A Sequential Projection Based Meta-cognitive Learning in RBF Network for Classification Problems

G. Sateesh Babu, Student Member, IEEE, S. Suresh, Senior Member, IEEE

Abstract—In this paper, we present a sequential projection based meta-cognitive learning algorithm in radial basis function network for classification problems, referred as PBL-McRBFN. The algorithm is inspired by human meta-cognitive learning principles and has two components namely a cognitive component and a meta-cognitive component. The cognitive component is a single hidden layer radial basis function network with evolving architecture. The meta-cognitive component controls the learning process in the cognitive component by choosing the best learning strategy for the current sample and adapts the learning strategies by implementing self-regulation. In addition, sample overlapping conditions and past knowledge of the samples in the form of pseudo samples are used for proper initialization of new hidden neurons, to minimize the misclassification. The parameter update strategy uses projection based direct minimization of hinge loss error. The interaction of the cognitive component and the meta-cognitive component addresses the what-to-learn, when-to-learn and how-to-learn human learning principles efficiently. The performance of the PBL-McRBFN is evaluated using a set of benchmark classification problems from UCI machine learning repository. The statistical performance evaluation on these problems has proven the superior performance of PBL-McRBFN classifier over results reported in the literature. Also, we evaluated the performance of the proposed algorithm on a practical Alzheimer’s disease detection problem. The performance results on open access series of imaging studies and Alzheimer’s disease neuroimaging initiative data sets which are obtained from different demographic regions clearly show that PBL-McRBFN can handle a problem with change in distribution.

Index Terms—Meta-cognitive learning, radial basis function network classifier, projection based learning, self-regulatory thresholds, Alzheimer’s disease.

I. INTRODUCTION

In machine learning, batch learning algorithms require complete training data to build the model. In most of the practical applications, especially in medical diagnosis, the complete training data describing the input-output relationship is not available a priori. For these problems, classical batch-learning algorithms are rather infeasible and instead sequential learning is employed. In a sequential learning framework, the training samples arrive one-by-one and the samples are discarded after the learning process. Hence, it requires less memory and computational time during the learning process. In addition, sequential learning algorithms automatically determine the minimal architecture that can accurately approximate the true decision function described by stream of training samples [1].

Radial Basis Function (RBF) networks have been extensively used in a sequential learning framework due to their universal approximation ability and simplicity of architecture [2]. Recently, there has been renewed interests in single-hidden layered RBF networks with least square error training criterion, partly due to their modeling ability and partly due to the existence of efficient learning algorithms such as Extreme Learning Machine (ELM) [3], and second order training methods [4]. In recent years, researchers have been focussing on sequential learning algorithms for RBF networks using streams of data. A brief discussion of existing research work in a sequential learning framework is given below.

The first sequential learning algorithm introduced in the literature was Resource Allocation Network (RAN) [5]. RAN uses novelty based neuron growth to evolve the network architecture automatically and linear mean square algorithm to update the network parameters. Minimal Resource Allocation Network (MRAN) [6] adapted the similar approach, but it uses extended Kalman filter (EKF) for parameter update and pruning strategy to determine the compact network architecture. Growing and Pruning Radial Basis Function Network [7] uses the neuron significance to select the neuron growth/prune criterion. On-line Sequential Extreme Learning Machine (OS-ELM) [8] is the sequential version of ELM using recursive least squares. OS-ELM randomly assigns input weights with fixed number of hidden neurons and uses minimum norm recursive least squares to determines the output weights analytically. The random selection of input weights with fixed number of hidden neurons affects the performance of OS-ELM significantly in case of sparse and imbalance data sets [9]. Sequential Multi-Category Radial Basis Function network (SMC-RBF) combines the misclassification error, similarity measure within the class and prediction error in neuron growth/parameter update criterion [2]. In SMC-RBF, it has been shown that growing/pruning criterion based on class specific conditions improves the classification performance than conditions based on approximation error alone.

Aforementioned sequential learning algorithms in neural networks gain knowledge about the information contained in stream of data by using all the samples in the training data set. Simply put, these algorithms have human information-processing abilities such as perception, learning, remembering, judging and problem-solving, and these abilities are cognitive in nature. On the other hand, human learning studies suggested
that the learning process is effective when the learners adopt self-regulation in the learning using meta-cognition [10], [11]. Meta-cognition means cognition about cognition. Precisely, in a meta-cognitive framework, learners think about their cognitive process, and improve/control it by developing new strategies using the information contained in their memory. If a RBF network is considered as a cognitive component in meta-cognitive framework to analyze its cognitive process and to choose suitable learning strategies adaptively, then it is referred to as ‘Meta-Cognitive Radial Basis Function Network’ (McRBFN). Hence, there is a need to develop such McRBFN that is capable of deciding what-to-learn, when-to-learn and how-to-learn from a stream of training data to devise the decision function by emulating the human self-regulated learning principles.

Self-adaptive Resource Allocation Network (SRAN) [12] is a sequential learning algorithm and also addresses the what-to-learn component of meta-cognition by selecting significant samples using misclassification error and hinge loss error. The complex version of above algorithm is Complex-valued Self-regulating Resource Allocation Network (CSRAN) [13]. It has been shown in [12], [13], that selecting significant samples and removing repetitive samples in learning helps to improve the generalization performance. Therefore, it is apparent that emulating the three components of meta-cognition with suitable learning strategies would improve the generalization ability of a neural network. The drawbacks in the above algorithms are: (a) The selection of significant samples from stream of training data are based on simple error criterion which is not sufficient; (b) The allocation of new hidden neuron center without considering the amount of overlap with already existing neuron centers leads to misclassification; (c) Knowledge gained from past trained samples is not used in further learning; (d) These algorithms use computationally intensive EKF for parameter update. Meta-cognitive Neural Network (McNN) [14] and Meta-cognitive Neuro-Fuzzy Inference System (McFIS) [15] address the first two issues efficiently by using three components of meta-cognition. However, McNN and McFIS use computationally intensive EKF for parameter update and does not utilize the knowledge acquired from past trained samples. Similar works of meta-cognition in complex domain are reported in [16], [17]. Recently proposed projection based learning in meta-cognitive radial basis function network [18] addresses the first three issues in batch framework except proper utilization of the past knowledge stored in the network and applied in detection of neurodegenerative diseases [19], [20]. Therefore in this paper, we propose a fast and efficient sequential projection based learning algorithm for McRBFN.

