Autism Spectrum Disorder Detection Using Projection Based Learning Meta-cognitive RBF Network

S.Vigneshwaran, B.S. Mahanand, S.Suresh and R.Savitha

Abstract—In this paper, we present an approach for the diagnosis of Autism Spectrum Disorder (ASD) from Magnetic Resonance Imaging (MRI) scans with Voxel-Based Morphometry (VBM) detected features using Projection Based Learning (PBL) algorithm for a Meta-cognitive Radial Basis Function Network (McRBFN) classifier. McRBFN emulates human-like meta-cognitive learning principles. As each sample is presented to the network, the 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. 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. Moreover, as samples with similar information are deleted, over-training is avoided. The PBL algorithm helps to reduce the computational effort used in training. For simulation studies, we have used MR images from the Autism Brain Imaging Data Exchange (ABIDE) data set. The performance of the PBL-McRBFN classifier is evaluated on complete morphometric features set obtained from the VBM analysis. The performance evaluation study clearly indicates the superior performance of PBL-McRBFN classifier over other classification algorithms.

I. INTRODUCTION

Autism Spectrum Disorder (ASD), is a highly genetic neuro-developmental condition that is relatively common, affecting 1 in 150 children. ASD is characterised by impaired social communication, social reciprocity and repetitive stereotypic behaviour. Motor function, attention and other cognitive domains may also be affected. Early detection of ASD using non-invasive methods play a major role in providing treatment that may slow down its progress. Traditionally, ASD is diagnosed solely on the basis of behavioural criteria. This can be time consuming and problematic. The gold-standard diagnostic instruments such as the Autism Diagnostic Observation Schedule [1] and the Autism Diagnostic Interview [2] are dependent on the skill of the examiner and eliciting a developmental history from informants who know the person well. Hence a non-invasive method of early detection is needed. One such non-invasive method of early detection of ASD is by brain imaging. Magnetic Resonance Imaging (MRI) is the most important brain imaging proce-
shown in SRAN and CSRAN that the selecting appropriate samples for learning and removing repetitive samples helps in improving the generalization performance. Therefore, it is imperative that emulating the three components of meta-cognition with suitable strategies would improve the generalization ability of a neural network. The Meta-cognitive Neural Network (McNN) in [11], Meta-cognitive Fully Complex-valued Radial Basis Function (Mc-FCRBF) network [12] and Meta-cognitive neuro-Fuzzy Inference System (McFIS) [13] 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. Recently a ‘Projection Based Learning Meta-cognitive Radial Basis Function Network’ (PBL-McRBFN) [14] [15] [16] was proposed that addresses the three components of meta-cognition with less computational effort. Unlike other algorithms that require the number of hidden neurons to be fixed a priori, the Projection Based Learning (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 [17]. The problem of finding optimal weights is first formulated as a linear programming problem. 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.

In this paper, we apply the PBL-McRBFN for accurate classification of ASD patients and healthy persons. In our work, VBM analysis is performed on the MRI volumes of 79 ASD patients and 105 healthy persons obtained from the ABIDE (Autism Brain Imaging Data Exchange) [18] consortium, contributed by New York University Langone Medical Centre. Voxel locations of the VBM detected brain regions are used as features and then used as input to the PBL-McRBFN classifier. Finally the performance of PBL-McRBFN on ASD classification has been compared with the results based on traditional classifiers. The results clearly indicate the better performance of the PBL-McRBFN classifier. This paper is structured as follows: Section 2 briefly describes the framework of the proposed method including VBM feature extraction and the PBL-McRBFN classifier. Section 3 describes the main results from this study using the PBL-McRBFN classifier and compares with other classification methods such as decision trees, lazy algorithms and the SVM. Section 4 summarizes the conclusion from this study.

II. MATERIALS AND METHODS
A. Materials and Image Acquisition
MRI scans used in this study were obtained from the New York University Langone Medical Centre, Autism Brain Imaging Data Exchange (ABIDE) data set. The dataset from New York University Langone Medical Center consists of MR images of 184 subjects. Among them, 79 persons were diagnosed with ASD while the remaining 105 were healthy. Among the 79 people diagnosed with ASD, 53 diagnosed with Autistic disorder, 21 with Asperger’s disorder and 5 with pervasive developmental disorder. The considered data set demographics are summarized in Table I.

