Fast Adaptive Gradient RBF Networks For Online Learning of Nonstationary Time Series

Tong Liu, Sheng Chen, *Fellow, IEEE*, Shan Liang, *Member, IEEE*, Shaojun Gan, and Chris J. Harris

***Abstract*—For a learning model to be effective in online modeling of nonstationary data, it must not only be equipped with high adaptability to track the changing data dynamics but also maintain low complexity to meet online computational restrictions. Based on these two important principles, in this paper, we propose a fast adaptive gradient radial basis function (GRBF) network for nonlinear and nonstationary time series prediction. Specifically, an initial compact GRBF model is constructed on the training data using the orthogonal least squares algorithm, which is capable of modeling variations of local mean and trend in the signal well. During the online operation, when the current model does not perform well, the worst performing GRBF node is replaced by a new node, whose structure is optimized to fit the current data. Owing to the local one-step predictor property of GRBF node, this adaptive node replacement can be done very efficiently. Experiments involving two chaotic time series and two real-world signals are used to demonstrate the superior online prediction performance of the proposed fast adaptive GRBF algorithm over a range of benchmark schemes, in terms of prediction accuracy and real-time computational complexity.**

***Index Terms*—Nonlinear and nonstationary signals, prediction, radial basis function (RBF) network, gradient RBF network, adaptive algorithm, tunable nodes.**

I. Introduction

MOST real-world time series are nonlinear to a large extent, and radial basis function (RBF) neural networks, as effective means of modeling nonlinear characteristics from data, have enjoyed considerable success in time series prediction [1]–[5]. One important advantage of RBF networks compared with other neural network models is that the orthogonal least squares (OLS) algorithm [4]–[9] can readily be applied to construct a parsimonious RBF model that generalizes well in prediction for stationary data. Like many other neural network models, the RBF model constructed on training data does not characterize temporal variability of unseen data well [10]. Therefore, using the fixed RBF model constructed on training data to predict nonstationary signals generally yields unsatisfactory performance.

As an alternative to the feedforward RBF neural network for nonlinear dynamic modeling, the recurrent neural networks (RNNs) have received considerable attention due to their better capabilities in capturing nonlinear dynamic characteristics in data [11]–[13]. Different from feedforward RBF networks, the nodes in RNNs are connected in a loop, and the internal state of the network exhibit dynamic temporal behavior, which makes RNNs particularly suitable for modeling nonlinear time series data with long memory or embedding dimension [14], [15]. Unlike the training of a RBF network, which offers a single global optimal solution, training of a RNN is much more challenging. In particular, traditional RNNs suffer from the problem of vanishing gradients and thus have difficulty to capture long-term dependencies [16]. Recently, long short-term memory (LSTM) network [17] and the gated recurrent unit (GRU) [18], [19] have been developed to combating this limitation. By adding multi-threshold gates, The LSTM and GRU are more effective in learning long-term temporal dependencies, and they have been successfully applied to many sequence modeling problems [20]–[23]. However, the underlying structure of RNNs deters the use of RNNs in many online scenarios, where data are fast arriving and the underlying dynamics of data are fast time-varying. This is because the structure and parameters of an RNN are fixed during online operation, since it is impossible to optimize its structure and parameters within a small sampling period in order to track the fast time-varying nonstationary characteristics.

Since most real-world signals are not only nonlinear but also nonstationary, Online learning in nonstationary environment has been a very hot topic in the machine learning community [24]–[31]. Predicting nonstationary time series imposes a set of challenges, including model adaptability for evolving environment and stringent real-time computational constraint. Specifically, a learning model must not only be equipped with effective adaptation mechanism for tracking changing data dynamics but also be sufficiently efficient to meet online complexity

Manuscript received August 15, 2019; revised February 4, 2020; accepted March 11, 2020. Date of publication March 18, 2020; date of current version April 10, 2020. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mark A. Davenport. This work was supported in part by the National Natural Science Foundation of China under Grant 61771077, and in part by the Key Research Program of Chongqing Science & Technology Commission under Grant CSTC2017jcyjBX0025. The work of T. Liu was supported by the Chinese Scholarship Council and School of Electronics and Computer Science, University of Southampton, UK. *(Corresponding author: Sheng Chen.)*

Tong Liu and Shan Liang are with the Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, Chongqing University, School of Automation, Chongqing University, Chongqing 400044, China (e-mail: tl3n18@soton.ac.uk; lightsun@cqu.edu.cn).

Chris J. Harris is with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: cjh@ecs.soton.ac.uk).

Sheng Chen is with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K., and also with the King Abdulaziz University, Jeddah 21589, Saudi Arabia (e-mail: sqc@ecs.soton.ac.uk).

Shaojun Gan is with the Beijing Key Laboratory of Traffic Engineering, College of Metropolitan Transportation, Beijing University of Technology, Beijing 100124, China (e-mail: s.gan@bjut.edu.cn).

Digital Object Identifier 10.1109/TSP.2020.2981197

restrictions. To cope with nonstationary data, a commonly used simple method is using adaptive recursive estimators, such as the recursive least square (RLS) [1], [32] or the online sequential extreme learning machine (OS-ELM) [33]–[36], to update the RBF model's weights in real time.

The OS-ELM of [33]–[36] is a popular online learning method due to its 'simplicity' in model construction. By randomly selecting a large number of training data as the RBF centers to fix the RBF model structure, only the model weights are adapted online using the RLS algorithm. Because the size of the RBF model has to be very large for OS-ELM, online adaptation of the model weights is computationally costly and, moreover, there is no guarantee that the fixed RBF nodes, no matter how dense they are in the training data space, will also cover the changing nonstationary data space well. Therefore, the OS-ELM only works for relatively slow time varying data with priori known data space. It can be seen that adapting the RBF model structure to track time-varying characteristics, not just adapting the RBF model weights, is the key to success for nonstationary data modeling. However, for fast arriving data with short sampling period, optimizing the whole RBF model structure online may not meet the real-time constraint.

Starting with a compact RBF model constructed for example in training by the OLS algorithm, the fast tunable RBF method [37] adjusts RBF nodes as well as the model weights online to adaptively model nonstationary signals. Specifically, the fast tunable RBF adapts the RBF model structure by replacing an 'insignificant' RBF node with a new node to better fit the current data. Equipped with this fast adaptive mechanism together with a small fixed model size, the fast tunable RBF learning is capable of tracking changing characteristics in nonstationary signals, while imposing a low online computational complexity. The experimental results of [37] show that this fast tunable RBF method outperforms the OS-ELM considerably, in terms of both prediction accuracy and real-time computational complexity.

It is well known that for nonstationary time series involving variations of local mean and trend, the series can be made stationary by applying an appropriate difference operation on the original signal [38]. Inspired by this property, the gradient RBF (GRBF) neural network is proposed for nonstationary signal prediction [39]. In the GRBF model, the input signal to the network is the difference of the original signal and each hidden node, which consists of a center and a width as well as a scalar, can be interpreted as a local one-step predictor [39]. The OLS algorithm can readily be applied to determine an appropriate model size and, therefore, to construct a compact GRBF model from the training data. Not surprisingly, this GRBF network trained by the OLS algorithm outperforms the classic RBF network also trained by the OLS algorithm in nonstationary time series prediction [39].

Obviously, nonstationary time series or signals also exhibit other time-varying characteristics, not just variations of local mean and trend. In a nonstationary environment, the signal dynamics can change significantly from the initial training data space and some new time-varying characteristics, other than variations of local mean and trend, may appear, which are unseen in the training data. The fixed GRBF model constructed on the training data is unlikely to track these unseen dynamics well and, consequently, its online prediction performance may degrade. A solution to this problem is to update the GRBF model structure online. This motivates our current work.

In this paper, we propose a fast adaptive or tunable GRBF network for online prediction of nonlinear and nonstationary time series. To be specific, starting from the initial GRBF network constructed from the training data using for example the OLS algorithm, the worst performing node during online operation is replaced with a new node when the current model does not fit the current data well. This enables the GRBF network to track the changing dynamics online with high adaptability. It turns out that online optimization of a GRBF node is far easier and simpler than online updating a RBF node as given in [37]. This is because owing to local one-step predictor interpretation of GRBF node, the center and scalar of the replacement GRBF node can simply be chosen to be the current input data and the output gradient, respectively, which is actually optimal, while the new width can be computed easily from the set of centers, and the new network weight vector is calculated using the least squares (LS) estimator. By comparison, the center and width of the replacement RBF node also need to be determined via an iterative optimization procedure for the fast tunable RBF [37]. Consequently, our proposed fast tunable GRBF method imposes significantly lower online computational complexity than the fast tunable RBF method of [37]. Experiments involving prediction of two chaotic time series and two real-world signals demonstrate the superior performance of the proposed fast tunable GRBF algorithm over a range of benchmark schemes, including the fast tunable RBF of [37], in terms of both prediction accuracy and online computational complexity.

II. The GRBF Network

Without loss of generality, consider the one-step-ahead time series prediction, which uses the past signal samples

$$\boldsymbol{x}_t = [y_{t-1}\ y_{t-2} \cdots y_{t-M'}]^{\mathrm{T}}, \tag{1}$$

to predict the current signal value y_t, where M' is referred to as the embedding vector length. The task of the online prediction therefore can be formulated as follows: given the observation data $\{\boldsymbol{x}_t, y_t\}$, construct an estimator $\widehat{y}_t = F_t(\boldsymbol{x}_t)$ to approximate the underlying dynamics at every sampling time t, where $F_t(\cdot)$ denotes the estimator mapping. All the our discussions equally apply to multi-step-ahead prediction.

A. GRBF Neural Network

The GRBF network [39], like the conventional RBF network, is a single-hidden-layer feedforward neural network. However, unlike the RBF network, where the network input is given by (1), the input vector to the GRBF network is generated by differencing the raw data. More specifically, given the M' past samples, the input vector of the first-order GRBF at time t is given by

$$\boldsymbol{x}_t = [y_{t-1} - y_{t-2}\ y_{t-2} - y_{t-3} \cdots y_{t-M} - y_{t-M-1}]^{\mathrm{T}}, \tag{2}$$

where $M = M' - 1$. The elements of $\boldsymbol{x}_t$ in (2) show the rate of change in the trajectory of the time series for the past M'

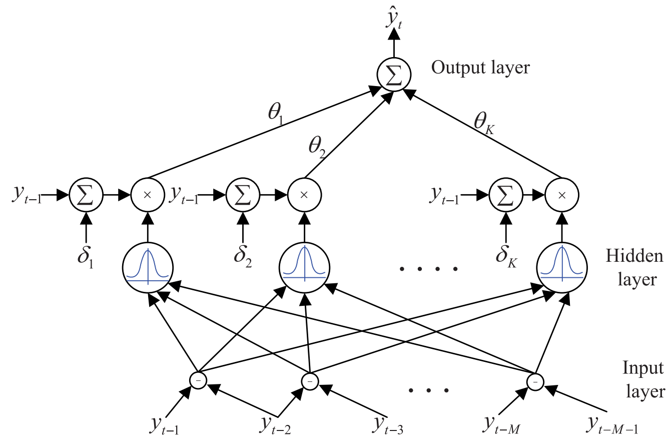


Fig. 1. Structure of the GRBF network [39].

samples. Differencing helps to eliminate or reduce trend and seasonality [38].

