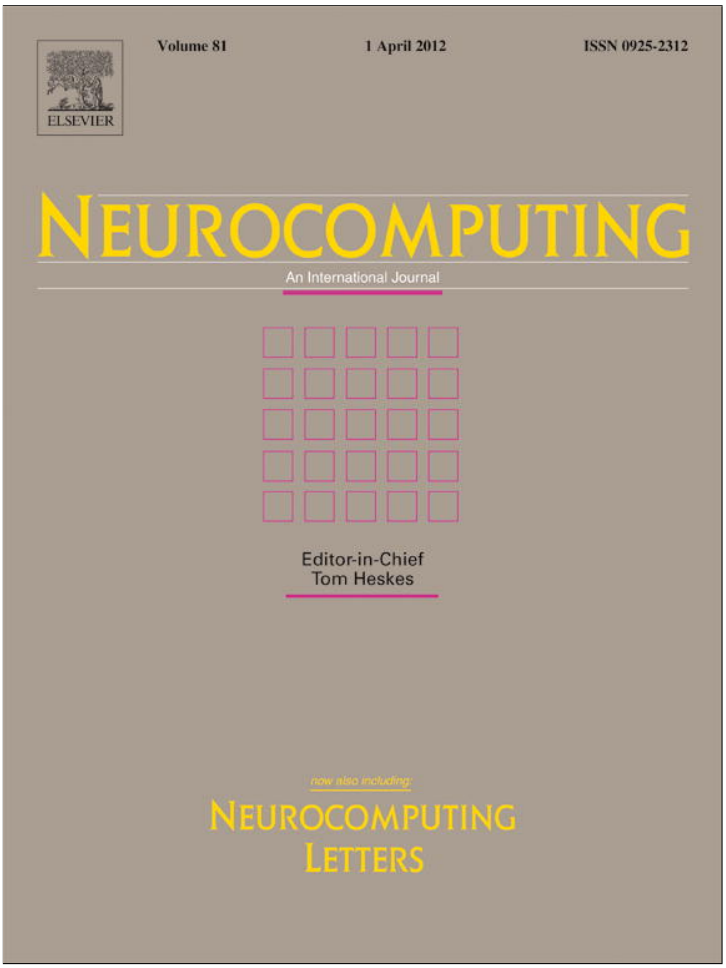

Neurocomputing 81 (2012) 86–96

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Meta-cognitive Neural Network for classification problems in a sequential learning framework

G. Sateesh Babu, S. Suresh *

School of Computer Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 29 June 2011
Received in revised form
7 October 2011
Accepted 13 December 2011
Communicated by G.-B. Huang
Available online 24 December 2011

Keywords:
Meta-cognitive learning
Self-regulatory thresholds
Radial basis function network
Multi-category classification problem
Sequential learning

ABSTRACT

In this paper, we propose a sequential learning algorithm for a neural network classifier based on human meta-cognitive learning principles. The network, referred to as Meta-cognitive Neural Network (McNN). McNN has two components, namely the cognitive component and the meta-cognitive component. A radial basis function network is the fundamental building block of the cognitive component. The meta-cognitive component controls the learning process in the cognitive component by deciding *what-to-learn*, *when-to-learn* and *how-to-learn*. When a sample is presented at the cognitive component of McNN, the meta-cognitive component chooses the best learning strategy for the sample using estimated class label, maximum hinge error, confidence of classifier and class-wise significance. Also sample overlapping conditions are considered in growth strategy for proper initialization of new hidden neurons. The performance of McNN classifier is evaluated using a set of benchmark classification problems from the UCI machine learning repository and two practical problems, viz., the acoustic emission for signal classification and a mammogram data set for cancer classification. The statistical comparison clearly indicates the superior performance of McNN over reported results in the literature.

1. Introduction

In supervised learning, artificial neural networks became powerful tools to model complex input–output relationships by learning. Hence they are being increasingly employed to solve classification tasks to learn the decision surface that maps set of input features to class labels [1,2]. 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 [3].

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 a stream of training samples. Radial basis function networks have been extensively used in a sequential learning framework due to its universal approximation ability and simplicity of architecture. Many sequential learning algorithms in radial basis function framework are available in the literature to solve function approximation or classification problems [4–10].

Resource Allocation Network (RAN) [4] was the one of the first sequential learning algorithm introduced in the literature. RAN evolves the network architecture required to approximate the true function using novelty based neuron growth criterion. The Minimal Resource Allocation Network (MRAN) [5] and the Extended Minimal Resource Allocation Network (EMRAN) [6] use a similar approach, but these algorithms incorporate error based neuron growing/pruning criterion. Hence, they determine compact network architecture than RAN algorithm. In Growing and Pruning Radial Basis Function Network (GAP-RBFN) [7], growing/pruning criteria of the network is selected based on the significance of a neuron. Aforementioned algorithms were developed for the function approximation problems, and may not work well for classification problems [9]. On the other hand, in Sequential Multi-Category Radial Basis Function Network (SMC-RBFN) [9], the similarity measures within class, misclassification rate and prediction error are used in neuron growing and parameter update criterion. It has been shown in [9] that updating the nearest neuron parameters in the same class as that of current sample helps in improving the performance than updating a nearest neuron in any class.

Another well known paradigm in fast learning neural network is extreme learning machine (ELM) [11]. It is a batch learning algorithm for a single-hidden layer feed forward neural network.

* Corresponding author. Tel.: +65 6790 6185.
E-mail address: ssundaram@ntu.edu.sg (S. Suresh).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.12.001

ELM chooses input weights randomly and analytically determines the output weights using minimum norm least-squares. It has been extended to sequential framework using recursive least squares in [8]. In case of sparse and imbalance data sets, the random selection of input weights in the ELM and its variants affects the performance significantly [12]. A complete survey of research works in ELM framework are presented in [13].

Another widely used algorithm to solve classification problems is support vector machines (SVM). A sequential learning version in support vector machine framework is called an incremental and decremental support vector machine learning was presented in [14]. It uses an on-line recursive algorithm for training support vector machines, and it handles one sample at a time by retaining Karush–Kuhn–Tucker conditions on all previously seen training data. Recently in [15], a multiple incremental decremental learning of support vector machines was proposed. Here, multiple samples are added or removed simultaneously and is faster than the conventional incremental decremental support vector machines presented in [14].

Aforementioned sequential learning algorithms in radial basis function network, ELM and SVM frameworks address *how-to-learn* the decision function from stream of training samples. In Self-adaptive Resource Allocation Network (SRAN) [10] significant samples are selected using misclassification error and hinge loss function. It has been shown in [10] that the selection of appropriate samples by removing repetitive samples helps in achieving better generalization performance. Therefore, an efficient classifier must also be capable of judging what samples to learn and when to learn, during the training process. Hence, there is a need to develop a learning algorithm which automatically selects appropriate samples for learning and adopt best learning strategy to learn them accurately. Similar observations in complex domain are also reported in the literature [16,17].

Recent studies in human learning suggested that the learning process is effective when the learners adopt self-regulation in learning process using meta-cognition [18]. The term *meta-cognition* is defined in [19] as 'one's knowledge concerning one's own cognitive processes or anything related to them'. Precisely the learner should control the learning process, by planning and selecting learning strategies and monitor their progress by analyzing the effectiveness of the proposed learning strategies. When necessary, these strategies should be adapted appropriately. Meta-cognition present in human-being provides a means to address *what-to-learn*, *when-to-learn* and *how-to-learn*, i.e., the ability to identify the specific piece of required knowledge, judge when to start and stop learning by emphasizing best learning strategy. Hence, there is a need to develop a Meta-cognitive Neural Network classifier that is capable of deciding *what-to-learn*, *when-to-learn* and *how-to-learn* the decision function from the training data.

