Meta-cognitive Regression Neural Network for Function Approximation: Application to Remaining Useful Life Estimation

G. Sateesh Babu, Xiao-Li Li, and S. Suresh

***Abstract*—In this paper, we present a novel approach for Remaining Useful Life (RUL) estimation problem in prognostics using a proposed 'sequential learning Meta-cognitive Regression Neural Network (McRNN) algorithm for function approximation'. The McRNN has two components, namely, a cognitive component and a meta-cognitive components. The cognitive component is an evolving single hidden layer Radial Basis Function (RBF) network with Gaussian activation functions. The meta-cognitive component present in McRNN helps to cognitive component in selecting proper samples to learn based on its current knowledge and evolve architecture automatically. The McRNN employs extended Kalman Filter (EKF) to find optimal network parameters in training. First, the performance of the proposed sequential learning McRNN algorithm has been evaluated using a set of benchmark function approximation problems and is compared with existing sequential learning algorithms. The performance results on these problems show the better performance of McRNN algorithm over the other algorithms. Next, the proposed McRNN algorithm has been applied to RUL estimation problem based on sensor data. For simulation studies, we have used Prognostics Health Management (PHM) 2008 Data Challenge data set and compared with the existing approaches based on state-of-the-art regression algorithms. The experimental results show that our proposed McRNN algorithm based approach can accurately estimate RUL of the system.**

I. Introduction

Prognostic technologies are very crucial in condition based maintenance for diverse application areas, such as manufacturing, aerospace, automotive, heavy industry, power generation, and transportation. Prognostic technologies predict the future performance of a component or a subsystem by accessing the degradation from its expected normal operating conditions and make their Remaining Useful Life (RUL) estimation. If we can accurately predict when an engine will fail, then we can make informed maintenance decision in advance to avoid disasters, reduce the maintenance cost, as well as streamline operational activities. This paper describes the development of a data-driven approach to predict RUL of a complex system as it degrades from an unknown initial state to failure. Clearly, data-driven approach for RUL estimation normally relies on the availability of run-to-failure data, based on which the RUL can be estimated. In literature, RUL is estimated either directly through a multivariate pattern matching process, or indirectly through damage estimation followed by extrapolation to the damage progression [1], [2], [3], [4], [5].

RUL estimation problem requires online, adaptive, and often real-time operation. Such problem pose a serious challenge, since most existing algorithms operated in batch, processing data off-line and requiring random/multiple access to training samples. This means that the incorporation of new data point may be difficult or even impossible to do. Worse still, the time complexity of the best of these algorithms scales super-linearly in the number of training samples, and the evaluation time on new data point often also scales linearly with the size of the training set. Furthermore, most of the existing methods are applicable to linear degradation models. In prognostics, degradation of mechanical systems is typically non-linear in nature, therefore for better and real-time estimation of RUL we need a non-linear and online/sequential learning method.

In a online/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. Many sequential learning algorithms in the literature to solve function approximation problems and depending on training method of the network and the structure, these learning algorithms could be broadly classified as belonging to one the these: error driven algorithms [6], neuron significance based algorithms [7], extreme learning machine based algorithms [8], spiking neural networks algorithms [9], kernel least mean square based algorithms [10], and on-line support vector machine algorithms [11]. In the families of artificial neural networks, Radial Basis Function (RBF) neural 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 RBF framework are available in the literature to solve function approximation problems [12], [6], [13], [7], [8].

All the existing sequential learning algorithms for radial basis function neural networks use all the samples in the training data set to gain knowledge about the information contained in the samples. In other words, they possess information-processing abilities of humans, including perception, learning, remembering, judging, and problem-solving, and these abilities are cognitive in nature. However, recent studies on human learning have revealed that the

Giduthuri Sateesh Babu, Xiao-Li Li are with the Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore (email: giduthurisb, xlli@i2r.a-star.edu.sg).

Sundaram Suresh with the School of Computer Engineering, Nanyang Technological University, Singapore (email: ssundaram@ntu.edu.sg).

learning process is effective when the learners adopt self-regulation in learning process using meta-cognition [14]. Meta-cognition means *cognition about cognition*. In a meta-cognitive framework, human-beings think about their cognitive processes, develop new strategies to improve their cognitive skills and evaluate the information contained in their memory. 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. In literature, meta-cognitive algorithms also available in neural networks and neuro-fuzzy domains [15], [16], [17], [18], [19], [20]. Aforementioned meta-cognitive algorithms were developed for the classification problems, and may not work well for function approximation problems. Hence, there is a need to develop a meta-cognitive neural network regression algorithm for function approximation problems that is capable of deciding *what-to-learn*, *when-to-learn* and *how-to-learn* the decision function from the stream of training data.