Fig. 1 depicts the structure of the GRBF network. Without loss of generality, we use the Gaussian function to serve as the hidden node's nonlinearity which compares the similarity of the input vector to the hidden node's center. In addition to differencing, the main difference between a GRBF node and a classic RBF node is that the Gaussian function response in a GRBF hidden node is further multiplied by an additional term $(y_{t-1}+\delta)$. Hence, the response of the jth hidden node to the input vector $\boldsymbol{x}_t$ is given by [39]

$$\phi_j(\boldsymbol{x}_t) = \exp\left(-\alpha \left\|\boldsymbol{x}_t - \boldsymbol{c}_j\right\|^2\right) \times (y_{t-1}+\delta_j), \quad (3)$$

where $\boldsymbol{c}_j$ is an M-dimensional center vector, α is a positive constant which determines the width of the symmetric response of the hidden node, and δ_j is a constant scalar associated with the jth hidden node. Because the network input is (2), the hidden nodes now sense the gradient of the time series rather than the series itself as in the case of the classic RBF model. The term $y_{t-1}+\delta_j$ also has a clear geometrical interpretation as a local one-step prediction of y_t by the jth hidden node. Observe from (3) that if $\boldsymbol{x}_t$ is very similar to $\boldsymbol{c}_j$, the response of the jth Gaussian hidden node will be very close to the maximum value of 1 and the local predictor $y_{t-1}+\delta_j$ becomes fully active.

The output layer of the GRBF network is a linear combiner with weights θ_j, just as in a classical RBF network. Thus, assuming a total of K hidden nodes and given the input vector $\boldsymbol{x}_t$ of (2), the relationship between the output of the GRBF network $\widehat{y}_t$ and the actual output y_t can be expressed as

$$y_t = \widehat{y}_t + e_t = \sum_{j=1}^{K}\phi_j(\boldsymbol{x}_t)\theta_j + e_t, \quad (4)$$

where e_t denotes the modeling error.

B. GRBF Neural Network Construction

The centers $\boldsymbol{c}_j$ and the scalars δ_j can be chosen during the training from the training data $\{\boldsymbol{x}_k, y_k\}_{k=1}^N$. First, for each training input vector $\boldsymbol{x}_k$, define

$$d_k = y_k - y_{k-1}. \quad (5)$$

If $\boldsymbol{x}_k$ is selected as the jth center $\boldsymbol{c}_j$, we set $\delta_j = d_k$ to ensure that the jth hidden node is a perfect predictor of y_k. Next assume that the width parameter α is chosen. Then the problem of constructing the GRBF network is equivalent to the task of selecting a K-term subset model $\{\boldsymbol{c}_j, \delta_j\}_{j=1}^K$ from the full N-term model $\{\boldsymbol{x}_k, d_k\}_{k=1}^N$, and the OLS method can readily be applied to complete this subset selection problem.

Specifically, according to (4), the full N-term GRBF model over the training data set can be expressed as

$$\boldsymbol{y} = \boldsymbol{\Phi\theta} + \boldsymbol{e}, \quad (6)$$

where $\boldsymbol{y} = [y_1\ y_2 \cdots y_N]^{\mathrm{T}} \in \mathbb{R}^N$ is the desired output vector, $\boldsymbol{\Phi} \in \mathbb{R}^{N\times N}$ is the full regression matrix whose (k,i)th entry is $\boldsymbol{\Phi}_{k,i} = \phi_i(\boldsymbol{x}_k)$, and $\boldsymbol{\theta} = [\theta_1\ \theta_2 \cdots \theta_N]^{\mathrm{T}} \in \mathbb{R}^N$ is the full model weight vector, while $\boldsymbol{e} = [e_1\ e_2 \cdots e_N]^{\mathrm{T}} \in \mathbb{R}^N$ represents the error vector over the training samples.

Let the orthogonal decomposition of the regression matrix be $\boldsymbol{\Phi} = \boldsymbol{WA}$, where $\boldsymbol{W} = [\boldsymbol{w}_1\ \boldsymbol{w}_2 \cdots \boldsymbol{w}_N] \in \mathbb{R}^{N\times N}$ denotes the orthogonal regression matrix that satisfies $\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{w}_j = 0$ for $i \neq j$, and $\boldsymbol{A}$ is an unit upper triangular matrix given by

$$\boldsymbol{A} = \begin{bmatrix} 1 & a_{1,2} & \cdots & a_{1,N} \\ 0 & 1 & \cdots & a_{2,N} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix}. \quad (7)$$

The regression model (6) can then be rewritten as

$$\boldsymbol{y} = \boldsymbol{Wg} + \boldsymbol{e}, \quad (8)$$

where the transformed weight vector $\boldsymbol{g} = [g_1\ g_2 \cdots g_N]^{\mathrm{T}} = \boldsymbol{A\theta}$, whose elements are given by $g_i = \boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{y}/\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{w}_i$ for $1 \leq i \leq N$. The sum of squares of the output $\boldsymbol{y}$ is therefore given by

$$\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} = \sum_{j=1}^{N} g_j^2 \boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{w}_j + \boldsymbol{e}^{\mathrm{T}}\boldsymbol{e}. \quad (9)$$

The contribution of the jth model term or the error reduction radio $[\mathrm{err}]_j$ is defined by

$$[\mathrm{err}]_j = \frac{\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{w}_j g_j^2}{\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}}. \quad (10)$$

The orthogonal forward selection (OFS) procedure [6] can readily be utilized to select a subset of K centers $\boldsymbol{c}_i$ and scalars δ_i one by one from the full model. At each step, a candidate with the maximum value of $[\mathrm{err}]_i$ is chosen. The selection procedure is terminated when some termination criterion is met, yielding a K-term subset model $\boldsymbol{W}_K\boldsymbol{g}_K$ which contains the selected $\{\boldsymbol{c}_i, \delta_i\}_{i=1}^K$, where $\boldsymbol{W}_K \in \mathbb{R}^{N\times K}$ is the selected subset orthogonal regression matrix and $\boldsymbol{g}_K \in \mathbb{R}^K$ is the associated weight vector. Let the corresponding upper triangular matrix be $\boldsymbol{A}_K \in \mathbb{R}^{K\times K}$. Then the weight vector of the selected K-term subset GRBF model $\boldsymbol{\theta}_K \in \mathbb{R}^K$ can readily be solved from $\boldsymbol{g}_K = \boldsymbol{A}_K\boldsymbol{\theta}_K$ by the backward substitution. Termination of the OFS procedure can base on Akaike's information criterion [40], regularization techniques [5], D-optimality experimental design [7] or leave-one-out mean square error [8].

C. Determining Width of Gaussian Node

A variety of ways exist to determine the width α of Gaussian hidden nodes. A simple way is to set the width to [41]

$$\alpha = \frac{1}{2d_{\max}^2}, \tag{11}$$

where $d_{\max}$ is the maximum distance between the centers. The number of centers K can also be taken into account by modifying (11) as [42]

$$\alpha = \frac{K}{d_{\max}^2}. \tag{12}$$

Furthermore, the dimension of the input M can also be considered by setting the width to [43]

$$\alpha = \frac{M\ \sqrt[2/M]{K}}{2d_{\max}^2}. \tag{13}$$

Unsupervised clustering [1] can also be applied to partition the input data into an appropriate number of clusters which also yields the cluster variance suitable as the width of hidden nodes. Individual hidden nodes can also have tunable widths, which can be optimized using gradient descend or evolutionary algorithms, and moreover this width optimization can naturally be embedded with the OFS procedure [44]–[50] to construct the GRBF network having individually optimized hidden nodes. Since we will consider online adaptation of the GRBF model, we only adopt the simple OLS algorithm of Subsection II-B to construct the initial GRBF model from the training data with a common width α.

III. Online Adaptation of the GRBF Network

At the beginning of online operation, we have the initial GRBF model with K hidden nodes constructed based on the training data. During online operation, the signal dynamics can vary dramatically from those seen in the training data. A simple way of adapting the GRBF model online is to apply the RLS algorithm to update the weights. However, this is clear inadequate to track highly time-varying signals. In order to capture the newly emerged signal's dynamics, it is necessary to adapt the GRBF network structure as well. To maintain low online computational complexity, we do not attempt to grow the GRBF model by adding new hidden nodes to capture the newly emerging signal's dynamics. Rather, we opt for maintaining the model size, similar to the fast tunable RBF of [37], and modify the hidden nodes of the GRBF network, namely, update the centers and associated scalars as well as individual widths of the hidden nodes, to track the signal's changing dynamics.

Therefore, for our fast adaptive GRBF, when the weight adaptation becomes inadequate at sample t, modification of the hidden nodes takes place. It is worth pointing out that it is generally not necessary, in fact, it is undesired to update all the K hidden nodes of the GRBF network. Recalling from the previous two subsections, each hidden node encodes a local signal state learnt from the history. Since the existing hidden nodes contain the previous system dynamics and they can be important in the future prediction, it is unwise to 'wipe out' them all. Rather, it is far better to identify the most out-of-date or most insignificant hidden node and replace it with a new hidden node that better represents the newly emerging local signal state. It can be seen that our fast adaptive GRBF is designed to have self-tuned fast tracking capability for capturing local characteristics of non-stationary signals, while maintaining low online computational complexity. To achieve these two objectives, three issues need to be addressed: 1) when the hidden node replacement takes place, and which node should be replaced; 2) how the structure of the new node is optimised; and 3) how the weight vector of the new network is optimized. We now provide the details below.

A. Node Replacement

At sample t, we have the data point $(\boldsymbol{x}_t, y_t)$ and the weight vector $\boldsymbol{\theta}_{t-1} = [\theta_1(t-1) \cdots \theta_K(t-1)]^{\mathrm{T}}$. We first need to decide whether a node replacement should take place. We can calculate the residual error e_t according to

$$e_t = y_t - \boldsymbol{\phi}_t^{\mathrm{T}} \boldsymbol{\theta}_{t-1}, \tag{14}$$

where $\boldsymbol{\phi}_t^{\mathrm{T}} = [\phi_1(\boldsymbol{x}_t) \cdots \phi_K(\boldsymbol{x}_t)]$ is the hidden layer's response to $\boldsymbol{x}_t$. The following squared relative error (SRE) is defined to measure the overall network performance

$$\mathrm{SRE}_t = \frac{e_t^2}{y_t^2}, \tag{15}$$

and we introduce the following decision rule

$$\begin{cases} \text{if } \mathrm{SRE}_t < \varepsilon, & \text{model structure remains unchanged,} \\ \text{if } \mathrm{SRE}_t \geq \varepsilon, & \text{node replacement takes place,} \end{cases} \tag{16}$$

where ε is a preset positive threshold. In general, a small ε leads to frequent node replacements that improves modeling accuracy but imposes more computational cost, and vice verse.

1) No node replacement: When $\mathrm{SRE}_t < \varepsilon$, we simply keep the GRBF network model structure unchanged.

2) Which node to replace: When $\mathrm{SRE}_t \geq \varepsilon$, we need to decide which node to be replaced. Similar to [37], we define the weighted node-output variance (WNV) that measures the individual node's contribution. Specifically, the WNV of the jth node is given by

$$\mathrm{WNV}_j = \left|\phi_j(\boldsymbol{x}_t)\theta_j(t-1)\right|^2. \tag{17}$$

Then find the node with the smallest WNV value:

$$m = \arg \min_{1 \leq i \leq K} \mathrm{WNV}_i. \tag{18}$$

Since the node m has the smallest WNV, it is the worst performing node and is replaced by a new node.

B. Optimizing New Replacement Node

At sample t, we have decided to replace the node m. We need to determine the new center $\boldsymbol{c}_m$, scalar δ_j and width α_m. By exploiting the geometric property of GRBF hidden node, the task of optimizing this new node becomes straightforward. We simply set $\boldsymbol{c}_m = \boldsymbol{x}_t$ and $\delta_m = y_t - y_{t-1}$ to ensure that the new replacement node m is a perfect local predictor of y_t and the new local signal state, represented by $\{\boldsymbol{x}_t, y_t\}$, is automatically encoded into the updated network structure. Since the set of centers now contains a new one, the new maximum distance $d_{\max}$ between the centers is recalculated, and the new width α_m is determined according to, for example, (11) based on this new $d_{\max}$.

