Cogn Comput (2014) 6:253–263
DOI 10.1007/s12559-013-9223-2

A Meta-Cognitive Learning Algorithm for an Extreme Learning Machine Classifier

R. Savitha · S. Suresh · H. J. Kim

Received: 29 June 2012 / Accepted: 22 March 2013 / Published online: 4 July 2013

Abstract This paper presents an efficient fast learning classifier based on the Nelson and Narens model of human meta-cognition, namely 'Meta-cognitive Extreme Learning Machine (McELM).' McELM has two components: a cognitive component and a meta-cognitive component. The cognitive component of McELM is a three-layered extreme learning machine (ELM) classifier. The neurons in the hidden layer of the cognitive component employ the *q*-Gaussian activation function, while the neurons in the input and output layers are linear. The meta-cognitive component of McELM has a self-regulatory learning mechanism that decides *what-to-learn*, *when-to-learn*, and *how-to-learn* in a meta-cognitive framework. As the samples in the training set are presented one-by-one, the meta-cognitive component receives the monitory signals from the cognitive component and chooses suitable learning strategies for the sample. Thus, it either deletes the sample, uses the sample to add a new neuron, or updates the output weights based on the sample, or reserves the sample for future use. Therefore, unlike the conventional ELM, the architecture of McELM is not fixed a priori, instead, the network is built during the training process. While adding a neuron, McELM chooses the centers based on the sample, and the width of the Gaussian function is chosen randomly. The output weights are estimated using the least square estimate based on the hinge-loss error function. The hinge-loss error function facilitates prediction of posterior probabilities better than the mean-square error and is hence preferred to develop the McELM classifier. While updating the network parameters, the output weights are updated using a recursive least square estimate. The performance of McELM is evaluated on a set of benchmark classification problems from the UCI machine learning repository. Performance study results highlight that meta-cognition in ELM framework enhances the decision-making ability of ELM significantly.

Keywords Extreme learning machine · Meta-cognition · Classification · Self-regulatory learning mechanism · Hinge-loss error function

R. Savitha · S. Suresh
School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore

H. J. Kim (✉)
Graduate School of International Management and Security, Korea University, Seoul, Korea
e-mail: khj-@korea.ac.kr

Introduction

Neural network classifiers can be used to perform important decision-making tasks. Owing to the localization properties of radial basis function (RBF) networks, they are widely used in the development of these classifiers. Earlier, RBF networks with gradient descent batch learning algorithm, in which the initial parameters are tuned iteratively, were used to learn the decision function. However, such networks require the network structure to be fixed a priori and also require the entire training data to be available a priori. Fixing the network structure a priori does not ensure accurate approximation of the decision surface. Moreover, training these networks is computationally intensive due to the iterative gradient descent approach.

To overcome the above difficulties, sequential learning algorithms that process the training data sample by sample and discard them after learning have been developed. These sequential learning algorithms do not need the network structure to be fixed a priori. They evolve the network

during the training process, by adding and pruning neurons or updating the network parameters using the extended Kalman filter. Some of the sequential learning algorithms available in the literature include the resource allocation network [1], minimal resource allocation network [2, 3], the growing and pruning RBF network [4], the sequential multi-category radial basis function classifier [5], the sequential adaptive resource allocation network [6] and the meta-cognitive neural network [7]. Although these networks learn the sample one-by-one and only once, they use the computationally intensive extended Kalman filter to update the network parameters.

On the other hand, the extreme learning machine (ELM) [8] is a fast learning single hidden layer network that can be used to solve both regression and classification problems [9]. In an ELM, the input parameters of the network are chosen randomly and the output weights are estimated analytically. As the output weights are calculated analytically, these networks are computationally less intensive and learn the training data in short duration. However, the original ELM is a batch learning algorithm that requires the network structure to be fixed a priori. In such a network, the number of hidden neurons has to be carefully chosen. Choosing too many neurons may result in over-fitting, while choosing too few neurons will result in inaccurate approximation of the decision surface. Later, a sequential learning algorithm for the ELM, namely online sequential extreme learning machine (OS-ELM) that processes data chunk-by-chunk online, has been developed in [10, 11], and an OS-ELM with a forgetting term, namely FOS-ELM, has been developed in [12]. However, the number of neurons is fixed a priori in OS-ELM, and it suffers from issues due to insufficient or excessive number of hidden neurons. Recently, an incremental extreme learning machine (I-ELM) has been developed in [13]. In an I-ELM, the network begins with zero hidden neurons, and neurons are added during the training process. When a new neuron is added to an I-ELM, the input parameters are chosen randomly, and the output weights are estimated analytically. Further, whenever a new neuron is added, the output weights of the remaining neurons are also updated. However, the randomly generated centers may not represent the training data set efficiently. Moreover, the samples that do not contribute to the addition of neurons do not take part in the learning process, effectively. For a complete survey of the ELM algorithms available in the literature, one must refer to [14]. ELM and its extensions have been used to solve classification problems like classification in P2P networks [15], intrusion detection [16], image quality assessment [17, 18], object tracking [19], protein secondary structure prediction [20], micro-array gene classification [21, 22].

All the above-mentioned algorithms use all the sample in the training data set to learn the decision function. These algorithms operate with an implicit assumption that samples are uniformly distributed in the input space. However, samples are usually not distributed uniformly in the input space, and learning all the samples in the training data set results in over-fitting of the samples in the densely populated region of the input space. Recently, it has been shown in human learning literature that self-regulated learning that is capable of assessing *what-to-learn*, *when-to-learn* and *how-to-learn* is the best learning strategy [23]. The aforementioned algorithms address only *how-to-learn* and lack the ability to assess their knowledge with respect to the knowledge in the training data set. Meta-cognition, meaning knowledge about knowledge, enables human to assess one's knowledge capabilities and helps to self-regulate learning in an individual. As all the machine learning algorithms available in the literature are inspired from the principles of human learning (For e.g., [24]), extending the principles of self-regulation in a meta-cognitive framework is an important aspect to develop efficient machine learning algorithms.