<table>
<thead>
<tr>
<th>Group</th>
<th>Healthy Persons</th>
<th>ASD Patients</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Persons</td>
<td>105</td>
<td>79</td>
</tr>
<tr>
<td>Age(Range)</td>
<td>31.8–6.5</td>
<td>39.1–7.1</td>
</tr>
<tr>
<td>Gender (Male/Female)</td>
<td>79/26</td>
<td>68/11</td>
</tr>
</tbody>
</table>

B. Feature Extraction using VBM
Voxel-Based Morphometry (VBM) is an automated image analysis approach for identification of regional differences in gray matter between groups of subjects [4]. In our VBM approach, unified segmentation, smoothing and statistical testing steps are performed. In the unified segmentation step, tissue segmentation, bias correction and image registration are combined in a single general model [19]. The segmented and registered gray matter images are then smoothed by convolving with a 10 mm full-width at half-maximum isotropic Gaussian kernel. In our statistical testing, the whole brain group comparison between normal persons and ASD patient is conducted with two-sample t-test and gray matter volume is used as the covariate. Maximum intensity projections ($P < 0.001$, uncorrected) of significant areas with increased gray matter density in the normal persons relative to the ASD patients are shown in Fig. 1.

To locate the above regions with respect to the spatial locations in the brain, these regions are overlaid on the sliced sections of the commonly used Montreal Neurological Institute (MNI) brain template and the results of the same are shown in Fig. 2.

From the voxel locations of the significant areas obtained from unified VBM, gray matter tissue probability values are extracted as features. A total of 268 morphometric features are obtained. These features are then used as input to the classifier.

C. Projection Based Learning Meta-cognitive Radial Basis Function Network Classifier
The ASD detection problem can be viewed as a mapping from the domain of the features to the domain of the class labels. A classifier is needed to approximate the underlying decision function that maps $x^f \in \mathbb{R}^m \rightarrow y^t \in \mathbb{R}^n$. The Meta-cognitive Radial Basis Function Network Classifier (McRBFN) is a neural network with single hidden layer of neurons with radial basis activation function. It begins with zero hidden neurons and selects suitable strategy for each sample (adds neurons or update output parameters or deletes
The McRBFN has two components, viz. the cognitive component and the meta-cognitive component. 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 employing the Gaussian activation function for the hidden layer. The input and output layers are linear.

We can assume, without loss of generality, that the McRBFN builds $K$ Gaussian neurons from $t-1$ training samples. For a given input $x_t$, the predicted output $\hat{y}_j^t$ of McRBFN is

$$\hat{y}_j^t = \sum_{k=1}^{K} w_{kj} h_t^k, \quad j = 1, 2, \ldots, n$$

(1)

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

$$h_t^k = \exp\left(-\frac{\|x_t - \mu_l^k\|^2}{(\sigma_l^k)^2}\right)$$

(2)

where $\mu_l^k \in \mathbb{R}^m$ is the center and $\sigma_l^k \in \mathbb{R}^+$ 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 parameters of the cognitive component are chosen by a learning process called Projection Based Learning (PBL) algorithm. The PBL algorithm is described as follows.

**Projection Based Learning Algorithm:** The PBL algorithm formulates the sum of squared errors (SSE) at the McRBFN output neurons as its energy function and finds the network output parameters for which the energy function is minimum. The SSE at the McRBFN output layer is given by,

$$J(W) = \frac{1}{2} \sum_{i=1}^{t} \sum_{j=1}^{n} \left( y_i^j - \sum_{k=1}^{K} w_{kj} h_t^k \right)^2$$

(3)

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

$$W^* = \arg\min_{W \in \mathbb{R}^{K \times n}} J(W)$$

(4)

In order to obtain the optimal $W^*$ corresponding to the minimum energy point of the energy function ($J(W^*)$), the first order partial derivative of $J(W)$ with respect to the output weight is equated to zero, i.e.,