There are several models of meta-cognition available in educational psychology and survey of these models of meta-cognition is reported in [21]. Nelson and Narens proposed a simple model of meta-cognition in [22] which clearly highlights the various self-regulated principles in human meta-cognition. Nelson and Narens model contains a cognitive component and a meta-cognitive component. The flow of information from the cognitive component to the meta-cognitive component is referred as a monitory signal, while the flow of information from the meta-cognitive component to the cognitive component is referred as a control signal. The control signal changes the state of the cognitive component or changes the cognitive component itself. On the other hand, monitory signal updates the meta-cognitive component about the state of cognitive component. Similar to the Nelson and Narens model of meta-cognition [22], McRBFN also has two components. A single hidden layer RBF network with evolving architecture is the cognitive component and the meta-cognitive component contains a dynamic model of the cognitive component, knowledge measures and self-regulated thresholds. The cognitive component learns from stream of training data by growing hidden neurons or updating the output weights of hidden neurons so as to approximate the decision function. When a new hidden neuron is added to the cognitive component, the Gaussian parameters (center and width) are determined based on the current training sample and the output weights are estimated using the Projection Based Learning (PBL) algorithm. Finding optimal output weights is first formulated as a linear programming problem from the principles of minimization and Real calculus. PBL algorithm then converts the linear programming problem into a system of linear equations and provides a solution for the optimal output weights, corresponding to the minima of the error function. For classification problems, it has been proven theoretically in [23], [24] that a classifier developed using a hinge-loss function estimates the posterior probability more accurately than a mean square error function. Hence, PBL algorithm in the cognitive component of McRBFN employs hinge-loss error function. In addition, PBL algorithm uses existing neurons in the cognitive component as pseudo-samples. There-by, the proposed algorithm exploits the knowledge stored in the network for proper initialization. When a sample is presented to the cognitive component of McRBFN, the meta-cognitive component of McRBFN measures the information contained in the current training sample with respect to the knowledge acquired in the cognitive component using its knowledge measures. Knowledge measures in the meta-cognitive component are predicted class label, maximum hinge error and class-wise significance. In kernel methods, spherical potential is widely used to determine whether all the trained samples are enclosed tightly by the Gaussian kernels [25]. Hence, the spherical potential is taken as class-wise significance of a sample and it is measured as the squared distance between the sample and the hyper-dimensional projection. In this paper, spherical potential is redefined to address the classification problems. Using the above mentioned knowledge measures along with self-regulatory thresholds, the meta-cognitive component constructs two sample based learning strategies and three neuron based learning strategies. As each training sample is presented, the meta-cognitive component selects one of the five strategies such that the cognitive component learns the decision function efficiently and achieves better classification performance. The meta-cognitive component also measures the amount of overlap between the current training sample and nearest neurons in the inter/intra-class to determine new hidden parameters. The PBL algorithm for McRBFN to obtain the network parameters is referred to as PBL-McRBFN.

The performance of the proposed PBL-McRBFN classifier
is evaluated using a set of benchmark binary/multi-category classification problems from University of California, Irvine (UCI) machine learning repository [26]. We consider nine multi-category and five binary classification problems with varying values of imbalance factor. In all these problems, the performance of PBL-McRBFN is compared against the best performing classifiers available in the literature. Further, we have also conducted a non-parametric Friedman test [27] to indicate the statistical significance in performance of PBL-McRBFN classifier. The results indicate PBL-McRBFN classifier outperforms the existing classifiers.

Finally, the performance of the PBL-McRBFN classifier has also been evaluated on a practical Alzheimer’s Disease (AD) detection problem using Magnetic Resonance Imaging (MRI). For performance evaluation, we consider well-known Open Access Series of Imaging Studies (OASIS) [28] and Alzheimer’s Disease Neuroimaging Initiative (ADNI) [29] MRI databases. These two MRI databases are generated in similar imaging environment with different demographic patient conditions. Hence, we use these data sets to demonstrate the capability of learning from changing distribution. The proposed classifier approximates the relationship between morphometry features and class label. First, we demonstrate the performance of the proposed classifier on these data sets and compare with the existing results reported in the literature. Later, we show the ability to capture the change in distribution by adapting the classifier developed using OASIS with ADNI images. The results show that the classifier has the ability to learn efficiently from a stream of data.

The structure of this paper is as follows: Section II describes the meta-cognitive radial basis network for classification problems. Section III presents the performance evaluation of PBL-McRBFN classifier on a set of benchmark and practical AD detection data sets in comparison to the best performing classifiers available in the literature. Section IV summarizes the conclusions from this study.

II. META-COGNITIVE RADIAL BASIS FUNCTION NETWORK FOR CLASSIFICATION PROBLEMS

In this section, we describe the meta-cognitive radial basis function network for solving classification problems. The classification problem in sequential framework can be defined as follow: Given a stream of training data, \( \{ (x^1, c^1), \cdots, (x^t, c^t), \cdots \} \), where \( x^t = [x^t_1, \cdots, x^t_m]^T \in \mathbb{R}^m \) is the \( m \)-dimensional input of the \( t \)-th sample, \( c^t \in \{1, \cdots, n \} \) is its class label and \( n \) is the total number of classes. The coded class labels \( y^t = [y^t_1, \cdots, y^t_j, \cdots, y^t_n]^T \in \mathbb{R}^n \) are given by:

\[
y^t_j = \begin{cases} 1 & \text{if } c^t = j, \cdots, n \end{cases} \quad j = 1, \cdots, n
\]

The objective of McRBFN classifier is to approximate the underlying decision function that maps \( x^t \in \mathbb{R}^m \rightarrow y^t \in \mathbb{R}^n \). McRBFN begins with zero hidden neuron and selects suitable strategy for each sample.

A. Meta-cognitive learning in RBF Network

The schematic diagram of Meta-cognitive learning in McRBFN is shown in Fig. 1. The cognitive component of McRBFN is a RBF network with evolving architecture. The meta-cognitive component of McRBFN contains a dynamic model of the cognitive component. When a new training sample is presented to the McRBFN, the meta-cognitive component of McRBFN 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 selecting suitable strategy for the current sample.

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

Without loss of generality, we assume that the McRBFN builds \( K \) neurons from \( t - 1 \) training samples. For a given input \( x^t \), the predicted output \( \hat{y}^t_j \) of McRBFN is

\[
\hat{y}^t_j = \sum_{k=1}^{K} w_{kj} h^t_k, \quad j = 1, \cdots, n
\]

where \( w_{kj} \) is the weight connecting the \( k \)-th hidden neuron to the \( j \)-th output neuron and \( h^t_k \) is the response of the \( k \)-th hidden neuron to the input \( x^t \) is given by

\[
h^t_k = \exp \left( -\frac{\|x^t - \mu^t_k\|^2}{\sigma^t_k^2} \right)
\]

where \( \mu^t_k \in \mathbb{R}^m \) is the center and \( \sigma^t_k \in \mathbb{R}^+ \) is the width of the \( k \)-th hidden neuron. Here, the superscript \( l \) represents the corresponding class of the hidden neuron.

Projection based learning algorithm: The projection based learning algorithm works on the principle of minimization of error function and finds the optimal network output weights for which the error is minimum.

\[
\frac{\partial E}{\partial \mu^t_k} = -\sum_{j=1}^{n} \frac{y^t_j - \hat{y}^t_j}{\sigma^t_k^2} (x^t - \mu^t_k) = 0
\]

\[
\frac{\partial E}{\partial \sigma^t_k} = \sum_{j=1}^{n} \frac{y^t_j - \hat{y}^t_j}{\sigma^t_k^2} (x^t - \mu^t_k) = 0
\]

where \( E \) is the error function. The previous two equations can be expressed in matrix form as:

\[
\begin{bmatrix}
\frac{\partial E}{\partial \mu^t_k} \\
\frac{\partial E}{\partial \sigma^t_k}
\end{bmatrix} = 0
\]

\[
\begin{bmatrix}
\sum_{j=1}^{n} (y^t_j - \hat{y}^t_j) (x^t - \mu^t_k)^T \\
\sum_{j=1}^{n} (y^t_j - \hat{y}^t_j) (x^t - \mu^t_k)
\end{bmatrix}
\]

This system of equations can be solved to obtain the optimal values of \( \mu^t_k \) and \( \sigma^t_k \) for each hidden neuron. The optimal values of the weights \( w_{kj} \) can be obtained by using the following equation:

\[
w_{kj} = \sum_{k=1}^{K} \frac{w_{kj}}{\sigma^t_k^2}
\]

The output of the McRBFN classifier is given by

\[
\hat{y}^t_j = \sum_{k=1}^{K} w_{kj} h^t_k
\]
The considered error function is the sum of squared errors at output neurons of McRBFN. The error function for \(i^{th}\) sample is defined as
\[
J_i = \sum_{j=1}^{n} \left( y^j_i - \sum_{k=1}^{K} w_{kj} h^i_k \right)^2
\]  
(4)

For \(t\) training samples, the overall error function is defined as
\[
J(W) = \frac{1}{2} \sum_{i=1}^{t} J_i = \frac{1}{2} \sum_{i=1}^{t} \sum_{j=1}^{n} \left( y^j_i - \sum_{k=1}^{K} w_{kj} h^i_k \right)^2
\]  
(5)

where \(h^i_k\) is the response of the \(k^{th}\) hidden neuron for \(i^{th}\) training sample.