C. Updating Network Weight Vector

1) $\text{SRE}_t < \varepsilon$*:* Since the model structure is unchanged, we simply update the weight vector using the RLS algorithm

$$\begin{cases} \boldsymbol{\psi}_t = \boldsymbol{P}_{t-1}\boldsymbol{\phi}_t\left(\lambda + \boldsymbol{\phi}_t^{\mathrm{T}}\boldsymbol{P}_{t-1}\boldsymbol{\phi}_t\right)^{-1}, \\ \boldsymbol{P}_t = \left(\boldsymbol{P}_{t-1} - \boldsymbol{\psi}_t\boldsymbol{\phi}_t^{\mathrm{T}}\boldsymbol{P}_{t-1}\right)\lambda^{-1}, \\ \boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \boldsymbol{\psi}_t e_t, \end{cases} \tag{19}$$

where $\boldsymbol{\psi}_t \in \mathbb{R}^K$ is the Kalman gain vector, $0.9 \le \lambda < 1$ is the forgetting factor, and the inverse of covariance matrix $\boldsymbol{P}_t \in \mathbb{R}^{K\times K}$ is usually initialized to $\boldsymbol{P}_0 = \vartheta\boldsymbol{I}_K$ in which ϑ is a large positive constant and $\boldsymbol{I}_K$ is the $K \times K$ identity matrix.

2) $\text{SRE}_t \ge \varepsilon$*:* Since the network structure has been changed, the previous weight vector $\boldsymbol{\theta}_{t-1}$ is no longer relavent, and we need to recompute the weight vector. We will compute the new weight vector $\boldsymbol{\theta}_t$ as the LS estimate based on the p latest data $D_p = \{\boldsymbol{x}_{t-i}, y_{t-i}\}_{i=0}^{p-1}$.

Specifically, define the desired output vector and the regression matrix over the data set D_p respectively as

$$\boldsymbol{y}_p = [y_t\ y_{t-1} \cdots y_{t-p+1}]^{\mathrm{T}}, \tag{20}$$

$$\boldsymbol{\Phi}_p = \begin{bmatrix} \phi_1(\boldsymbol{x}_t) & \phi_2(\boldsymbol{x}_t) & \cdots & \phi_K(\boldsymbol{x}_t) \\ \phi_1(\boldsymbol{x}_{t-1}) & \phi_2(\boldsymbol{x}_{t-1}) & \cdots & \phi_K(\boldsymbol{x}_{t-1}) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(\boldsymbol{x}_{t-p+1}) & \phi_2(\boldsymbol{x}_{t-p+1}) & \cdots & \phi_K(\boldsymbol{x}_{t-p+1}) \end{bmatrix}. \tag{21}$$

Then the regularized LS estimate of $\boldsymbol{\theta}_t$ is readily given by

$$\boldsymbol{\theta}_t = \left(\boldsymbol{\Phi}_p^{\mathrm{T}}\boldsymbol{\Phi}_p + \beta\boldsymbol{I}_K\right)^{-1}\boldsymbol{\Phi}_p^{\mathrm{T}}\boldsymbol{y}_p, \tag{22}$$

where β is a small positive regularization parameter, e.g., $\beta = 10^{-6}$. To achieve the smooth transition from one mode to another at the next sample, the inverse of the covariance matrix $\boldsymbol{P}_t$ for $\boldsymbol{\theta}_t$ is needed. Therefore, we always set $\boldsymbol{P}_t$ to

$$\boldsymbol{P}_t = \left(\boldsymbol{\Phi}_p^{\mathrm{T}}\boldsymbol{\Phi}_p + \beta\boldsymbol{I}_K\right)^{-1}, \tag{23}$$

after the regularized LS estimation of (22). The value of p trades off estimation accuracy with complexity. It can be seen that the main computational complexity of our fast adaptive GRBF algorithm comes from this online LS estimation.

D. Algorithm Summary

The proposed fast adaptive GRBF algorithm is summarized in Algorithm 1. We highlight that our fast adaptive GRBF network is capable of effectively address the well-known stability-plasticity dilemma [51]. The hidden nodes of our GRBF network basically encode the past local data dynamics, which provides the ability to retain acquired knowledge, i.e., stability. During online learning in a nonstationary environment, a learning model must be able to forget part of the previous knowledge in order to capture the newly emerging data pattern as fast as possible, i.e., plasticity. Basically, the number of hidden nodes in the GRBF network can be regarded as its memory depth, and the size K of the fast adaptive GRBF network can be quite small in order to have excellent plasticity of not keeping too much past knowledge. Our fast adaptive GRBF network forgets the most out-of-date hidden node to free 'space' for encoding the newly emerging local data characteristics as fast as these new dynamics appear. Sensitivity of the algorithmic parameters, ε and p, to the achievable prediction accuracy and online computational complexity will be further investigated in the experimental study.

Compared with the work of [37], [52], our fast adaptive GRBF algorithm imposes significantly lower online computational complexity per sample. This is because for the tunable RBF network, the center and width of the replacement node must be optimized, which is an $(M+1)$-dimensional nonlinear optimization problem. In [37], [52], this optimization is solved using a gradient descent iterative procedure at each sampling time to minimize the square of the current error. This iterative optimization procedure imposes considerably online computational complexity per sample. In fact, to reliably determine the center and width, the multi-innovation gradient descent [53] should be employed, i.e., the optimization should be based on a

Algorithm 1: Fast Adaptive GRBF Algorithm.

```
1:  **Initialization**
2:  Construct initial $K$-term GRBF model
    $\{\boldsymbol{c}_j, \delta_j, \alpha_j = \alpha, \theta_j\}_{j=1}^K$ based on training data set
    $\{\boldsymbol{x}_k, y_k\}_{k=1}^N$ using OLS algorithm of Subsection II-B.
3:  Set $\boldsymbol{\theta}_0$ to weight vector of initial GRBF model,
    $\boldsymbol{P}_0 = \vartheta\boldsymbol{I}_K$, and collect $p$ latest data points
    $D_p = \{\boldsymbol{x}_{-i}, y_{-i}\}_{i=0}^{p-1}$.
4:  Set sample index $t = 1$.
5:  **Online prediction**
6:  Given input $\boldsymbol{x}_t$, compute prediction of $y_t$ as
    $\widehat{y}_t = \boldsymbol{\phi}_t^{\mathrm{T}}\theta_{t-1}$.
7:  **Online adaptation**
8:  When actual output $y_t$ is available, shift oldest data
    point out of $D_p$, and add $(\boldsymbol{x}_t, y_t)$ to $D_p$.
9:  Calculate $\text{SRE}_t$ using (14) and (15).
10: **IF** $\text{SER}_t < \varepsilon$:
11:   Update weight vector $\boldsymbol{\theta}_t$ with RLS algorithm (19).
12: **ELSE IF** $\text{SER} \ge \varepsilon$:
13:   Calculate the WNV values for all $K$ nodes using
      (17), and find node with smallest WNV using (18).
14:   Node $m$ has smallest WNV, then replace it:
15:     Set new center to $\boldsymbol{c}_m = \boldsymbol{x}_t$ and new scalar
        $\delta_m = y_t - y_{t-1}$.
16:     Recalculate maximum distance $d_{\max}$ between
        centers, and set new width $\alpha_m$ according to (11)
17:   Recalculate network weight vector $\boldsymbol{\theta}_t$ as regularized
      LS estimate, and update $\boldsymbol{P}_t$ with (23).
18: **END IF**
19: Set $t = t + 1$ and go to step 5.
```

block of p latest data points. This, however, would impose even heavier computational complexity. By contrast, the optimization of the replacement hidden node for our GRBF network involve very little computation as can be clearly seen in the previous subsection. To be specific, during online operation if only weight adaptation is performed, the complexity comes from the RLS algorithm (19), which is on the order of $\mathsf{O}(K^2)$, while if the node replacement occurs, the WNV calculation in (17) costs $\mathsf{O}(K)$ and the weigh adaptation costs $\mathsf{O}(p^3)$ for the regularized LS estimation (22).Thus, the online computational complexity per sample of the proposed algorithm is no more than $\max\{\mathsf{O}(p^3), \mathsf{O}(K^2)\}$. Since p and K are typically very small, this is clearly affordable.

IV. Experimental Results and Discussions

Extensive experiments involving several nonlinear and nonstationary benchmark time series are conducted to evaluate the proposed fast adaptive GRBF network. The first case study includes online prediction of two chaotic time series, Rossler series [54] and Lorenz series [55], while two real-world time series, the monthly recorded sunspot number [56], [57] and an electroencephalographic (EEG) time series [58]–[60], are predicted in the second case study. The mean square error (MSE) and mean absolute error (MAE), defined as

$$\mathrm{MSE}_t = \frac{1}{t}\sum_{i=1}^{t}\left(y_i - \widehat{y}_i\right)^2, \tag{24}$$

$$\mathrm{MAE}_t = \frac{1}{t}\sum_{i=1}^{t}\left|y_i - \widehat{y}_i\right|, \tag{25}$$

are used to evaluate the online prediction performance, where $\widehat{y}_i$ is the model prediction for y_i. The online computational complexity is quantified by its online averaged computation time per sample (ACTpS). The experiments are carried out on Matlab 2017a, running on a PC with i7-3770 3.40 GHz processor of 4 cores and 16 GB of RAM.

The performance of our proposed approach is compared with those of typical approaches including the LSTM [17], [20]–[22], the GRU [18], [19], the OS-ELM [33]–[36], the RBF network [4], the GRBF network [39] and the fast tunable RBF [37]. For the OS-ELM, a RBF network is initialized during training by randomly selecting a large number of input data points as its centers, and the online adaptation of the OS-ELM involves the weight updating using the RLS algorithm. For the RBF model, a small RBF model is constructed during training using the OLS algorithm, and the RBF network structure as well as its weights are fixed throughout the online prediction. Similar, the initial GRBF model is constructed during training using the OLS algorithm, and the model structure as well as weights are fixed during online prediction. For the fast tunable RBF, the RBF model is initialized during training using the OLS algorithm, and adaptation takes place during online modeling and prediction according to the algorithm described in [37]. For both the LSTM and GRU with single hidden layer, the mini batch stochastic gradient descend is used to tune the parameters of the LSTM and GRU during training, and their network structures and parameters are fixed throughout online prediction.

A. *Chaotic Time Series Prediction*

This case study involves prediction of Rossler and Lorenz time series. For each chaotic time series, after removing a large number of initial data points, 2100 samples are generated, with the first 100 samples as the training set for initial modeling and the remaining 2000 samples as the test dataset for adaptive prediction. Also for each time series, 100 independent realizations are generated. The performance of each method are presented by its mean and standard deviation (STD) of the test MSE and ACTpS over the 100 realizations.

The embedding dimension is set to $M = 6$. Hence, the input dimension of the GRBF model is $M' = 5$. For our fast adaptive GRBF, the decision threshold and the number of latest data points for regularized LS estimate are empirically chosen to be $\varepsilon = 10^{-6}$ and $p = 7$, respectively. For a fair comparison, we also use $\varepsilon = 10^{-6}$ and $p = 7$ for the fast tunable RBF. The regularization parameter for regularized LS estimator should be a very small positive number and we set $\beta = 10^{-6}$. Additionally, for the fast tunable RBF, the step size and the maximum number of iterations are empirically chosen to be 0.01 and 5, respectively, for its gradient descent iterative search procedure. For the LSTM and GRU, the learning rate and batch size are carefully tuned to be 0.005 and 1, respectively, for gradient descend training, while the number of maximum training epochs is set to 50 for the both RNNs.