Recently, a class of meta-cognitive learning algorithms [7, 25–28] has been developed based on the simple Nelson and Narens model of human meta-cognition [29]. These algorithms address *what-to-learn*, *when-to-learn* and *how-to-learn* in a meta-cognitive framework. It has been shown in the literature that these algorithms perform better than the learning algorithms available in the literature and have a better generalization ability. A complete review of the meta-cognitive machine learning algorithms available in the literature is presented in Sect. 2.

In this paper, we extend the principles of meta-cognition to the ELM and develop a fast meta-cognitive extreme learning machine (McELM). Similar to the Nelson and Narens model of meta-cognition, McELM has two components, namely a cognitive component and a meta-cognitive component. An ELM with the q-Gaussian activation function at the hidden layer forms the cognitive component. A q-Gaussian function parameterizes standard Gaussian distribution by replacing exponential expressions with q-exponential expressions [30]. Thus, the modification of the q parameter allows the representation of different basis functions (Cauchy, inverse multi-quadratic, etc.) and helps the q-Gaussian function to match the shape of the kernel and the distribution of the distances better [30]. On the other hand, a self-regulatory learning mechanism is the meta-cognitive component of McELM. The meta-cognitive component has a dynamic model of the cognitive component and chooses suitable learning strategies for every sample in the training data set. When a new sample is presented to McELM, the meta-cognitive component compares the knowledge contained in the sample with that of the cognitive component using the instantaneous hinge-loss error [31, 32] and the predicted class label as the

knowledge measures. Based on these knowledge measures, the meta-cognitive component chooses either the sample deletion strategy or the sample learning strategy or the sample reserve strategy. The sample deletion strategy addresses *what-to-learn*, while the sample learning and sample reserve strategies address *when-to-learn* and *how-to-learn* components of meta-cognition, respectively. The sample learning strategy includes the neuron addition strategy and parameter update strategy. When a new neuron is added, the center of the neuron is chosen based on the input of the sample, the Gaussian width and the q-parameters are chosen randomly. The output weight is estimated using the least squares estimate based on the hinge-loss error function, as the hinge-loss error function estimates the posterior probability much better than the mean-squared error function [32]. The performance of McELM is studied on a set of multi-category and binary classification problems from the UCI machine learning repository [33], in comparison with the support vector machine (SVM), ELM [8], self-regulatory resource allocation network (SRAN) [6] and the recently developed meta-cognitive neural network (McNN) [7]. It can be observed from the performance study results that the McELM performs better than the existing classifiers and also helps to improve the generalization ability of ELM.

The paper is organized as follows: In Sect. 2, the various meta-cognitive learning algorithms available in the literature are reviewed. Section 3 presents the architecture and learning algorithm of the McELM classifier. The performance of McELM in comparison with SVM, ELM, SRAN and McNN on benchmark multi-category and binary classification problems is studied in detail in Sect. 4 Finally, Sect. 5 summarizes the main contributions of the paper.

Review on Meta-Cognitive Machine Learning Algorithms

Meta-cognition is defined as knowledge about knowledge. In the literature of human learning, meta-cognition enables individuals to assess their own knowledge with respect to the global knowledge and helps to self-regulate one's learning process. There are several models of human meta-cognition and a review of these models is presented in [34]. Nelson and Naren proposed a simple, imitable model of meta-cognition [29] as shown in Fig. 1.

Inspired by this simple model of meta-cognition, several meta-cognitive machine learning algorithms have been developed for real-valued neural networks, complex-valued neural networks and neuro-fuzzy inference systems. SRAN [6] is the first self-regulated sequential learning algorithm for the real-valued RBF neural network. It uses a self-regulatory learning mechanism that decides *what-to-learn*, *when-to-learn* and *how-to-learn* for a RBF network with a Gaussian activation function. However, it uses only the misclassification error and maximum hinge-loss error as the knowledge measures and does not represent the knowledge contained in the training sequence, efficiently. To overcome this issue, a meta-cognitive neural network that uses the confidence of the classifier and the class-wise significance of the classifier, in addition to the estimated class label and maximum hinge-loss error as the knowledge measure, has been developed in [7]. Since McNN uses class-wise knowledge measures, it can represent the knowledge in training data efficiently. However, it uses the computationally intensive extended Kalman filter to update the parameters of the network. Therefore, although SRAN and McNN are sequential in nature, it takes huge computational effort to train these classifiers.

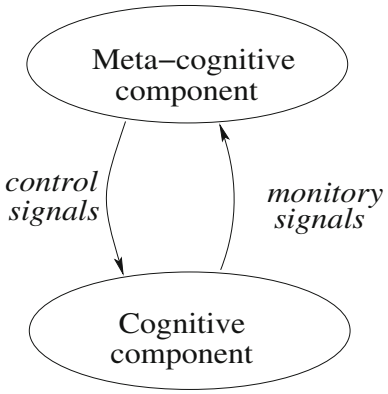

Fig. 1 Nelson and Narens model of meta-cognition

In the complex domain, a sequential learning complex-valued self-regulatory resource allocation network that is similar to the SRAN has been developed in [35]. The self-regulatory learning mechanism of CSRAN uses the instantaneous magnitude and phase errors of the sample as the error measure and selects suitable learning strategies for each sample presented to a fully complex-valued radial basis function (FC-RBF) [36] network. FC-RBF uses a fully complex-valued *sech* activation function and fully complex-valued gradients to update the network parameters using a fully complex-valued extended Kalman filter [37]. Later, a Meta-cognitive Fully Complex-valued Radial Basis Function network (Mc-FCRBF) with the FC-RBF as the cognitive component and a self-regulatory learning mechanism, using magnitude and phase errors as the knowledge measures, as the meta-cognitive component, has been developed in [25]. The self-regulatory learning mechanism of Mc-FCRBF chooses appropriate samples to participate in the training process and uses a fully complex-valued gradient descent batch learning algorithm to update the parameters of the network iteratively. Recently, a fast learning fully complex-valued meta-cognitive learning algorithm namely, Meta-cognitive Fully Complex-valued

Relaxation Network (McFCRN), has been developed in [26]. The cognitive component of McFCRN is a fully complex-valued relaxation network [38]. When a new neuron is added, the input/hidden layer parameters of the neuron are chosen randomly and its output weights are estimated based on a logarithmic error function that represents the errors in magnitude and phase, explicitly. Thus, McFCRN is a fast learning fully complex-valued meta-cognitive neural network.