The optimal network output weights \((W^* \in \mathbb{R}^{K \times n})\) are estimated such that the total error reaches its minimum.
\[
W^* = \text{arg} \min_{W \in \mathbb{R}^{K \times n}} J(W)
\]  
(6)

The optimal \(W^*\) corresponding to the minima of the error function \((J(W^*))\) is obtained by equating the first order partial derivative of \(J(W)\) with respect to the output weight to zero, i.e.,
\[
\frac{\partial J(W)}{\partial w_{pj}} = 0, \quad p = 1, \ldots, K; \quad j = 1, \ldots, n
\]  
(7)

Equating the first partial derivative to zero and re-arranging
\[
\sum_{k=1}^{K} \sum_{i=1}^{t} h^i_k h^i_p w_{kj} = \sum_{i=1}^{t} h^i_k y^i_j
\]  
(8)

which can be represented in matrix form as
\[
AW = B
\]  
(9)

where the projection matrix \(A \in \mathbb{R}^{K \times K}\) is given by
\[
a_{kp} = \sum_{i=1}^{t} h^i_k h^i_p, \quad k = 1, \ldots, K; \quad p = 1, \ldots, K
\]  
(10)

and the output matrix \(B \in \mathbb{R}^{K \times n}\) is
\[
b_{pj} = \sum_{i=1}^{t} h^i_k y^i_j, \quad p = 1, \ldots, K; \quad j = 1, \ldots, n
\]  
(11)

Eq. (9) gives the set of \(K \times n\) linear equations with \(K \times n\) unknown output weights \(W\).

The solution for \(W\) obtained as a solution to the set of equations as given in Eq. (9) is minimum, if \(\frac{\partial^2 J}{\partial w_{ip}^2} > 0\). The second derivative of the error function \((J)\) with respect to the output weights is given by
\[
\frac{\partial^2 J(W)}{\partial w_{ip}^2} = \sum_{i=1}^{t} h^i_k h^i_p = \sum_{i=1}^{t} |h^i_p|^2 > 0
\]  
(12)

As the second derivative of the error function \(J(W)\) is positive, the following observations can be made from Eq. (12):

1) \(J\) is a convex function.
2) The obtained output weight \(W^*\) as a solution to the set of linear equations (Eq. (9)) is the weight corresponding to the minima of the error function \((J)\).

The solution for the system of equations in Eq. (9) can be determined as follows:
\[
W^* = A^{-1}B
\]  
(13)

2) Meta-cognitive component of McRBFN: The meta-cognitive component contains a dynamic model of the cognitive component, knowledge measures and self-regulated thresholds. During the learning process, the meta-cognitive component monitors the cognitive component and updates its dynamic model of the cognitive component. When a new training sample \((x', y')\) is presented to McRBFN, the meta-cognitive component of McRBFN estimates the knowledge present in the new training sample with respect to the cognitive component using its knowledge measures. The meta-cognitive component uses predicted class label \((\hat{c}^t)\), maximum hinge error \((E^t)\), confidence of classifier \((\hat{p}(c^t|x'))\) and class-wise significance \((\psi_c)\) as the measures of knowledge in the new training sample. Self-regulated thresholds are adapted to capture the knowledge present in the new training sample. Based on the knowledge measures and self-regulated thresholds, the meta-cognitive component chooses one of the two sample based learning strategies or three neuron based learning strategies to learn the current sample accurately.

The knowledge measures are defined as shown below:

**Predicted class label \((\hat{c}^t)\):** Using the predicted output \((\hat{y}^t)\) it can be obtained as
\[
\hat{c}^t = \text{arg} \max_{j \in \{1, \ldots, n\}} \hat{y}^j_t
\]  
(14)

**Maximum hinge error \((E^t)\):** McRBFN uses the hinge loss error
\[
é^t = [e^t_1, \ldots, e^t_j, \ldots, e^t_n]^T \in \mathbb{R}^n\]  
and is defined as
\[
e^t_j = \begin{cases} 
\hat{y}^t_j - \hat{y}^j_t & \text{if } \hat{y}^t_j - \hat{y}^j_t < 1 \\
0 & \text{otherwise}
\end{cases}, \quad j = 1, \ldots, n
\]  
(15)

Using the hinge loss error \(e^t\), the maximum hinge error \((E^t)\) can be obtained as
\[
E^t = \max_{j \in \{1, 2, \ldots, n\}} |e^t_j|
\]  
(16)

**Confidence of classifier \((\hat{p}(c^t|x'))\):** The confidence level of classification or predicted posterior probability is given as
\[
\hat{p}(j|x') = \frac{\min(1, \max(-1, \hat{y}^j_t) + 1)}{2}, \quad j = c^t
\]  
(17)

**Class-wise significance \((\psi_c)\):** The class-wise distribution plays a vital role and it will influence the performance the classifier significantly [2]. Hence, we use the measure of the spherical potential of the new training sample \(x', y'\) 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 \((\psi_c)\) is defined as
\[
\psi_c = \frac{1}{K^c} \sum_{k=1}^{K^c} h(x^t, \mu^c_k)
\]  
(18)

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. For more details on the class-wise significance, one may refer to [14].

3) Learning strategies: The meta-cognitive component devises various learning strategies using the knowledge measures and self-regulated thresholds that address the basic principles of self-regulated learning (i.e., what-to-learn, when-to-learn and how-to-learn). The meta-cognitive part controls the learning process in the cognitive component by selecting one of the following five learning strategies for the new training sample.

- **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. During neuron addition, sample overlapping conditions are identified to allocate a new hidden neuron appropriately.

- **Parameter update strategy**: The new training sample is used to update the parameters of the cognitive component. PBL is used to update the parameters.

- **Neuron pruning strategy**: Very less contributed neurons in a class over a period of same class samples are deleted from the cognitive component.

- **Sample reserve strategy**: If 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.

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

**Sample delete strategy**: When the predicted class label of the new training sample is same as the actual class label and the estimated posterior probability is close to 1, 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

\[ c^t = e^t \land \hat{p}(c^t | x^t) \geq \beta_d \]  \tag{19}

The meta-cognitive deletion threshold \((\beta_d)\) controls number of samples participating in the learning process. The sample deletion strategy prevents learning of samples with similar information, and thereby, avoids over-training and reduces the computational effort. In our simulation studies, it is selected in the interval of \([0.9-0.95]\).