1) Rossler Chaotic Time Series: Rossler process is a system of three ordinary differential equations

$$\begin{cases} \frac{\mathrm{d}\,x(t)}{\mathrm{d}\,t} &= -y(t) - z(t), \\ \frac{\mathrm{d}\,y(t)}{\mathrm{d}\,t} &= x(t) - ay(t), \\ \frac{\mathrm{d}\,z(t)}{\mathrm{d}\,t} &= b + z(t)(x(t) - c), \end{cases} \tag{26}$$

which define a continuous dynamical map that exhibits chaotic dynamics associated with the fractal properties of the Rossler attractor [54]. The fourth-order Runge-Kutta method with a step size of 0.01 is used to generate the samples, and only Y-dimension samples $\{y_t\}$ are used for time series prediction.

First consider this chaotic time series with the fixed controlling parameters $a = 0.2$, $b = 0.2$ and $c = 5.7$. Table I compares the performance of the seven methods, in terms of test predication accuracy and ACTpS. For the RBF, GRBF, LSTM and GRU methods, no online adaptation takes place, and they do not impose the computational complexity of online adaptation. Fig. 2 depicts the MSE learning curves for all the seven methods, where it can be seen that the performance of the OS-ELM is poor, only attaining a prediction accuracy similar to the small nonadaptive RBF model. The LSTM and GRU are only slightly better than the OS-ELM and the nonadaptive RBF, with the GRU attaining lower test MSE than the LSTM. Also the small nonadaptive GRBF model is more than 20 dB better in the test MSE than the GRU in this case. The test MSE of the fast tunable RBF is about 23 dB lower than the fixed GRBF, while our fast adaptive GRBF attains the best test MSE performance, about 19 dB lower than the fast tunable RBF. Observe from Table I

TABLE I
COMPARISON OF ONE-STEP PREDICTION PERFORMANCE (AVERAGE±STD) OF DIFFERENT PREDICTORS FOR ROSSLER SERIES WITH FIXED PARAMETERS

Method	Model Type	Model Size	MSE (dB)	MAE	Online ACTpS (ms)
LSTM	All fixed	50	12.92±0.14	3.359±0.101	-
		100	12.58±0.17	3.216±0.109	
GRU	All fixed	50	11.03±0.15	2.854±0.087	-
		100	10.04±0.21	2.476±0.105	
OS-ELM	Structure fixed, weights tunable	100	13.37±0.84	3.519±0.569	0.30±0.009
RBF	All fixed	10	13.42±0.61	3.600±0.375	-
		20	**13.36±0.64**	**3.560±0.328**	
GRBF	All fixed	10	**-14.69±4.84**	**0.120±0.030**	-
		20	-14.11±2.74	0.115±0.027	
Fast tunable RBF	All tunable	10	**-37.50±3.99**	**0.001±2.75e-04**	0.09±0.021
Proposed	All tunable	10	**-56.31±12.16**	**7.64e-04±6.13e-04**	0.04±0.014

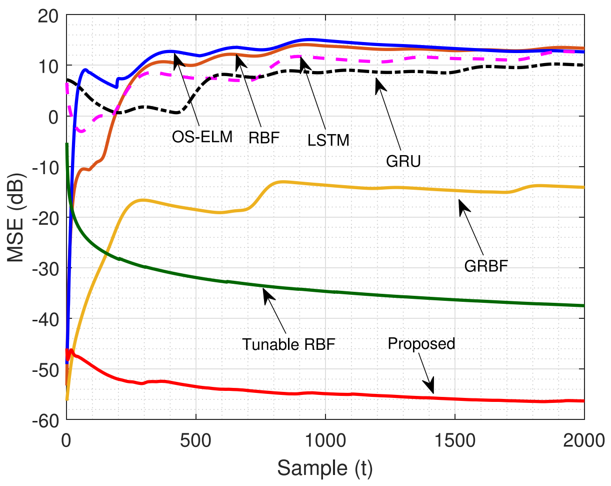


Fig. 2. Comparison of average MSE learning curves of various one-step predictors for Rossler time series with fixed control parameters. The RBF and GRBF models both have 20 hidden nodes, and the LSTM and GRU models both have 100 hidden nodes.

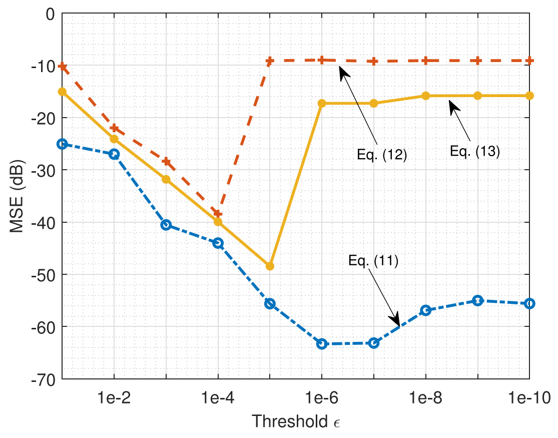


Fig. 3. Impact of the threshold ε with different calculations of Gaussian width α on the test MSE for the proposed method in online prediction of Rossler time series with fixed control parameters, given $p = 7$ and $K = 10$.

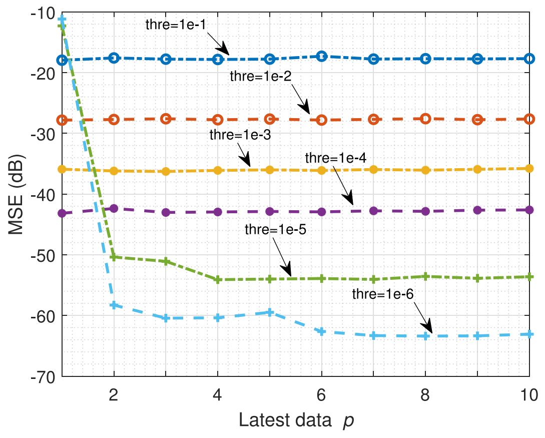


Fig. 4. Impact of number of latest data points p on the test MSE for the proposed method in online prediction of Rossler time series with fixed control parameters, given different thresholds ε and $K = 10$ hidden nodes.

that the ACTpS imposed by the fast tunable RBF is significantly lower than that of the OS-ELM, and our fast adaptive GRBF is considerably better than the fast tunable RBF, in terms of ACTpS.

The three methods of determining the node width α in Subsection II-C, (11) to (13), are well investigated in the literature, each performing better than the others in different situations. In our application, we find (11) is better, and it is employed in Algorithm 1. To demonstrate this, Fig. 3 shows the achievable test MSE performance by three adaptive GRBF models with these three Gaussian width calculations. It can be seen that the adaptive GRBF predictor based on the width calculation (11) attains the best prediction accuracy.

We now investigate how the algorithmic parameters of Algorithm 1, the latest data points p and threshold ε, impact on the achievable test MSE performance. Fig. 3 also depicts the achievable test MSE performance as the function of ε, given $p = 7$ and $K = 10$, where it can be seen that the lowest test MSE is attained at $\varepsilon = 10^{-6}$ (when (11) is used). Fig. 4 investigates the impact of p on prediction performance, given various threshold values and $K = 10$. Observing the test MSE curve related to $\varepsilon = 10^{-6}$, it can be seen that $p = 7$ is appropriate in this case, as further increasing p does not lead further improvement in MSE but will impose higher online computational complexity.

TABLE II
COMPARISON OF ONE-STEP PREDICTION PERFORMANCE (AVERAGE±STD) OF DIFFERENT PREDICTORS FOR ROSSLER SERIES WITH TIME-VARYING PARAMETERS

Method	Model Type	Model Size	MSE (dB)	MAE	Online ACTpS (ms)
LSTM	All fixed	50	14.09±0.09	3.737±0.089	-
		100	13.84±0.19	3.644±0.167	
GRU	All fixed	50	12.48±0.15	3.135±0.056	-
		100	12.22±0.17	3.013±0.079	
OS-ELM	Structure fixed, weights tunable	100	15.03±0.39	4.214±0.198	0.31±0.008
RBF	All fixed	10	15.07±0.25	4.344±0.153	-
		20	**15.04±0.26**	**4.283±0.176**	
GRBF	All fixed	10	**-10.95±4.22**	**0.172±0.032**	-
		20	-10.73±3.62	0.104±0.028	
Fast tunable RBF	All tunable	10	**-25.61±8.91**	**0.014±0.002**	0.19±0.021
Proposed	All tunable	10	**-40.09±10.43**	**0.002±0.001**	0.04±0.001

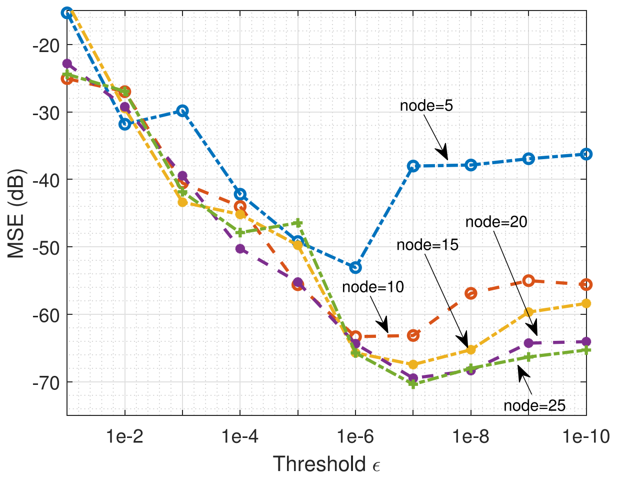


Fig. 5. Impact of threshold ε on the test MSE for the proposed method in online prediction of Rossler time series with fixed control parameters, given different numbers of hidden nodes K and $p = 7$.

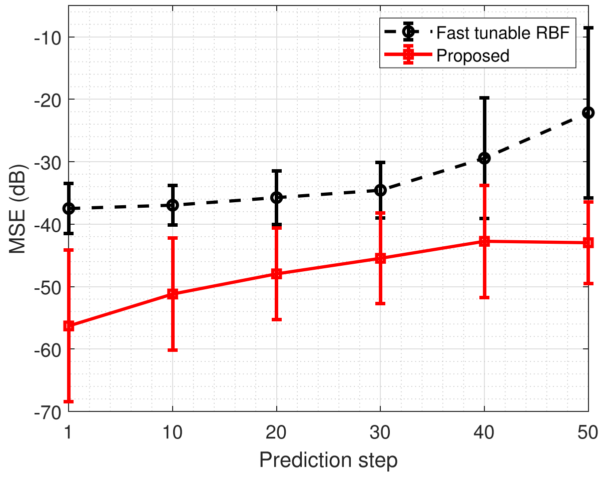


Fig. 6. Average MSE performance and associated STDs of two multi-step prediction models for Rossler time series with fixed control parameters. Both the fast tunable RBF and fast adaptive GRBF have 10 hidden nodes.

Note that it is impossible to determine the GRBF predictor's size K via its online test performance. The size of the GRBF predictor K can only be determined in the initial training, with the aim to construct a small predictor that imposes a small online computational complexity while attaining adequate training performance. Fig. 5 shows the impact of the threshold ε with different numbers of hidden nodes K on the test MSE performance, given $p = 7$. Observing the test MSE curve related to $K = 10$, again it is seen that $\varepsilon = 10^{-6}$ is appropriate in this case. Also from Fig. 5, it can be seen that better test MSE performance can be achieved by using $K = 15, 20$ or 25, with approximate optimal value of $\varepsilon = 10^{-7}$. However, these predictors will increase the online computational complexity considerably, compared the case of $K = 10$.

