A New Discrete-Continuous Algorithm for Radial Basis Function Networks Construction

Long Zhang, Student Member, IEEE, Kang Li, Senior Member, IEEE, Haibo He, Senior Member, IEEE, and George W. Irwin, Fellow, IEEE

Abstract—The construction of a radial basis function (RBF) network involves the determination of the model size, hidden nodes, and output weights. Least squares-based subset selection methods can determine a RBF model size and its parameters simultaneously. Although these methods are robust, they may not achieve optimal results. Alternatively, gradient methods are widely used to optimize all the parameters. The drawback is that most algorithms may converge slowly as they treat hidden nodes and output weights separately and ignore their correlations. In this paper, a new discrete-continuous algorithm is proposed for the construction of a RBF model. First, the orthogonal least squares (OLS)-based forward stepwise selection constructs an initial model by selecting model terms one by one from a candidate term pool. Then a new Levenberg–Marquardt (LM)-based parameter optimization is proposed to further optimize the hidden nodes and output weights in the continuous space. To speed up the convergence, the proposed parameter optimization method considers the correlation between the hidden nodes and output weights, which is achieved by translating the output weights to dependent parameters using the OLS method. The correlation is also used by the previously proposed continuous forward algorithm (CFA). However, unlike the CFA, the new method optimizes all the parameters simultaneously. In addition, an equivalent recursive sum of squared error is derived to reduce the computation demanding for the first derivatives used in the LM method. Computational complexity is given to confirm the new method is much more computationally efficient than the CFA. Different numerical examples are presented to illustrate the effectiveness of the proposed method. Further, Friedman statistical tests on 13 classification problems are performed, and the results demonstrate that RBF networks built by the new method are very competitive in comparison with some popular classifiers.

Index Terms—Forward stepwise selection, Levenberg–Marquardt (LM), model generalization, orthogonal least squares (OLS), radial basis function (RBF) networks.

I. INTRODUCTION

SINCE the radial basis function (RBF) is first introduced into neural networks design by Broomhead and Lowe [1], RBF neural networks are widely studied and used in system identification, regression, and classification [2], [3]. A RBF network is a nonlinear mapping in which each basis function maps a multivariable input to a scalar value [4]. It is proved that a RBF network is capable of universal approximation and thus can approximate any multivariate continuous function if a sufficient number of RBF nodes are provided [5], [6]. Its whole structure is a single hidden layer type of artificial neural networks. For a RBF network, the adjustable parameters are hidden layer parameters (centers and widths) and output weights. If hidden layer parameters have fixed values and can be determined in advance, then the output weight parameters can be obtained using least squares methods. If all the parameters are to be adjusted in the continuous space, it becomes a nonlinear optimization problem. Another issue is to determine the model size (the number of hidden layer nodes), based on the parsimony principle where a model with good generalization performance and fewer parameters is always preferred [7]. Parameter adjustment and model size determination are the two key issues for constructing a RBF network.

A simple approach for training a RBF network is to consider every training data as a center, this however can lead to over-fitting and ill-conditioned problems. Another conventional approach is to randomly select the centers and widths with a predetermined model size, and then use least squares to estimate the output weights [1], [8]. Such approaches cannot produce a parsimonious network with desirable generalization performance. Further, ill-conditioned problems may occur because of the linear dependency caused by highly correlated variables [2].

An extensively researched approach is the hybrid method combining both unsupervised and supervised learning introduced by Moody and Darken [9], where the unsupervised self-organizing clustering method is used to determine the centers and widths in the hidden layer nodes, and then the supervised least squares method is used to update the output weights. The clustering method partitions and constructs homogeneous datasets according to the distance similarity among them. This can effectively reduce the number of the hidden layer nodes. However, the drawback of the hybrid method is that the unsupervised scheme does not have effect on the formation of the prototypes or does not fully utilize the information.
to control the agglomeration process [11]. A supervised clustering method is an agglomerative clustering algorithm that merges redundant cluster points and then use cluster means as the centers, but it has an additional clustering parameter to control the agglomeration process [11]. A supervised hierarchical clustering method is employed by Lee for RBF networks construction [12], where the construction of neural networks starts with one hidden layer node and additional nodes are added to the network when they are desired [13]. However, all these clustering-based methods may not be able to produce an optimal or parsimonious network.