In this paper, we propose a Meta-cognitive Regression Neural Network (McRNN) which employs human-like meta-cognition to regulate the sequential learning process. Similar to Nelson and Narens model of meta-cognition [21], McRNN has two components, namely, a cognitive component and a meta-cognitive component. The cognitive component of McRNN is the an evolving single hidden layered RBF network with Gaussian activation functions in the hidden layer. As a new sample is presented to the network, the meta-cognitive component monitors the knowledge in the current sample to decide on *what-to-learn*, *when-to-learn* and *how-to-learn*. These three actions are realized by sample delete strategy, sample reserve strategy, and sample learn strategy, respectively. During sample delete strategy, a sample is deleted if similar knowledge exists in the network. Sample learn strategy proceeds by adding a new neuron to the network or updating the parameters of the network depending on the knowledge present in the sample. Sample reserve strategy will result in the current sample being reserved to be considered for learning at a later stage.

The performance of McRNN algorithm has been evaluated on five benchmark function approximation problems, namely, the system identification problem I [22], the system identification problem II [23], two variants of the Mackey-Glass chaotic time-series prediction problem [24], and the Box-Jenkins gas furnace problem [25]. The performance of McRNN algorithm has been compared to existing sequential learning algorithms, namely, the Minimal Resource Allocation Network (MRAN) [6], Growing and Pruning Radial Basis Function Network (GAP-RBFN) [7] and recently developed Quantized Kernel Least Mean Square (QKLMS) [10] algorithms. MRAN algorithm [6] is similar to the Resource Allocation Network (RAN) algorithm [12]. RAN [12] was the one of the first sequential learning algorithm introduced in the literature. RAN evolves the network architecture using a very simple novelty based neuron growth criterion. While, MRAN algorithm incorporates error based neuron growing and pruning criteria. In GAP-RBFN algorithm [7], growing and pruning criteria of the network is selected based on the significance of a neuron. QKLMS algorithm [10] is based on simple online vector quantization method. In QKLMS algorithm, quantization is applied to compress the input (or feature) space of the kernel adaptive filters so as to control the growth of the RBF network structure. The performance comparison results with these three algorithms show the better prediction ability of McRNN on function approximation problems. Finally, we apply the proposed McRNN algorithm for practical RUL estimation problem from sensor data. For RUL estimation problem, we have used Prognostics Health Management (PHM) 2008 Data Challenge data set [26]. This data set contains simulated data produced using a model based simulation program C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) developed by NASA [27]. The performance of the proposed McRNN on the considered data set for RUL estimation is compared with the results reported in the literature. Results clearly demonstrates that the proposed McRNN algorithm is accurately predicts RUL than existing approaches significantly.

This paper is organized as follows: Section 2 describes the proposed McRNN algorithm. Section 3 presents performance evaluation of the proposed algorithm on benchmark function approximation problems. Section 4 presents the application of the proposed McRNN algorithm to estimate RUL from sensor data. Section 5 summarizes the conclusions from this study.

II. Meta-cognitive Regression Neural Network Algorithm

In the sequential learning algorithms, the training samples are presented to the network only once, and the network adapts the structure and parameters based on the knowledge difference between existing network and current training sample. Meta-cognitive Regression Neural Network (McRNN) consists of two components, viz., a cognitive component and meta-cognitive component. The cognitive component of McRNN is an evolving single hidden layered RBF network with Gaussian activation functions in the hidden layer, while the meta-cognitive component is a self-regulatory learning mechanism that controls the learning ability of the cognitive component. The learning mechanism is an adaptive sequential learning algorithm that starts with zero hidden neurons in the cognitive component and builds the network with sufficient number of hidden neurons based on the information contained in the training samples.

Given a stream of training data represented as input-output pair given by $\{(\mathbf{x}(1),\mathbf{y}(1)),\cdots,(\mathbf{x}(t),\mathbf{y}(t)),\cdots\}$ where $\mathbf{x}(t)=[x_1(t),\cdots,x_m(t)]\in\Re^m$ is the m-dimensional input vector of the t^{th} sample, and $\mathbf{y}(t)=[y_1(t),\cdots,y_n(t)]\in\Re^n$ is its corresponding n-dimensional output vector. For large range multi-input and multi-output dynamic system, the $\mathbf{x}(t)$ could be expressed as $\mathbf{x}(t) = [\mathbf{y}(t-1),\cdots,\mathbf{y}(t-\nu-1);\mathbf{u}(t-1),\cdots,\mathbf{u}(t-\chi-1)]$, where $\mathbf{u}$ is the input to the dynamic system, ν and χ are