To show that all the methods can be extended to multi-step prediction, we consider the multi-step adaptive prediction, which uses the embedding vector $\boldsymbol{x}_t$ of (1) to provide the m-step ahead prediction for y_{t+m-1}. In Fig. 6, we compare the multi-step prediction performance for the fast tunable RBF and our fast adaptive GRBF. As expected, our fast adaptive GRBF significantly outperforms the fast tunable RBF. Observe that the STD of prediction accuracy for our method is much smaller than that of the fast tunable RBF, indicating that our method attains much more accurate and reliable prediction.

Next, we let the controlling parameters of Rossler map vary with time to obtain an even more nonlinear and nonstationary time series. Specifically, we set

$$\begin{cases} a = & 0.2, \\ b = & 0.1 + 0.1\left(1 + \sin(0.1t)\right), \\ c = & 3.7 + 2\left(1 + \cos\left(2^{0.1t}\right)\right). \end{cases} \tag{27}$$

Table II and Fig. 7 compare the one-step prediction performance of the seven methods for this Rossler chaotic time series with time-varying control parameters. Again it can be clearly seen that the proposed fast adaptive GRBF model attains the best performance, in terms of both prediction accuracy and ACTpS. The average MSE learning curves and associated STDs of the multi-step fast tunable RBF and fast adaptive GRBF predictors

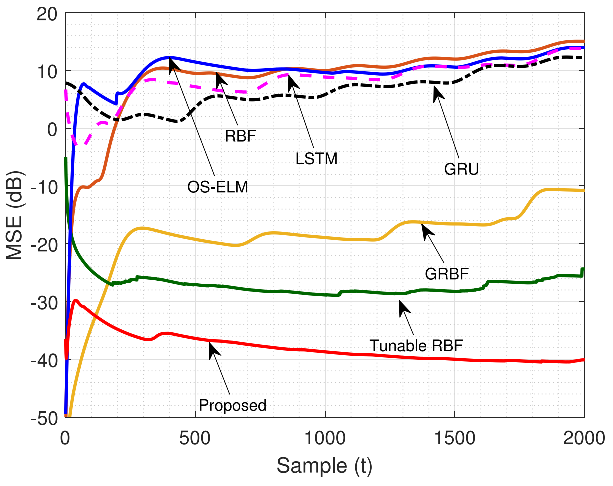


Fig. 7. Comparison of average MSE learning curves of various one-step prediction models for Rossler time series with time-varying control parameters. The RBF and GRBF models both have 20 hidden nodes, and the LSTM and GRU models both have 100 hidden nodes.

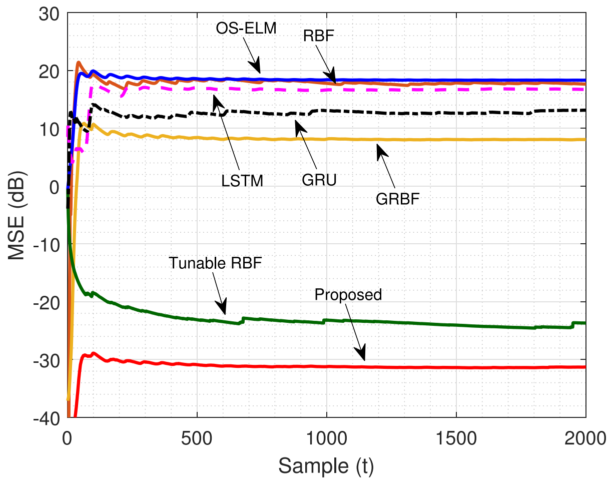


Fig. 9. Comparison of average MSE learning curves of various one-step prediction models for Lorenz time series with fixed control parameters. The RBF and GRBF models both have 20 hidden nodes, and the LSTM and GRU models both have 100 hidden nodes.

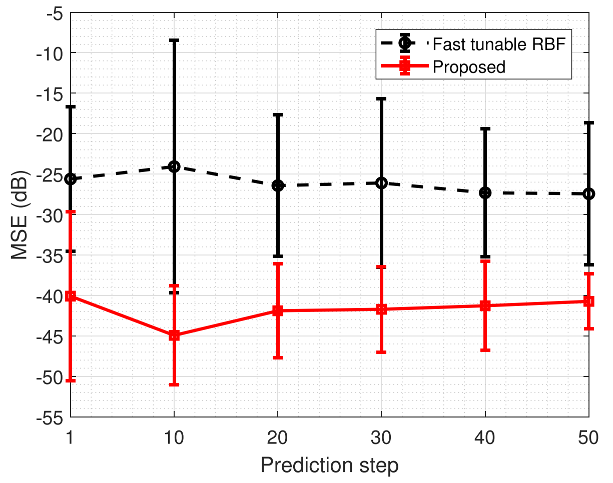


Fig. 8. Average MSE performance and associated STDs of two multi-step prediction models for Rossler time series with time-varying control parameters. Both the fast tunable RBF and fast adaptive GRBF have 10 hidden nodes.

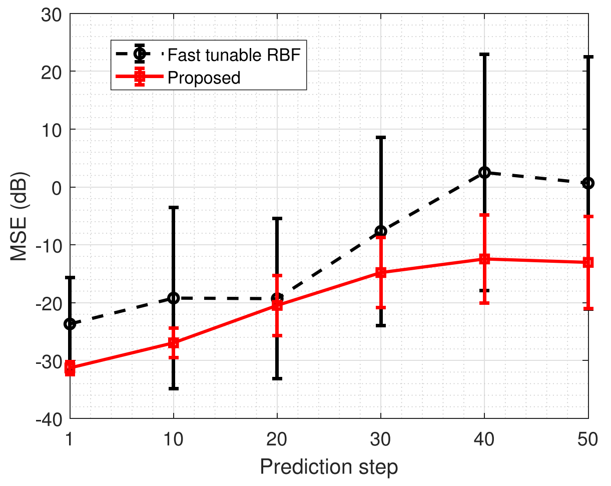


Fig. 10. Average MSE performance and associated STDs of two multi-step prediction models for Lorenz time series with fixed control parameters. Both the fast tunable RBF and fast adaptive GRBF have 10 hidden nodes.

are shown in Fig. 8, which again demonstrates the superior performance of our method.

2) Lorenz Chaotic Time Series: Lorzen process [55] is a nonlinear dynamic system that exhibits chaotic flow. It is governed by the three differential equations as

$$\begin{cases} \frac{\mathrm{d}\,x(t)}{\mathrm{d}\,t} = a(y(t)-x(t)), \\ \frac{\mathrm{d}\,y(t)}{\mathrm{d}\,t} = cx(t)-x(t)z(t)-y(t), \\ \frac{\mathrm{d}\,z(t)}{\mathrm{d}\,t} = x(t)y(t)-bz(t). \end{cases} \tag{28}$$

The fourth-order Runge-Kutta method with a step size of 0.01 is used to generate the samples, and only Y-dimension samples $\{y_t\}$ are used for the time-series prediction.

Again, we first consider the fixed controlling parameters with $a = 10$, $b = 8/3$ and $c = 28$. Table III compares the performance of the seven one-step predictors, in terms of prediction accuracy and ACTpS, while Fig. 9 depicts the average test MSE learning curves of these seven one-step prediction models. The results obtained again show that both RNN predictors and the classic RBF model are inferior to the fixed GRBF network. The fast tunable RBF achieves the second best prediction performance, which is about 31 dB better than the nonadaptive GRBF predictor. Our proposed adaptive GRBF is dramatically better than the fast tunable RBF, in terms of both online prediction accuracy and ACTpS.

The results of multi-step ahead prediction performance for the two best predictors, the fast tunable RBF and our fast adaptive GRBF, are shown in Fig. 10. It can be seen that our proposed method consistently outperforms the fast tunable RBF. In particular, the average MSE of the 50-step-ahead fast adaptive GRBF predictor is more than 15 dB lower than that of the 50-step-ahead fast tunable RBF predictor. Moreover, the STDs

TABLE III
COMPARISON OF ONE-STEP PREDICTION PERFORMANCE (AVERAGE±STD) OF DIFFERENT PREDICTORS FOR LORENZ SERIES WITH FIXED PARAMETERS

Method	Model Type	Model Size	MSE (dB)	MAE	Online ACTpS (ms)
LSTM	All fixed	50	16.79±0.24	4.737±0.440	-
		100	16.70±0.31	4.388±0.556	
GRU	All fixed	50	13.87±0.19	3.252±0.450	-
		100	13.10±0.33	3.459±0.671	
OS-ELM	Structure fixed, weights tunable	100	17.91±0.35	5.387±0.422	0.33±0.009
RBF	All fixed	10	18.71±0.14	6.702±0.165	-
		20	**18.30±0.26**	**6.016±0.189**	
GRBF	All fixed	10	9.41±0.74	1.343±0.194	-
		20	**8.04±0.73**	**1.023±0.132**	
Fast tunable RBF	All tunable	10	**-23.68±8.01**	**0.017±0.034**	0.24±0.011
Proposed	All tunable	10	**-31.28±1.13**	**0.014±0.001**	0.11±0.009

TABLE IV
COMPARISON OF ONE-STEP PREDICTION PERFORMANCE (AVERAGE±STD) OF DIFFERENT PREDICTORS FOR LORENZ SERIES WITH TIME-BASED DRIFT

Method	Model Type	Model Size	MSE (dB)	MAE	Online ACTpS (ms)
LSTM	All fixed	50	38.50±0.15	55.614±1.587	-
		100	38.22±0.21	52.424±2.067	
GRU	All fixed	50	37.33±0.18	46.156±1.897	-
		100	36.54±0.34	45.297±1.765	
OS-ELM	Structure fixed, weights tunable	100	38.88±0.13	59.846±1.156	0.30±0.008
RBF	All fixed	10	38.89±0.14	60.356±1.112	-
		20	**38.88±0.13**	**59.844±1.234**	
GRBF	All fixed	10	38.66±0.17	53.878±1.367	-
		20	**38.58±0.19**	**52.897±1.987**	
Fast tunable RBF	All tunable	10	**7.77±8.95**	**0.987±0.112**	0.36±0.034
Proposed	All tunable	10	**-8.60±5.86**	**0.129±0.091**	0.14±0.009

of the test MSE for the multi-step fast adaptive GRBF predictor are consistently much smaller than those for the multi-step fast tunable RBF predictor. This confirms that the fast adaptive GRBF is much more accurate and reliable than the fast tunable RBF.

Next, we create a new series by modifying Lorenz chaotic time series with a time-based drift. Specifically, Lorenz time series samples $\{y_t\}$ are weighted by an exponential time-based drift to obtain a new series $\{\widetilde{y}_t\}$ according to

$$\widetilde{y}_t = 1.1^{0.01t} y_t. \tag{29}$$

The new time series $\{\widetilde{y}_t\}$ is then used for the time series prediction. Note that $\{\widetilde{y}_t\}$ is seriously nonstationary, and the dynamic range of $\widetilde{y}_t$ change from initially around $[-10,\ 10]$ to about $[-400,\ 400]$ in the end. Table. IV summarizes the performance of the seven one-step predictor models, in terms of prediction accuracy and online computational complexity, while Fig. 11 compares the average test MSE learning curves for different one-step predictors. Clearly, except for the fast tunable RBF and our method, the other five models all have difficulty to predict this time series accurately, as evidenced by their large testing MSEs. Unlike the previous example, the GRBF can hardly improve the performance over the RBF. This is because most of the nonstationary features of Lorenz time series with time-based drift are not nearly variations of local mean and trend. Moreover, the dynamics of this series varies significantly beyond the initial modeling space. Thus fixed predictors, such as the RBF, GRBF, LSTM and GRU perform poorly. It can be seen that the average test MSE of the one-step fast tunable RBF predictor is more than 30 dB lower than those of the RBF, GRBF, LSTM and GRU. Also observe that the average test MSE of our fast adaptive GRBF is more than 15 dB lower than the fast tunable RBF, while imposing a significantly smaller ACTpS than the latter.