If the nonlinear parameters, i.e., widths and centers in the hidden layer nodes are predetermined, the construction of a RBF network is to select a representative subset from the candidate term pool. Subset selection algorithms are widely studied and used for model selection or variable selection. Subset selection algorithms, such as the orthogonal least squares (OLS) methods and the fast recursive algorithm (FRA) [14], are useful in determining the network size, selecting the hidden layer nodes and estimating the output weights simultaneously. They mainly include forward selection and backward selection. The former gradually builds a model by adding one term that gives the largest decrease in the cost function at a time while the latter involves starting with all candidate terms and deleting one term which is least significant in terms of the cost function at a time [15], [16]. The forward selection is computationally much more efficient than the backward selection. However, neither of them could guarantee a global optimal solution. To improve model compactness and generalization performance, a two-stage algorithm is proposed by Li for re-evaluating the initial results produced by forward selection [17]. It firstly selects an initial model and then reviews the significance of each term in it, replacing the insignificant ones. The process continues until no further reduction in the cost function is possible by replacing any term, thereby improving model compactness and accuracy significantly [17]. More recently, a new two-stage algorithm using OLS which is computationally even more efficient than the original FRA-based two-stage algorithm is proposed [18]. However, existing subset selection methods and their variants are based on a candidate term pool of hidden nodes and the optimal values for the candidate terms may not be included. Hence, these methods may not produce an optimal model.

To optimize all the parameters in the RBF networks, evolutionary and gradient-based methods are useful technology. Unlike conventional calculus-based methods, evolutionary algorithms are stochastic optimization techniques. A significant advantage is that they may escape local minima and converge to global minimum [19]. In [20] and [21], genetic algorithms (GAs) were used for RBF network construction, and the results show that the GA-based method can outperform forward selection and clustering methods. However, the disadvantages are that it may cause over-fitting and has a high computational overhead. To build a sparse RBF networks, Mao proposed to combine the GA with forward selection method [22]. More recently, a new evolutionary algorithm, particle swarm optimization, is integrated into forward OLS selection and the two-stage algorithm to produce more sparse neural models with reduced model size [23], [24]. However, the drawbacks of evolutionary methods are also identified. First, some of these methods are computational very demanding, which prohibits their wide applications. Secondly, the analysis of their convergence is still problematic, which means the convergence cannot always be guaranteed [17]. Finally, the results cannot be repeated due to their stochastic nature.

Gradient-based methods like gradient descent, Gauss–Newton, Levenberg–Marquardt (LM), and Newton are useful nonlinear optimization algorithms [3], [10], [25], [26]. The gradient descent, Gauss–Newton, and LM do not explicitly compute the Hessian matrix (second derivatives matrix), but approximate it by an identity matrix, normal equation of the Jacobian matrix (first derivatives matrix) and regularized normal equation of the Jacobian matrix, respectively, and only the Newton method computes the Hessian matrix explicitly. Among them, LM tends to be the most powerful and widely used approach as it combines the good local convergence property of the Newton method when the initial guess is near the solution and the consistent error decrease property when the initial guess is far from the solution [25], [27]. However, these algorithms have a high computational overhead and they often converge very slowly as they treat hidden layer nodes and output weights separately and ignore their correlation. To simplify the computational complexity and speed up the convergence, an alternative continuous forward algorithm (CFA) is proposed recently and it is a combination of conjugate gradient algorithm and the least squares method [28], [29]. More specifically, firstly, the output weights are converted into dependent parameters on the hidden layer parameters by using the FRA least squares approach [14], and then the hidden layer parameters with dependent output weights are optimized by the conjugate gradient algorithm in the continuous space, followed by updating its corresponding output weight again by the least squares method. The CFA method is a piecewise algorithm as it increases and optimizes one hidden node at a time, repeating this until the model performance is satisfied. Its advantage is that the model size and model parameters can be determined simultaneously, but its parameters tend to be not optimal as the previously optimized hidden layer nodes are fixed while optimizing the new hidden node. In other words, it does not optimize all the hidden layer parameters simultaneously but instead optimize one hidden node each time.

It should be noted that if large noise presents in the modeling data, all these above mentioned methods may suffer from the over-fitting problems when they only aim to minimize the sum squared error (SSE). Thus, a resulting model with a large number of hidden nodes is generally too complex. To avoid over-fitting, regularization methods are among the most popular approaches. Most regularization methods adopt an extra penalty term to the sum squares errors and typical regularization methods include ridge regression and least absolute shrinkage and selection operator (LASSO), which use $l_2$ norm and $l_1$ norm penalties on model output weights, respectively [30]. The regularized forward selection using ridge regression is introduced for RBF construction by Orr
and later a fast implementation of regularized forward OLS selection is given by Chen [31]. More recently, the ridge regularization is extended into forward selection and two-stage (forward and backward) subset selection algorithm for building a compact network [32], [33]. Though all these methods can avoid over-fitting problems, the aforementioned drawbacks of subset selection still exist.