the maximum lags of the ouput and input, respectively. The above relationship could be expressed as:

$$\mathbf{y}(t) = \mathbf{f}[\mathbf{x}(t)] = \mathbf{f}[\mathbf{y}(t-1), \cdots, \mathbf{y}(t-\nu-1); \mathbf{u}(t-1), \cdots, \mathbf{u}(t-\chi-1)] \quad (1)$$

where $\mathbf{f}[.]$ is the functional relationship that maps the input to their respective targets ($\mathbf{x} \rightarrow \mathbf{y}$). The aim of McRNN is to approximate $\mathbf{f}[.]$ such that for a given input $\mathbf{x}(t)$, the predicted output

$$\widehat{\mathbf{y}}(t) = \widehat{\mathbf{f}}[\mathbf{x}(t), \boldsymbol{\theta}] \quad (2)$$

is as close as possible to the desired target $\mathbf{y}(t)$. It must be noted that $\boldsymbol{\theta}$ represents the parameter vector of McRNN. For a given training training sample $\mathbf{x}(t)$, the predicted output $\widehat{\mathbf{y}}(t) = [\widehat{y}_1(t), \cdots, \widehat{y}_j(t), \cdots, \widehat{y}_n(t)]$ of McRNN with K hidden neurons is

$$\widehat{y}_j(t) = \sum_{k=1}^{K} w_{kj}\phi_k(\mathbf{x}(t)), \qquad j = 1, 2, \cdots, n \quad (3)$$

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

$$\phi_k\left(\mathbf{x}(t)\right) = exp\left(-\frac{\|\mathbf{x}(t) - \boldsymbol{\mu}_k\|^2}{\sigma_k^2}\right) \quad (4)$$

where $\boldsymbol{\mu}_k$ is the center and σ_k is the width of the k^{th} hidden neuron.

The error $\mathbf{e}(t) = [e_1(t), \cdots, e_n(t)]$ for the t^{th} sample is defined as the difference between the actual and predicted output

$$e_j(t) = y_j(t) - \widehat{y}_j(t) \quad j = 1, 2, \cdots, n \quad (5)$$

During the sequential learning, as each sample is presented to the network, the meta-cognitive component monitors the knowledge in the current sample (by employing error and spherical potential) with respect to the knowledge present in the network to decide whether to delete the sample without learning (*sample delete strategy*) or learn the knowledge in the sample (*sample learn strategy*) or reserve (*sample researve strategy*) the sample for future use.

In order to measure the knowledge difference between the current sample and the existing network, McRNN employs two measures: prediction error and spherical potential [28]. The prediction error $E(t)$ for the current sample at instant is given as

$$E(t) = \sqrt{\sum_{j=1}^{n} e_j^2(t)} \quad (6)$$

Spherical potential[28] is a direct measure of novelty in the current sample. It is defined as the average distance of the current sample from already added K hidden neurons in a hyperdimensional feature space

$$\psi(t) = \frac{1}{K}\sum_{k=1}^{K} \phi\left(\mathbf{x}(t), \boldsymbol{\mu}_k\right) \quad (7)$$

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

In the following section, we present the detailed description of the three strategies.

A. Sample Delete Strategy

The *what-to-learn* by meta-cognitive component in McRNN is achieved by the *sample delete strategy*. If the predicted error for current sample is less than the delete threshold β_d, then the knowledge content of the sample is similar to the knowledge present in the network. Hence, the sample is deleted from the training sequence without being used in learning. It prevents the network from over-training. Delete threshold is chosen based on the required absolute prediction error of the network. A lower value of this threshold (close to zero) will result in no samples being deleted, leading to over-fitting, whereas a higher value will lead to deletion of too many samples from the training sequence. But, the resultant network may not satisfy the required absolute prediction error. For the problems chosen in this work, it is chosen in the range [0.0001, 0.001].

B. Sample Learn Strategy

The *how-to-learn* by meta-cognitive component in McRNN is achieved by the *sample learn strategy*. A sample is learnt when it contains new knowledge, this strategy works by either adding a new hidden neuron (Neuron Growth Criterion) or updating the network parameters (Parameters Update Criterion).