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

(5)
After solving Eq. (5), we can obtain
\[ \sum_{k=1}^{K} x_t k w_{kj} = \sum_{i=1}^{t} h_i p_l y_j \]  
Eq. (6) can be written as
\[ \sum_{k=1}^{K} a_{kp} w_{kj} = b_{pj}, \quad p = 1, \ldots, K; \quad j = 1, \ldots, n \]  
which can be represented in matrix form as
\[ AW = B \]  
where the projection matrix \( A \in \mathbb{R}^{K \times K} \) is given by
\[ a_{kp} = \sum_{i=1}^{t} h_i p_l k, \quad k = 1, \ldots, K; \quad p = 1, \ldots, K \]  
and the output matrix \( B \in \mathbb{R}^{K \times n} \) is
\[ b_{pj} = \sum_{i=1}^{t} h_i p_l y_j, \quad p = 1, \ldots, K; \quad j = 1, \ldots, n \]  
Eq. (7) gives the set of \( K \times n \) linear equations with \( K \times n \) unknown output weights \( W \). Note that the projection matrix is always a square matrix of order \( K \times K \).

The solution for the system of equations in Eq. (8) can be determined as follows:
\[ W^* = A^{-1} B \]  

2) Meta-cognitive component of McRBFN: The meta-cognitive component uses three measures of knowledge for every training sample, namely, the estimated class label \( \hat{c} \), maximum hinge error \( (E^i) \) and spherical potential based class-wise significance. Using these measures, the meta-cognitive component controls the learning process of the cognitive component by addressing, what-to-learn, when-to-learn and how-to-learn properly. A suitable strategy is selected for each training sample when it is presented to the network for learning.

Estimated Class label \( \hat{c} \): Using the predicted output \( \hat{y}^t \), the estimated class label \( \hat{c} \) can be obtained as
\[ \hat{c} = \text{arg} \max_{j \in 1,2,\ldots,n} \hat{y}_j^t \]  

Maximum Hinge Error \( (E^i) \): The objective of the classifier is to minimize the error between the predicted output \( \hat{y}_j^t \) and actual output \( y_j^t \). It has been shown in [17] hinge loss function can be used to arrive at more accurate estimates of the posterior probability compared to mean square error function for classification problems. Hence, the McRBFN uses the hinge loss error \( e_t^i = [e_1^t, \ldots, e_j^t, \ldots, e_n^t]^T \in \mathbb{R}^n \) defined as
\[ e_j^t = \begin{cases} 0 & \text{if } y_j^t - \hat{y}_j^t > 1 \\ y_j^t - \hat{y}_j^t & \text{otherwise} \end{cases} \quad j = 1,2,\ldots,n \]  
The maximum absolute hinge error \( (E^i) \) is given by
\[ E^t = \max_{j \in 1,2,\ldots,n} |e_j^t| \]  

Class-wise Significance \( \psi_c \): It has been shown in [20] that the class-wise distribution will influence the performance the classifier significantly. Hence, the McRBFN uses the measure of the spherical potential of the new training sample \( x^t \) 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 \( \psi_c \) is defined as
\[ \psi_c = \frac{1}{K^c} \sum_{k=1}^{K^c} h(x^t, \mu_k^c) \]  
The spherical potential explicitly indicates the knowledge contained in the sample, a spherical potential closer to one indicates that the sample is similar to the existing knowledge in the cognitive component and a spherical potential closer to zero indicates that the sample is novel. For more details on the class-wise significance of McRBFN, one can refer to [11].

3) Learning Strategies: The meta-cognitive part controls the learning process in cognitive component by selecting one of the following four learning strategies.

- 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: The new training sample contains significant novel information to necessitate the addition 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.

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

Sample Delete Strategy: When the class label of the new training sample is predicted accurately and the maximum hinge error is very small, it can be inferred that 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. Sample deletion helps to avoid over-training and reduces the computational effort. The sample deletion criterion is given by
\[ e^t = c^t \quad \text{AND} \quad E^t \leq \beta_d \]  
The meta-cognitive deletion threshold \( \beta_d \) is used to determine the number of samples participating in the learning process. If \( \beta_d \) is close to 0 then almost all of the training samples will participate in the learning process which can result in over-training due to similar samples. Increasing \( \beta_d \) beyond the desired accuracy results in deletion of too many samples from the training sequence which might hinder the resultant accuracy. Hence, this threshold is chosen based on
the expected accuracy level, \( \beta_d \) is selected in the interval [0.1 - 0.2].