In this paper, we introduce a Meta-cognitive Neural Network (McNN) classifier which employs human-like meta-cognition to regulate the sequential learning process. McNN has two components namely the cognitive component and the meta-cognitive component. The cognitive component of McNN is a single hidden layer radial basis function network. The cognitive component adds neurons and updates the parameters of the network so as to approximate the decision surface described by the stream of training data. The meta-cognitive component of McNN measures the knowledge contained in the current training sample with respect to the cognitive component using estimated class label, maximum hinge error, confidence of classifier and class-wise significance. Class-wise significance is obtained from spherical potential, which is used widely in kernel methods to determine whether all the data points are enclosed tightly by the Gaussian kernels [20]. Here, the squared distance between the current sample and the hyper-dimensional projection helps in measuring the novelty in the data. Since, McNN address classification problems, we redefine the spherical potential in class-wise and is used in devising the learning strategy. In addition, the meta-cognitive component identifies the overlapping/non-overlapping criterion by measuring the distance from nearest neuron in the inter/intra-class. Using the above-mentioned measures the meta-cognitive component constructs two sample based learning strategies and two neuron based learning strategies. One of these strategies is selected for the current training sample such that the cognitive component learn them accurately and achieves better generalization performance.

Performance of the proposed McNN classifier is evaluated using set of benchmark binary/multi-category classification problems from the University of California, Irvine (UCI) machine learning repository [21]. For this purpose, we consider five multi-category classification problems and five binary classification problems with varying values of imbalance factor. The training sample imbalance between different classes are measured using imbalance factor [9]. The performance of McNN classifier on these benchmark data sets are compared with the existing sequential learning algorithms in the literature using class-wise performance measures like overall/average efficiency and a non-parametric statistical significance test [22]. The non-parametric Friedman test based on the mean ranking of each algorithm over multiple data sets [22] indicate the statistical significance of the proposed McNN classifier. Finally, the performance of McNN classifier has also been evaluated using two practical classification problems viz., the acoustic emission signal classification problem [23] and the mammogram classification problem for breast cancer detection [24]. The results clearly highlight that McNN classifier provides a better generalization performance than the results reported in the literature.

This paper is organized as follows: Section 2 describes the proposed McNN classifier. Section 3 presents performance evaluation of the proposed classifier for benchmark multi-category problems, binary classification problems and practical applications. Section 4 summarizes the conclusions from this study.

2. Meta-cognitive Neural Network (McNN) classifier

In this section, we describe the Meta-cognitive Neural Network classifier for solving multi-category classification problems. First, we define the classification problem. Next, we present the Meta-cognitive Neural Network architecture. Finally, we present the learning algorithm which addresses the three fundamental principles of meta-cognition in the learning process.

2.1. Problem definition

Let us consider the sequence of training samples $\{(\mathbf{x}^1,c^1),\ldots,(\mathbf{x}^i,c^i),\ldots,(\mathbf{x}^N,c^N)\}$, where $\mathbf{x}^i=[x_1^i,\ldots,x_m^i]^T\in\Re^m$ be the m-dimensional feature vector of ith training sample and c^i be its class label. Note, N is the total training samples available in the data stream. In a standard classification problem, the class label (c^i) of the sample i is converted into a coded class label ($\mathbf{y}^i=[y_1^i,\ldots,y_j^i,\ldots,y_n^i]^T$) as

$$y_j^i=\begin{cases}1 & \text{if } j=c^i,\\ -1 & \text{otherwise,}\end{cases}\quad j=1,2,\ldots,n \tag{1}$$

where n is the number of distinct class labels present in the training samples.

The goal is to determine a suitable discriminant function $f(\mathbf{x}):\mathbf{x}\in\Re^m\rightarrow\mathbf{y}\in\Re^n$, from a class of functions $\mathbb{F}$, such that $f(\mathbf{x})$ accurately predicts the class labels with a certain degree of confidence. In this paper, we employ a meta-cognitive neural network as a classifier. The learning algorithm finds an

appropriate number of hidden neurons required to approximate the discriminant function.

2.2. Meta-cognitive Neural Network architecture

In this section, we present the architecture of the Meta-cognitive Neural Network (McNN) classifier and its working principles. McNN architecture is developed based on the Nelson and Narens meta-cognition model [25]. Fig. 1(a) shows the Nelson and Narens meta-cognition model which is analogous to the meta-cognition in human-beings and has two components, a cognitive component and a meta-cognitive component. The information flow from the cognitive component to meta-cognitive component is considered monitoring, while the information flow in the reverse direction is considered control. Similar to Nelson and Narens model, McNN has two components as shown in Fig. 1(b), namely the cognitive component and the meta-cognitive component. The cognitive component of McNN is a three layered feed forward radial basis function network with Gaussian activation function in the hidden layer. The meta-cognitive component contains copy of the cognitive component. When a new training sample arrives, the meta-cognitive component of McNN predicts the class label and estimate the knowledge present in the new training sample with respect to the cognitive component. Based on this information, the meta-cognitive component selects a suitable learning strategy, for the current sample. Thereby, addressing the three fundamental issues in learning process: (a) *what-to-learn*, (b) *when-to-learn* and (c) *how-to-learn*. First, we present the cognitive component and next we highlight various learning strategies of the meta-cognitive component.

2.2.1. Cognitive component of McNN

The cognitive component of McNN is a three layered feed forward radial basis function network. The input layer maps all features to the hidden layer without doing any transformation, the hidden layer employs Gaussian activation function and the output layer uses a linear activation function.

Without loss of generality, we assume that the meta-cognitive learning algorithm builds K Gaussian neurons from $i-1$ training samples. For given training sample $\mathbf{x}^i$, the predicted output ($\hat{\mathbf{y}}^i=[\hat{y}^i_1,\ldots,\hat{y}^i_j,\ldots,\hat{y}^i_n]^T$) of McNN classifier with K hidden neurons is

$$\hat{y}^i_j=\alpha_{j0}+\sum_{k=1}^{K}\alpha_{jk}\phi_k(\mathbf{x}^i),\quad j=1,2,\ldots,n \tag{2}$$

where α_{j0} is the bias to the jth output neuron, α_{jk} is the weight connecting the kth hidden neuron to the jth output neuron and $\phi_k(\mathbf{x}^i)$ is the response of the kth hidden neuron to the input $\mathbf{x}^i$ is given by

$$\phi_k(\mathbf{x}^i)=\exp\left(-\frac{\|\mathbf{x}^i-\boldsymbol{\mu}^l_k\|^2}{(\sigma^l_k)^2}\right) \tag{3}$$

where $\boldsymbol{\mu}^l_k$ is the center and σ^l_k is the width of the kth hidden neuron. Here, the superscript l represent the class that hidden neuron belongs to.

2.2.2. Meta-cognitive component of McNN

The meta-cognitive component uses estimated class label ($\hat{c}$), maximum hinge error (E), posterior probability as confidence measure ($\hat{p}(j|\mathbf{x}^i)$) and spherical potential [26,20] based class-wise significance as a measure of knowledge in the new training sample. Using these measures, the meta-cognitive component devices various learning strategies. First, we describe these measures in detail.