The principles of meta-cognition have also been extended to develop a Meta-cognitive neuro-Fuzzy Inference System (McFIS) [39, 40, 41]. The cognitive component of McFIS is a four-layered Takagi–Sugeno–Kang type-0 neuro-fuzzy inference system. The self-regulatory learning mechanism of McFIS, which is its meta-cognitive component, uses the class specific knowledge measure and the maximum hinge-loss error of a sample to choose suitable learning strategy for a given sample.

From the studies on the various meta-cognitive machine learning algorithms available in the literature discussed above, it can be inferred that meta-cognitive component of these models help them to improve their generalization ability, by preventing the network from learning repetitive knowledge. As an ELM is a fast learning algorithm, it is necessary to extend the principles of meta-cognition to ELMs, so as to develop a fast meta-cognitive learning algorithm. In the next section, we develop a meta-cognitive learning algorithm for an ELM classifier with a q-Gaussian activation function at the hidden layer of the cognitive component.

A Meta-Cognitive Extreme Learning Machine Classifier

In this section, we describe the architecture and learning algorithm of a McELM classifier. The classification problem can be defined as: Given N training samples, $\{(\mathbf{x}^1, c^1), \ldots, (\mathbf{x}^t, c^t), \ldots, (\mathbf{x}^N, c^N)\}$, where $\mathbf{x}^t = [x_1^t, \ldots, x_m^t]^T \in \Re^m$ are the m-dimensional real-valued input features of tth observation and $c^t \in \{1, 2, \ldots, C\}$ is its class label. The coded class label $\mathbf{y}^t = [y_1^t, \ldots, y_C^t]^T \in \Re^C$ is obtained using:

$$y_l^t = \begin{cases} 1, & \text{if } c_t = l, \\ -1, & \text{otherwise} \end{cases} \quad l = 1, 2, \ldots, C \tag{1}$$

Now, the objective of a classifier can be defined as approximating the decision function ($\mathbb{F}$) that maps the input features to the coded class labels, i.e., $\mathbb{F} : \Re^m \rightarrow \Re^C$ as close as possible, and then predicting the class labels of new, unseen samples with certain accuracy.

To accomplish this objective, McELM has two components: a cognitive component and a meta-cognitive component. The cognitive component of McELM is an ELM employing q-Gaussian activation function, while a self-regulatory learning mechanism is its meta-cognitive component. In this section, we describe the architecture and the working principle of these components.

Cognitive Component: Extreme Learning Machine

The cognitive component of the McELM is an ELM with m input neurons, K hidden neurons and C output neurons. The neurons in the input and output layers of McELM are linear, while the neurons in the hidden layer employ the q-Gaussian radial basis function.

All the real-valued meta-cognitive learning algorithms discussed in Sect. 2 use the Gaussian activation function in the hidden layer. However, the Cauchy radial basis function is preferred in applications like image retrieval [42] and computerized tomography [43], while the inverse multi-quadratic radial basis function is preferred in real-time signal processing applications [44]. Therefore, it is desirable to employ an activation function like the q-Gaussian function that helps to employ radial basis functions with kernels, whose shape can be relaxed or contracted.

The response of the neurons in the hidden layer (h_j^t) of the cognitive ELM employing a q-Gaussian RBF is given by:

$$h_j^t = \begin{cases} (r_j^t)^{\frac{1}{1-q}}, & \text{if } r_j^t > 0 \\ 0 & \text{otherwise} \end{cases}; \quad j = 1, \ldots, K \tag{2}$$

$$\text{where,} \quad r_j^t = \left(1 - (1-q)\, d_j(\mathbf{x}^t)\right) \quad \text{and}$$

$$d_j(\mathbf{x}^t) = \frac{\|\mathbf{x}^t - \mathbf{u}_j\|^2}{\sigma_j^2} \tag{3}$$

is the radial distance of the sample $\mathbf{x}^t$ from the center of the jth neuron ($\mathbf{u}_j \in \Re^m$) with width $\sigma_j \in \Re^1$. The parameter $q \in \Re^1$ helps the q-Gaussian RBF to reproduce different RBFs [30]. For example, when $q \rightarrow 1$, the q-Gaussian converges to a Gaussian RBF, while $q \rightarrow 2$ and $q \rightarrow 3$ converge to a Cauchy RBF and inverted multi-quadratic RBF, respectively. Figure 2 presents the response of hidden neurons in a single dimension with $q = 1, \ldots, 8$. As it can be seen from the figure, the widely used Gaussian RBF presents a very selective response, with high activation for samples that are close to the center and very small activation for the samples far away from the center. However, as q increases, the activation for samples far from the center of the RBF is greater than those of the Gaussian RBF for the same samples. This is clearly seen in the figure that the Cauchy RBF and the inverted multi-quadratic RBF have longer tails than the Gaussian RBF. Thus, although all the activations (for different values of q) do not fall asymptotically to zero, as q increases, the decay of the RBF is very