**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

\[
(\hat{c}^t \neq c^t \text{ OR } E^t \geq \beta_a) \text{ AND } \psi_c(x^t) \leq \beta_c
\]

where \( \beta_c \) is the meta-cognitive knowledge measurement threshold and \( \beta_a \) is the self-adaptive meta-cognitive addition threshold. The thresholds \( \beta_c \) and \( \beta_a \) are used to select samples with significant knowledge for learning first and then use the other samples for fine tuning. If \( \beta_c \) is closer to zero and the initial value of \( \beta_a \) is closer to the maximum value of hinge error, then very few neurons will be added and result in an inaccurate network. If \( \beta_a \) is chosen closer to one and the initial value of \( \beta_a \) is chosen closer to the minimum value of hinge error, it may result in too many neurons and a network with poor generalization ability. Hence, \( \beta_a \) can be selected in the interval \([0.3 - 0.7]\) and the initial value of \( \beta_a \) can be selected in the interval \([1.3 - 1.7]\). \( \beta_a \) is adapted based on the prediction error as:

\[
\beta_a := \delta \beta_a + (1 - \delta) E^t
\]

where \( \delta \) 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 either be covered partially by existing neurons (overlap) for other classes or it could be from a distinct cluster far away from the nearest neuron in the same class. These conditions affects the classification performance of a classifier significantly. 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

\[
nrS = \arg \min_{t=c \neq k} ||x^t - \mu^c_t||; \quad nrI = \arg \min_{t \neq c \neq k} ||x^t - \mu^c_t||
\]

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

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

Using the nearest neuron distances, the overlapping condition can be determined as follows:

- **No-overlap with any class**: If the training sample is far away from both intra/inter class nearest neurons \((d_S >> \sigma^c_{nrS} \text{ AND } d_I >> \sigma^c_{nrI})\) then it is from a distinct cluster and does not overlap with any class cluster. In this case, the new hidden neuron center \((\mu^c_{K+1})\) and width \((\sigma^c_{K+1})\) parameters are determined as

\[
\mu^c_{K+1} = x^t; \quad \sigma^c_{K+1} = \kappa \sqrt{x^t x^t}
\]

where \( \kappa \) 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-overlap with the inter-class**: For a new training sample, if the intra/inter class distance ratio is less than 1, i.e it is closer to the nearest neuron of the same class than a neuron from other class, then the sample does not overlap with the other classes. In this case, the new hidden neuron center \((\mu^c_{K+1})\) and width \((\sigma^c_{K+1})\) parameters are determined as

\[
\mu^c_{K+1} = x^t; \quad \sigma^c_{K+1} = \kappa ||x^t - \mu^c_{nrS}||
\]

- **Minimum Overlap with the inter-class**: For a new training sample, if the intra/inter class distance ratio is in the range 1 to 1.5, it is closer to the nearest neuron of a different class, compared to the nearest neuron in the same class. In this case, the sample has a small overlap with the other class and hence the center of the new hidden neuron is shifted away from the inter-class nearest neuron and shifted towards the intra-class nearest neuron as

\[
\mu^c_{K+1} = \mu^c_{nrS} + \zeta (\mu^c_{nrS} - \mu^c_{nrI})
\]

\[
\sigma^c_{K+1} = \kappa ||\mu^c_{K+1} - \mu^c_{nrS}||
\]

where \( \zeta \) 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 matrix \( A \) is increased from \( K \times K \) to \((K + 1) \times (K + 1)\)

\[
A_{(K+1)\times(K+1)} = \begin{bmatrix}
A_{K\times K} + (h^T)^T h^T & a^T_{K+1} \\
\text{a}_{K+1} & \text{a}_{K+1}^T
\end{bmatrix}
\]

where \( h^T = [h^T_1, h^T_2, \ldots, h^T_t] \) is a vector of the existing \( K \) hidden neurons response for current \((t^{th})\) training sample. \( a_{K+1} \in \mathbb{R}^{1 \times K} \) is assigned as

\[
a_{K+1,p} = \sum_{i=1}^{t} h^T_{K+1} h^T_i, \quad p = 1, \ldots, K
\]