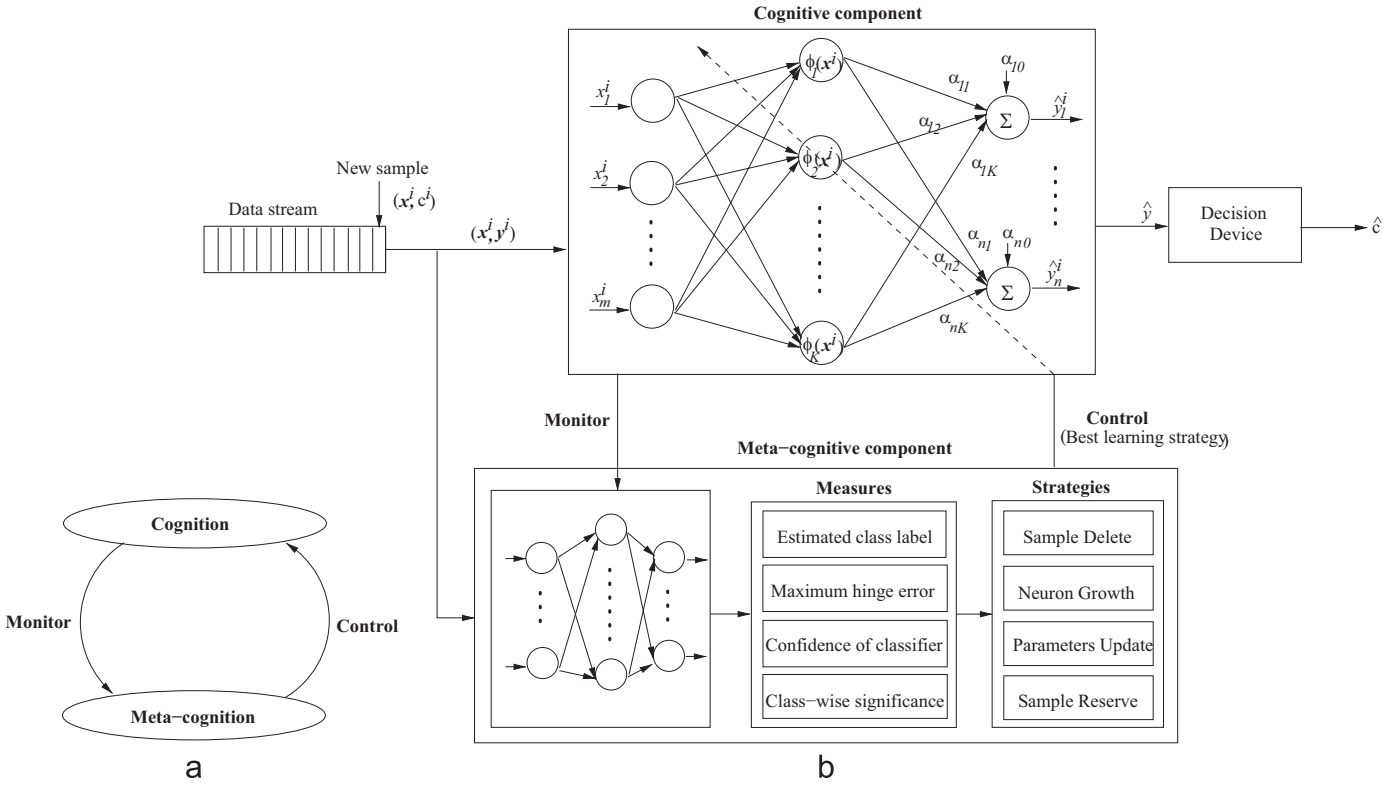

Fig. 1. (a) Nelson and Narens Model of meta-cognition. (b) Schematic diagram of McNN learning algorithm.

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

$$\hat{c} = \arg \max_{j \in 1,2,\ldots,n} \hat{y}_j^i \tag{4}$$

Maximum hinge error (E): The objective of the classifier is to minimize the error between the predicted output ($\hat{\mathbf{y}}^i$) and actual output ($\mathbf{y}^i$). In classification problems, it has been shown that the classifier developed using hinge loss function estimates the posterior probability more accurately than the classifier developed using mean square error function [27,28]. Hence, in McNN, we use the hinge loss error ($\mathbf{e} = [e_1, \ldots, e_j, \ldots, e_n]^T$) defined as

$$e_j = \begin{cases} 0 & \text{if } \hat{y}_j^i y_j^i > 1, \\ \hat{y}_j^i - y_j^i & \text{otherwise,} \end{cases} \quad j = 1, 2, \ldots, n \tag{5}$$

The maximum absolute hinge error (E) is given by

$$E = \max_{j \in 1,2,\ldots,n} |e_j| \tag{6}$$

Confidence of classifier: The confidence level of classification or predicted posterior probability is given as

$$\hat{p}(c|\mathbf{x}^i) = \frac{\min(1, \max(-1, \hat{y}_j^i)) + 1}{2} \tag{7}$$

Class-wise significance (ψ_c): In general, the input feature ($\mathbf{x}$) is mapped on to a hyper-dimensional spherical feature space $\mathbb{S}$ using K Gaussian neurons, i.e., $\mathbf{x} \rightarrow \boldsymbol{\phi}$. Therefore, all $\boldsymbol{\phi}(\mathbf{x})$ lie on a hyper-dimensional sphere [26]. The knowledge or spherical potential of any sample in original space is expressed as a squared distance from the hyper-dimensional mapping $\mathbb{S}$ centered at ϕ_0 [20].

In McNN, the center ($\boldsymbol{\mu}$) and width (σ) of the Gaussian neurons describe the feature space $\mathbb{S}$. Let the center of the K-dimensional feature space be $\phi_0 = (1/K) \sum_{k=1}^{K} \phi(\boldsymbol{\mu}_k)$. The knowledge present in the new data $\mathbf{x}^i$ can be expressed as the potential of the data in the original space, which is squared distance from the K-dimensional feature space to the center ϕ_0. The potential (ψ) is given as

$$\psi = \|\phi(\mathbf{x}^i) - \phi_0\|^2 \tag{8}$$

Using the steps shown in [20], the above equation can be expressed as

$$\psi = \phi(\mathbf{x}^i, \mathbf{x}^i) - \frac{2}{K} \sum_{k=1}^{K} \phi(\mathbf{x}^i, \boldsymbol{\mu}_k^l) + \frac{1}{K^2} \sum_{k,r=1}^{K} \phi(\boldsymbol{\mu}_k^l, \boldsymbol{\mu}_r^l) \tag{9}$$

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

$$\psi \approx -\frac{2}{K} \sum_{k=1}^{K} \phi(\mathbf{x}^i, \boldsymbol{\mu}_k^l) \tag{10}$$

Since we are addressing classification problems, the class-wise distribution plays a vital role and it will influence the performance of the classifier significantly [9]. Hence, we use the measure of the spherical potential of the new training sample $\mathbf{x}^i$ belonging to class c with respect to the neurons associated to same class (i.e., $l = c$). Let K^c be the number of neurons associated with the class c, then class-wise spherical potential or class-wise significance (ψ_c) is defined as

$$\psi_c = \frac{1}{K^c} \sum_{k=1}^{K^c} \phi(\mathbf{x}^i, \boldsymbol{\mu}_k^c) \tag{11}$$

The spherical potential directly 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.

2.2.3. Learning strategies

Based on the above-mentioned measures, the meta-cognitive component devices various learning strategies which directly addresses the basic principles of self-regulated human learning (i.e., *what-to-learn*, *when-to-learn* and *how-to-learn*). The meta-cognitive part controls the sequential learning process by selecting one of the following 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.
- *Parameter update strategy*: The new training sample is used to update the parameters of the cognitive component.
- *Sample reserve strategy*: The new training sample contains some information but not significant, they can be used at later stage of the learning process for fine tuning the parameters of the cognitive component. These sample may be discarded without learning or used for fine tuning the cognitive component parameters in a later stage.

Most of the existing sequential learning algorithms address only neuron addition/pruning and parameter update. In case of proposed McNN classifier, these learning strategies help to achieve the best human learning ability. Also, these strategies are adapted such that it suits current situation. Since the meta-cognitive component addresses *what-to-learn*, *when-to-learn* and *how-to-learn*, it improves the generalization ability of the cognitive component.