In this paper, a new discrete-continuous method based on OLS and LM methods is introduced to construct a compact RBF network, and it provides an alternative solution to the tradeoff between the performance of subset selection methods and the convergence of the gradient-based methods. The proposed method is a two-stage algorithm. In the first stage (initial model construction), the OLS-based subset selection method is employed to build an initial model, in which the model size and the initial hidden layer parameters are determined preliminarily. In the second stage, all the parameters are further optimized by the LM method. This paper has four main contributions.

1) The parameter optimization not only considers the parameter correlations by transforming the output weights to dependent parameters on hidden nodes, but also optimizes all the parameters simultaneously, a distinctive difference from the early proposed CFA that optimizes only one hidden node each time.

2) A recursive SSE is developed for effective and efficient computation of both the stepwise forward selection criterion and the associated first derivatives. Computational complexity analysis confirms its superiority to the CFA in terms of computational time.

3) The ridge and LASSO regularization techniques are incorporated into this new discrete-continuous method to avoid over-fitting problems.

4) A coarse-to-fine scheme for tuning the regularized parameters is proposed to find a suitable value automatically. Compared with the conventional scheme, this new method does not need set an initial guess manually or tune the step size by try-and-error.

Finally, two numerical examples on regression problems show that the new method outperforms the OLS and CFA methods and needs less computational time than the CFA. Further, the Friedman statistical test on 13 classification problems shows that the RBF networks built by the new method are very competitive with some popular classifiers including regularization AdaBoost (ABR), support vector machine (SVM), and kernel Fisher discriminant (KFD).

II. PROBLEM FORMULATION AND PRELIMINARIES

A. RBF Networks

The RBF network structure is a linear combination of nonlinear basis function and it is a single hidden layer network. When it is used for system identification, regression and classification, its mathematical formula is the same and the only difference lies in the model inputs and outputs. In the following, the mathematical formulation is first introduced, and then the differences for applications are detailed.

A RBF network can be formulated as [28]

\[
y(t) = \sum_{i=1}^{M} p_i(x(t), d_i, s_i) \theta_i + \zeta(t)
\]

where \(x(t)\) and \(y(t)\) are the model input and output variables, \(x(t) = [x_1(t), x_2(t), \ldots, x_q(t)]\) is of assumed known dimension of \(q\), here \(t = 1, 2, \ldots, N\), \(N\) being the size of the training dataset. \(p_i(x(t), d_i, s_i)\) denotes the RBF network function of the \(i\)th hidden layer node with the width \(d_i \in \mathbb{R}\) and centers \(s_i \in \mathbb{R}^q\), \(i = 1, 2, \ldots, M\), \(M\) being the total number of RBF nodes. \(\theta_i\) is the output weight parameter. \(\zeta(t)\) is a model residual sequence. For a RBF network, all adjustable variables include the hidden layer parameters and output weights. All the hidden layer parameters can be expressed as follows:

\[
V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{bmatrix} = \begin{bmatrix} d_1 & s_{11} & s_{12} & \ldots & s_{1q} \\ d_2 & s_{21} & s_{22} & \ldots & s_{2q} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_M & s_{M1} & s_{M2} & \ldots & s_{Mq} \end{bmatrix}
\]

where the vector \(v_i\) includes the \(i\)th hidden layer node parameters. \(d_i\) and the set \([s_{i1}, \ldots, s_{iq}]\) are the width and centers of the \(i\)th hidden layer node, respectively. For brevity, \(v_i = [d_i, s_{i1}, \ldots, s_{iq}]\) is redefined as \(v_i = [v_{i1}, \ldots, v_{ir}]\) where \(r = q + 1\). A schematic of the RBF network is shown in Fig. 1.