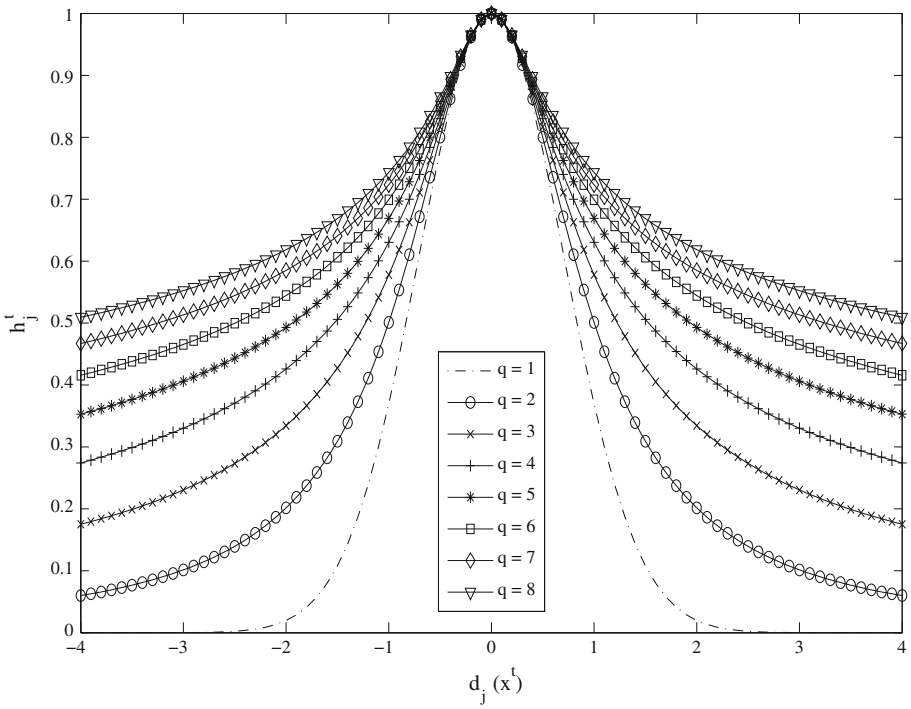

Fig. 2 Responses of q-Gaussian RBF in one-dimensional space with $u = 0$ and $\sigma = 1$ for different values of q

slow for sufficiently large distance norms. Therefore, in this paper, we use the q-Gaussian RBF that considers different types of local functions for different values of q.

The neurons in the output layer of the ELM are linear, and the output of the lth neuron is given by:

$$\hat{y}_l^t = w_{lj} h_j^t; \quad l = 1, \ldots, C; j = 1, \ldots, K \tag{4}$$

the output of the tth sample is given by Eq. (4). Equation (4) can be written in the matrix form as

$$\hat{Y} = WH, \tag{5}$$

where H is the hidden layer output matrix ($H \in \Re^{((K-1)\times t)}$) as shown below:

$$H = \begin{pmatrix} h_1^1(\mathbf{u}_1, \sigma_1, q_1, \mathbf{x}^1) & \ldots & h_1^t(\mathbf{u}_1, \sigma_1, q_1, \mathbf{x}^t) \\ \vdots & \ldots & \vdots \\ h_j^1(\mathbf{u}_j, \sigma_j, q_j, \mathbf{x}^1) & \ldots & h_j^t(\mathbf{u}_j, \sigma_j, q_j, \mathbf{x}^t) \\ \vdots & \ldots & \vdots \\ h_{(K-1)}^1(\mathbf{u}_{(K-1)}, \sigma_{(K-1)}, q_{(K-1)}, \mathbf{x}^1) & \ldots & h_{(K-1)}^t(\mathbf{u}_{(K-1)}, \sigma_{(K-1)}, q_{(K-1)}, \mathbf{x}^t) \end{pmatrix} \tag{6}$$

where $w_{lj} \in \Re$ is the output weight connecting the jth hidden neuron and the lth output neuron.

Learning Algorithm of the Cognitive ELM

The cognitive component of McELM begins with zero hidden neurons, adds neurons and updates the parameters of the existing network using the training samples. Without loss of generality, let us assume that $K - 1$ hidden neurons are added with $t - 1$ samples. The response of the jth hidden neuron for the tth sample is given by Eq. (2), and

The output weight W can be computed analytically as:

$$W = YH^{\dagger} \tag{7}$$

where the operator † represents the matrix pseudo-inverse operator.

Meta-Cognitive Component: Self-Regulatory Learning Mechanism

A self-regulatory learning mechanism that decides *what-to-learn*, *when-to-learn* and *how-to-learn* the training samples

based on the relative knowledge of the cognitive ELM with respect to the training data set is the meta-cognitive component of McELM. It uses the predicted class label of the cognitive ELM and the maximum hinge-loss error of the sample to choose one of the following learning strategies, for a given sample:

(1) **Sample Deletion:** Delete the samples without learning.
(2) **Sample Learning:** Use the sample to add a neuron or to update the existing network parameters.
(3) **Sample Reserve:** Reserve the sample for future use.

The knowledge measures used to choose an appropriate learning strategy for the current sample are defined as follows:

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

$$\widehat{c}^t = \arg \max_{j \in 1,2,\ldots,C} \widehat{y}_j^t \tag{8}$$

- **Maximum Hinge-loss Error** (E^t): The objective of the classifier is to minimize the error between the predicted output $(\widehat{\mathbf{y}}^t)$ and actual output $(\mathbf{y}^t)$. It has been shown in the literature [32] that a classifier developed using the hinge-loss function estimates the posterior probability more accurately than that developed using the mean-squared error function. Hence, we use the hinge-loss error $\left(\mathbf{e^t} = \left[e_1^t, \ldots, e_j^t, \ldots, e_C^t\right]^T\right) \in \Re^C$ defined as

$$e_j^t = \begin{cases} 0 & \text{if } y_j^t \widehat{y}_j^t > 1 \\ y_j^t - \widehat{y}_j^t & \text{otherwise} \end{cases} \quad j = 1, 2, \ldots, C \tag{9}$$

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

$$E^t = \max_{j \in 1,2,\ldots,C} \left| e_j^t \right| \tag{10}$$

Based on these knowledge measures, when a new sample is presented to McELM, it selects one of the following learning strategies:

(1) **Sample Deletion**: When the predicted class label of a new training sample t is the same as the actual class label and the maximum hinge-loss error is less than the delete threshold (E_d), the training sample does not contain novel knowledge and can be deleted from training sequence without being used in learning process. The sample deletion criterion is given by

$$c^t == \widehat{c}^t \quad \text{AND} \quad E^t \leq E_d \tag{11}$$

The delete threshold (E_d) decides the approximation accuracy of the McELM classifier. Therefore, if it is selected close to zero, then all the samples will participate in the training process, which might result in over-fitting. On the other hand, increasing E_d results in deletion of too many samples, and hence, inaccurate approximation. Hence, E_d is fixed at the expected accuracy level. In our simulation studies, it is selected in the range of [0.1, 0.2]. The sample deletion strategy prevents learning of samples with similar knowledge and avoids over-fitting. As it deletes a few training samples, it also reduces the computational effort.