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

- *Sample delete strategy*: Prevents similar samples from being learnt, which avoids over-training and reduces the computational effort. When the predicted class label of the new training sample is same as the actual class label and the confidence level (estimated posterior probability) is greater than expected value 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 delete* criterion is given by

$$\hat{p}(c|\mathbf{x}^i) \geq \beta_d \quad \textbf{AND} \quad c == \hat{c} \tag{12}$$

  The meta-cognitive deletion threshold (β_d) is related to the expected confidence level in the class identification, i.e., if McNN predicts the class label ($\hat{c}$) of a sample with the higher confidence level (say, 0.9) then the sample contains information similar to McNN classifier. Hence, such samples can be deleted without being used in the learning process. The lowest limit of β_d is user defined expected confidence level. The term β_d controls number of samples participating in the learning process.
- *Neuron growth strategy*: When the new training sample contains significant information and the estimated class label is different from the actual class label then one needs to add new hidden neuron to capture the knowledge. The neuron growth

criterion is given by

$$\hat{c} \neq c \quad \textbf{AND} \quad \psi_c(\mathbf{x}^i) \leq \beta_c \quad \textbf{AND} \quad E \geq \beta_a \tag{13}$$

where β_c is the meta-cognitive knowledge measurement threshold and β_a is the self-adaptive meta-cognitive addition threshold. The β_a is adapted as follows:

$$\beta_a := \delta\beta_a + (1-\delta)E \tag{14}$$

where δ is the slope that controls rate of self-adaptation and is set close to 1. β_a allows samples with significant knowledge for learning first then uses the other samples for fine tuning.

If growth criterion given in Eq. (13) is satisfied, then a new hidden neuron $K+1$ is added and its parameters are initialized as explained below. Existing learning algorithms in the literature do not consider overlapping and distinct cluster criterion in assigning the new neuron parameters. However, the overlapping condition will influence the performance significantly. The new training sample may have overlap with other classes or will be from a distinct cluster far away from the nearest neuron in the same class. Hence, McNN measures inter/intra-class nearest neuron distances from the current sample in assigning the new neuron parameters.

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

$$nrS = \arg\min_{l==c;\forall k} \|\mathbf{x}^i - \boldsymbol{\mu}_k^l\|, \quad nrI = \arg\min_{l\neq c;\forall k} \|\mathbf{x}^i - \boldsymbol{\mu}_k^l\| \tag{15}$$

Let the Euclidian distances between the new training sample to *nrS* and *nrI* are given as follows:

$$d_S = \|\mathbf{x}^i - \boldsymbol{\mu}_{nrS}^c\|, \quad d_I = \|\mathbf{x}^i - \boldsymbol{\mu}_{nrI}^l\| \tag{16}$$

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

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

$$\boldsymbol{\mu}_{K+1}^c = \mathbf{x}^i, \quad \sigma_{K+1}^c = \kappa\sqrt{\mathbf{x}^{iT}\mathbf{x}^i}, \quad \boldsymbol{\alpha}_{K+1} = \mathbf{e} \tag{17}$$

  where κ is a positive constant which controls the overlap of the responses of the hidden units in the input space, which lies in the range $0.5 \leq \kappa \leq 1$.

- *no-overlapping with the inter-class*: When a new training sample is close to the intra-class nearest neuron then the sample does not overlap with the other classes, i.e., the intra/inter-class distance ratio is less than 1, then the sample does not overlap with the other classes. In this case, the new hidden neuron center ($\boldsymbol{\mu}_{K+1}^c$), width (σ_{K+1}^c) and weight ($\boldsymbol{\alpha}_{K+1}$) parameters are determined as

$$\boldsymbol{\mu}_{K+1}^c = \mathbf{x}^i, \quad \sigma_{K+1}^c = \kappa\|\mathbf{x}^i - \boldsymbol{\mu}_{nrS}^c\|, \quad \boldsymbol{\alpha}_{K+1} = \mathbf{e} \tag{18}$$

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

$$\boldsymbol{\mu}_{K+1}^c = \mathbf{x}^i + \zeta(\boldsymbol{\mu}_{nrS}^c - \boldsymbol{\mu}_{nrI}^l), \quad \sigma_{K+1}^c = \kappa\|\boldsymbol{\mu}_{K+1}^c - \boldsymbol{\mu}_{nrS}^c\| \tag{19}$$

  where ζ is center shift factor which determines how much center has to be shifted from the new training sample location. It lies in the range [0.01–0.1]. The weight parameter of the new hidden neuron is calculated as

$$\boldsymbol{\alpha}_{K+1} = \mathbf{e}/\phi_{K+1}(\mathbf{x}^i) \tag{20}$$

- *significant overlapping with the inter-class*: When a new training sample is very close to the inter-class nearest neuron compared to the intra-class nearest neuron, i.e., the intra/inter-class distance ratio is more than 1.5, then the sample has significant overlapping with the other class. In this case, the center of the new hidden neuron is shifted away from the inter-class nearest neuron and is initialized as

$$\boldsymbol{\mu}_{K+1}^c = \mathbf{x}^i - \zeta(\boldsymbol{\mu}_{nrI}^l - \mathbf{x}^i), \quad \sigma_{K+1}^c = \kappa\|\boldsymbol{\mu}_{K+1}^c - \boldsymbol{\mu}_{nrI}^l\| \tag{21}$$

  The weight parameter of new hidden neuron is calculated as given in Eq. (20).

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

- *Network parameters update strategy*: Cognitive component parameters $\mathbf{w} = [\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \boldsymbol{\mu}^l, \sigma_1^l, \ldots, \boldsymbol{\alpha}_K, \boldsymbol{\mu}_K^l, \sigma_K^l]^T$) are updated if the following criterion is satisfied.

$$c == \hat{c} \quad \textbf{AND} \quad E \geq \beta_u \tag{22}$$

where β_u is the self-adaptive meta-cognitive update threshold. The β_u is adapted based on the prediction error as

$$\beta_u := \delta\beta_u + (1-\delta)E \tag{23}$$

where δ is the slope that controls the rate of self-adaption of update threshold and is set close to 1. The advantage of self-adaptive meta-cognitive thresholds is that, they help in selecting the samples for adding as a hidden neuron or to update parameters.

McNN uses extended Kalman filter (EKF) to update the cognitive component parameters

$$\mathbf{w}' = \mathbf{w} + \mathbf{G}\mathbf{e} \tag{24}$$

where $\mathbf{e}$ is the error obtained from the hinge loss function and $\mathbf{G} \in \Re^{z\times n}$ is the Kalman gain matrix given by

$$\mathbf{G} = \mathbf{P}\mathbf{B}[\mathbf{R} + \mathbf{B}^T\mathbf{P}\mathbf{B}]^{-1} \tag{25}$$

where $z = K(m+n+1)+n$, $\mathbf{R} = r_0\mathbf{I}_{n\times n}$ is the variance of the measurement noise, $\mathbf{P} \in \Re^{z\times z}$ is the error covariance matrix, $\mathbf{B}$ is the partial derivatives for the output with respect to the parameters ($\mathbf{w}$) given by

$$\mathbf{B} = \begin{bmatrix} \mathbf{I}_{n\times n}, \phi_1\mathbf{I}_{n\times n}, \phi_1\frac{2\boldsymbol{\alpha}_1}{(\sigma_1^l)^2}(\mathbf{x}^i - \boldsymbol{\mu}_1^l)^T, \phi_1\frac{2\boldsymbol{\alpha}_1}{(\sigma_1^l)^3}\|\mathbf{x}^i - \boldsymbol{\mu}_1^l\|^2 \ldots, \\ \phi_K\mathbf{I}_{n\times n}, \phi_K\frac{2\boldsymbol{\alpha}_K}{(\sigma_K^l)^2}(\mathbf{x}^i - \boldsymbol{\mu}_K^l)^T, \phi_K\frac{2\boldsymbol{\alpha}_K}{(\sigma_K^l)^3}\|\mathbf{x}^i - \boldsymbol{\mu}_K^l\|^2 \end{bmatrix}^T \tag{26}$$

The error covariance matrix is updated by

$$\mathbf{P}' = [\mathbf{I}_{z\times z} - \mathbf{G}\mathbf{B}^T]\mathbf{P} + q_0\mathbf{I}_{z\times z} \tag{27}$$

The addition of artificial process noise (q_0) helps in avoiding convergence to local minima.