The multi-step-ahead prediction performance of the fast tunable RBF and our fast adaptive GRBF are shown in Fig. 12. Except for the prediction steps of 10 and 30, our fast adaptive GRBF predictor consistently outperforms the fast tunable RBF predictor, in terms of both average test MSE and STD. Although the average MSEs of the fast tunable RBF predictor are lower than those of our fast adaptive GRBF predictor at the prediction steps of 10 and 30, the corresponding STDs of the former are much larger than those of the latter. Therefore, our multi-step fast adaptive GRBF predictor is much more reliable than the

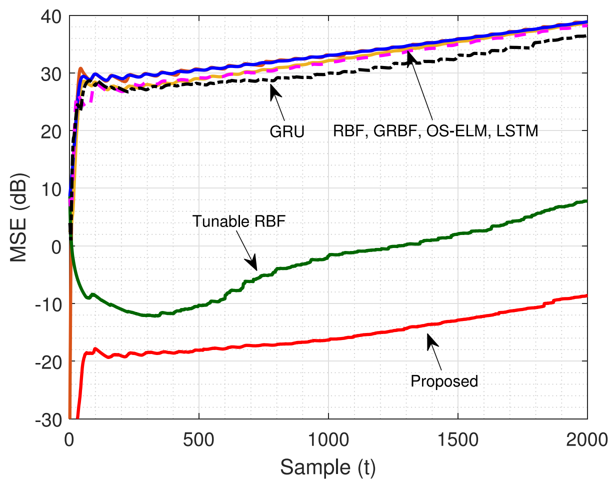


Fig. 11. Comparison of average MSE learning curves of various one-step prediction models for Lorenz time series with time-based drift. The RBF and GRBF models both have 20 hidden nodes, and the LSTM and GRU models both have 100 hidden nodes.

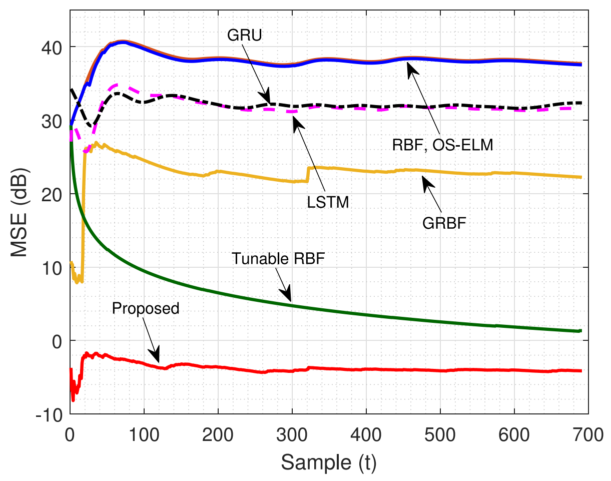


Fig. 13. Comparison of MSE learning curves of various one-step prediction models for sunspot time series. The RBF and GRBF models both have 50 hidden nodes, and the LSTM and GRU models both have 500 hidden nodes.

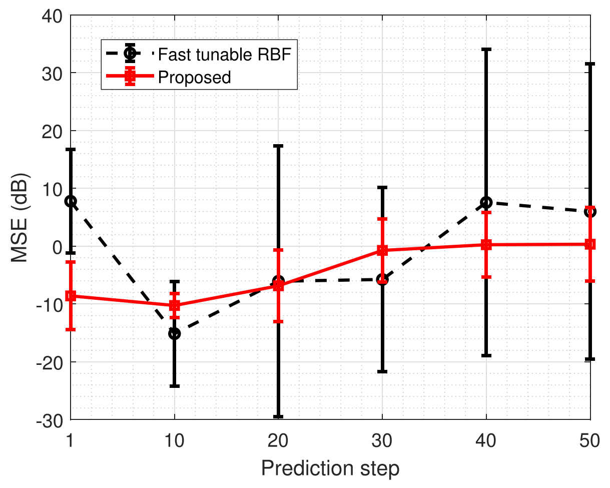


Fig. 12. Average MSE performance and associated STDs of two multi-step prediction models for Lorenz time series with time-based drift. Both the fast tunable RBF and fast adaptive GRBF have 10 hidden nodes.

multi-step fast tunable RBF predictor over the entire range of the prediction steps tested.

B. Real-World Time Series Prediction

To further illustrate its applicability and efficiency for real data prediction, we apply our method to two real-life time series, namely, the sunspot number time series [56] and an EEG signal [60].

1) Prediction of Sunspot Number Series: The sunspot time series is an annual averaged relative number of sunspots observed, which exhibits highly complex and nonstationary characteristics. It is widely used as a benchmark for nonlinear time series analysis and prediction [57]. The monthly recorded sunspot time series from 1945 to 2017 s is considered here. In constructing one-step predictor, the entire sunspot series is divided into the training set, from 1945 to 1953, and the test set, from 1954 to 2017. The embedding vector's dimension is chosen to be $M = 4$ as in [57]. For our fast adaptive GRBF, $\varepsilon = 10^{-2}$ and $p = 7$ are empirically chosen. The same ε and p are also used for the fast tunable RBF. The regularization parameter is again set to $\beta = 10^{-6}$. Additionally, the step size for gradient descent and the maximum number of iterations for RBF node optimization are chosen to be 0.01 and 5, respectively, for the fast adaptive RBF. For the LSTM and GRU, the learning rate and batch size are empirically chosen to be 0.005 and 1, respectively, for its gradient descend training, while the maximum epochs during training is 50. Since the sampling rate of this time series is month, one-step-ahead prediction is one-month-ahead prediction, one-year-ahead prediction is 12-step-ahead, and two-year-ahead prediction is 24-step-ahead, etc. Consequently, when constructing one-year-ahead predictor, the desired outputs are the sunspot number series from 1946 to 1954, and the test data are from 1955 to 2017. Thus, the test data becomes shorter as the prediction step increases.

One-step prediction performance of the seven predictors are compared in Table V, and their corresponding test MSE learning curves are depicted in Fig. 13. It can be seen again that the OS-ELM and the nonadaptive RBF are the worst predictors, as evidenced by their large test MSEs. Both the LSTM and GRU also find hard to track this sunspot number time series. The fixed GRBF predictor is much better but its prediction accuracy is still poor. This is further confirmed in Fig. 14, where it is clearly seen that the fixed GRBF one-step predictor cannot track the monthly recorded sunspot number adequately. Again the results demonstrate that our fast adaptive GRBF attains the best prediction accuracy, while imposing the lowest ACTpS.

Fig. 15 depicts the multi-year prediction performance using the two best methods, the fast tunable RBF and fast adaptive GRBF. Clearly, our method significantly outperforms the fast tunable RBF. It is interesting to note that the both methods achieve the lowest prediction errors in the 10-year-ahead prediction. This may be due to the fact that the sunspot number exhibits

TABLE V
COMPARISON OF ONE-STEP PREDICTION PERFORMANCE OF DIFFERENT PREDICTORS FOR SUNSPOT NUMBER TIME SERIES

Method	Model Type	Model Size	MSE (dB)	MAE	Online ACTpS (ms)
LSTM	All fixed	100	35.8631	49.7342	-
		200	34.3286	39.5836	
		500	31.2763	31.3537	
GRU	All fixed	100	33.1912	39.4791	-
		200	33.1240	39.2050	
		500	32.2065	35.6460	
OS-ELM	Structure fixed, weights tunable	100	37.5291	64.3362	0.34
RBF	All fixed	30	37.7580	66.5073	-
		50	**37.7064**	**66.6155**	
GRBF	All fixed	30	22.9400	6.8343	-
		50	**22.2310**	**7.4127**	
Fast tunable RBF	All tunable	10	**1.2514**	**0.1126**	0.43
Proposed	All tunable	10	**-4.3712**	**0.4756**	0.06

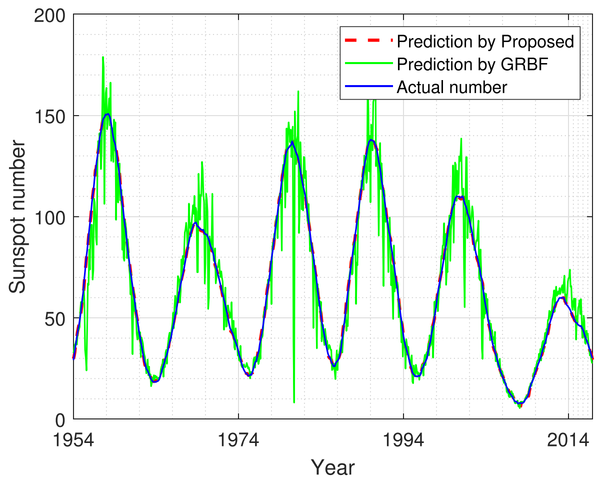


Fig. 14. One-step sunspot number predictions using the GRBF of size 50 and the fast adaptive GRBF of size 10.

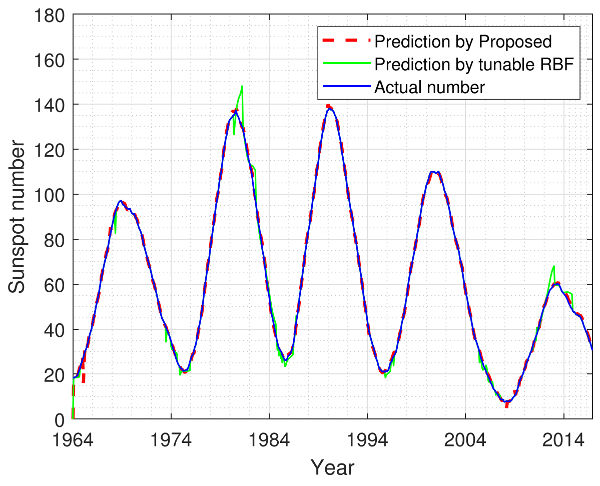


Fig. 16. Comparison of 10-year-ahead sunspot number predictions using the fast tunable RBF and the proposed fast adaptive GRBF. The both models have 10 hidden nodes.

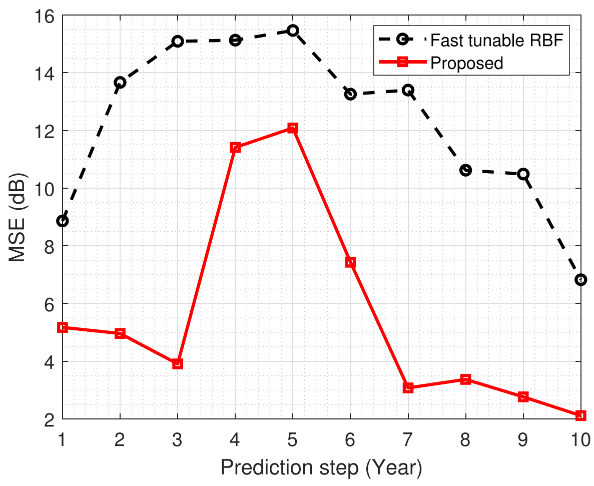


Fig. 15. MSE performance of two multi-year-ahead prediction models for sunspot time series. Both the fast tunable RBF and fast adaptive GRBF have 10 hidden nodes. A-year-ahead corresponds to 12-step-ahead.

approximately a 10-year cycle feature as can be seen from the sunspot series shown in Fig. 14. Fig. 16 further compares the 10-year predictions obtained by the fast tunable RBF and fast adaptive GRBF. It can be seen that the fast tunable RBF has difficulty to predict the sunspot series points where rates and gradient signs are changing. By contrast, our fast adaptive GRBF is inherently immune to this difficult.