(2) **Sample Learning**: The meta-cognitive component uses the current sample either to add a neuron or to update the parameters of an existing network. McELM begins with zero hidden neurons, adds neurons and updates the parameters of the existing neurons to achieve an optimal network structure with optimum parameters. Below, we describe these two strategies:

(a) ***Neuron Addition Strategy:*** If the predicted class label of a new training sample $(\widehat{c}^t)$ is different from the actual class label (c^t), or the maximum hinge-loss error of the current sample (E^t) is greater than the neuron addition threshold (E_a), then the sample contains novel knowledge and is used to add a new neuron to the cognitive ELM, i.e.,

$$\begin{aligned} &\text{If } \widehat{c}^t \neq c^t \text{ OR } E^t > E_a, \\ &\quad \text{Then, add a new neuron} \end{aligned} \tag{12}$$

where E_a is the self-regulating neuron addition threshold. When a new neuron is added, E_a is self-regulated according to:

$$E_a := \delta E_a + (1 - \delta) E^t \tag{13}$$

where δ is the slope that controls rate of self-adaptation and is set close to 1. The maximum value that E^t can take is 2. If E_a is chosen closer to 100 % of the maximum hinge error, then very few samples will be used for adding neurons, and the resultant network will not be able to approximate the decision function accurately. If a lower value is chosen, then many neurons will be added to the network, and the generalization ability of the network will be compromised. Hence, the range for the initial value of meta-cognitive neuron addition threshold is usually selected in the interval [1.3, 1.7].

When a new neuron is added, the centers $(\mathbf{u}_K)$ are chosen as the input features of the sample $(\mathbf{x}^t)$, and the width (σ_K) and q parameter are chosen randomly. With the new hidden neuron, the dimension of the hidden neuron response matrix (H) increases from $(t-1) \times (K-1)$ to $t \times K$, i.e.,

$$H^t = \left[\begin{array}{c|c} & h_1^t \\ & \vdots \\ H^{t-1} & h_j^t \\ & \vdots \\ & h_{(K-1)}^t \\ \hline h_K^1, \cdots, h_K^{(t-1)} & h_K^t \end{array}\right] \quad (14)$$

where h_j^t can be calculated using Eq. (2). The output weights can then be obtained by:

$$W^t = YH^{t^\dagger} \quad (15)$$

It must be noted here that the pseudo-inverse of the hidden layer matrix H is computed whenever a neuron is added and could result in increased computational effort. However, the computational cost involved could be considerably reduced by computing the generalized LM-inverse of the matrix H^t [45].

(b) ***Parameter Update Strategy:*** The current training sample $(\mathbf{x}^t, \mathbf{y}^t)$ is used to update the output weights of the cognitive component $(\mathbf{W}_K = [\mathbf{w}_1, \ldots, \mathbf{w}_C]^T)$ if the following criterion is satisfied:

$$c^t == \widehat{c}^t \quad \textbf{AND} \quad E^t \geq E_u \quad (16)$$

where E_u is the self-adaptive meta-cognitive parameter update threshold. If E_u is chosen closer to 50 % of the maximum hinge error, then very few samples will be used for adapting the network parameters, and most of the samples will be pushed to the end of the training sequence. The resultant network will not approximate the function accurately. If a lower value is chosen, then all samples will be used in updating the network parameters without altering the training sequence. Hence, the range for the initial value of meta-cognitive parameter update threshold is selected within the interval [0.4, 0.7]. The E_u is adapted based on the prediction error as:

$$E_u := \delta E_u + (1 - \delta)E^t \quad (17)$$

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

When a sample is used to update the output weight parameters, McELM estimates the hidden layer response of the K neurons for the sample t $(h_j^t; j = 1, \ldots, K)$ and updates the output weights using recursive least squares according to:

$$W^t = W^{t-1} + P^t \mathbf{h}^t \mathbf{e}^t \quad (18)$$

where, $\mathbf{h}^t = \left[h_1^t(\mathbf{u}_1, \sigma_1, q_1, \mathbf{x}^t), \ldots, h_K^t(\mathbf{u}_K, \sigma_K, q_K, \mathbf{x}^t)\right]$

and $P^t = P^t - \dfrac{P^{t-1}\mathbf{h}^t(\mathbf{h}^t)^T P^{t-1}}{1 + (\mathbf{h}^t)^T P^{t-1}\mathbf{h}^t}$ (19)

Here, the superscript T represents the matrix transpose operator, $\mathbf{e}^t$ is the hinge-loss error defined in Eq. (9), and $P^0 = (\mathbf{h}^{1^T}\mathbf{h}^1)^{-1}$. It must be noted here that the parameter update equations are similar to the recursive least squares estimate given in [10]. However, the update derived in [10] is based on the mean-squared error, while in McELM, the update is derived based on the hinge-loss error defined in Eq. (9). This is because McELM is a classifier, and it has been shown in [32] that the hinge-loss error function helps to estimate the posterior probability more accurately than the mean-squared error in classification problems.

(3) **Sample Reserve**: When a sample does not satisfy the sample deletion strategy and the sample learning strategy, it is reserved in the training set for future use. After the network is sufficiently trained, the reserved samples might satisfy either the neuron addition or the parameter updated strategies, based on the updated values of the neuron addition and parameter update thresholds. Thus, the reserved samples may be used at a later time.

The learning algorithm of McELM can be summarized as follows:

Step 1. For a sample $(\mathbf{x}^t, c^t)$,
Step 2. Compute the hidden layer output using Eq. (2)
Step 3. Compute the network output $(\widehat{\mathbf{y}}^t)$ using Eq. (4), the predicted class label $(\widehat{c}^t)$ using Eq. (8) and the maximum hinge-loss error (E^t) using Eq. (10).
Step 4. For a given sample $(\mathbf{x}^t, c^t)$,

a. If **Sample Deletion** criterion $(\widehat{c}^t == c^t$ AND $E^t < E_d)$ is satisfied, then delete the sample from the training sequence.
b. Elseif **Neuron addition** criterion $(\widehat{c}^t \neq c^t$ OR $E^t > E_a)$ is satisfied, then add a neuron $(K = K + 1)$. Update the matrix H [Eq. (14)] and compute the output weight according to Eq. (15).
c. Elseif **Parameter update** criterion $(\widehat{c}^t == c^t$ AND $E^t > E_u)$ is satisfied, then update the existing network parameters using Eq. (18).
d. Else, **Reserve** the sample for future use.