When a new hidden neuron is added, the dimensionality of error covariance matrix $\mathbf{P}$ is increased to

$$\begin{bmatrix} \mathbf{P}_{z\times z} & \mathbf{0}_{z\times(m+n+1)} \\ \mathbf{0}_{(m+n+1)\times z} & p_0\mathbf{I}_{(m+n+1)\times(m+n+1)} \end{bmatrix} \tag{28}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{0}$ is the zero matrix and p_0 is the initial estimated uncertainty.

- *Sample reserve strategy*: If the new training sample does not satisfy either the deletion or the neuron growth or the

cognitive component parameters update criterion, then the sample is pushed to the rear of the training sequence. Since McNN modifies the strategies based on current sample knowledge, these samples may be used in later stage.

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

2.3. McNN algorithm

McNN algorithm can be summarized as below:

1. For each new training sample input ($\mathbf{x}^i$) compute the cognitive component output ($\hat{\mathbf{y}}$) using Eqs. (2) and (3).
2. The meta-cognitive component finds the estimated class label ($\hat{c}$), maximum hinge error (E), confidence ($\hat{p}(c|\mathbf{x}^i)$) and class-wise significance (ψ_c) measures for the new training sample ($\mathbf{x}^i$) using Eqs. (4), (6), (7) and (11).
3. The meta-cognitive component selects one of the following strategies based on the above calculated measures.
   (a) *Sample delete strategy*: If $\hat{p}(c|\mathbf{x}^i) \geq \beta_d$ **AND** $c == \hat{c}$ then delete the sample from the sequence without learning.
   (b) *Neuron growth strategy*: If $\hat{c} \neq c$ **AND** $\psi_c(\mathbf{x}^i) \leq \beta_c$ **AND** $E \geq \beta_a$, then allocate a new hidden neuron in the cognitive component and its parameters are calculated based on the intra- and inter-class nearest neurons distances using equations from (17) to (21). Also, update the self-adaptive meta-cognitive addition threshold using Eq. (14) and increase the dimensionality of $\mathbf{P}$.
   (c) *Parameters update strategy*: If $c == \hat{c}$ **AND** $E \geq \beta_u$, then update the cognitive component parameters using EKF. Also, update the self-adaptive meta-cognitive update threshold using Eq. (23).
   (d) *Sample reserve strategy*: When the new sample does not satisfy deletion, growth and update criterion, then push the sample to the reserve to be used later for learning.
4. The cognitive component executes the above selected strategy.
5. Continue steps 1–4 until there are no more samples in the training data stream or number of samples remains same in the reserve.

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

2.4. Guidelines for McNN thresholds initialization

In this section, we explain the influence of β_c, β_d, β_a and β_u thresholds on the performance of the McNN and provide some guidelines for their initialization.

The knowledge threshold (β_c) helps in identifying novelty of current sample and it depends on spherical potential, the range of spherical potential is between 0 and 1. If spherical potential is close to 1 then sample is similar to existing knowledge, lesser value of spherical potential indicates that the sample is novel. If one selects the threshold β_c close to zero then the network does not allow addition of neurons. Similarly, if one selects the threshold β_c close to one then all samples will be identified as novel sample. Hence, β_c can be selected in the range [0.4–0.8].

The delete threshold (β_d) prevents over training by removing samples predicted accurately with high confidences. The self-regulated addition threshold (β_a) and update threshold (β_u) are used to select appropriate samples for efficient learning. β_d, β_a and β_u thresholds depend on hinge error (E). Note that the hinge error E is between $[0, 2]$. The characteristics of thresholds and their influence on McNN performance can be explained by dividing the error region into three subregions, namely, the sample deleting region, the parameter updating region, and the neuron growing region, as shown in Fig. 2.

If confidence of classifier (predicted posterior probability) is greater than β_d and predicted class label ($\hat{c}$) is same as actual class label then the current sample is similar to existing knowledge in the cognitive component. In that case the current sample is deleted from the training sequence without being used in the learning process. Confidence of classifier decreases from 1 to 0 as hinge error (E) increases from 0 to 2 as shown in Fig. 2. Suppose one selects the deletion threshold to be 0.5 then many samples satisfying the condition will be deleted without being used in learning. Hence, the resultant classifier may not provide better generalization performance. If one selects close to 1, say 0.99 then most of the similar samples will be used in learning, which will result in over training. Hence, the β_d can be selected in the range [0.85–0.95].

The addition threshold β_a is combined with other conditions, which measure misclassification and knowledge. The minimum possible prediction error one can get when there is a misclassification is 1. Hence, β_a should be greater than 1. If one selects the threshold β_a close to 1 then all samples misclassified will be used for neuron addition. If one selects the threshold β_a close to 2 then very few neurons will be added and the resultant network may not approximate the decision surface. Note that the meta-cognitive component adapts addition threshold such that new hidden neuron added for a sample with higher error. Hence, the initial value of addition threshold (β_a) can be selected in the range [1.2–1.5].

McNN updates the parameters when the predicted class is accurate and the hinge error (E) is greater than β_u. When predicted class label is accurate then value of E is in between 0 and 1. If one selects the threshold β_u close to 1 then no sample will be used in updating and hence the resultant network does not approximate the decision function. If one selects the threshold β_u close to 0 then all samples will be used for updating. McNN

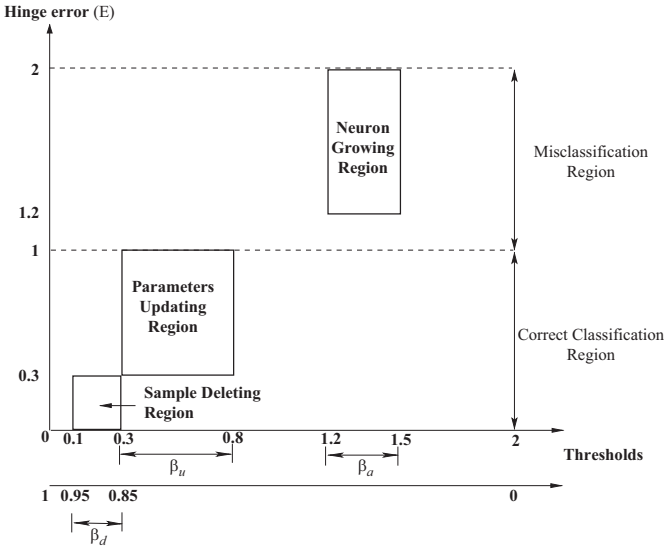

Fig. 2. Error regions of various thresholds in McNN.

updates the parameters using samples, which produces higher error. Hence, the initial value of update threshold (β_u) can be selected in the range [0.3–0.8].

3. Performance evaluation of McNN classifier

The performance of McNN classifier is evaluated using benchmark binary and multi-category classification problems from the UCI machine learning repository. The performance is compared with the best performing sequential learning algorithm reported in the literature (SRAN), batch ELM classifier and also with the standard support vector machines. The data sets are chosen with varying sample imbalance. The sample imbalance is measured using imbalance factor (IF) as

$$\mathrm{IF} = 1 - \frac{n}{N} * \min_{j=1\cdots n} N_j \tag{29}$$

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

The description of these data sets including the number of classes, the number of input features, the number of samples in the training/testing and the imbalance factor are presented in Table 1. From Table 1, it can be observed that the problems chosen for the study have both balanced and unbalanced data sets and the imbalance factors of the data sets vary widely.