2) Modeling of EEG Data: The EEG data set used is available publicly from University of Bonn [60], which is sampled at a sampling rate of 173.61 Hz. When constructing one-step predictor, we use 6 seconds of the EEG signals. The first second of observations are used for the initial training, while the signals from 2 s to 6 s are used for online prediction. The embedding vector's dimension is chosen to be $M = 4$. The algorithmic parameters of the fast adaptive GRBF are empirically chosen to be $\varepsilon = 10^{-4}$ and $p = 2$, with the regularization parameter set to $\beta = 10^{-6}$. The same ε and p are also used for the fast

TABLE VI
COMPARISON OF ONE-STEP PREDICTION PERFORMANCE OF DIFFERENT PREDICTORS FOR EEG SIGNAL

Method	Model Type	Model Size	MSE (dB)	MAE	Online ACTpS (ms)
LSTM	All fixed	100	28.8516	18.1790	-
		200	26.4762	11.6579	
		500	24.2097	6.5774	
GRU	All fixed	100	24.8655	10.1818	-
		200	24.6749	9.9119	
		500	24.3787	9.5523	
OS-ELM	Structure fixed, weights tunable	100	31.6739	30.1018	0.36
RBF	All fixed	30	31.6839	30.0054	-
		50	**31.6715**	**30.1061**	
GRBF	All fixed	30	32.0200	30.3387	-
		50	**31.7180**	**24.0683**	
Fast tunable RBF	All tunable	10	**18.2253**	**3.3985**	1.27
Proposed	All tunable	10	**10.2885**	**2.0574**	1.31

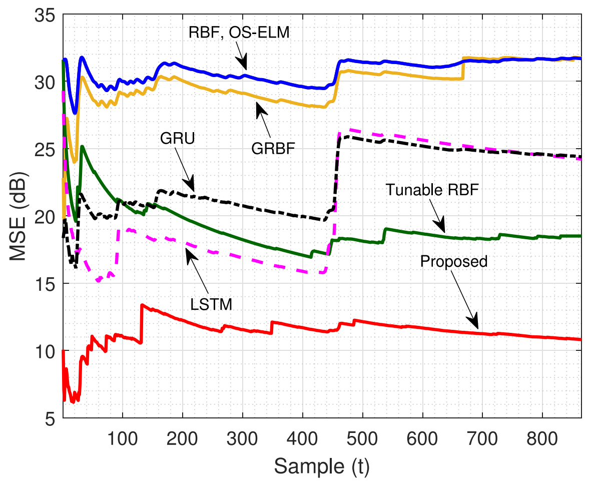


Fig. 17. Comparison of MSE learning curves of various one-step prediction models for EEG signal. The RBF and GRBF models both have 50 hidden nodes, and the LSTM and GRU models both have 500 hidden nodes.

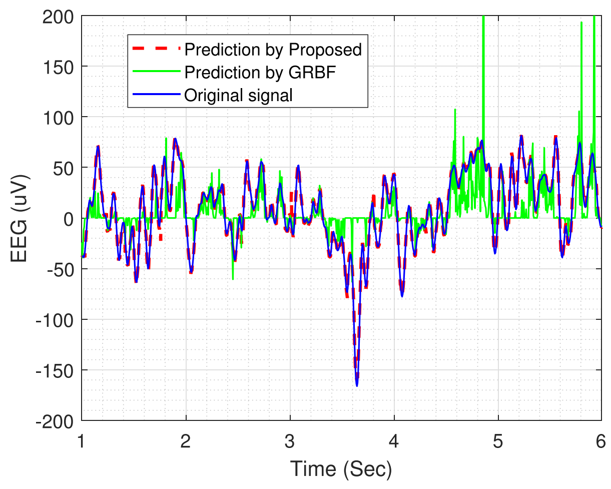


Fig. 18. One-step EEG signal predictions using the GRBF of size 50 and the fast adaptive GRBF of size 10.

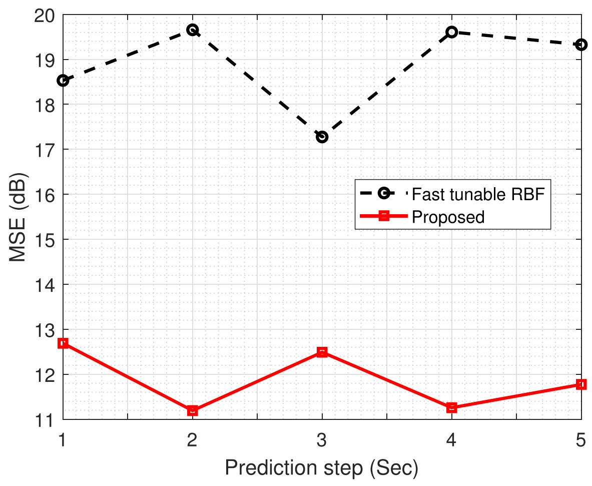


Fig. 19. Prediction accuracy comparison of two multi-second-ahead predictors for EEG signal. Both the fast tunable RBF and fast adaptive GRBF have 10 hidden nodes.

tunable RBF, while its step size and the maximum number of iterations are set to 0.01 and 5, respectively. Again the LSTM and GRU predictors employ the same structure settings with the previous simulation. Because the sampling rate is 173.61 Hz, one-second-ahead prediction corresponds to 173-step-ahead. As a result, when constructing one-second-ahead predictor, the desired outputs are the EEG series from 1 s to 2 s, and we use the EEG series from 2 s to 7 s for evaluating the test performance. Likewise, for 5-second-ahead prediction, the EEG series from 6 s to 11 s are user for test evaluation.

Table VI compares the performance of different one-step prediction models, while the corresponding MSE learning curves are depicted in Fig. 17. Not surprisingly, the OS-ELM as well as fixed RBF and fixed GRBF predictors can hardly capture the highly time-varying nonlinear dynamics of this EEG signal. Even the nonadaptive GRBF predictor performs poorly as can be clearly seen from Fig. 18. The LSTM and GRU have better

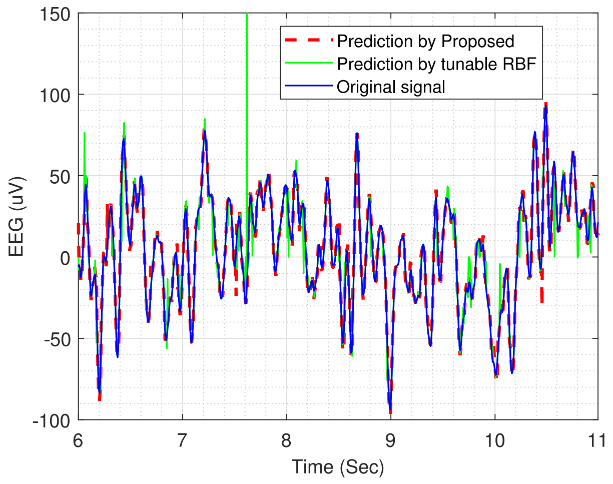


Fig. 20. Comparison of 5-second-ahead EEG signal predictions using the fast tunable RBF and fast adaptive GRBF. The both models have 10 hidden nodes.

prediction performance than the OS-ELM, fixed RBF and fixed GRBF, but they are still considerably inferior to the fast tunable RBF. Our fast adaptive GRBF predictor tracks this EEG signal extremely well. From Table VI, it is seen that the one-step fast adaptive GRBF predictor outperforms the one-step fast tunable RBF predictor by about 8 dB in prediction accuracy, although it imposes a slightly higher ACTpS in this case.

The multi-second-ahead prediction performance of the two best methods are shown in Fig. 19, where it can be seen that our fast adaptive GRBF predictor consistently outperforms the fast tunable RBF predictor. The 5-second predictions of the two models are illustrated in Fig. 20, which again conforms the superior performance of our fast adaptive GRBF predictor.

V. Conclusion

In this paper, we have proposed a novel fast adaptive GRBF network with adaptive tunable nodes for nonlinear and non-stationary time series modeling and prediction. By exploiting the local predictor property of hidden GRBF node, a compact initial GRBF network can readily be constructed using the OLS algorithm during initial training, which encodes the underlying dynamics seen from the training data in its hidden nodes. With the number of hidden nodes fixed, during online operation, our fast adaptive GRBF model automatically replaces the worst performing hidden node in the current signal environment with a new hidden node which encodes the newly emerging signal dynamics. It has been shown that the optimization of this new replacement node is straightforward and imposes little computation. We have demonstrated that the proposed fast adaptive GRBF model has excellent adaptability and plasticity. Extensive experiments have been conducted, involving two chaotic time series and two real-world signals. The results obtained have shown that our proposed fast adaptive GRBF network consistently outperforms the existing state-of-the-art fast tunable RBF network, in terms of both prediction accuracy and online computational complexity.

References

[1] S. Chen, “Nonlinear time series modelling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning,” *Electron. Lett.*, vol. 31, no. 2, pp. 117–118, Jan. 1995.

[2] I. Rojas *et al.*, “Time series analysis using normalized PG-RBF network with regression weights,” *Neurocomputing*, vol. 42, no. 1–4, pp. 267–285, Jan. 2002.

[3] C.-M. Lee and C.-N. Ko, “Time series prediction using RBF neural networks with a nonlinear time-varying evolution PSO algorithm,” *Neurocomputing*, vol. 73, no. 1–3, pp. 449–460, Dec. 2009.

[4] S. Chen, C. F. N. Cowan, and P. M. Grant, “Orthogonal least squares learning algorithm for radial basis function networks,” *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[5] S. Chen, “Local regularization assisted orthogonal least squares regression,” *Neurocomputing*, vol. 69, no. 4-6, pp. 559–585, Jan. 2006.

[6] S. Chen, S. A. Billings, and W. Luo, “Orthogonal least squares methods and their application to non-linear system identification,” *Int. J. Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[7] S. Chen, X. Hong, and C. J. Harris, “Sparse kernel regression modelling using combined locally regularized orthogonal least squares and D-optimality experimental design,” *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 1029–1036, Jun. 2003.

[8] S. Chen, X. Hong, C. J. Harris, and P. M. Sharkey, “Sparse modelling using orthogonal forward regression with PRESS statistic and regularization,” *IEEE Trans. Syst., Man Cybern., Part B*, vol. 34, no. 2, pp. 898–911, Apr. 2004.

[9] X. Hong *et al.*, “Model selection approaches for non-linear system identification: A review,” *Int. J. Syst. Sci.*, vol. 39, no. 10, pp. 925–946, Oct. 2008.

[10] E. Levin, “Hidden control neural architecture modeling of nonlinear time varying systems and its applications,” *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 109–116, Jan. 1993.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 9, pp. 533–536, Oct. 1986.

[12] P. J. Werbos, “Backpropagation through time: What it does and how to do it,” *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Mar. 1990.

[13] J. L. Elman, “Distributed representations, simple recurrent networks, and grammatical structure,” *Mach. Learn.*, vol. 7, no. 2, pp. 195–225, Sep. 1991.

[14] Y. Gao and M. J. Er, “NARMAX time series model prediction: Feedforward and recurrent fuzzy neural network approaches,” *Fuzzy Sets Syst.*, vol. 150, no. 2, pp. 331–350, Mar. 2005.