**Neuron growth strategy**: When a new training sample contains significant information and the predicted class label is different from the actual class label, then one need to add a new hidden neuron to represent the knowledge contained in the sample. The neuron growth criterion is given by

\[ (c^t \neq e^t \lor E^t \geq \beta_u) \land \psi_c(x^t) \leq \beta_c \]  \tag{20}

where \(\beta_c\) is the meta-cognitive knowledge measurement threshold and \(\beta_u\) is the self-adaptive meta-cognitive, addition threshold. The terms \(\beta_c\) and \(\beta_u\) allows samples with significant knowledge for learning first then uses the other samples for fine tuning. The \(\beta_u\) is adapted as follows

\[ \beta_u := \delta \beta_u + (1 - \delta) E^t \]  \tag{21}

where \(\delta\) is the slope that controls rate of self-adaptation and is set close to one. The \(\beta_u\) adaptation allows McRBFN to add neurons only when presented samples to the cognitive network contains significant information. In our simulation studies, \(\beta_u\) is selected in the interval of \([0.3-0.7]\) and \(\beta_u\) is selected in the interval of \([1.3-1.7]\). Details on influence of these parameters are given in [14].

When a new neuron \((K+1)\) is added, the existing sequential learning algorithms initialize center based on the new training sample and width based on the distance with nearest neuron. The new training sample may have overlap with existing neurons in other classes or will be a distinct cluster far away from the nearest neuron in the same class. Therefore, one has to consider the amount of overlap with respect to existing neurons while initializing the new neuron parameters. Also, the existing sequential learning algorithms estimates new neuron output weights based on the error of the new training sample. In sequential learning framework, the output weight initialization using only the new training sample influences the performance significance significantly, due to absence of knowledge contained in the past samples. In the proposed McRBFN, the above mentioned issues are dealt as

- The new neuron output weights will be estimated using existing knowledge of past trained samples stored in the network.
- The new neuron center and width parameters will be initialized based on new training sample distances to existing intra/inter class nearest neurons.

Let \(nrS\) be the nearest hidden neuron in the intra-class and \(nrI\) be the nearest hidden neuron in the inter-class as

\[ nrS = \underset{i=\in c \forall k}{\arg\min} | | x^t - \mu^\ell_k | |; \quad nrI = \underset{i \neq c \forall k}{\arg\min} | | x^t - \mu^\ell_k | | \]  \tag{22}

Let the Euclidian distances between the new training sample to \(nrS\) and \(nrI\) are given as follows

\[ d_S = | | x^t - \mu^\ell_{nrS} | |; \quad d_I = | | x^t - \mu^\ell_{nrI} | | \]  \tag{23}

We can determine the new hidden neuron center \((\mu^\ell_{K+1})\) and width \((\sigma^\ell_{K+1})\) parameters based on overlapping/no-overlapping conditions using the nearest neuron distances as follows:

- **Distinct sample**: If \((d_S >> \sigma^\ell_{nrS} \land d_I >> \sigma^\ell_{nrI})\) then the new training sample does not overlap with any class cluster, and is from a distinct cluster.

\[ \mu^\ell_{K+1} = x^t; \quad \sigma^\ell_{K+1} = \kappa \sqrt{ (x^t)^T x^t }, \quad l = c; \]  \tag{24}

where \(\kappa\) is a overlap factor of the hidden units, which lies in the range \(0.5 \leq \kappa \leq 1\).

- **No-overlapping**: If the intra/inter class distance ratio is less than 1, then the sample does not overlap with the
other classes.

\[ \mu_{l+1}^i = x^i; \quad \sigma_{l+1}^i = \kappa \| x^i - \mu_{nrS}^{l+1} \|, \quad l = c; \quad (25) \]

- **Minimum overlapping with the inter-class:** If the intra-inter class distance ratio is in range 1 to 1.5, then the sample has minimum overlapping with the other class.

\[ \mu_{l+1}^i = x^i + \zeta (\mu_{nrL}^l - \mu_{nrS}^l), \quad l = c; \quad (26) \]

where \( \zeta \) is center shift factor. In our simulation studies, it is selected in the interval of [0.01-0.1].

- **Significant overlapping with the inter-class:** If the intra-inter class distance ratio is more than 1.5, then the sample has significant overlapping with the other class.

\[ \mu_{l+1}^i = x^i - \zeta (\mu_{nrL}^l - x^i), \quad l = c; \quad (27) \]

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

When a neuron is added to McRBFN, the output weights are estimated using PBL with past knowledge stored in the network as pseudo-samples, i.e., centers and associated class labels as pseudo-samples.

When a new neuron is added to the cognitive component, the size of matrix \( A \) as defined in Eq. (10) is increased from \( K \times K \) to \((K + 1) \times (K + 1)\)

\[ A_{(K+1)\times(K+1)} = \begin{bmatrix} A_{K\timesK} + (h^i)^T h^i & a_{K+1}^T \\ a_{K+1} & 1 \end{bmatrix}, \quad (28) \]

where \( h^i = [h_1^i, h_2^i, \ldots, h_K^i] \) is a vector of the existing \( K \) hidden neurons response for new \((t^{th})\) training sample. In sequential learning samples are discarded after learning, but the information present in the past samples are stored in the network. The centers of neuron provides the distribution of past samples in feature space. These centers \( \{\mu_1^i, \cdots, \mu_K^i\} \) can be used as pseudo-samples to capture the effect of past samples. Hence, existing hidden neurons are used as pseudo-samples to calculate \( a_{K+1} \) and \( a_{K+1, K+1} \) terms. \( a_{K+1} \in \mathbb{R}^{1 \times K} \) is assigned as

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

and \( a_{K+1, K+1} \in \mathbb{R}^{1} \) value is assigned as

\[ a_{K+1, K+1} = \sum_{i=1}^{K+1} h_{K+1}^i h_{K+1}^i \quad (30) \]

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

\[ B_{(K+1)\times n} = \begin{bmatrix} B_{K\times n} + (h^i)^T (y^i)^T \\ b_{K+1} \end{bmatrix}, \quad (31) \]

and \( b_{K+1} \in \mathbb{R}^{1 \times n} \) is a row vector assigned as

\[ b_{K+1, j} = \sum_{i=1}^{K+1} h_{K+1}^i \tilde{y}^i_j, \quad j = 1, \cdots, n \quad (32) \]

where \( \tilde{y}^i_j \) is the pseudo-output for the \( i^{th} \) pseudo sample or hidden neuron \( (\mu_j^i) \) given as

\[ \tilde{y}^i_j = \begin{cases} 1 & \text{if } l = j \\ -1 & \text{otherwise} \end{cases}, \quad j = 1, \cdots, n \]

Finally the output weights are estimated as

\[ W_{K+1}^i = \begin{bmatrix} A_{(K+1)\times(K+1)}^{-1} B_{(K+1)\times n} \end{bmatrix}, \quad (34) \]

where \( W_{K+1}^i \) is the output weight matrix for \( K \) hidden neurons, and \( W_{K+1}^i \) is the vector of output weights for new hidden neuron after learning from \((t^{th})\) sample. The inverse of a matrix \( A_{(K+1)\times(K+1)} \) is calculated recursively using matrix identities as

\[ \left( A_{(K+1)\times(K+1)} \right)^{-1} = \begin{bmatrix} (A_{K\timesK})^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} (A_{K\timesK})^{-1} a_{K+1}^T \\ a_{K+1} \end{bmatrix} \begin{bmatrix} (A_{K\timesK})^{-1} a_{K+1}^T \\ a_{K+1} \end{bmatrix}^T \quad (35) \]

where \( A_{K\timesK} = A^{t-1} + (h^t)^T h^t, \quad \Delta = a_{K+1, K+1} - a_{K+1} \left( A_{K\timesK}^{-1} a_{K+1} \right) \) and \( A_{K\timesK}^{-1} \) is calculated as

\[ A_{K\timesK}^{-1} = (A^{t-1})^{-1} - \frac{(A^{t-1})^{-1} (h^t)^T h^t (A^{t-1})^{-1}}{1 + h^t (A^{t-1})^{-1} (h^t)^T} \quad (36) \]