Finally, McNN classifier is used to solve two real-world classification problems: the acoustic emission signal processing for health monitoring [23] and the mammogram classification for breast cancer detection [29].

All the simulations are conducted in MATLAB 2010 environment on a desktop PC with Intel Core 2 Duo, 2.66 GHz CPU and 3 GB RAM. The simulations for batch SVM are carried out using the LIBSVM package in C [30]. The performance measures used to compare the classifiers are described below.

3.1. Performance measures

The class-wise performance measures like overall/average efficiencies and a statistical significance test on performance of multiple classifiers on multiple data sets are used for performance comparison.

3.1.1. Class-wise measure

The confusion matrix Q is used to obtain the class-level performance and global performance of the various classifiers. Class-level performance is measured by the percentage classification (η_i) which is defined as

$$\eta_j = \frac{q_{jj}}{N_j} \times 100\% \tag{30}$$

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

$$\eta_a = \frac{1}{n}\sum_{j=1}^{n} \eta_j \tag{31}$$

$$\eta_o = \frac{\sum_{j=1}^{n} q_{jj}}{\sum_{j=1}^{n} N_j} \times 100\% \tag{32}$$

3.1.2. Statistical significance test

The classification efficiency itself is not a conclusive measure of a classifier performance [22]. Since the developed classifier is compared with multiple classifiers over multiple data sets, the Friedman test followed by the Benferroni–Dunn test is used to establish the statistical significance of McNN classifier. A brief description of the conducted test is given below.

The Friedman test is used to compare multiple classifiers (U) over multiple data sets (V). Let r_i^j be the rank of the jth classifier on the ith data set. The Friedman test compares the average ranks of classifiers, $R_j = (1/V)\sum_i r_i^j$. Under the null-hypothesis, which states that all the classifiers are equivalent and so their ranks R_j should be equal, the Friedman statistic is given by

$$\chi_F^2 = \frac{12V}{U(U+1)}\left[\sum_j R_j^2 - \frac{U(U+1)^2}{4}\right] \tag{33}$$

which follows the χ^2 (Chi-square distribution) distribution with $U-1$ degrees of freedom. A χ^2 distribution is the distribution of a sum of squares of U independent standard normal variables.

Iman and Davenport showed that Friedman's statistic (χ_F^2) is more conservative and derived a better statistic in [31]. It is given by

$$F_F = \frac{(V-1)\chi_F^2}{V(U-1)-\chi_F^2} \tag{34}$$

which follows the F-distribution with $U-1$ and $(U-1)(V-1)$ degrees of freedom is used in this paper. F-distribution is defined as the probability distribution of the ratio of two independent χ^2 distributions over their respective degrees of freedom. The aim of the statistical test is to prove that the performance of McNN

Table 1
Description of benchmark data sets selected from UCI machine learning repository for performance study.

Data sets	No. of features	No. of classes	No. of samples		IF	
			Training	Testing	Training	Testing
Image segmentation (IS)	19	7	210	2100	0	0
IRIS	4	3	45	105	0	0
WINE	13	3	60	118	0	0.29
Vehicle classification (VC)	18	4	424[a]	422	0.1	0.12
Glass identification (GI)	9	6	109[a]	105	0.68	0.77
HEART	13	2	70	200	0.14	0.1
Liver disorders (LD)	6	2	200	145	0.17	0.14
PIMA	8	2	400	368	0.22	0.39
Breast cancer (BC)	9	2	300	383	0.26	0.33
Ionosphere (ION)	34	2	100	251	0.28	0.28

[a] As suggested in [9], the samples are repeated three times.

classifier is substantially different from the other classifiers with a confidence level of value $1-\alpha$. If calculated $F_F > F_{\alpha/2,(U-1),(U-1)(V-1)}$ or $F_F < F_{1-\alpha/2,(U-1),(U-1)(V-1)}$, then the null-hypothesis is rejected. The Statistical tables for critical values can be found in [32].

The Benferroni–Dunn test [33] is a post-hoc test that can be performed after rejection of the null-hypothesis. It is used to compare McNN classifier against all the other classifiers. This test assumes that the performances of two classifiers are significantly different if the corresponding average ranks differ by at least the critical difference (CD)

$$CD = q_\alpha \sqrt{\frac{U(U+1)}{6V}} \tag{35}$$

where critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$ as given in [22].

3.2. Performance evaluation on UCI benchmark data sets

The class-wise performance measures (average/overall) testing efficiencies, number of hidden neurons and samples used for McNN, SRAN, ELM and SVM classifiers are reported in Table 2.

The table contains results of both the binary and the multi-category classification data sets from the UCI machine learning repository. From Table 2, we can see that McNN classifier performs slightly better than the best performing SRAN classifier and significantly better than ELM and SVM classifiers on all the 10 data sets. On the well balanced data sets (IS, IRIS, WINE) the generalization performance in McNN is improved by 1% compared to the SRAN and 2–3% compared to the batch ELM and SVM classifiers. On all the three well balanced data sets McNN takes less number of samples to build the classifier and also McNN requires less number of hidden neurons to approximate decision surface.

The McNN classifier selects the best sequence of training samples from a given training sequence such that the classifier produces better generalization performance. On IS data set, McNN uses only 93 samples out of 210 training samples to build the best classifier. Now, we conduct a study using the ELM classifier on IS data set to highlight the advantages of proposed McNN classifier. On IS data set we use the best 93 sample sequence generated by the McNN in the ELM algorithm. We call this classifier as ELM*. The number of neurons, number of samples used, overall training efficiency and testing efficiency are reported in Table 3. From the results, we can see that the training and testing performance of the ELM* classifier (which uses 93 sample sequence) is better than the ELM classifier. Also, the ELM* achieves better generalization performance with smaller number of hidden neurons (ELM* requires only 33 hidden neurons to achieve 91.62% testing efficiency whereas ELM requires 100 hidden neurons to achieve 90.67%). This study clearly indicates that one needs to remove the similar samples present in training data to achieve better decision making ability.

On the imbalanced data sets (VC, GI, HEART, LD, PIMA, BC, ION) the generalization performance of McNN is approximately 2–7% more than SRAN, and 2–10% more than the ELM and SVM classifiers. On all the seven imbalanced data sets McNN takes less number of samples to build the classifier and also McNN requires less number of hidden neurons. The GI data set has imbalance factor of 0.68 in training and 0.77 in testing. Such high imbalance influences the performance of SRAN, ELM and SVM classifiers. On GI data set, the difference between the overall testing efficiency (η_o) and the average testing efficiency (η_a) of SRAN, ELM and SVM classifiers is approximately 6–8%. This is due to the fact that the classifier are not able to capture the knowledge for the classes which contain smaller number of samples accurately. In case of proposed McNN classifier, the difference between average and overall performance is around 1%. The overlapping condition and class specific criterion in McNN helps in capturing the knowledge accurately in case of high sample imbalance problems. From Table 2, we can say that the proposed McNN improves average/overall efficiency even under high sample imbalance.

3.2.1. Statistical significance analysis

Now, we conduct the Friedman test followed by the Benferroni–Dunn test to statistically compare McNN classifier performance with multiple classifiers over multiple data sets. The Friedman test checks whether the measured average ranks are significantly different from the mean rank (in present case mean rank is 2.5) expected under the null-hypothesis. The Benferroni–Dunn test checks whether a proposed McNN classifier is better than the existing classifiers.

Table 3
Performance comparison for ELM and ELM* classifiers on image segmentation (IS) data set.

Classifier	K neurons	Samples used	Training η_o	Testing η_o
ELM	100	210	94.76	90.67
ELM*	33	93	95.24	91.62
McNN	49	93	97.62	**93.38**

Table 2
Performance comparison of McNN with SRAN, ELM and SVM.