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

\[
a_{K+1,K+1} = \sum_{i=1}^{t} h^T_{K+1} h^T_i
\]

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

\[
B_{(K+1)\times n} = \begin{bmatrix}
B_{K \times n} + (h^T)^T (y^T)^T & \text{b}_{K+1}^T
\end{bmatrix}
\]
where \( b_{K+1} \in \mathbb{R}^{1 \times n} \) is a row vector assigned as
\[
b_{K+1,j} = \sum_{i=1}^{t} h_{K+1,i}^j, \quad j = 1, \ldots, n
\] (28)

Finally the output weights are estimated as
\[
\begin{bmatrix} W_K \\ w_{K+1} \end{bmatrix} = (A_{(K+1) \times (K+1)})^{-1} B_{(K+1) \times n}
\] (29)

After calculating inverse of the matrix \( A_{(K+1) \times (K+1)} \) recursively using matrix identities, the resultant equations are
\[
W_K = \left[ I_{K \times K} + \frac{(A_{K \times K})^{-1} a_{K+1}^T a_{K+1}}{\Delta} \right]^{-1}
\]
\[
\left[ W_K + (A_{K \times K})^{-1} (h^i)^T (y^i)^T \right] - \frac{(A_{K \times K})^{-1} a_{K+1}^T b_{K+1}}{\Delta}
\]
\[
w_{K+1} = -\frac{a_{K+1}}{\Delta}
\]
\[
\Delta = a_{K+1,1} \cdots a_{K+1,n} \left( A_{K \times K} + (h^i)^T h^i \right)^{-1} a_{K+1,1}^T \cdots a_{K+1,n}^T
\] (30)

**Parameters Update Strategy** The current \((t^{th})\) training sample is used to update the output weights of the cognitive component \((W_K = [w_1, w_2, \cdots, w_K]^T)\) if the following criterion is satisfied.
\[
c_t \leq c \quad \text{AND} \quad E_t \geq \beta_u
\] (32)

where \( \beta_u \) is the self-adaptive meta-cognitive parameter update threshold. If \( \beta_u \) is chosen closer to 50% of maximum hinge error, then very few samples will be used for adapting the network parameters hence reducing the accuracy of the network. If a lower value is chosen, most of the samples will be used for updating the network parameters without altering the training sequence, resulting in higher computational load. Hence, the initial value of \( \beta_u \) can be selected in the interval [0.4 - 0.7]. The \( \beta_u \) is adapted based on the prediction error as:
\[
\beta_u := \delta \beta_u + (1 - \delta) E_t
\] (33)

The PBL algorithm updates the output weight parameters as follows:

The matrices \( A \in \mathbb{R}^{K \times K} \) and \( B \in \mathbb{R}^{K \times n} \) are updated as
\[
A = A + (h^i)^T h^i; \quad B = B + (h^i)^T (y^i)^T
\] (34)

and the output weights are updated as
\[
W_K = W_K + A^{-1} (h^i)^T (e^i)^T
\] (35)

**Sample Reserve Strategy** If the new training sample does not satisfy any of the deletion, neuron growth or the 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. A more detailed description of the PBLMcRBFN is available at [14].

### III. EXPERIMENTAL RESULTS

In our study, ASD detection using PBL-McRBFN classifier is performed on the MRI scans obtained from the New York University Langone Medical Centre, ABIDE data set. We have considered 184 subjects, of which 105 are normal persons and 79 are ASD patients. VBM analysis was performed on normal persons and ASD patients using Statistical Parametric Map (SPM) software package [21]. VBM detected brain regions are used as mask for feature extraction. The 268 morphometric features extracted from the MRI scans using the VBM analysis [4] represent the regions in the whole brain where gray matter volume changes are significant. The extracted features are used as input features to train the PBL-McRBFN classifier. In our classification experiments, 75% of the samples are chosen randomly as the training set and the rest as testing set at each trial. The performance of PBL-McRBFN classifier is studied by generating 10 random trials of the training and testing sets. The PBL-McRBFN classifier is implemented in JAVA and its performance is compared with some well known classification algorithms implemented on WEKA 3.7 [22]. The simulations are run on a laptop with Intel Core i7 quad core processor, 2.1 GHz CPU and 4 GB RAM.