[15] M. Han, J. Xi, S. Xu, and F.-L. Yin, “Prediction of chaotic time series based on the recurrent predictor neural network,” *IEEE Trans. Signal Process.*, vol. 52, no. 12, pp. 3409–3416, Dec. 2004.

[16] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[17] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[18] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” 2014, *arXiv:1409.1259*.

[19] K. Cho *et al.*, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” 2014, *arXiv:1406.1078*.

[20] Y. Li *et al.*, “EA-LSTM: Evolutionary attention-based LSTM for time series prediction,” *Knowl.-Based Syst.*, vol. 181, pp. 1–8, Oct. 2019.

[21] H. Wan *et al.*, “CTS-LSTM: LSTM-based neural networks for correlated time series prediction,” *Knowl.-Based Syst.*, vol. 191, pp. 1–10, Mar. 2020.

[22] D. Hsu, “Time series forecasting based on augmented long short-term memory,” 2017, *arXiv:1707.00666*.

[23] X. Yuan, L. Li, and Y. Wang, “Nonlinear dynamic soft sensor modeling with supervised long short-term memory network,” *IEEE Trans. Ind. Informat.*, vol. 16, no. 5, pp. 3168–3176, May 2020.

[24] Z. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams, “Big data opportunities and challenges: Discussions from data analytics perspectives,” *IEEE Comput. Intell. Mag.*, vol. 9, no. 4, pp. 62–74, Nov. 2014.

[25] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, “Learning in nonstationary environments: A survey,” *IEEE Comput. Intell. Mag.*, vol. 10, no. 4, pp. 12–25, Oct. 2015.

[26] N. I. Sapankevych and R. Sankar, “Time series prediction using support vector machines: A survey,” *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, pp. 24–38, May 2009.

[27] J. L. Lobo *et al.*, “Evolving spiking neural networks for online learning over drifting data streams,” *Neural Netw.*, vol. 108, pp. 1–19, Dec. 2018.

[28] J. Liu and D.-S. Chen, “Nonstationary fault detection and diagnosis for multimode processes,” *AIChE J.*, vol. 56, no. 1, pp. 207–219, Jan. 2010.

[29] J. Shan, H. Zhang, W. Liu, and Q. Liu, “Online active learning ensemble framework for drifted data streams,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 486–498, Feb. 2019.

[30] I. Bogunovic, J. Scarlett, and V. Cevher, “Time-varying Gaussian process bandit optimization,” in *Proc. 19st Int. Conf. Artif. Intell. Statist.* (Cadiz, Spain), May 9–11, 2016, pp. 314–323.

[31] F. M. Nyikosa, *Adaptive Bayesian Optimization for Dynamic Problems*. Ph.D. thesis, Univ. Oxford, Oxford, U. K., 2018.

[32] S. Chen and S. Billings, “Recursive prediction error parameter estimator for non-linear models,” *Int. J. Control*, vol. 49, no. 2, pp. 569–594, 1989.

[33] G. B. Huang, Q. Y. Zhu, and C. K. Siew, “Extreme learning machine: Theory and applications,” *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, Dec. 2006.

[34] N. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, “A fast and accurate online sequential learning algorithm for feedforward networks,” *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.

[35] X. Wang and M. Han, “Online sequential extreme learning machine with kernels for nonstationary time series prediction,” *Neurocomputing*, vol. 145, pp. 90–97, Dec. 2014.

[36] X. Wang and M. Han, “Improved extreme learning machine for multivariate time series online sequential prediction,” *Eng. Appl. Artif. Intell.*, vol. 40, pp. 28–36, Apr. 2015.

[37] H. Chen, Y. Gong, X. Hong, and S. Chen, “A fast adaptive tunable RBF network for nonstationary systems,” *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 2683–2692, Dec. 2016.

[38] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control* (5th edition). Hoboken, NJ, USA: Wiley, 2015.

[39] E. S. Chng, S. Chen, and B. Mulgrew, “Gradient radial basis function networks for nonlinear and nonstationary time series prediction,” *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 190–194, Jan. 1996.

[40] H. Akaike, “A new look at the statistical model identification,” *IEEE Trans. Autom. Control*, vol. AC-19, no. 6, pp. 716–723, Dec. 1974.

[41] D. Lowe, “Adaptive radial basis function nonlinearities, and the problem of generalisation,” in *Proc. 1st IEE Int. Conf. Artif. Neural Netw.*, Oct. 16-18, 1989, pp. 171–175.

[42] S. Haykin, *Neural Networks and Learning Machines* (3rd Edition). Upper Saddle River, NJ, USA: Prentice Hall, 2009.

[43] S. Kitayama and K. Yamazaki, “Simple estimate of the width in Gaussian kernel with adaptive scaling technique,” *Appl. Soft Comput.*, vol. 11, no. 8, pp. 4726–4737, Dec. 2011.

[44] S. Chen, X. X. Wang, and D. J. Brown, “Orthogonal least squares regression with tunable kernels,” *Electron. Lett.*, vol. 41, no. 8, pp. 484–486, Apr. 2005.

[45] S. Chen, X. X. Wang, and D. J. Brown, “Sparse incremental regression modeling using correlation criterion with boosting search,” *IEEE Signal Process. Lett.*, vol. 12, no. 3, pp. 198–201, Mar. 2005.

[46] S. Chen, X. Hong, C. J. Harris, and X. X. Wang, “Identification of nonlinear systems using generalized kernel models,” *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 3, pp. 401–411, May 2005.

[47] S. Chen, X. X. Wang, and C. J. Harris, “NARX-based nonlinear system identification using orthogonal least squares basis hunting,” *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 1, pp. 78–84, Jan. 2008.

[48] S. Chen, X. Hong, B. L. Luk, and C. J. Harris, “Non-linear system identification using particle swarm optimisation tuned radial basis function models,” *Int. J. Bio-Inspired Comput.*, vol. 1, no. 4, pp. 246–258, 2009.

[49] S. Chen, X. Hong, B. L. Luk, and C. J. Harris, “Construction of tunable radial basis function networks using orthogonal forward selection,” *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 39, no. 2, pp. 457–466, Apr. 2009.

[50] S. Chen, X. Hong, and C. J. Harris, “Particle swarm optimization aided orthogonal forward regression for unified data modelling,” *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 477–499, Aug. 2010.

[51] S. Grossberg, “Nonlinear neural networks: Principles, mechanisms, and architectures,” *Neural Netw.*, vol. 1, no. 1, pp. 17–61, 1988.

[52] H. Chen, Y. Gong, and X. Hong, “Online modeling with tunable RBF network,” *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 935–947, Jun. 2013.

[53] F. Ding and T. Chen, “Performance analysis of multi-innovation gradient type identification methods,” *Automatica*, vol. 43, no. 1, pp. 1–14, Jan. 2007.

[54] K. Lüdge (editor), *Nonlinear Laser Dynamics: From Quantum Dots to Cryptography*. Weinheim, Germany: Wiley, 2012.

[55] E. N. Lorenz, “Deterministic nonperiodic flow,” *J. Atmospheric Sci.*, vol. 20, no. 2, pp. 130–141, Mar. 1963.

[56] H. Tong, *Threshold Models in Non-Linear Time Series Analysis*. New York, NY, USA: Springer, 1983.

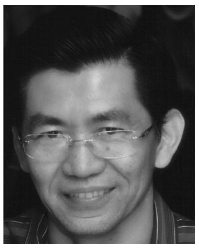
[57] A. Miranian and M. Abdollahzade, “Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 207–218, Feb. 2013.

[58] Y. Li, W. Cui, Y. Guo, T. Huang, X. Yang, and H. Wei, “Time-varying system identification using an ultra-orthogonal forward regression and multiwavelet basis functions with applications to EEG,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 2960–2972, Jul. 2018.

[59] Y. Li, H.-L. Wei, S. A. Billings, and P. G. Sarrigiannis, “Identification of nonlinear time-varying systems using an online sliding-window and common model structure selection (CMSS) approach with applications to EEG,” *Int. J. Syst. Sci.*, vol. 47, no. 11, pp. 2671–2681, 2016.

[60] R. G. Andrzejak *et al.*, “Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state,” *Physical Rev. E*, vol. 64, no. 6, pp. 061907-1–061907-8, Nov. 2001.

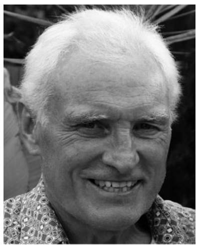
Tong Liu received the B.Sc. degree in automation from the College of Automation, Chongqing University, Chongqing, China, in 2016, where he is currently working toward the Ph.D. degree in control theory and control engineering. From September 2018 to September 2019, he was a Visiting Ph.D. Student with the School of Electronics and Computer Science, University of Southampton, Southampton, U.K. His current research interests include online learning, system identification, neural networks, machine learning, and intelligent control system design.



Sheng Chen (Fellow, IEEE) received the B.Eng. degree from the East China Petroleum Institute, Dongying, China, in 1982, and the Ph.D. degree from the City University, London, in 1986, both in control engineering. He received the higher doctoral degree, Doctor of Sciences (D.Sc.), from the University of Southampton, Southampton, U.K. From 1986 to 1999, he held research and academic appointments with the Universities of Sheffield, Edinburgh and Portsmouth, all in U.K. Since 1999, he has been with the School of Electronics and Computer Science, the University of Southampton, U.K., where he holds the post of Professor in Intelligent Systems and Signal Processing. He has authored more than 650 research papers. He has 14,100+ Web of Science citations with h-index 53, and 29,500+ Google Scholar citations with h-index 75. His research interests include neural network and machine learning, wireless communications, and adaptive signal processing. Dr. Chen is a Fellow of the United Kingdom Royal Academy of Engineering, a Fellow of IET, a Distinguished Adjunct Professor with King Abdulaziz University, Jeddah, Saudi Arabia, and an original ISI highly cited Researcher in engineering (March 2004).

Shan Liang (Member, IEEE) received the M.Sc. degree in control science and engineering from the College of Automation, Chongqing University, Chongqing, China, in 1995, and the Ph.D. degree from the Department of Mechanical Systems Engineering, Kumamoto University, Kumamoto, Japan, in 2004. His current research interests include numerical modeling, electromagnetic theory, nonlinear systems, adaptive control, and sensor networks.

Shaojun Gan received the B.Eng. degree in automation from Huainan Normal University, Huainan, China, in 2010, and the Ph.D. degree in control engineering from the School of Automation, Chongqing University, Chongqing, China, in 2016. From 2014 to 2016, he was a Visiting Ph.D. Student with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, U.K. From 2017 to 2019, he was a Research Fellow with the School of Electronic and Electrical Engineering, University of Leeds, U.K. He is currently an Assistant Professor with the College of Metropolitan Transportation, Beijing University of Technology, Beijing, China. His main research interests include system modeling and machine learning methods, with the applications to manufacturing energy systems and intelligent transportation systems.



Chris J. Harris received the B.Sc. and M.A. degrees from the University of Leicester, Leicester, U.K. and the University of Oxford, Oxford, U.K., respectively, and the Ph.D. degree from the University of Southampton, Southampton, U.K, in 1972. He received the higher doctoral degree, the Doctor of Sciences (D.Sc.), from the University of Southampton in 2001. He is Emeritus Research Professor with the University of Southampton, having previously held senior academic appointments with Imperial College, Oxford and Manchester Universities, as well as Deputy Chief Scientist for the U.K. Government. He is the coauthor of more than 500 scientific research papers during a 50 year research career. Dr. Harris was awarded the IEE Senior Achievement Medal for Data Fusion research and the IEE Faraday Medal for distinguished international research in Machine Learning. He was elected to the UK Royal Academy of Engineering in 1996.