Data sets	McNN				SRAN				ELM			SVM		
	K	Samples used	Testing		K	Samples used	Testing		K	Testing		SV[a]	Testing	
			η_o	η_a			η_o	η_a		η_o	η_a		η_o	η_a
IS	49	93	93.38	93.38	47	113	92.29	92.29	100	90.67	90.67	96	90.62	90.62
IRIS	5	22	97.14	97.14	8	29	96.19	96.19	10	96.19	96.19	13	96.19	96.19
WINE	9	27	98.3	98.49	12	46	96.61	97.19	10	97.46	98.04	36	97.46	98.04
VC	146	359	77.72	78.72	113	437	75.12	76.86	150	77.01	77.59	234	68.72	67.99
GI	73	117	85.71	87.03	59	159	86.21	80.95	60	73.00	65.46	102	64.23	60.01
HEART	26	46	80.50	79.65	28	56	78.50	77.53	36	76.50	75.91	42	75.50	75.10
LD	68	110	73.79	71.60	91	151	66.90	65.78	100	72.41	71.41	141	71.03	70.21
PIMA	76	193	80.16	77.31	97	230	78.53	74.90	400	76.63	75.25	77.45	76.43	
BC	9	27	97.39	97.85	7	91	96.87	97.26	300	96.35	96.48	24	96.61	97.06
ION	20	39	95.62	95.60	21	86	90.84	91.88	32	89.64	87.52	43	91.24	88.51

[a] Number of support vectors.

Table 4
Ranks based on the overall testing efficiency (η_o).

Data sets	McNN	SRAN	ELM	SVM
IS	1	2	3	4
IRIS	1	3	3	3
WINE	1	4	2.5	2.5
VC	1	3	2	4
GI	2	1	3	4
HEART	1	2	3	4
LD	1	4	2	3
PIMA	1	2	4	3
BC	1	2	4	3
ION	1	3	4	2
Average rank (R_j)	1.1	2.6	3.05	3.25

Table 5
Ranks based on the average testing efficiency (η_a).

Data sets	McNN	SRAN	ELM	SVM
IS	1	2	3	4
IRIS	1	3	3	3
WINE	1	4	2.5	2.5
VC	1	3	2	4
GI	1	2	3	4
HEART	1	2	3	4
LD	1	4	2	3
PIMA	1	4	3	2
BC	1	2	4	3
ION	1	2	4	3
Average rank (R_j)	1	2.8	2.95	3.25

- *Comparison based on the overall testing efficiency* (η_o): Ranks of all four classifiers based on the overall testing efficiency for each data set are provided in Table 4. The Friedman statistic (χ_F^2 as in Eq. (33)) is 17.01 and modified (Iman and Davenport) statistic (F_F as in Eq. (34)) is 11.79. For four classifiers and 10 data sets, the modified statistic is distributed according to the F-distribution with 3 and 27 degrees-of-freedom. The critical value for rejecting the null hypothesis at significance level of 0.05 is 3.65. Since, modified statistic is greater than the critical value ($11.79 \gg 3.65$), we can reject the null hypothesis.
  Hence, we can say that the proposed McNN classifier performs better than the existing classifiers on these data sets.
  Next, we conduct the Benferroni–Dunn test to compare the proposed McNN classifier with all other classifiers. From Eq. (35), the critical difference is calculated as 1.382 for a significance level of 0.05 ($q_{0.05} = 2.394$). From Table 4, we can see that the difference in average rank between the proposed McNN classifier and the other three classifiers are 1.5, 1.95 and 2.15. The difference in average rank is greater than the critical difference. Hence, based on the overall testing efficiency the Benferroni–Dunn test shows that the proposed McNN classifier is significantly better than the SRAN, ELM and SVM classifiers.
- *Comparison based on the average testing efficiency* (η_a): Ranks of all four classifiers based on the average testing efficiency for each data set are provided in Table 5. The Friedman statistic (χ_F^2 as in Eq. (33)) is 18.63 and modified statistic (F_F as in Eq. (34)) is 14.75. Since, modified statistic is greater than the critical value ($14.75 \gg 3.65$), we can reject the null hypothesis. Hence, we can say that the proposed McNN classifier performs better than the other classifiers on these data sets. From Table 5, we can see that the difference in average rank between the proposed McNN classifier and the other three classifiers are 1.8, 1.95 and 2.25. The difference in average rank is greater than the critical difference (1.382). Hence, based on the average testing efficiency, the Benferroni–Dunn test also shows that the proposed McNN classifier performs better than the other well known classifiers.

Next, we present the performance results of McNN classifier on the two real-world classification problem data sets, viz., an acoustic emission data set for health monitoring [23] and the mammogram classification data set for breast cancer detection [29].

3.3. Acoustic emission signal classification for health monitoring

The stress or pressure waves produced by the sensitive transducer due to the transient energy released by the irreversible deformation in the material are called as acoustic emission signals. These signals are produced by various sources and classification/identification of sources using the acoustic emission signals is a very difficult problem. The presence of ambient noise and pseudo-acoustic emission signals in practical situations increases the complexity further. In addition, the superficial similarities between the acoustic emission signals produced by different sources increases the complexity further. In this section, we address classification of such acoustic emission signals using the proposed McNN classifier. The experimental data provided for the burst type acoustic emission signals from the metallic surface is considered for our study [23]. The burst type acoustic emission signal is characterized by five features and these signals are classified into one of the four sources, namely, the pencil source, the pulse source, the spark source and the noise source. Out of 199 samples, 62 samples are used for training (as highlighted in [23]) and the remaining samples are used for testing the classifier. For details on characteristics of input features and the experimental setup, one should refer to [23].

The performance study results of McNN classifier are compared against the SRAN, ELM, SVM, fuzzy K-means clustering algorithm [23] and ant colony optimization algorithm [34], presented in Table 6. It can be seen that McNN classifier requires only five neurons to achieve an over-all testing efficiency of 99.27%. Thus, McNN classifier performs an efficient classification of the acoustic emission signals using a compact network.

3.4. Mammogram classification for breast cancer detection

Mammogram is a better means for early diagnosis of breast cancer, as tumors and abnormalities show up in mammogram much before they can be detected through physical examinations [35]. Clinically, identification of malignant tissues involves detecting the abnormal masses or tumors, if any, and then classifying the mass as either malignant or benign [24]. However, once a tumor is detected, the only method of determining whether it is benign or malignant is by conducting a biopsy, which is an invasive procedure that involves the removal of the cells or tissue from a patient. A non-invasive method of identifying the

Table 6
Performance comparison for acoustic emission signal problem.

Method	N_H neurons	Samples used	Testing	
			η_o	η_a
Ant colony optimization	–	62	90.51	89.27
Fuzzy C-means clustering	–	62	–	93.34
SVM	22[a]	62	98.54	97.95
ELM	10	62	99.27	98.91
SRAN	10	39	99.27	98.91
McNN	5	20	**99.27**	**98.91**

[a] Number of support vectors.

Table 7
Performance comparison for mammogram classification problem.

Method	N_H neurons	Samples used	Testing	
			η_o	η_a
SVM	26[a]	97	90.91	91.67
ELM	30	97	90.91	90.0
SRAN	25	45	90.91	91.67
McNN	19	34	**100**	**100**

[a] Number of support vectors.

abnormalities in a mammogram can reduce the number of unnecessary biopsies, thus sparing the patients of inconvenience and saving medical costs.

In this study, mammogram database available in [29] has been used. The nine input features extracted from the mammogram of the identified abnormal mass are used to classify the tumor as either malignant or benign. Here, 97 samples are used to develop McNN classifier and the performance of McNN classifier is evaluated using the remaining 11 samples. For further details on the input features and the data set, one should refer to [29].