**A. Performance of PBL-McRBFN**

The complete data set consists of 184 samples with 268 features. For classification study, 10 random trials of the training and testing tests were used. The outcome of a classifier is usually represented in the form of a confusion matrix. There are many measures derived from this matrix for evaluating the performance of a classifier. These performance measures are summarised in the Table II. Table III presents the mean, standard deviation (STD) and the maximum value of classification accuracy obtained for various algorithms, on the training and testing datasets. Similarly Table IV, Table V and Table VI summarize the specificity, sensitivity and the F-measure for the various algorithms. We compare the performance of the PBL-McRBFN with the J48[23] decision tree algorithm, Random Forest (R-Forest) [24] algorithm with 100 trees, SVM [25] and lazy algorithms like Naive Bayes [26] algorithm and the K-nearest neighbour (K-NN) [27] algorithm with two different values of K (1 and 5). For SVM, an RBF kernel with the parameters (c=1 and gamma=0.07) is chosen. All the parameters are set using the WEKA explorer GUI.

Next, we compare the performance of PBL-McRBFN with the other algorithms based on these performance measures.

**Accuracy:** The training and testing accuracies of the various classifiers used in the study is reported in Table III. From the table, we can observe that the random forest, J48 and the 1-NN algorithm achieve a mean accuracy very close to 100% when evaluated on the training dataset itself. The PBL-McRBFN closely follows them with a mean accuracy of 97%. The random forest, 1-NN and the PBL-McRBFN achieve the maximum possible accuracy of 100% in at
least one of the training datasets. When the algorithms are evaluated on the testing dataset, the PBL-McRBFN achieves significantly better mean accuracy of 70% followed by SVM with a mean accuracy of 62%, and the random forest algorithm 61%. The PBL-McRBFN also achieves the highest value for accuracy 78% on the testing dataset. SVM, random forest and the KNN (K=5) methods achieved a peak accuracy of 74% on one of the trials. Thus PBL-McRBFN is clearly more accurate than the other classifiers used in this study.

### TABLE II
CLASSIFIER PERFORMANCE MEASURES.

<table>
<thead>
<tr>
<th>Test result +ve</th>
<th>Actually positive (tp)</th>
<th>Actually negative (fn)</th>
<th>precision = ( \frac{tp}{tp + fp} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-ve</td>
<td>True positive (tp)</td>
<td>False positive (fp)</td>
<td>F-measure = ( \frac{2tp}{2tp + fn + fp} )</td>
</tr>
</tbody>
</table>

Sensitivity: The training and testing sensitivity of the various classifiers used in the study is reported in Table III. From the table, we can observe that the random forest, J48 and the 1-NN algorithm achieve a mean specificity very close to 1 when evaluated on the training dataset itself. The PBL-McRBFN closely follows with a mean specificity of 0.97. The random forest, J48, 1-NN and the PBL-McRBFN achieve the maximum possible specificity of 1 in at least one of the training datasets. However, the PBL-McRBFN achieves significantly better mean specificity when evaluated on the testing dataset. The PBL-McRBFN has a mean specificity of 0.72 while the next best is the Naive Bayes method with a mean specificity of 0.65. The PBL-McRBFN also achieves the highest possible specificity (0.8) on the testing dataset.

Specificity: The training and testing specificity of the various classifiers used in the study is reported in Table IV. From the table, we can observe that the random forest, J48 and the 1-NN algorithm achieve a mean specificity very close to 1 when evaluated on the training dataset itself. The PBL-McRBFN closely follows with a mean specificity of 0.97. The random forest, J48, 1-NN and the PBL-McRBFN achieve the maximum possible specificity of 1 in at least one of the training datasets. However, the PBL-McRBFN achieves significantly better mean specificity when evaluated on the testing dataset. The PBL-McRBFN has a mean specificity of 0.72 while the next best is the Naive Bayes method with a mean specificity of 0.65. The PBL-McRBFN also achieves the highest possible specificity (0.8) on the testing dataset.