For nonlinear system modeling, the model output often represents the system output but the model input is often given by the following:

\[
x(t) = [y(t-1), \ldots, y(t-n_y), u_1(t-1), \ldots, u_1(t-n_{u_1}), \ldots, u_r(t-1), \ldots, u_r(t-n_{u_r})]
\]

where \(u_1(t), \ldots, u_r(t)\) and \(y(t)\) are the real system input and output, respectively. Their largest lagged input and output are \(n_{u_1}, \ldots, n_{u_r}\) and \(n_y\), respectively. The RBF networks with these input and output variables are also referred to as nonlinear autoregressive with exogenous input model [34], which is widely used for nonlinear system identification, time-series prediction and regression.
For binary classification and discriminant problems, the model input \( \mathbf{x}(t) \in \mathbb{R}^q \) represents the \( q \)-dimensional pattern vector and \( y(t) \in \{-1, +1\} \) denotes the class label for \( \mathbf{x}(t) \). The training process has no differences from nonlinear system modeling. However, the classification test or prediction is different, the model output needs to be mapped to \( \{-1, +1\} \). That is
\[
\hat{y}(t) = \text{sgn}(\tilde{y}(t)) = \begin{cases} 
-1, & \tilde{y}(t) \leq 0 \\
+1, & \tilde{y}(t) > 0
\end{cases}
\]
where
\[
\tilde{y}(t) = \sum_{i=1}^{M} p_i(\mathbf{x}(t), d_i, s_i) \theta_i
\]
and the \( \mathbf{x}(t) \) is one test pattern vector.

When training a RBF network, the choice of RBF should be considered first. The value of a RBF only relies on the Euclidean distance of the argument from the origin point and any rotations thereof make no difference to the RBF value [35]. Popular basis functions include thin-plate-spline, multiquadric, inverse multiquadric, and Gaussian functions, which are defined as follows:
\[
\begin{align*}
p(\mathbf{x}, d, s) &= \delta^2 \log(\delta/d) \\
p(\mathbf{x}, d, s) &= (\delta^2 + d^2)^{1/2} \\
p(\mathbf{x}, d, s) &= (\delta^2 + d^2)^{-1/2} \\
p(\mathbf{x}, d, s) &= \exp(-\delta^2/d^2)
\end{align*}
\]
respectively, where \( \delta = \| \mathbf{x} - \mathbf{s} \| = \| (\mathbf{x} - \mathbf{s})^T (\mathbf{x} - \mathbf{s}) \|^{1/2} \) is the Euclidean distance between the input variables \( \mathbf{x} \) and centers \( \mathbf{s} \), and \( d \) is the width.

The thin-plate-spline is introduced for multivariate interpolation problem and it has a physical background for bending energy minimization associated with warping a thin elastic plate. The thin-plate-spline is a global basis function and performs well on the interpolation problems. Multiquadric and its inverse are originally employed as global basis functions for topographical mappings and surfaces. Their main difference is that the multiquadric function increases monotonically and its inverse decreases. Furthermore, the multiquadric is quite stable with the parameter width while the reverse multiquadric is sensitive to the parameter width. Although they are different in forms, they can work equally well for some problems. The Gaussian function has a symmetric bell curve that quickly falls off toward infinity [36], [37]. Theoretical work has proved that these basis functions can approximate any continuous function if the sufficient hidden layer nodes are given. However, there is no further theoretical framework to guide how to select the basis function and numerical examples could not give a consistent suggestion as none of them can be dominatingly better than others. In general, the Gaussian function is the most widely known and well-studied basis function.

Once the basis function was chosen, the initialization for centers \( \mathbf{s} \) and widths \( d \) shown in (2) has to be given. Generally, the centers \( \mathbf{s} \) are initialized by the input data samples, namely \( \mathbf{s}_i = \mathbf{x}(i), i = 1, \ldots, N \). The width is generally chosen as a fixed value for all initial widths \( d \) [2]. The initial value can be set as some multiple of the average distance between centers [38].

### B. SSE and Regularization Methods

Equation (1) can be expressed in the matrix form as follows:
\[
y = \mathbf{P} \Theta + \Xi
\]
where \( y = [y(1), \ldots, y(N)]^T \) is a \( N \)-by-1 output vector, \( \mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_M] \) is a \( N \)-by-\( M \) matrix, here \( \mathbf{p}_i = [p_i(\mathbf{x}(1), v_i), \ldots, p_i(\mathbf{x}(N), v_i)]^T \) is formulated by the \( i^{th} \) hidden layer parameter vector \( \mathbf{v}_i \). \( \Theta = [\theta_1, \ldots, \theta_M]^T \) is the output weights and \( \Xi = [\zeta(1), \ldots, \zeta(N)]^T \) is the residual vector. The RBF network modeling aims to minimize the SSE
\[
\Xi^T \Xi = (y - \mathbf{P} \Theta)^T (y - \mathbf{P} \Theta)
\]
by optimizing all unknown parameters \( \mathbf{v}_i \)s and \( \Theta \)s.