1) Neuron Growth Criterion: When a sample contains significant novel knowledge, a new hidden neuron $(K+1)^{th}$ is added to the network. Novelty measured by the prediction error and spherical potential. If the prediction error of the network is very high and the spherical potential is below the novelty threshold. The neuron growth criterion is given as

$$E(t) \geq \beta_a \quad \textbf{AND} \quad \psi(t) \leq \beta_n \quad (8)$$

where β_a and β_n are the self-adaptive addition and novelty thresholds. A lower value of β_n indicates higher resistance to the neuron growth. In this study, β_a and β_n are initially set in the range [0.1, 0.3] and [0.3, 0.7] and when a new hidden neuron is added to the network, β_a is self-adapted according to:

$$\beta_a := \delta\beta_a + (1-\delta)E(t) \quad (9)$$

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. In McRNN, when the $(K+1)^{th}$ hidden neuron is added to the network the parameters of the new hidden neuron are initialized as

$$\boldsymbol{\mu}_{K+1} = \mathbf{x}(t) \quad (10)$$

$$\sigma_{K+1} = \kappa \min_{\forall k} \|\mathbf{x}(t) - \boldsymbol{\mu}_k\| \quad k = 1, 2, \cdots, K \quad (11)$$

$$\mathbf{w}_{K+1} = \mathbf{e}(t) \quad (12)$$

where κ is a positive constant which controls the overlap of the responses of the hidden units in the input space, in this study κ in the range [0.5, 1.0].

2) Parameters Update Criterion: The parameters of the cognitive component ($\boldsymbol{\theta} = [\mathbf{w}_1, \boldsymbol{\mu}_1, \sigma_1, \cdots, \mathbf{w}_K, \boldsymbol{\mu}_K, \sigma_K] \in \Re^{K(m+n+1)\times n}$) are updated, when the prediction error $E(t)$ is greater than self-adaptive update threshold β_u as given as

$$E(t) \geq \beta_u \quad (13)$$

In this study, β_u is initially set in the range [0.001, 0.01]. The β_u is adapted based on the prediction error as:

$$\beta_u := \delta\beta_u + (1-\delta)E(t) \quad (14)$$

The advantage of self-adaptive thresholds is that, they help in selecting the samples for adding as a hidden neuron or to update parameters.

McRNN uses Extended Kalman Filter (EKF) to update the cognitive component parameters

$$\boldsymbol{\theta} := \boldsymbol{\theta} + \mathbf{e}(t)\mathbf{G}^T \quad (15)$$

where $\mathbf{e}(t)$ is the error obtained as defined in Eq. (5) and $\mathbf{G} \in \Re^{z\times n}$ is the Kalman gain matrix given by:

$$\mathbf{G} = \mathbf{PB}\left[\mathbf{R} + \mathbf{B}^T\mathbf{PB}\right]^{-1} \quad (16)$$

where $z = K(m+n+1)$, $\mathbf{R} = r_0\mathbf{I}_{n\times n}$ 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 ($\boldsymbol{\theta}$) given by

$$\mathbf{B} = \begin{bmatrix} \phi_1\mathbf{I}_{n\times n}, \phi_1\frac{2\mathbf{w}_1}{\sigma_1^2}(\mathbf{x}(t)-\boldsymbol{\mu}_1)^T, \\ \phi_1\frac{2\mathbf{w}_1}{\sigma_1^3}\|\mathbf{x}(t)-\boldsymbol{\mu}_1\|^2, \\ \vdots \\ \phi_K\mathbf{I}_{n\times n}, \phi_K\frac{2\mathbf{w}_K}{\sigma_K^2}(\mathbf{x}(t)-\boldsymbol{\mu}_K)^T, \\ \phi_K\frac{2\mathbf{w}_K}{\sigma_K^3}\|\mathbf{x}(t)-\boldsymbol{\mu}_K\|^2 \end{bmatrix}^T \quad (17)$$

The error covariance matrix is updated by

$$\mathbf{P} := \left[\mathbf{I}_{z\times z} - \mathbf{GB}^T\right]\mathbf{P} + q_0\mathbf{I}_{z\times z} \quad (18)$$

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

$$\begin{bmatrix} \mathbf{P}_{z\times z} & \mathbf{0}_{z\times(m+n+1)} \\ \mathbf{0}_{(m+n+1)\times z} & p_0\mathbf{I}_{(m+n+1)\times(m+n+1)} \end{bmatrix} \quad (19)$$

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

C. Sample Reserve Strategy

The *when-to-learn* by meta-cognitive component in McRNN is achieved by the *sample reserve strategy*. If the current sample does not satisfy sample delete or sample learn strategy, it is reserved to be considered for learning at a later stage of the learning process. This is achieved by pushing the current sample to the rear end of data stream. These samples may be used at a later stage in learning process by the virtue of self-regulatory nature of McRNN.

