neuron growth or the cognitive component parameters update criterion, then the sample is pushed to the rear of the training sequence. Since McRBFN modifies the strategies based on current sample knowledge, these samples may be used in later stage.

C. PBL Algorithm for McRBFN Classifier

PBL algorithm for McRBFN Classifier can be summarized as below:

1) For each new training sample input ($\mathbf{x}^t$) compute the cognitive component output ($\widehat{\mathbf{y}}^t$) using Eqs. (2) & (3).
2) The meta-cognitive component finds the estimated class label ($\widehat{c}^t$), maximum hinge error (E^t) and class-wise significance (ψ_c) measures for the new training sample ($\mathbf{x}^t$) using Eqs.(13),(15) & (19).
3) The meta-cognitive component selects one of the following strategies based on above calculated measures.
   a) **Sample Delete Strategy**: If $c^t == \widehat{c}^t$ **AND** $E^t \leq \beta_d$ then delete the sample from the training data set with-out learning.
   b) **Neuron Growth Strategy**: If $(\widehat{c}^t \neq c^t$ **OR** $E^t \geq \beta_a)$ **AND** $\psi_c(\mathbf{x}^t) \leq \beta_c$, then allocate a new hidden neuron in the cognitive component. New hidden neuron width and center parameters are determined based on the intra and inter class nearest neurons distances using Eqs. from (25) to (27). Output weight parameters for all hidden neurons are estimated based on PBL algorithm using Eq. (33). Also, update the self-adaptive meta-cognitive addition threshold using Eq. (22).
   c) **Parameters Update Strategy**: If $c^t == \widehat{c}^t$ **AND** $E^t \geq \beta_u$, then update the cognitive component output weight parameters based on PBL algorithm using Eq. (40). Also, update the self-adaptive meta-cognitive update threshold using Eq. (37).
   d) **Sample Reserve Strategy**: When the new sample does not satisfy deletion, growth and update criterion, then push the sample to the reserve to be used later for learning.
4) The cognitive component executes the above selected strategy.
5) Continue steps 1 to 4 until there are no more samples in the training data set.

In McRBFN, sample delete strategy address the *what-to-learn* by deleting insignificant samples from training data set, neuron growth strategy and parameters update strategy address the *how-to-learn* by which the cognitive component learns from the samples, and self-adaptive nature of meta-cognitive thresholds in addition to the sample reserve strategy address the *when-to-learn* by presenting the samples in the learning process according to the knowledge present in the sample.

III. Performance results

The performance of PBL-McRBFN classifier is evaluated using benchmark binary and multi-category classification problems from the UCI machine learning repository. The performance is compared with the best performing learning algorithms reported in the literature, viz., SRAN, ELM classifier and also with the standard support vector machines. The data sets are chosen with varying sample imbalance. The sample imbalance is measured using Imbalance Factor (I.F) as

$$\text{I.F} \quad = \quad 1 - \frac{n}{N} \times \min_{j=1\cdots n} N_j \tag{41}$$

Note that $N = \sum_{j=1}^{n} N_j$, where N_j is the total number of training samples belonging to the class j.

TABLE I
Description of benchmark data sets from UCI machine learning repository

Problem	No. of features	No. of classes	No. of Samples		I.F	
			Training	Testing	Training	Testing
Image - Segmentation (IS)	19	7	210	2100	0	0
Iris	4	3	45	105	0	0
Wine	13	3	60	118	0	0.29
Vehicle - Classification (VC)	18	4	424	422	0.1	0.12
Glass - Identification (GI)	9	6	336	105	0.68	0.77
Heart	13	2	70	200	0.14	0.1
Liver - Disorders (LD)	6	2	200	145	0.17	0.14
Breast - Cancer (BC)	9	2	300	383	0.26	0.33
Ionosphere (ION)	34	2	100	251	0.28	0.28

The description of these data sets including the number of classes, the number of input features, the number of samples in the training/testing and the imbalance factor are presented in Table I. From Table I, it can be observed that the problems chosen for the study have both balanced and unbalanced data sets and the imbalance factors of the data sets vary widely.