If the modeling data has a large noise, the over-fitting problem may occur if only minimizing the SSE. To avoid over-fitting, regularization techniques use additional penalty term to smooth the modeling curve surface or classification boundaries. Though this leads to a biased model, it attempts to impose the principle of parsimony on the solution [39]. The cost function for regularization is given by the following:
\[
\Xi^T \Xi + \lambda C
\]
where the parameter \( \lambda \) controls the fitting smoothness and the model size (the number of hidden nodes). The penalty terms for ridge regression and LASSO are given by
\[
C = \| \Theta \|_2^2 \\
C = \| \Theta \|_1
\]
respectively, and \( \| \Theta \|_2^2 = \sum_{i=1}^{M} \theta_i^2 \) is the \( l_2 \) norm of \( \Theta \) and \( \| \Theta \|_1 = \sum_{i=1}^{M} | \theta_i | \) is the \( l_1 \) norm of \( \Theta \). An equivalent way to describe the ridge regression is given as follows [40]:
\[
\min \| \Xi \|^2 \text{ subject to } \sum_{i=1}^{M} \theta_i^2 < t_r
\]
while the LASSO can also be reformulated as
\[
\min \| \Xi \|^2 \text{ subject to } \sum_{i=1}^{M} | \theta_i | < t_l
\]
where \( t_r \) and \( t_l \) are tuning parameters that control the shrinkage of the output weights.

### C. OLS Method

If widths and centers in the hidden layer nodes have fixed values, the RBF network model can then be represented by a linear-in-the-parameters model. This can be solved by forward stepwise selection methods. The forward stepwise selection using OLS method selects a subset model
\[
\mathbf{P}_{m} = [\mathbf{p}_1, \ldots, \mathbf{p}_m]
\]
from the full model
\[
\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_M]
\]
in a forward stepwise manner by choosing one model term each time for which the model error is maximally reduced, where \( m \) is the number of the subset model terms.
The forward stepwise selection using the OLS method involves a series of the orthogonal decompositions. Each decomposition, the $M^{th}$ say, is given by [41]

$$P = WA$$

where $A$ is a $M \times M$ triangular matrix

$$A = \begin{bmatrix}
1 & a_{12} & \cdots & a_{1M} \\
0 & 1 & \cdots & a_{2M} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1
\end{bmatrix}$$

and $W = [w_1, \ldots, w_M]$ is a $N \times M$ matrix with orthogonal columns $w_i$

$$W = \begin{bmatrix}
w_{11} & w_{12} & \cdots & w_{1M} \\
w_{21} & w_{22} & \cdots & w_{2M} \\
\vdots & \vdots & \ddots & \vdots \\
w_{N1} & w_{N2} & \cdots & w_{NM}
\end{bmatrix}$$

which satisfies

$$W^T W = \text{diag}(w_1^T w_1, \ldots, w_M^T w_M).$$

The orthogonal decomposition involved in the OLS method can be carried out by the Gram-Schmidt algorithm. It computes one column of $A$ at a time and factorizes $P$ as follows:

$$w_1 = p_1$$

$$a_{ik} = \frac{\langle w_i, p_k \rangle}{\langle w_i, w_i \rangle}, \quad 1 \leq i < k$$

$$w_k = p_k - \sum_{i=1}^{k-1} a_{ik} w_i$$

$$k = 2, \ldots, M.$$  (23)

The model (10) can thus be expressed as follows:

$$y = (PA^{-1})(A \Theta) + \Xi = Wg + \Xi$$

where $g = [g_1, g_2, \ldots, g_M]^T = A \Theta$ is the orthogonal weight vector. The original model weight vector $g$ can then be calculated by [34]

$$g = (W^T W)^{-1} W^T y - (W^T W)^{-1} W^T \Xi$$

and the sum squares of $y$ is given by the following:

$$y^T y = g^T W^T W g + \Xi^T \Xi + \Xi^T W g.$$  (26)

As $\Xi$ is uncorrelated with $P$ and all stochastic processes of interest are ergodic, then $W^T \Xi = 0$ [34]. Hence, (25) and (26) are simplified to

$$g = (W^T W)^{-1} W^T y$$  (27) and

$$y^T y = g^T W^T W g + \Xi^T \Xi$$  (28)

respectively. Using the property of $W$ shown in (22), the SSE can be transformed into

$$\Xi^T \Xi = y^T y - g^T W^T W g$$

$$= y^T y - \sum_{j=1}^{M} g_j^2 < w_j, w_j >$$  (29)

which is widely used as a cost function in subset selection [34], [41].