After calculating inverse in matrix in Eq. (34) using Eqs. (35) and (36), the resultant equations are

\[ W_{K+1}^i = \left[ I_{K\timesK} + \left( \frac{a_{K+1}^T}{\Delta} A_{K\timesK}^{-1} \right) b_{K+1} \right] \]

\[ W_{K+1}^i - \left( \frac{a_{K+1}^T}{\Delta} A_{K\timesK}^{-1} (h^t)^T (y^i)^T \right) - \frac{A_{K\timesK}^{-1} a_{K+1}^T b_{K+1}}{\Delta} \quad (37) \]

\[ w_{K+1}^i = \frac{a_{K+1}^T W_{K+1}^i + \left( \frac{a_{K+1}^T}{\Delta} A_{K\timesK}^{-1} (h^t)^T (y^i)^T \right) + b_{K+1}^T}{\Delta} \quad (38) \]

**Parameters update strategy:** The current \((t^{th})\) training sample is used to update the output weights of the cognitive component \((W_K = [w_1, w_2, \cdots, w_K])\) if the following criterion is satisfied.

\[ c^t = c^t \quad \text{AND} \quad E^t \geq \beta_u \quad (39) \]

where \( \beta_u \) is the self-adaptive meta-cognitive parameter update threshold. For the selection of \( \beta_u \) value one may refer to [14].

The \( \beta_u \) is adapted based on the maximum hinge error as:

\[ \beta_u := \delta \beta_u + (1 - \delta) E^t \quad (40) \]

where \( \delta \) is the slope that controls the rate of self-adaption of parameter update and is set close to one.

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

\[ \frac{\partial J(W^k_K)}{\partial w_{pj}} = \frac{\partial J(W^k_K)}{\partial w_{pj}} + \frac{\partial J_i(W^k_K)}{\partial w_{pj}} = 0, \quad \text{where} \quad p = 1, \cdots, K; \quad j = 1, \cdots, n \quad (41) \]
Equating the first partial derivative to zero and re-arranging the Eq. (41), we get
\[
(A^{t-1} + (h^t)^T h^t) W^t_K - (B^{t-1} + (h^t)^T (y^t)^T) = 0
\]
By substituting \(B^{t-1} = A^{t-1} W^{-1}_K\) and \(A^t = A^{t-1} + (h^t)^T h^t\) and adding/subtracting the term \((h^t)^T h^t W^{-1}_K\) on both sides the Eq. (42) reduced to
\[
W^t_K = (A^t)^{-1} (A^t W^{-1}_K + (h^t)^T ((y^t)^T - h^t W^{-1}_K))
\]
Finally the output weights are updated as
\[
W^t_K = W^{t-1}_K + (A^t)^{-1} (h^t)^T (e^t)^T
\]
where \(e^t\) is the hinge loss error for \(t^{th}\) sample obtained from Eq. (15). From Eqs. (15) and (44), we can see that only the non-zero hinge loss error producing output neurons connections weights are updated. Since, the parameter update employs hinge error, the resultant network approximates the posterior probability very well.

**Neuron pruning strategy:** If the contribution of a neuron in the same class is lesser than a pruning threshold \(\beta_p\) for \(N_p\) consequent samples in the same class \(l\) then that neuron is insignificant and can be removed from the network. The contribution of the \(k^{th}\) neuron is defined as
\[
r^l_k = \frac{h (x^l_i, \mu^l_k)}{\max_j h (x^l_i, \mu^l_j)}
\]
where \(\beta_p\) is the meta-cognitive neuron pruning threshold. If \(\beta_p\) is chosen closer to zero, then pruning occurs seldom and all the added neurons will remain in the network irrespective of their contribution to the network output. If \(\beta_p\) is chosen closer to one, then pruning occurs frequently, resulting in oscillations and insufficient neurons to capture the distribution from stream of training samples. When a neuron is pruned from the network, the dimensionality of the \(A^t\) and \(B^t\) is reduced by removing the respective rows/columns corresponding to the pruned neuron.

**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 or neuron pruning, then the current sample is pushed to the rear of the training sequence. Since McRBFN modifies the strategies based on the current sample knowledge, these samples may be used in later stage.

Ideally, training process stops when no further sample is available in data stream. However, in real-time, training stops when samples in the reserve remains same.

In PBL-McRBFN, sample delete strategy address the what-to-learn by deleting insignificant samples from training data set, neuron growth strategy, parameters update strategy and neuron pruning strategy address the how-to-learn efficiently 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 evaluation of PBL-McRBFN classifier

In this section, we first present the performance comparison of the proposed PBL-McRBFN with the best performing sequential learning algorithm reported in the literature McNN [14], batch ELM [9] and standard Support Vector Machine (SVM) [30] on real-world benchmark binary and multi-category classification data sets from the UCI machine learning repository [26]. Next, we use the PBL-McRBFN to detect Alzheimer’s disease from a stream of MRI scans.

#### A. Performance comparison on benchmark data sets

1) **Benchmark data sets:** In order to extensively verify the performance of the proposed algorithm, we have chosen data sets with small and large number of samples, low and high dimensional features, and balanced and unbalanced data sets in both binary and multi classification problems. The detailed specifications of five binary and nine multi classification data sets are given in Table I. Note that the data sets are taken from UCI machine learning repository, except for satellite imaging [24], global cancer mapping using micro-array gene expression [31] and acoustic emission data sets [32]. The sample imbalance in training and testing is measured using Imbalance Factor (I.F) and is defined as
\[
I.F = 1 - \frac{n}{N} \times \min_j N_j
\]
where \(N_j\) is the total number of training samples in class \(j\) and \(N = \sum_{j=1}^{n} N_j\).

For efficient comparison, we present them under the following categories as described below:

**Binary class data sets:** All the considered binary class data sets have high sample imbalance and are grouped into two categories

- **Low dimensional:** Liver disorders (LD), pima indian diabetes (PIMA) and breast cancer (BC) are having low dimensional features with relatively smaller number of training samples.
- **High dimensional:** Heart disease (HEART) and ionosphere (ION) data sets are having smaller number of training samples with high dimensional features.

**Multi class data sets:** Considered nine multi class data sets are grouped into three categories:

- **Well balanced:** Image segmentation (IS), iris classification (IRIS) and wine determination (WINE) data sets have equal number of training samples per class. These data sets are having varying number of features and training/testing samples.
- **Imbalanced:** Vehicle classification (VC), glass identification (GI) and acoustic emission classification (AE) data sets have lower dimensional features and highly imbalanced training samples. The global cancer mapping (GCM) using micro-array gene expression is having high dimensional features with high sample imbalance.
- **Large number of samples:** Letter recognition (LETTER) and satellite image classification (SI) data sets have relatively large number of samples and classes.
2) **Simulation Environment:** For this performance comparison study, experiments are conducted for the PBL-McRBFN, McNN, ELM and SVM classifiers on all the data sets in MATLAB 2011 on a desktop PC with Intel Core 2 Duo, 2.66 GHz CPU and 3 GB RAM. The tunable parameters of PBL-McRBFN and McNN are chosen using cross-validation on the training data sets. For ELM classifier [3], the number of hidden neurons are obtained using the constructive-destructive procedure presented in [33]. The simulations for batch SVM with Gaussian kernels are carried out using the LIBSVM package in C [34]. For SVM classifier, the parameters ($c, \gamma$) are optimized using grid search technique.