Sensitivity: The training and testing values of sensitivity for the various classifiers used in the study is reported in Table V. From the table, we can observe that the PBL-McRBFN has a mean sensitivity of 0.95, over the 10 trials when evaluated on the training dataset itself. Other algorithms such as 1-NN and decision trees have a sensitivity close to one. Considering the maximum sensitivity obtained by the algorithms, the PBL-McRBFN, Random forest, J48 and the KNN (K=1) obtain the highest possible value of 1. However, the performance of all algorithms drop significantly when evaluated on the testing dataset. When evaluated on the testing dataset, the Naive Bayes algorithm has the highest value of mean sensitivity over all the 10 trials (0.59) while PBL-McRBFN closely follows it (0.53). PBL-McRBFN achieves the highest possible sensitivity (0.83) on the testing dataset. SVM has the lowest value of sensitivity due to a large number of false-negatives.

### Table III
PERFORMANCE STUDY OF PBL-McRBFN CLASSIFIER BASED ON ACCURACY.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Training Accuracy %</th>
<th>Testing Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Mean</td>
</tr>
<tr>
<td>McRBFN</td>
<td>100</td>
<td>97</td>
</tr>
<tr>
<td>SVM</td>
<td>90</td>
<td>85</td>
</tr>
<tr>
<td>R-Forest</td>
<td>100</td>
<td>99.96</td>
</tr>
<tr>
<td>J48</td>
<td>99</td>
<td>98</td>
</tr>
<tr>
<td>1-NN</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>5-NN</td>
<td>82</td>
<td>74</td>
</tr>
<tr>
<td>N-Bayes</td>
<td>83</td>
<td>73</td>
</tr>
</tbody>
</table>

F-Measure: The training and testing F-measure of the various classifiers used in the study is reported in Table VI. From the table, we can observe that the random forest, J48 and the 1-NN algorithm achieve a mean F-measure very close to 1 when evaluated on the training dataset itself. The PBL-McRBFN closely follows with a mean F-measure of 0.97. If we look at the maximum possible training specificity obtained, these 4 algorithms obtain the highest possible value of 1. However, the PBL-McRBFN achieves significantly better mean F-measure when evaluated on the testing dataset. The PBL-McRBFN has a mean F-measure of 0.58 while the next best is the Naive Bayes algorithm with a mean F-measure of 0.53. The PBL-McRBFN also achieves the highest possible F-measure (0.76) on the testing dataset while...
the next highest is 0.68. Thus the PBL-McRBFN achieves a significantly higher performance when compared with the other methods.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Training F-measure</th>
<th>Testing F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Mean</td>
</tr>
<tr>
<td>McRBFN</td>
<td>1</td>
<td>0.97</td>
</tr>
<tr>
<td>SVM</td>
<td>0.87</td>
<td>0.79</td>
</tr>
<tr>
<td>R-Forest</td>
<td>1</td>
<td>0.996</td>
</tr>
<tr>
<td>J48</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>1-NN</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5-NN</td>
<td>0.78</td>
<td>0.69</td>
</tr>
<tr>
<td>N-Bayes</td>
<td>0.79</td>
<td>0.71</td>
</tr>
</tbody>
</table>

Over the 10 trials, the PBL-McRBFN used an average of 70 neurons in the hidden layer, while the SVM used an average of 131 support vectors to build the classification model.

Thus from the results above we can see that the PBL-McRBFN achieves around 5 to 8% improvement in classification accuracy and the F-measure, when compared with the existing methods. While conventional methods achieve excellent performance on the training dataset, they tend to falter when evaluated on the testing dataset. The PBL-McRBFN demonstrates better performance when evaluated on the testing dataset.

IV. CONCLUSIONS

This paper has presented an approach for ASD detection using the PBL-McRBFN classifier, based on complete VBM detected features from MRI scans. 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. The PBL algorithm helps to reduce the computational effort used in training. The performance of PBL-McRBFN is compared with some well-known classifiers in the literature. The results clearly indicate the superior performance of the PBL-McRBFN classifier for diagnosis of individuals with or without ASD.

REFERENCES