These three strategies are repeated for every sample from data stream. It helps the McRNN learn and generalize the knowledge efficiently. The training process stops when no further sample is available in the data stream or number of samples in the reserve remains same.

III. Performance Evaluation of McRNN Algorithm on Benchmark Problems

In this section, the effectiveness of the proposed McRNN is evaluated on a set of benchmark function approximation problems, namely, the non-linear system identification problem I [22], the non-linear system identification problem II [23], two variants of the Mackey-Glass chaotic time-series prediction problem [24], and the Box-Jenkins gas furnace problem [25]. The performance of McRNN algorithm is compared with existing sequential learning algorithms MRAN, GAP-RBFN and QKLMS algorithms. The tunable parameters including the number of hidden neurons of all the four sequential algorithms, McRNN, MRAN, GAP-RBFN and QKLMS sequential algorithms are chosen using standard 10-fold cross-validation procedure based on the *training set* only, where we tune their parameter values for training these models on the randomly selected nine-folds and choose their final values that give best results in the last-fold. First, we describe the considered benchmark problems first and then present the performance comparison results.

A. Non-linear System Identification Problem I

The non-linear system identification problem is a two-input one-input problem and is given as

$$y(t+1) = \frac{y(t)}{1+y^2(t)} + u^3(t) \quad (20)$$

The objective is to predict the one step ahead output $y(t+1)$ of the system, based on the current output $y(t)$ and a sinusoidal input $u(t) = sin(\frac{2\pi t}{100})$. As suggested in [22], 50200 data samples are generated in the range [-1.5, 1.5], out of which 50000 data samples are used as training and remaining 200 as testing.

B. Non-linear System Identification Problem II

The input and output relation of this second non-linear system identification problem is given as

$$y(t) = \frac{y(t-1)y(t-2)(y(t-1)-0.5)}{1+y^2(t-1)+y^2(t-2)} + u(t-1) \quad (21)$$

The objective is to predict the current output $y(t)$ of the system, based on the input at $(t-1)^{th}$ instant and output at

$(t-1)^{th}$,$(t-2)^{th}$ instants. Where the input to the system is given by $u(t) = sin(\frac{2\pi t}{25})$. 5200 data samples are generated, out of which 5000 data samples are used as training and remaining 200 as testing.

C. Mackey-Glass Time-Series Prediction Problem

The Mackey-Glass chaotic time-series data [24] are generated using the differential equation is given by

$$\frac{\partial x}{\partial t} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{22}$$

The objective is to predict the future values of the system based on past and current values. A fourth-order Runge-Kutta method is used to approximate the numerical solution of the differential equation. Two different variants of this problem is considered in this study: 1) Mackey-Glass time-series problem with six steps ahead prediction (Mackey-Glass-6); and 2) Mackey-Glass time series problem with 85 steps ahead prediction (Mackey-Glass-85).

1) Mackey-Glass-6 Problem: The objective of this problem is to predict the value of $x(t+6)$ based on past four values:$[x(t-18), x(t-12), x(t-6), x(t)]$. In order to extract the data, the following initial conditions are employed: $x(0) = 1.2$ and $\tau = 30$. 1000 data samples are generated in the interval $t = 124$ to $t = 1123$, out of which 500 data samples are used for training and remaining 500 for testing.

2) Mackey-Glass-85 Problem: The objective of this problem is to predict the value of $x(t+85)$ based on inputs $[x(t-18), x(t-12), x(t-6), x(t)]$. In this variant, the parameters are set as $\tau = 17$ and $x(0) = 1.2$. 3000 data samples are generated in the interval $t = 201$ to $t = 3200$ for training and 500 data samples in the interval $t = 5001$ to $t = 5500$ for testing.

D. Box-Jenkins Gas Furnace Problem

The objective of this Box-Jenkins gas furnace problem [25] is to predict the output CO_2 concentration from input gas flow rate in furnace. The system is modeled using a series-parallel model given by $\widehat{y}(t) = f(y(t-1), u(t-4))$. This data set consists of 290 data samples, out of which 200 data samples are used for training and remaining 90 for testing.

E. Performance Measures

In this study, the inputs and targets are normalized in the range [-1, 1] for all the aforementioned problems and for all the algorithms considered. In this study, two performance measures are employed for performance comparison of the proposed McRNN algorithm with other considered sequential learning algorithms. The two performance measures are Root Mean Square Error (RMSE) and the percentage of samples (PS) used.