The performance results of McNN classifier, in comparison with the SRAN, ELM and SVM are presented in Table 7. From the table, it is seen that McNN classifier performs a highly efficient classification with 100% classification accuracy with smaller number of hidden neurons. When compared to SRAN, ELM, SVM classifiers, performance of McNN is improved considerably.

Thus, from the performance study of McNN conducted with SRAN, ELM, SVM for chosen benchmark data sets and practical classification problems, it can be observed that the proposed McNN classifier performs better than other classifiers.

4. Conclusions

In this paper, we have presented the sequential learning algorithm for Meta-cognitive Neural Network (McNN) based on human meta-cognitive learning principles for classification problems. The meta-cognitive component in McNN is helpful in choosing suitable strategy for training the cognitive component in McNN. Also, the learning strategies consider sample overlapping condition for proper initialization of new hidden neurons, which helps in minimization of misclassification. The meta-cognitive component adapts the learning strategies appropriately and hence it decides *what-to-learn*, *when-to-learn* and *how-to-learn* efficiently. In addition, the overlapping conditions present in neuron growth strategy helps in proper initialization of new hidden neuron parameters and also minimizes the misclassification error. The performance of the proposed McNN classifier has been evaluated using the benchmark multi-category, binary classification problems from the UCI machine learning repository with wide range of imbalance factor and two practical classification problems. The statistical comparison with the well-known classifiers in the literature clearly indicates the superior performance of the proposed McNN classifier.

Acknowledgment

The authors would like to thank the reviewers for their valuable suggestions which helped in improving the quality of this paper. The authors acknowledge the support of Academic Research Funding (AcRF) Tier I (No. M58020020) grant.

References

[1] T.M. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.
[2] V.N. Vapnik, The Nature of Statistical Learning Theory, 2nd ed., Springer-Verlag, New York, 2000.
[3] T.G. Dietterich, Machine learning for sequential data: a review, in: Structural, Syntactic, and Statistical Pattern Recognition, Springer Verlag, New York, 2002, pp. 15–30.
[4] J.C. Platt, A resource allocation network for function interpolation, Neural Comput. 3 (2) (1991) 213–225.
[5] L. Yingwei, N. Sundararajan, P. Saratchandran, A sequential learning scheme for function approximation using minimal radial basis function neural networks, Neural Comput. 9 (2) (1997) 461–478.
[6] L. Yan, N. Sundararajan, P. Saratchandran, Analysis of minimal radial basis function network algorithm for real-time identification of nonlinear dynamic systems, IEE Proc. Control Theory Appl. 147 (4) (2000) 476–484.
[7] G.-B. Huang, P. Saratchandran, N. Sundararajan, An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks, IEEE Trans. Syst. Man Cybern. Part B Cybern. 34 (6) (2004) 2284–2292.
[8] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Networks 17 (6) (2006) 1411–1423.
[9] S. Suresh, N. Sundararajan, P. Saratchandran, A sequential multi-category classifier using radial basis function networks, Neurocomputing 71 (1) (2008) 1345–1358.
[10] S. Suresh, K. Dong, H.J. Kim, A sequential learning algorithm for self-adaptive resource allocation network classifier, Neurocomputing 73 (16–18) (2010) 3012–3019.
[11] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks. in: IEEE International Joint Conference on Neural Networks. Proceedings, vol. 2, 2004, pp. 985–990.
[12] S. Suresh, R.V. Babu, H.J. Kim, No-reference image quality assessment using modified extreme learning machine classifier, Appl. Soft Comput. 9 (2) (2009) 541–552.
[13] G.-B. Huang, D. Wang, Y. Lana, Extreme learning machines: a survey, Int. J. Mach. Learn. Cybern. 2 (2) (2011) 107–122.
[14] G. Cauwenberghs, T. Poggio, Incremental and decremental support vector machine learning. in: Advances in Neural Information Processing Systems (NIPS 2000), vol. 13, MIT Press, Cambridge, MA, 2001.
[15] M. Karasuyama, I. Takeuchi, Multiple incremental decremental learning of support vector machines, IEEE Trans. Neural Networks 21 (7) (2010) 1048–1059.
[16] S. Suresh, R. Savitha, N. Sundararajan, A sequential learning algorithm for complex-valued self-regulating resource allocation network—CSRAN, IEEE Trans. Neural Networks 22 (7) (2011) 1061–1072.
[17] R. Savitha, S. Suresh, N. Sundararajan, A self-regulated learning in fully complex-valued radial basis function networks, in: The 2010 International Joint Conference on Neural Networks (IJCNN), 2010, pp. 1–8.
[18] R. Isaacson, F. Fujita, Metacognitive knowledge monitoring and self-regulated learning: academic success and reflections on learning, J. Scholarship Teach. Learn. 6 (1) (2006) 39–55.
[19] J.H. Flavell, Metacognitive aspects of problem solving, in: L.B. Resnick (Ed.), The Nature of Intelligence, Lawrence Erlbaum Associates, Hillsdale, NJ, 1976, pp. 231–236.
[20] H. Hoffmann, Kernel PCA for novelty detection, Pattern Recognition 40 (3) (2007) 863–874.
[21] C.M.C. Blake, UCI Repository of Machine Learning Databases, University of California, Irvine, Department of Information and Computer Sciences, 1998, ⟨http://archive.ics.uci.edu/ml/⟩.
[22] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[23] S.N. Omkar, S. Suresh, T.R. Raghavendra, V. Mani, Acoustic emission signal classification using fuzzy c-means clustering, in: Proceedings of the ICONIP'02, 9th International Conference on Neural Information Processing, vol. 4, 2002, pp. 1827–1831.
[24] C. Aize, Q. Song, X. Yang, S. Liu, C. Guo, Mammographic mass detection by vicinal support vector machine, in: Proceedings of the ICNN'04, International Conference on Neural Networks, vol. 3, 2004, pp. 1953–1958.
[25] T.O. Nelson, L. Narens, Metamemory: a theoretical framework and new findings, Psychol. Learn. Motiv. 26 (1990) 125–173.
[26] B. Scholkopf, A.J. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002.
[27] T. Zhang, Statistical behavior and consistency of classification methods based on convex risk minimization, Ann. Stat. 32 (1) (2004) 56–85.
[28] S. Suresh, N. Sundararajan, P. Saratchandran, Risk-sensitive loss functions for sparse multi-category classification problems, Inf. Sci. 178 (12) (2008) 2621–2638.
[29] J. Suckling, J. Parker, D.R. Dance, S. Astley, I. Hutt, C. Boggis, I. Ricketts, E. Stamatakis, N. Cerneaz, S. Kok, et al., The mammographic image analysis society digital mammogram database, in: Experta Medica International Congress Series, vol. 1069, 1994, pp. 375–378.
[30] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intelligent Syst. Technol. 2 (2011). 27:1–27:27 (software available at ⟨http://www.csie.ntu.edu.tw/cjlin/libsvm⟩).

[31] R.L. Iman, J.M. Davenport, Approximations of the critical region of the Friedman statistic, Commun. Stat. 1000 (1980) 571–595.
[32] J.H. Zar, Biostatistical Analysis, 4th ed., Prentice-Hall, Englewood Cliffs, NJ, 1999.
[33] O.J. Dunn, Multiple comparisons among means, J. Am. Stat. Assoc. 56 (293) (1961) 52–64.
[34] S.N. Omkar, U. Raghavendra Karanth, Rule extraction for classification of acoustic emission signals using ant colony optimisation, Eng. Appl. Artif. Intell. 21 (8) (2008) 1381–1388.
[35] P. Gamigami, Atlas of Mammography: New Early Signs in Breast Cancer, Blackwell Science, Cambridge, MA, USA, 1996.

Giduthuri Sateesh Babu 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 neural networks, pattern recognition, and machine learning.

Sundaram Suresh 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. His research interest includes flight control, unmanned aerial vehicle design, machine learning, optimization and computer vision.