3) **Performance measures:** The overall and average classification efficiencies are used as quantitative evaluation measures in this study. 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 ($\eta_j$) which is defined as:

$$\eta_j = \frac{q_{jj}}{N_j} \times 100\% \quad (47)$$

where $q_{jj}$ is the total number of correctly classified samples in the class $j$. The global measures used in the evaluation are the average per-class classification accuracy ($\eta_a$) and the over-all classification accuracy ($\eta_o$) defined as:

$$\eta_a = \frac{1}{n} \sum_{j=1}^{n} \eta_j \quad , \quad \eta_o = \frac{\sum_{j=1}^{n} q_{jj}}{N} \times 100\% \quad (48)$$

4) **Performance Comparison:** **Binary class data sets:** The performance measures such as overall ($\eta_o$), average ($\eta_a$) testing efficiencies, number of neurons and samples used for PBL-McRBFN, McNN, ELM and SVM classifiers on all the 5 binary class data sets are reported in Table II. From the performance comparison results in Table II, one can see that in case of low dimensional LD and PIMA data sets, the proposed PBL-McRBFN uses fewer samples for training and achieves significantly better generalization performance approximately 1% improvement over McNN, 1-2% improvement over ELM and SVM with less number of neurons. In case of simple BC data set, PBL-McRBFN uses fewer samples for training and achieves slightly better generalization performance approximately 1% improvement over ELM and SVM with less number of neurons and same performance as McNN. In case of large dimensional HEART and ION data sets, PBL-McRBFN uses fewer samples for training and achieves better generalization performance 1-2% improvement over McNN, 6-9% improvement over SVM and ELM. The overlapping conditions and class specific criterion in learning strategies of PBL-McRBFN helps in capturing the knowledge accurately in case of high sample imbalance problems.

### Multi-category data sets:

The overall ($\eta_o$) and average ($\eta_a$) testing efficiencies, number of neurons and samples used for PBL-McRBFN, ELM and SVM classifiers on all the 9 multi-category data sets are reported in Table III. From Table III, we can see that PBL-McRBFN performs significantly better than the ELM and SVM on all the 9 multi-category data sets. In case of well balanced IS, IRIS, and WINE data sets, PBL-McRBFN uses only 42-50% training samples to achieve better generalization performance approximately 2-3% improvement over McNN, SVM and ELM with the less number of neurons. Meta-cognitive sample deletion criteria helps in removing redundant samples from the training set and thereby improves the generalization performance. For highly unbalanced data sets, one can see that the proposed PBL-McRBFN is able to achieve significantly better performance than the other classifiers. In case of VC and GI data sets, PBL-McRBFN uses fewer samples and achieves better generalization performance approximately 1-5% improvement over

---

### Table I: Specification of benchmark binary and multi class data sets

<table>
<thead>
<tr>
<th>Data sets</th>
<th># Features</th>
<th># Classes</th>
<th># Samples</th>
<th>I/F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Training</td>
<td>Testing</td>
</tr>
<tr>
<td>LD</td>
<td>6</td>
<td>2</td>
<td>200</td>
<td>145</td>
</tr>
<tr>
<td>PIMA</td>
<td>8</td>
<td>2</td>
<td>400</td>
<td>368</td>
</tr>
<tr>
<td>BC</td>
<td>9</td>
<td>2</td>
<td>300</td>
<td>383</td>
</tr>
<tr>
<td>HEART</td>
<td>13</td>
<td>2</td>
<td>70</td>
<td>200</td>
</tr>
<tr>
<td>ION</td>
<td>34</td>
<td>2</td>
<td>100</td>
<td>251</td>
</tr>
<tr>
<td>IS</td>
<td>19</td>
<td>7</td>
<td>210</td>
<td>2100</td>
</tr>
<tr>
<td>IRIS</td>
<td>4</td>
<td>3</td>
<td>45</td>
<td>105</td>
</tr>
<tr>
<td>WINE</td>
<td>13</td>
<td>3</td>
<td>60</td>
<td>118</td>
</tr>
<tr>
<td>VC</td>
<td>18</td>
<td>4</td>
<td>424$^a$</td>
<td>422</td>
</tr>
<tr>
<td>GI</td>
<td>9</td>
<td>6</td>
<td>109$^a$</td>
<td>105</td>
</tr>
<tr>
<td>AE</td>
<td>5</td>
<td>4</td>
<td>62</td>
<td>137</td>
</tr>
<tr>
<td>GCM</td>
<td>98</td>
<td>14</td>
<td>144</td>
<td>46</td>
</tr>
<tr>
<td>LETTER</td>
<td>16</td>
<td>26</td>
<td>13333</td>
<td>6667</td>
</tr>
<tr>
<td>SI</td>
<td>6</td>
<td>9</td>
<td>9108</td>
<td>262144</td>
</tr>
</tbody>
</table>

$^a$As suggested in [2], the samples are repeated three times.

### Table II: Performance comparison of PBL-McRBFN with McNN, ELM and SVM on binary class data sets

<table>
<thead>
<tr>
<th>Data set</th>
<th>Classifier</th>
<th># Neurons</th>
<th># Samples used</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD</td>
<td>SVM</td>
<td>141$^a$</td>
<td>200</td>
<td>71.03</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>100</td>
<td>200</td>
<td>72.41</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>68</td>
<td>110</td>
<td>73.79</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>87</td>
<td>116</td>
<td>73.10</td>
</tr>
<tr>
<td>PIMA</td>
<td>SVM</td>
<td>221$^a$</td>
<td>400</td>
<td>77.45</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>100</td>
<td>400</td>
<td>76.63</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>76</td>
<td>193</td>
<td>80.16</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>100</td>
<td>162</td>
<td>79.62</td>
</tr>
<tr>
<td>BC</td>
<td>SVM</td>
<td>24$^a$</td>
<td>300</td>
<td>96.61</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>66</td>
<td>300</td>
<td>96.35</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>9</td>
<td>27</td>
<td>97.39</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>13</td>
<td>45</td>
<td>97.39</td>
</tr>
<tr>
<td>HEART</td>
<td>SVM</td>
<td>42$^a$</td>
<td>70</td>
<td>75.5</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>36</td>
<td>70</td>
<td>76.50</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>26</td>
<td>46</td>
<td>80.50</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>20</td>
<td>69</td>
<td>81.50</td>
</tr>
<tr>
<td>ION</td>
<td>SVM</td>
<td>43$^a$</td>
<td>100</td>
<td>91.24</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>32</td>
<td>100</td>
<td>89.64</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>20</td>
<td>39</td>
<td>95.62</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>18</td>
<td>58</td>
<td>96.41</td>
</tr>
</tbody>
</table>