- RMSE: It is defined as

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N}\sum_{j=1}^{n}\left(y(t)-\widehat{y}(t)\right)^2}{N}} \tag{23}$$

where N is the total number of samples.

- Percentage of samples employed in training (PS): It is given by

$$PS = \frac{\textit{\# samples employed in training}}{\textit{\# samples available in training data set}} \times 100\% \tag{24}$$

F. Preliminary Results

TABLE I
PERFORMANCE COMPARISON OF THE PROPOSED MCRNN WITH EXISTING MRAN, GAP-RBFN AND QKLMS ALGORITHMS ON BENCHMARK APPROXIMATION PROBLEMS

Data set	Algorithm	No. of Neurons	PS %	RMSE	
				Train	Test
Non-linear	MRAN	7	100	0.00247	0.00275
System	GAP-RBFN	9	100	0.00230	0.00255
Identification	QKLMS	41	100	0.05699	0.05551
Problem I	McRNN	9	10.79	**0.00159**	**0.00170**
Non-linear	MRAN	3	100	0.00695	0.00819
System	GAP-RBFN	8	100	0.00278	0.00359
Identification	QKLMS	26	100	0.01136	0.01138
Problem II	McRNN	5	82.22	**0.00135**	**0.00222**
Mackey-Glass-6	MRAN	5	100	0.02419	0.01498
Time-Series	GAP-RBFN	10	100	0.02008	0.01348
Prediction	QKLMS	50	100	0.02588	0.01769
Problem	McRNN	7	76.00	**0.01649**	**0.01198**
Mackey-Glass-85	MRAN	8	100	0.00669	0.00694
Time-Series	GAP-RBFN	10	100	0.00697	0.00727
Prediction	QKLMS	79	100	0.01043	0.01079
Problem	McRNN	8	76.77	**0.00530**	**0.00538**
Box-Jenkins	MRAN	3	100	0.08703	0.05610
Gas	GAP-RBFN	3	100	0.03687	0.05082
Furnace	QKLMS	26	100	0.05996	0.06358
Problem	McRNN	3	71.00	**0.02527**	**0.03669**

The performance comparison of the proposed McRNN algorithm with the existing MRAN, GAP-RBFN and QKLMS algorithms on benchmark approximation problems is given in Table I. Table I gives the number of hidden neurons, percentage of samples used by each of the algorithm, and training and testing RMSE values. From the Table I, it can be seen that on all the considered problems McRNN chooses less number of data samples for training and attains a network which produces significantly lower training and testing errors. In non-linear system identification problem I, which has more number of data samples compared to other problems, McRNN used only 10.79% (5399 samples employed in training out of 50000 samples available in training data set) and still achieves lowest training and testing errors compared with MRAN, GAP-RBFN and QKLMS sequential learning algorithms. Among the four algorithms, on all the problems QKLMS achieves higher testing error and also employs more number of hidden neurons in the network. GAP-RBFN algorithm achieves better performance than MRAN algorithm on the all the problems except Box-Jenkins gas furnace problem. It can also be seen that on all the considered

problems McRNN required less or equal number of neurons compared to the GAP-RBFN and QKLMS algorithms to achieve significantly better performance. When compared to MRAN algorithm, on non-linear system identification problems I & II and Mackey-Glass-6 problem, even though McRNN requires two more number of neurons in the network to approximate the complex input-output functional relationship and achieves significantly better performance than MRAN. When compared to MRAN algorithm, on Mackey-Glass-85 and Box-Jenkins gas furnace problems, McRNN uses exactly same number of neurons as of MRAN and achieves significantly better performance than MRAN. From the above comparison results based on the percentage of samples (PS) employed in training, and training and testing RMSE values, we can see that the use of meta-cognitive learning strategies has helped proposed McRNN sequential learning algorithm and attains better performance than the other well known sequential learning algorithms.

IV. Application of McRNN Algorithm to Remaining Useful Life Estimation

In this section, we present the application of proposed McRNN algorithm on RUL estimation problem based on sensor data and its performance is compared with the existing approaches based on state-of-the-art regression algorithms. Clearly, accurate estimation of RUL has great benefits and advantages in many real-world applications across different industrial verticals. RUL estimation problem is the most common task in the research field of prognostics and health management. In its simplest form RUL estimation problem in prognostics is similar to a regression problem. In this study, we have used PHM 2008 Data Challenge data set [26]. This data set contain simulated data produced using a model based simulation program C-MAPSS developed by NASA [27]. This data set can be downloaded from NASA prognostics data repository web-site (http://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/).