Step 5. Increment $t = t + 1$. Go to step 2.

Table 1 Description of the various classification problems used in the performance study

Type of data set	Prob.	No. of features	No. of classes	No. of samples		IF	
				Train	Test	Train	Test
Multi-categ.	Image segmentation	19	7	210	2,100	0	0
	Vehicle classification	18	4	424	422	0.1	0.12
	Glass identification	9	6	109	105	0.68	0.73
	Iris	4	3	45	105	0	0
	Wine	13	3	60	118	0	0.29
Binary	Liver disorder	6	2	200	145	0.17	0.145
	PIMA data	8	2	400	368	0.225	0.39
	Breast cancer	9	2	300	383	0.26	0.33
	Heart disorder	14	2	70	200	0.14	0.1
	Ionosphere	34	2	100	251	0.28	0.29

Thus, the sample deletion strategy, the sample learning strategy and the sample reserve strategy address the *what-to-learn*, *how-to-learn* and *when-to-learn* components of self-regulation in a meta-cognitive environment and help to achieve an improved generalization ability.

In the next section, we study the classification ability of McELM using benchmark classification problems from the UCI machine learning repository [33].

Performance Study

In this section, the classification ability of McELM is evaluated in comparison with SVM, ELM and McNN on a set of multi-category and binary classification problems from the UCI machine learning repository. 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 done using the LIBSVM package in C [46]. The results of SVM, ELM and McNN are reproduced from [7]. The problems used to evaluate the performances of the classifiers are listed in Table 1. The table presents the number of features, the number of classes, the number of samples used in training and testing and the imbalance factor (IF) of the data sets used in the performance study. The availability of small number of samples, the sampling bias and the overlap between classes introduce complexity in the classification and may affect the classification performance of the classifier [32]. To study the effect of these factors, we also consider the IF of the training and testing data sets, defined as:

$$\text{IF} = 1 - \frac{C}{N} \min_{l=1\ldots C} N_l \tag{20}$$

where N_l is the number of samples belonging to a class l. It must be noted that $N = \sum_{l=1}^{C} N_l$. The IF gives a measure of the sample imbalance in the various classes of the training data set, and the IF of each data set considered in this study is also presented in Table 1. The training data set of the image segmentation and iris problems is balanced data sets, while the remaining data sets are unbalanced in nature. As it can be observed from the table, the IFs of the training data sets vary widely in the range from 0.1 to 0.68 and those of the testing data sets vary widely between 0.1 and 0.73. It must also be noted that the imbalance factors of the training data set and that of the testing data set are not very similar. For example, the training data set for the wine problem is balanced, while its testing data set is unbalanced.

Performance Measures

The overall and average classification efficiencies are used to compare the performance of McELM classifier against the other classifiers available in the literature.

Average classification efficiency: The average classification efficiency (η_a) is defined as the average ratio of number of correctly classified samples in each class to the total number of samples in each class.

$$\eta_a = \frac{1}{C} \sum_{l=1}^{C} \frac{p_{ll}}{N_l} \times 100\,\% \tag{21}$$

where p_{ll} is the total number of correctly classified samples in the training/testing data set.

Overall classification efficiency: The overall classification efficiency (η_o) is defined as the ratio of total number of correctly classified samples to the total number of samples available in the training/testing data set.

$$\eta_o = \frac{\sum_{l=1}^{C} p_{ll}}{N} \times 100\,\% \tag{22}$$

Table 2 Performance comparison for the multi-category classification problems

Problem	Classifier	No. of neurons (K)	Training time (S)	Testing	
				η_o	η_a
Image segmentation	SVM	127[a]	721	91.38	91.38
	ELM	49	0.25	90.23	90.23
	SRAN	48	22	93	93
	McNN	49	10.67	93.38	93.38
	McELM	50	41	93.76	93.76
Vehicle classification	SVM	340[a]	550	70.62	68.51
	ELM	150	0.4	77.01	77.59
	SRAN	113	55	75.12	76.86
	McNN	146	562	77.72	78.72
	McELM	120	40	81.04	81.3
Glass identification	SVM	183[a]	320	70.47	75.61
	ELM	80	0.05	81.31	87.43
	SRAN	59	28	86.2	80.95
	McNN	73	13.6	85.71	87.03
	McELM	72	3.2	82.86	87.4
Iris	SVM	13[a]	0.02	96.19	96.19
	ELM	10	0.01	96.19	96.19
	SRAN	8	24.3	96.19	96.19
	McNN	5	0.23	97.14	97.14
	McELM	6	0.03	98.1	98.1
Wine	SVM	13[a]	0.1	97.46	98.04
	ELM	10	0.25	97.46	98.04
	SRAN	12	76	96.61	97.19
	McNN	9	0.26	98.3	98.49
	McELM	9	0.02	98.31	98.69

[a] Number of support vectors

Table 3 Performance comparison for the binary classification problems

Problem	Classifier	No. of neurons (K)	Training time (S)	Testing		$F1$ Score
				η_o	η_a	
LD	SVM	141[a]	0.1	71.03	70.21	0.6553
	ELM	100	0.17	72.41	71.41	0.6655
	SRAN	91	3.38	66.9	65.8	0.5996
	McNN	68	2	73.79	71.60	0.6434
	McELM	50	0.95	74.48	73.83	0.6975
PD	SVM	221[a]	0.21	77.45	76.33	0.8276
	ELM	400	0.29	76.63	75.25	0.8219
	SRAN	97	12.24	78.53	74.9	0.8447
	McNN	76	6.45	80.16	77.31	0.8550
	McELM	25	0.47	80.43	78.49	0.8543
BC	SVM	24[a]	0.11	96.6	97.06	0.9492
	ELM	66	0.14	96.35	96.5	0.9455
	SRAN	7	0.17	96.87	97.3	0.9531
	McNN	9	0.3	97.39	97.84	0.9606
	McELM	10	0.05	97.39	97.84	0.9606
HD	SVM	42[a]	0.04	75.5	75.1	0.7238
	ELM	36	0.15	76.5	75.9	0.7296
	SRAN	28	0.53	78.5	77.53	0.7430
	McNN	26	0.55	80.5	79.65	0.7696
	McELM	26	0.19	81.5	80.15	0.7701
Isp.	SVM	43[a]	0.02	91.24	88.51	0.9372
	ELM	32	0.04	89.64	87.52	0.9231
	SRAN	21	3.7	90.84	91.88	0.9228
	McNN	20	1.02	95.62	95.60	0.9651
	McELM	18	0.1	94.82	93.76	0.9610