$^a$Number of support vectors.
McNN and ELM, 7-12% improvement over SVM. In case of low dimensional AE data set, PBL-McRBFN achieves slightly better generalization performance 1% improvement over SVM, similar to McNN and ELM. In case of large dimensional GCM data set, PBL-McRBFN achieves significantly better generalization performance approximately 13% improvement over SVM and ELM. In case of large samples LETTER and SI data sets, the generalization performance of PBL-McRBFN is better than ELM and SVM by approximately 2-3% and 3% using less number of training samples and neurons. Due to computational intensive EKF for parameter update, McNN experiences memory problem for large problems like GCM, Letter and SI. Hence, the results for McNN in these problem are not presented here.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Classifier</th>
<th># Neurons</th>
<th># Samples used</th>
<th>Testing Efficiency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>SVM</td>
<td>127(^a)</td>
<td>210</td>
<td>91.38</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>49</td>
<td>210</td>
<td>90.23</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>49</td>
<td>93</td>
<td>93.38</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>50</td>
<td>89</td>
<td>94.19</td>
</tr>
<tr>
<td>IRIS</td>
<td>SVM</td>
<td>13(^a)</td>
<td>45</td>
<td>96.19</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>10</td>
<td>45</td>
<td>96.19</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>5</td>
<td>22</td>
<td>97.14</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>6</td>
<td>20</td>
<td>98.10</td>
</tr>
<tr>
<td>WINE</td>
<td>SVM</td>
<td>36(^a)</td>
<td>60</td>
<td>97.46</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>10</td>
<td>60</td>
<td>97.46</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>9</td>
<td>27</td>
<td>98.3</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>11</td>
<td>29</td>
<td>98.31</td>
</tr>
<tr>
<td>VC</td>
<td>SVM</td>
<td>340(^a)</td>
<td>620</td>
<td>70.62</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>150</td>
<td>620</td>
<td>77.01</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>146</td>
<td>359</td>
<td>77.72</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>175</td>
<td>318</td>
<td>78.91</td>
</tr>
<tr>
<td>GI</td>
<td>SVM</td>
<td>183(^a)</td>
<td>336</td>
<td>70.47</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>80</td>
<td>336</td>
<td>81.31</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>73</td>
<td>117</td>
<td>85.71</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>71</td>
<td>115</td>
<td>84.76</td>
</tr>
<tr>
<td>AE</td>
<td>SVM</td>
<td>22(^a)</td>
<td>62</td>
<td>98.54</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>10</td>
<td>62</td>
<td>99.27</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>5</td>
<td>20</td>
<td>99.27</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>5</td>
<td>9</td>
<td>99.27</td>
</tr>
<tr>
<td>GCM</td>
<td>SVM</td>
<td>107(^a)</td>
<td>98</td>
<td>76.08</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>55</td>
<td>98</td>
<td>76.08</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>72</td>
<td>109</td>
<td>93.47</td>
</tr>
<tr>
<td>LETTTER</td>
<td>SVM</td>
<td>4129(^a)</td>
<td>13333</td>
<td>92.94</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>13333</td>
<td>92.94</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>McNN</td>
<td>1654</td>
<td>3346</td>
<td>93.51</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>1274</td>
<td>2445</td>
<td>90.14</td>
</tr>
<tr>
<td>SI</td>
<td>SVM</td>
<td>1298(^a)</td>
<td>9108</td>
<td>92.21</td>
</tr>
<tr>
<td></td>
<td>ELM</td>
<td>1500</td>
<td>9108</td>
<td>88.39</td>
</tr>
<tr>
<td></td>
<td>PBL-McRBFN</td>
<td>1274</td>
<td>2445</td>
<td>90.14</td>
</tr>
</tbody>
</table>

\(^a\) Number of support vectors

From the Tables II and III, we can say that the proposed PBL-McRBFN improves generalization performance under wide range of sample imbalance data sets.

5) Statistical Comparison: In order to compare the performance of the proposed PBL-McRBFN classifier over McNN, ELM and SVM classifiers on various benchmark data sets, we employ a non-parametric Friedman test followed by the Benfornron-Dunn test as described in [27]. The Friedman test compares whether the mean of individual experimental condition differs significantly from the aggregate mean across all conditions. If the measure F-score is greater than the F-Statistic at 95% confidence level then one rejects the equality of mean hypothesis, i.e., the classifiers used in our study perform similarly on different data set. If non-parametric Friedman test rejects the equality hypothesis, then pair-wise post-hoc should be conducted to test which mean is different from others. In our study, we have used 4 different classifiers ranks based on average testing efficiency on 11 different data sets from the Table I.

The F-score obtained using non-parametric Friedman test is 42.75, which is greater than the F-statistic at 95% confidence level (F\(_{3,30,0.025}\) is 3.542), i.e, 42.75 > 3.542. Hence, one can reject mean equality hypothesis at a confidence of level 95%.

Next, we conduct a pair-wise comparison using a Benfornoni-Dunn test to highlight the performance significance of PBL-McRBFN classifier with respect to other classifiers. Here, the proposed PBL-McRBFN classifier is used as a control. The Critical Difference (CD) is calculated is 1.317 at 95% confidence level. From the Tables II and III, we can obtain the average ranks for all four classifiers and are PBL-McRBFN: 1.25, McNN: 1.86, ELM: 3.18, and SVM: 3.72. The difference in average rank between the proposed PBL-McRBFN classifier and the other three classifiers are PBL-McRBFN-McNN: 0.61, PBL-McRBFN-ELM: 1.93 and PBL-McRBFN-SVM: 2.47. Note that difference in average rank for PBL-McRBFN-ELM and PBL-McRBFN-SVM pairs are greater than CD at 95% confidence level, i.e, 1.93 > 1.317 and 2.47 > 1.317. The difference in average rank for PBL-McRBFN-McNN pair is less than CD at 95% confidence level, i.e, 0.61 < 1.317. Hence, we can say that PBL-McRBFN performs slightly better than the McNN classifier and significantly better than ELM and SVM classifiers with a confidence of 95%.

B. Alzheimer’s disease detection using PBL-McRBFN

Alzheimer’s Disease (AD) is one of the most common causes of dementia. AD is a progressive, neuro-degenerative disorder that leads to memory loss, problems in learning, confusion and poor judgment. Early detection of AD using non-invasive methods (brain imaging) plays a major role in providing treatment that may slow down its progress. MRI is the most important brain imaging procedure that provides accurate information about the shape and volume of the brain. The problem of early detection of AD using MRI can be formulated as binary classification and can be solved using machine learning techniques [35].

1) Related works: The studies of analyzing MRI scans can be categorized into two classes: Region-of-Interest (ROI) methods [36] and whole brain morphometric methods [37]. In ROI methods, the volumetric measurements of specific brain regions are used to detect AD. Studies have shown that the tissue loss in the hippocampus and the entorhinal cortex could be indicators for early AD. Major shortcomings in the use...
of the ROI methods are dependency on tracer expertise and are erroneous. To overcome these shortcomings, whole brain morphometric methods have been employed for accurate AD detection. Voxel Based Morphometry (VBM) is a completely automatic image analysis approach for identifying the amount of gray matter or white matter differences between the normal persons and AD patients [38]. The steps involved in VBM to identify significant differences between the groups of images are spatial normalization, segmentation, smoothing and statistical analysis [39]. The VBM analysis is used in this paper to identify the regional differences in gray matter between groups of persons and to extract morphometric features from MRI scans.