PHM 2008 Data Challenge data set consists of training and testing data sets which are arranged in an l-by-26 matrix where l corresponds to the number of data points in each component. Each row is a snapshot of data taken during a single operating time cycle and in 26 columns, where 1^{st} column represents the engine number, 2^{nd} column represents the operational cycle number, 3 - 5 columns represent the three operating settings, and 6 - 26 columns represent the 21 sensor values. Training data set consists of 218 engines data and testing data set consists of another 218 engines data. More information about the 21 sensors can be found in [29]. Engine performance can be effected by three operating settings in the data significantly. Each trajectory within the train and test trajectories is assumed to be life-cycle of an engine. While each engine is simulated with different initial conditions, these conditions are considered to be of normal conditions (no faults). For each engine trajectory within the training data set, the last data entry corresponds to the moment the engine is declared unhealthy or failure status. On the other hand, test data set contains data some time before the failure and aim here is to predict RUL in the test data set for each engine (Total 218 engines in test data set, therefore prediction result of an algorithm on this test data set must be a vector of length 218). The actual RUL value of the test trajectories in PHM 2008 Data Challenge data set is not available.

We performed following data pre-processing steps before learning the model: Plotting the three operating setting values in this data set, the data points are clustered into six different distinct clusters. These clusters are assumed to correspond to the six different operating conditions. It is therefore possible to include the operating condition history as a feature. Hence, operating conditions history is included as 6 extra features [2]. Next, standard score normalization is applied to normalize the data points to be within uniform scale range [2]. Finally, piece-wise linear degradation model (RUL target function) is used which limits the maximum value of the RUL function [1]. In this study, we set this maximum value to 130.

The *penalty score* function used in this study to compare the proposed McRNN with existing algorithms is identical to that used in PHM 2008 Data Challenge. This penalty score function is illustrated in Eq. (25), where M is the number of engines in test set, S is the computed *score*, and $h = (Estimated\ RUL - True\ RUL)$.

$$S = \begin{cases} \sum_{i=1}^{M} \left(e^{-\frac{h_i}{13}} - 1\right) & for \quad h_i < 0 \\ \sum_{i=1}^{M} \left(e^{\frac{h_i}{10}} - 1\right) & for \quad h_i \geq 0 \end{cases} \tag{25}$$

This penalty score function penalizes late predictions (too late to perform maintenance) more than early predictions (no big harms although it could waste maintenance resources). This is in line with the risk adverse attitude in aerospace industries.

TABLE II
McRNN algorithm performance on RUL problem using PHM 2008 Data Challenge data set

Algorithm	Penalty Score
Multi-Layer Perceptron [4]	118338
Support Vector Regression	15886
Relevance Vector Regression	8242
Kalman Filter Ensemble [4]	5590
Gibbs Filtering [30]	4170
Switching Kalman Filter [4]	2922
McRNN	**2255**

After running the McRNN algorithm to compute the estimated RUL's of 218 engines in the test data set, their RUL's were then uploaded to the NASA data repository web-site and a single penalty score was then calculated by the web-site as the final output. This penalty score of McRNN and the existing approaches penalty scores in the literature is given in Table II. Based on the penalty score results in Table II, we observe that the proposed McRNN

based approach outperforms the other existing approaches for RUL estimation significantly by producing much lower *penalty score*, indicating that the predicted failure time from our proposed McRNN approach is very near to the actual failure time or their ground truth values. Also, McRNN based approach outperforms recently proposed Gibbs filtering [30] based approach and switching Kalman filter neural network ensemble [4] based approach. Hence, we can conclude that McRNN based RUL estimation approach is better than the approaches based on state-of-the-art regression algorithms.

V. Conclusions

This paper has presented a novel approach for RUL estimation problem based on sensor data. For this, we have proposed sequential learning Meta-cognitive Regression Neural Network (McRNN) algorithm for function approximation problems inspired from human meta-cognitive learning principles. McRNN has two components, namely, a cognitive component and a meta-cognitive component. The cognitive component of McRNN is an evolving single hidden layered RBF network with a Gaussian activation functions. The meta-cognitive component of McRNN has a self-regulatory learning system that decides *what-to-learn*, *when-to-learn* and *how-to-learn*. These three actions are realized by sample delete strategy, sample reserve strategy, and sample learn strategy, respectively. First, the performance of the proposed sequential learning McRNN algorithm has been evaluated on set of benchmark function approximation problems. The performance comparison with the well-known sequential learning algorithms in the literature clearly indicates the better performance of the proposed McRNN on function approximation problems. Next, the proposed sequential learning McRNN algorithm has been applied to RUL estimation problem based on sensor data. The performance comparison with the literature results on RUL estimation problem based on PHM 2008 Data Challenge data set indicates that the predicted failure time from our proposed McRNN approach is very near to the actual failure time or their ground truth values.