All the simulations are conducted in MATLAB 2010 environment on a desktop PC with Intel Core 2 Duo, 2.66 GHz CPU and 3 GB RAM. In ELM simulations, the number of hidden neurons are obtained using the constructive-destructive procedure as presented in [23]. The simulations for batch SVM are carried out using the LIBSVM package in C [24]. In SVM simulations, the parameters (c,γ) are optimized using grid search technique. The performance measures used to compare the classifiers are described below.

A. Performance Measures

The class-wise performance measures like overall/average efficiencies are used for performance comparison. The confusion matrix Q is used to obtain the class-level performance and global performance of the various classifiers. Class-level performance is measured by the percentage classification (η_i) which is defined as:

$$\eta_j = \frac{q_{jj}}{N_j} \times 100\% \tag{42}$$

where q_{jj} is the total number of correctly classified samples in the class j and N_j is the total number of samples belonging to a class j in the training/testing data set. The global measures used in the evaluation are the average per-class classification accuracy (η_a) and the over-all classification accuracy (η_o) defined as:

$$\eta_a \quad = \quad \frac{1}{n}\sum_{j=1}^{n} \eta_j \tag{43}$$

$$\eta_o \quad = \quad \frac{\sum_{j=1}^{n} q_{jj}}{\sum_{j=1}^{n} N_j} \times 100\% \tag{44}$$

B. Performance Evaluation on UCI Benchmark data sets

The number of hidden neurons and the average/overall testing efficiencies for PBL-McRBFN, SRAN, ELM and SVM classifiers for the benchmark classification problems described in Table I are reported in Table II. From the table, we can see that PBL-McRBFN classifier performs slightly better than the best performing SRAN classifier on well-balanced data sets (approximately $1-2\%$ improvement), where as it performs significantly better than SRAN on unbalanced data sets (approximately $2-10$ % improvement). From the table, we can also see that the proposed PBL-McRBFN classifier performs significantly better than ELM and SVM classifiers on all the 9 data sets. Hence, we can say that PBL-McRBFN classifier performs better than the existing classifiers on these data sets.

TABLE II
Performance comparison on benchmark data sets

Data sets	PBL-McRBFN			SRAN			ELM			SVM		
	K	Testing		K	Testing		K	Testing		SV^a	Testing	
		η_o	η_a		η_o	η_a		η_o	η_a		η_o	η_a
IS	48	**94.19**	**94.19**	48	93	93	49	90.23	90.23	127	91.38	91.38
Iris	7	**97.14**	**97.14**	8	96.19	96.19	10	96.19	96.19	13	96.19	96.19
Wine	10	**98.3**	**98.69**	12	96.61	97.18	10	97.46	98.04	36	97.46	98.04
VC	161	**79.15**	**79.66**	113	75.12	76.86	150	77.01	77.59	340	70.62	68.51
GI	72	85.71	**91.63**	59	**86.21**	80.95	80	81.31	87.43	183	70.47	75.61
Heart	26	**80.5**	**80.5**	28	78.50	77.53	36	76.50	75.91	42	75.50	75.10
LD	87	71.03	**72.66**	91	66.90	65.78	100	**72.41**	71.41	141	71.03	70.21
BC	12	**97.39**	**97.85**	7	96.87	97.26	66	96.35	96.48	24	96.61	97.06
ION	15	**96.02**	**95.67**	21	90.84	91.88	32	89.64	87.52	43	91.24	88.51

[a] Support vectors

IV. Conclusions

In this paper, we have presented a Meta-cognitive Radial Basis Function Network (McRBFN) using a Projection Based Learning (PBL) algorithm for classification problems. The meta-cognitive component in McRBFN controls the learning of the cognitive component in McRBFN. The meta-cognitive component adapts the learning process appropriately and hence it decides *what-to-learn*, *when-to-learn* and *how-to-learn* efficiently. In addition, the overlapping conditions present in neuron growth strategy helps in proper initialization of new hidden neuron parameters and also minimizes the misclassification error. The PBL algorithm helps to reduce the computational effort used in training. The performance of the proposed PBL-McRBFN classifier has been evaluated using the benchmark multi-category, binary classification problems from the UCI data base with wide range of imbalance factor. The performance comparison with the well-known classifiers in the literature clearly indicates the superior performance of the proposed PBL-McRBFN classifier.