2) AD data sets: OASIS database [28] has cross-sectional collection of 416 persons covering the adult life-span including persons with AD in an early-stage. The data includes 218 persons aged between 18 to 59 years, and 198 between 60 to 96 years. Among 198 elderly persons considered in this study, 98 had no AD i.e, Clinical Dementia Rating (CDR) is zero. 70 persons were diagnosed as very mild AD (CDR=0.5), 28 persons were diagnosed as mild AD (CDR=1) and 2 persons as moderate AD (CDR=2). ADNI database [29] has cross-sectional collection of 422 elderly persons aged between 55 and 91 years. Among 422 elderly persons consider in this database, 226 were diagnosed as normal (CDR=0); 95 persons were diagnosed as very mild AD (CDR=0.5); 101 persons were diagnosed as mild AD (CDR=1).

3) Feature extraction using voxel based morphometry: Voxel Based Morphometry (VBM) is a voxel-wise comparison of local tissue volume of gray matter within or across groups of persons using MRI scans. Here, MRI scans undergo various preprocessing steps before the voxel-wise parametric tests are carried out on them.

The steps involved in VBM analysis are: unified segmentation, smoothing and statistical testing, in that order as in unified segmentation model proposed by [38]. The unified segmentation step is a generative modeling approach, in which tissue segmentation, bias correction and image registration are combined in a single model. The unified segmentation framework combines deformable tissue probability maps with a Gaussian mixture model. The MR brain images of both the AD patients and healthy persons are segmented into gray matter tissue class. The segmented and normalized gray matter images are then smoothed by convolving with an isotropic Gaussian kernel. VBM was performed using the Statistical Parametric Map (SPM) software package [40]. A 10 mm full-width at half maximum Gaussian kernel was employed. In Fig. 2, we have shown three planar views (sagittal, coronal and axial) of the original images and images after every stages in VBM analysis. The maximum intensity projections of the significant voxels obtained for OASIS data set in sagittal, coronal and axial views are shown in Fig. 3.

4) Performance comparison: Individual OASIS and ADNI data sets: Significant voxels extracted from VBM analysis are taken as features. VBM extracted 19789 features on OASIS data set and 23797 features on ADNI data set. For classification study, 10 random trials of features sets are used as input to the PBL-McRBFN classifier. In each trial, 50% of the samples are randomly chosen for training set, the rest as testing set. Table IV presents the mean results obtained from 10 random trials on both OASIS and ADNI data sets. The best PBL-McRBFN classification accuracy on OASIS data set is 76.77%. Compared to the results reported in [41] and [42], the PBL-McRBFN classification accuracy on 19879 features set is 10% more than the PCA-SVM approach (66.98%) [41], 7% more than the IPCA-SVM approach (69.7%) [41] and 14% more than the ICA-SVM approach (62.8%) [42].

The best PBL-McRBFN classification accuracy on ADNI data set is 86.26%. Compared to the results reported in [43] and [44], the PBL-McRBFN classification accuracy on complete 23797 features set is 8% more than the ICA-SVM approach on ADNI whole brain images (78.4%) [43], 4% more than the ICA-SVM approach on ADNI gray matter images (82.4%) [43], 5% more than the SVM approach on ADNI images with hippocampal volume features (81%) [44] and 4%
more than the SVM approach on ADNI images with cortical thickness features (82%) [44].

**Non-stationary ADNI data set from OASIS SPM:** From the description of OASIS and ADNI data sets we can see that these data sets are collected from the different demographic people with different geographic locations. Hence, the databases represent variation in the data distribution. In this study, we study the performance of proposed classifier under varying distribution. For this purpose, SPM obtained using OASIS database is used to extract the VBM features from ADNI database. The extracted 19879 features from ADNI samples are tested with best classifier developed using OASIS database. The schematic diagram of cross-sectional study is shown in Fig. 4. Such cross-sectional study will prevent computationally intensive VBM feature extraction and unify the diagnosis mechanism. When complete 422 samples from the ADNI data set are used as testing samples to the best PBL-McRBFN classifier which is trained on OASIS database, then the testing accuracy is 62.3%. Next, 25% of samples from the ADNI database are used to adapt the PBL-McRBFN classifier using meta-cognitive principles and tested on the remaining 75% of samples then the testing accuracy is 77.4%. From the above results, we can infer that best PBL-McRBFN classifier trained on one data distribution (OASIS database) with adaptation using fewer samples from ADNI achieves significant testing accuracy on unseen samples.

**IV. Conclusions**

In this paper, we have presented a sequential learning algorithm using human meta-cognitive principles. The meta-cognitive component selects appropriate learning strategy for the cognitive RBF classifier to achieve better generalization performance with minimal computational effort. The meta-cognitive component adapts the learning process appropriately by implementing self-regulation and hence it decides *what-to-learn, when-to-learn and how-to-learn* efficiently. In addition, projection based learning accurately estimates the output weight by direct minimization of hinge loss error and the overlapping conditions present in neuron growth strategy helps in proper initialization of new hidden neuron parameters which minimizes the misclassification error. The performance of the proposed PBL-McRBFN classifier has been evaluated using the benchmark binary and multi classification data sets and also on a practical problem of Alzheimer’s disease detection. The statistical non-parametric Friedman test based on eleven benchmark data sets clearly indicated that the proposed PBL-McRBFN classifier achieves slightly better performance than McNN, and significantly better performance than ELM and SVM classifiers. In practical Alzheimer’s disease detection problem the proposed PBL-McRBFN classifier shows 7-14% improvement over reported results on OASIS and 4-8% improvement over ADNI databases. Based on simulation studies conducted on benchmark and practical problems, we can infer that human meta-cognitive principles in learning algorithm improves the performance significantly. Finally the performance evaluation on ADNI database with PBL-McRBFN classifier trained on OASIS database shows that the proposed PBL-McRBFN can also achieves significant results on the non-stationary problems.

**ACKNOWLEDGEMENT**

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. The authors are grateful to Dr. B. S. Mahanand at Sri Jayachamarajendra college of engineering, Mysore, India for guidance and help in VBM feature extraction technique. The authors also acknowledge the Nanyang Technological University-Ministry of Defence, Singapore, for the financial support (MINDEF-NTU-JPP/11/02/05) to conduct this study.

**REFERENCES**


Giduthuri Sateesh Babu (S’12) received the B.Tech degree in electrical and electronics engineering from Jawaharlal Nehru Technological University, India, in 2007, and M.Tech degree in electrical engineering from Indian Institute of Technology Delhi, India, in 2009. From 2009 to 2010, he worked as a senior software engineer in Samsung R&D centre, India. He is currently a Ph.D. student with School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include machine learning, cognitive computing, neural networks, control systems, optimization and medical informatics.

Sundaram Suress (SM’08) received the B.E degree in electrical and electronics engineering from Bharathiyar University in 1999, and M.E (2001) and Ph.D. (2005) degrees in aerospace engineering from Indian Institute of Science, India. He was post-doctoral researcher in school of electrical engineering, Nanyang Technological University from 2005 to 2007. From 2007 to 2008, he was in INRIA-Sophia Antipolis, France as ERCIM research fellow. He was in Korea University for a short period as a visiting faculty in Industrial Engineering. From January 2009 to December 2009, he was in Indian Institute of Technology-Delhi as an Assistant Professor in Department of Electrical Engineering. Currently, he is working as an Assistant Professor in School of Computer Engineering, Nanyang Technological University, Singapore since 2010. He was awarded best young Faculty for the year 2009 by IIT-Delhi. His research interests include machine learning, cognitive modeling, flight control, applied game theory, optimization and computer vision.