Acknowledgment

The last author would like to acknowledge Ministry of Education Singapore Tier 1 Fund (RGS 24/14) for their financial support.

References

[1] F. O. Heimes, "Recurrent neural networks for remaining useful life estimation," in *International Conference on Prognostics and Health Management, 2008. PHM 2008.*, Oct 2008, pp. 1–6.

[2] L. Peel, "Data driven prognostics using a kalman filter ensemble of neural network models," in *International Conference on Prognostics and Health Management, 2008. PHM 2008.*, Oct 2008, pp. 1–6.

[3] T. Wang, J. Yu, D. Siegel, and J. Lee, "A similarity-based prognostics approach for remaining useful life estimation of engineered systems," in *International Conference on Prognostics and Health Management, 2008. PHM 2008.*, Oct 2008, pp. 1–6.

[4] P. Lim, C. K. Goh, K. C. Tan, and P. Dutta, "Estimation of remaining useful life based on switching kalman filter neural network ensemble," in *Annual conference of the Prognostics and Health Management Society*, 2014, pp. 1–8.

[5] E. Ramasso and A. Saxena, "Review and analysis of algorithmic approaches developed for prognostics on CMAPSS dataset," in *Annual Conference of the Prognostics and Health Management Society 2014*, 2014, pp. 1–11.

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

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

[8] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks." *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[9] S. Wysoski, L. Benuskova, and N. Kasabov, "On-Line Learning with Structural Adaptation in a Network of Spiking Neurons for Visual Pattern Recognition," in *Artificial Neural Networks - ICANN 2006*, ser. Lecture Notes in Computer Science, S. Kollias, A. Stafylopatis, W. Duch, and E. Oja, Eds. Springer Berlin Heidelberg, 2006, vol. 4131, pp. 61–70.

[10] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, "Quantized Kernel Least Mean Square Algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 1, pp. 22–32, 2012.

[11] M. Martin, *On-Line Support Vector Machine Regression*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, ch. Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings, pp. 282–294.

[12] J. C. Platt, "A resource allocation network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, 1991.

[13] L. Yan, N. Sundararajan, and P. Saratchandran, "Analysis of minimal radial basis function network algorithm for real-time identification of nonlinear dynamic systems," *IEE Proceedings - Control Theory and Applications*, vol. 147, no. 4, pp. 476–484, 2000.

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

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

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

[17] R. Savitha, S. Suresh, and N. Sundararajan, "Projection-based fast learning fully complex-valued relaxation neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 4, pp. 529–541, 2013.

[18] G. Sateesh Babu and S. Suresh, "Meta-cognitive RBF Network and Its Projection Based Learning algorithm for classification problems," *Applied Soft Computing*, vol. 13, no. 1, pp. 654–666, 2013.

[19] ——, "Sequential Projection-Based Metacognitive Learning in a Radial Basis Function Network for Classification Problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 2, pp. 194–206, 2013.

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

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

[22] C.-F. Juang and C.-T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 1, pp. 12–32, 1998.

[23] H.-J. Rong, N. Sundararajan, G.-B. Huang, and P. Saratchandran, "Sequential adaptive fuzzy inference system (safis) for nonlinear system identification and prediction," *Fuzzy sets and systems*, vol. 157, no. 9, pp. 1260–1275, 2006.

[24] R. S. Crowder, "Predicting the mackey-glass time series with cascade-correlation learning," in *Connectionist Models: Proceedings of the 1990 Summer School*. San Mateo, CA, 1990, pp. 117–123.

[25] G. E. Box and G. M. Jenkins, "Time series analysis forecasting and control-rev," 1976.

[26] A. Saxena and K. Goebel, "PHM08 challenge data set, NASA AMES prognostics data repository," Moffett Field, CA, Tech. Rep., 2008.

[27] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *International Conference on Prognostics and Health Management, 2008. PHM 2008.*, Oct 2008, pp. 1–9.

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

[29] P. Wang, B. D. Youn, and C. Hu, "A generic probabilistic framework for structural health prognostics and uncertainty management," *Mechanical Systems and Signal Processing*, vol. 28, pp. 622–637, 2012.

[30] K. L. Son, M. Fouladirad, and A. Barros, "Remaining useful life estimation on the non-homogenous gamma with noise deterioration based on gibbs filtering: A case study," in *IEEE conference on Prognostics and Health Management*, 2012, pp. 1–6.