[a] Number of support vectors

Performance Study on Multi-Category Classification Problems

In this section, we study the performance of the McELM with the q-Gaussian radial basis function on multi-category benchmark classification problems listed in Table 1. It can be observed from the table that the generalization performance of McELM is better than other classifiers used in comparison on all the multi-category classification problems. Another point worth noting is that McELM uses only fewer samples to approximate the decision function defined by these problems. For example, it requires only 412 samples in the vehicle classification problem and 100 samples in the glass identification problem. In the IRIS problem, the meta-cognitive component of McELM deletes 24 samples, while in the wine classification problem, it deletes 22 samples and achieves this generalization performance.

It can also be observed that McELM boosts the classification performance of ELM at least by 2 %. This is attributed to both the q-Gaussian activation function that helps to realize different radial basis functions for different values of the parameter q and the meta-cognitive component of McELM. Moreover, it requires lesser training time compared to the SRAN and McNN, which are self-regulating in nature. This is because SRAN and McNN use the computationally intensive EKF to update the network parameters, while McELM updates the network parameters analytically.

Performance Study on Binary Classification Problems

The performance results of McELM in comparison with SVM, ELM, SRAN and McNN for the binary benchmark classification problems chosen for the study are presented in Table 3. From the Table, it can be seen that McELM

performs better than the other classifiers in these problems, although its performance is similar to that of McNN in the breast cancer and ionosphere problems. It can be clearly seen that McELM outperforms ELM in all these problems. The improvement in performance over ELM is at least 2 %. This is due to the use of q-Gaussian activation function and the meta-cognitive component of McELM. Similar to the multi-category classification problems, the meta-cognitive component deletes similar samples from the training data set in the binary classification problems. The number of samples deleted during training from the liver disorder, PIMA data, breast cancer data set, heart disorder and ionosphere data sets is 9, 41, 114, 8 and 14, respectively. Deleting samples with similar information helps to improve the generalization ability of the McELM classifier. It can also be observed that McELM requires less computational effort than the McNN and SRAN in performing the binary classifications.

Thus, from the performance study on different classification problems, it can be inferred that McELM outperforms existing algorithms in literature, and the meta-cognitive component helps to improve the generalization ability of ELM.

Conclusions

In this paper, we have presented a meta-cognitive learning algorithm for an ELM classifier. McELM has two components: a cognitive component and a meta-cognitive component. The cognitive component of McELM is a three-layered ELM classifier with a q-Gaussian activation function at the hidden layer. The q-Gaussian activation function helps to realize different radial basis functions for different values of the q-parameter and helps in better approximation of the decision function. The meta-cognitive component of McELM has a self-regulatory learning mechanism that decides *what-to-learn*, *when-to-learn* and *how-to-learn* in a meta-cognitive framework. As the samples in the training set are presented one-by-one, the meta-cognitive component uses the hinge-loss error and the class label predicted by the cognitive component for the sample to choose suitable learning strategies for the sample. The various strategies that may be chosen include sample delete, sample learn or sample reserve. The performance of McELM is evaluated on a set of benchmark classification problems from the UCI machine learning repository. Performance study results show that McELM performs better than or similar to existing meta-cognitive classifiers with lesser computational effort. It can also be observed that McELM helps to improve the generalization ability of the cognitive ELM.

References

1. Platt JC. A resource allocation network for function interpolation. Neural Comput. 1997;2(2):213–25.
2. Yingwei L, Sundararajan N, Saratchandran P. A sequential learning scheme for function approximation using minimal radial basis function neural networks. Neural Comput. 1997;9(2):461–78.
3. Yingwei L, Sundararajan N, Saratchandran P. Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm. IEEE Trans Neural Netw. 1998;9(2):308–18.
4. Huang GB, Saratchandran P, Sundararajan N. An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. IEEE Trans Syst Man Cybern Part B Cybern. 2004;34(6):2284–92.
5. Suresh S, Sundararajan N, Saratchandran P. A sequential multi-category classifier using radial basis function networks. Neurocomputing. 2008;71(7–9): 1345–58.
6. Suresh S, Dong K, Kim HJ. A sequential learning algorithm for self-adaptive resource allocation network classifier. Neurocomputing. 2010;73(16–18):3012–9.
7. Babu GS, Suresh S. Meta-cognitive neural network for classification problems in a sequential learning framework. Neurocomputing. 2012;81:86–96.
8. Huang GB, Siew CK. Extreme learning machine with randomly assigned RBF kernels. Int J Inf Technol. 2005;11(1):16–24.
9. Huang G.-B, Zhou H, Ding X, Zhang R. Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B Cybern. 2012;42(2):513–29.
10. Liang N-Y, Huang GB, Saratchandran P, Sundararajan N. A fast and accurate on-line sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw. 2006;17(6): 1411–23.
11. Rong HJ, Huang GB, Sundararajan N, Saratchandran P. Online sequential fuzzy extreme learning machine for function approximation and classification problems. IEEE Trans Syst Man Cybern Part B Cybern. 2009;39(4):1067–72.
12. Zhao J, Wang Z, Park DS. Online sequential extreme learning machine with forgetting mechanism. Neurocomputing. 2012;87:79–89.
13. Huang GB, Chen L, Siew CK. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw. 2006;17(4):879–92.
14. Huang G.-B, Wang DH, Lan Y. Extreme learning machines: a survey. Int J Mach Learn Cybern. 2011;2(2):107–22.
15. Sun Y, Yuan Y, Wang G. An OS-ELM based distributed ensemble classification framework in p2p networks. Neurocomputing. 2011;74(16):2438–43.
16. Cheng C, Tay WP, Huang GB. Extreme learning machines for intrusion detection. International Joint Conference on Neural Networks 2012 (IJCNN 2012) (Art no. 6252449) 2012.
17. Suresh S, Babu RV, Kim HJ. No-reference image quality assessment using modified extreme learning machine classifier. Appl Soft Comput. 2009;9(2):541–52.
18. Decherchi S, Gastaldo P, Zunino R, Cambria E, Redi J. Circular-ELM for the reduced-reference assessment of perceived image quality. Neurocomputing. 2013;102:78–89.
19. Babu RV, Suresh S, Makur A. Online adaptive radial basis function networks for robust object tracking. Comput Vis Image Underst. 2010;114(3):297–10.
20. Wang G, Zhao Y, Wang DD. A protein secondary structure prediction framework based on the extreme learning machine. Neurocomputing. 2008;72(1–3):262–8.
21. Suresh S, Saraswathi S, Sundararajan N. Performance enhancement of extreme learning machine for multi-category sparse data classification problems. Eng Appl Artif Intell. 2010;23(7):149–57.