Acknowledgement

The authors would like to thank the MOE, Singapore for their financial support through the Academic Research Funding (AcRF) Tier I (No. M58020020) grant.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagation errors," *Nature*, vol. 323, pp. 533–536, 1986.

[2] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *IEEE International Joint Conference on Neural Networks. Proceedings*, vol. 2, 2004, pp. 985–990.

[3] L. Yingwei, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Computation*, vol. 9, no. 2, pp. 461–478, 1997.

[4] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, vol. 34, no. 6, pp. 2284–2292, 2004.

[5] G. B. Zhang, "Neural network for classification: A survey," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 30, no. 4, pp. 451–462, 2000.

[6] S. Suresh, N. Sundararajan, and P. Saratchandran, "A sequential multi-category classifier using radial basis function networks," *Neurocomputing*, vol. 71, no. 1, pp. 1345–1358, 2008.

[7] G.-B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1-3, pp. 155–163, 2010.

[8] A. L. Wenden, "Metacognitive knowledge and language learning," *Applied Linguistics*, vol. 19, no. 4, pp. 515–537, 1998.

[9] W. P. Rivers, "Autonomy at All costs: An Ethnography of Metacognitive Self-Assessment and Self-management among Experienced Language Learners," *The Modern Language Journal*, vol. 85, no. 2, pp. 279–290, 2001.

[10] R. Isaacson and F. Fujita, "Metacognitive knowledge monitoring and self-regulated learning: Academic success and reflections on learning," *Journal of the Scholarship of Teaching and Learning*, vol. 6, no. 1, pp. 39–55, 2006.

[11] S. Suresh, K. Dong, and H. J. Kim, "A sequential learning algorithm for self-adaptive resource allocation network classifier," *Neurocomputing*, vol. 73, no. 16–18, pp. 3012–3019, 2010.

[12] S. Suresh, R. Savitha, and N. Sundararajan, "A Sequential Learning Algorithm for Complex-valued Self-regulating Resource Allocation Network-CSRAN," *IEEE Transactions on Neural Networks*, vol. 22, no. 7, pp. 1061–1072, 2011.

[13] G. Sateesh Babu and S. Suresh, "Meta-cognitive Neural Network for classification problems in a sequential learning framework," *Neurocomputing*, vol. 81, pp. 86–96, 2012.

[14] R. Savitha, S. Suresh, and N. Sundararajan, "Metacognitive learning in a Fully Complex-valued Radial Basis Function Neural Network," *Neural Computation*, vol. 24, no. 5, pp. 1297–1328, 2012.

[15] S. Suresh and K. Subramanian, "A sequential learning algorithm for meta-cognitive neuro-fuzzy inference system for classification problems," in *The International Joint Conference on Neural Networks (IJCNN)*, 2011, pp. 2507–2512.

[16] M. T. Cox, "Metacognition in computation: A selected research review," *Artificial Intelligence*, vol. 169, no. 2, pp. 104–141, 2005.

[17] T. O. Nelson and L. Narens, *Metamemory: A theoritical framework and new findings*. Boston: Allyn and Bacon, 1992.

[18] S. Suresh, N. Sundararajan, and P. Saratchandran, "Risk-sensitive loss functions for sparse multi-category classification problems," *Information Sciences*, vol. 178, no. 12, pp. 2621–2638, 2008.

[19] C. M. C. Blake, *UCI repository of machine learning databases*. University of California, Irvine, Department of Information and Computer Sciences, 1998, http://archive.ics.uci.edu/ml/.

[20] T. Zhang, "Statistical behavior and consistency of classification methods based on convex risk minimization," *Annals of Statistics*, vol. 32, no. 1, pp. 56–85, 2004.

[21] B. Scholkopf and A. J. Smola, *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[22] H. Hoffmann, "Kernel PCA for Novelty Detection," *Pattern Recognition*, vol. 40, no. 3, pp. 863–874, 2007.

[23] S. Suresh, S. Omkar, V. Mani, and T. G. Prakash, "Lift coefficient prediction at high angle of attack using recurrent neural network," *Aerospace Science and Technology*, vol. 7, no. 8, pp. 595–602, 2003.

[24] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, http://www.csie.ntu.edu.tw/~cjlin/libsvm.