22. Saraswathi S, Sundaram S, Sundararajan N, Zimmermann M, Nilsen-Hamilton M. ICGA-PSO-ELM approach for accurate multiclass cancer classification resulting in reduced gene sets in which genes encoding secreted proteins are highly represented. IEEE/ACM Trans Comput Biol Bioinform. 2011;8(2):452–63.
23. Isaacson R, Fujita F. Metacognitive knowledge monitoring and self-regulated learning: academic success and reflections on learning. J Scholarsh Teach Learn. 2006;6(1):39–55.
24. Cambria E, Olsher D, Kwok, K. Sentic activation: A two-level affective common sense reasoning framework. In: Proceedings of association for the advancement of artificial intelligence conference, Toronto; 2012. p. 186–92.
25. Savitha R, Suresh S, Sundararajan N. Meta-cognitive learning in fully complex-valued radial basis function network. Neural Comput. 2012;24(5):1297–28.
26. Savitha R, Suresh S, Sundararajan N. A meta-cognitive learning algorithm for a fully complex-valued relaxation network. Neural Netw. 2012;32:209–18.
27. Babu GS, Suresh S. Meta-cognitive RBF network and its projection based learning algorithm for classification problems. Appl Soft Comput. 2013;13(1):654–66.
28. Babu GS, Suresh S. Sequential projection-based metacognitive learning in a radial basis function network for classification problems. IEEE Trans Neural Netw. 2012;24(2):194–206.
29. Nelson TO, Narens L. Metacognition: core readings, ch. Metamemory: a theoretical framework and new findings: Allyn and Bacon: Boston, T. O. Nelson (ed.) ed., 1980. p. 9–24.
30. Fernndez-Navarro F, Hervs-Martnez C, Gutirrez PA, Pea-Barragn JM, Lpez-Granados F. Parameter estimation of q-Gaussian Radial Basis Functions Neural Networks with a Hybrid Algorithm for binary classification. Neurocomputing. 2012;75:123–34.
31. Zhang T. Statistical behavior and consistency of classification methods based on convex risk minimization. Ann Stat. 2003;32(1):56–85.
32. Suresh S, Sundararajan N, Saratchandran P. Risk-sensitive loss functions for sparse multi-category classification problems. Inf Sci. 2008;178(12):2621–38.
33. Blake C, Merz C. UCI repository of machine learning databases. Department of Information and Computer Sciences, University of California, Irvine, [http://archive.ics.uci.edu/ml/] 1998.
34. Cox MT. Metacognition in computation: a selected research review. Artif Intell. 2005;169(2):104–41.
35. Suresh S, Savitha R, Sundararajan N. A sequential learning algorithm for complex-valued resource allocation network-CSRAN. IEEE Trans Neural Netw. 2011;22(7):1061–72.
36. Savitha R, Suresh S, Sundararajan N. A fully complex-valued radial basis function network and its learning algorithm. Int J Neural Syst. 2009;19(4):253–67.
37. Goh SL, Mandic DP. An augmented extended Kalman filter algorithm for complex-valued recurrent neural networks. Neural Comput. 2007;19(4):1039–55.
38. Savitha R, Suresh S, Sundararajan N. A fast learning fully complex-valued relaxation network (FCRN). In: Proceedings of the international joint conference on neural networks 2011; p. 1372–7.
39. Subramanian K, Suresh S. A sequential learning algorithm for meta-cognitive neuro-fuzzy inference system for classification problems. In: Proceedings of the international joint conference on neural networks 2011; p. 2507–12.
40. Subramanian K, Suresh S. Human action recognition using meta-cognitive neuro-fuzzy inference system. Proc Int Jt Conf Neural Netw. 2012;22(6):1250028-1–15.
41. Subramanian K, Suresh S. A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system. Appl Soft Comput. 2012;12(11):3603–14.
42. Shkurko K, Qi X. A radial basis function and semantic learning space based composite learning approach to image retrieval. Proc ICASSP IEEE Int Conf Acoust Speech Signal Process. 2007;1:945–8.
43. Zhang J, Li H. A reconstruction approach to ct with cauchy rbfs network. Advances in Neural Networks: ISNN2004, Lecture Notes in Computer Science 2004; 3174:234–6.
44. Saranli A, Baykal B. Complexity reduction in radial basis function (RBF) networks by using radial B-spline functions. Neurocomputing. 1998;18(1–3):183–94.
45. Udwadia FE, Phohomsiri P. Generalized LM-inverse of a matrix augmented by a column vector. Appl Math Comput. 2007;190:999–06.
46. Chang C.-C, Lin C.-J. LIBSVM: a library for support vector machines (software available at /http://www.csie.ntu.edu.tw/cjlin/libsvms). ACM Trans Intell Syst Technol 2011;2(27):1–27.