III. NEW RECURSIVE SSE

This section presents a new equivalent SSE and its recursive form. Substituting $g$ in (24) and (28) using (27) gives

$$\Xi = y - W(W^T W)^{-1} W^T y$$

and

$$\Xi^T \Xi = y^T y - y^T W(W^T W)^{-1} W^T y.$$  (30)

It can be seen that the SSE is only determined by the orthogonal basis $W$ of the model terms $P$. Further, the new SSE has a recursive form

$$\Xi^T \Xi = y^T y - \sum_{j=1}^{M} \frac{(w_j^T y)^2}{w_j^T w_j}.$$  (32)

In particular, the following can be established:

$$W(W^T W)^{-1} W^T = \sum_{j=1}^{M} \frac{w_j w_j^T}{w_j^T w_j}$$  (33)

where $W(W^T W)^{-1} W^T \in \mathbb{R}^{N \times N}$, $w_j w_j^T \in \mathbb{R}^{N \times N}$, and $w_j^T w_j \in \mathbb{R}$.

Proof: To prove (33), firstly define a recursive matrix $R_M \in \mathbb{R}^{N \times N}$ given by the following:

$$R_M = W(W^T W)^{-1} W^T$$

where $W = [w_1, \ldots, w_i, \ldots, w_M]$ and $w_i = [w_i1, \ldots, w_iN]^T$. Because of the orthogonal basis property

$$W^T W = \text{diag}(w_1^T w_1, \ldots, w_M^T w_M)$$

then $R_M$ can be reformulated as follows:

$$R_M = \begin{bmatrix}
w_1^T w_1 & \cdots & w_M^T w_1 \\
w_1^T w_2 & \cdots & w_M^T w_2 \\
\vdots & \ddots & \vdots \\
w_1^T w_M & \cdots & w_M^T w_M
\end{bmatrix} [w_1, \ldots, w_M]^T.$$  (36)

Denote $r_{ij}$, $i = 1, \ldots, N$, $j = 1, \ldots, M$ is the element of the matrix $R_M$ in the $i$th row and $j$th column, hence

$$r_{ij} = \begin{bmatrix}
\frac{w_{i1}^T w_{j1}}{w_j^T w_j} & \cdots & \frac{w_{i1}^T w_{jM}}{w_j^T w_j} \\
\frac{w_{i2}^T w_{j1}}{w_j^T w_j} & \cdots & \frac{w_{i2}^T w_{jM}}{w_j^T w_j} \\
\vdots & \ddots & \vdots \\
\frac{w_{iM}^T w_{j1}}{w_j^T w_j} & \cdots & \frac{w_{iM}^T w_{jM}}{w_j^T w_j}
\end{bmatrix} [w_{j1}, \ldots, w_{jM}]^T$$

$$= \sum_{h=1}^{M} \frac{w_{ih} w_{jh}}{w_j^T w_j}.$$  (37)

The right side of the formula is equal to the element of $\sum_{j=1}^{M} (w_j w_j^T / w_j^T w_j)$ in the $i$th row and $j$th column, hence, (33) holds.

Compare the conventional SSE in (29) and the recursive SSE in (32), they are numerically equal. However, the conventional SSE is the function of both $W$ and $g$ while the new one is the function of only $W$. The advantages of introducing the new SSE are as follows: first, the contribution of the $j$th term $w_j$ to the SSE is $(w_j^T y)^2/(w_j^T w_j)$, and it can be directly used for selecting terms one by one via subset selection procedure,

This section has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.
IV. PROPOSED ALGORITHM

The proposed discrete-continuous method first builds an initial model by forward stepwise selection using the OLS method, which determines the model size and preliminary values for all the parameters, and then employs the LM and OLS methods to further optimize the parameters of RBF networks in the continuous space. When large noise presents in the data, the ridge and LASSO regularization techniques can be easily incorporated into the proposed parameters optimization framework to prevent over-fitting.

A. Initial Model Construction

The algorithm first selects the RBF node from the candidate set that gives the largest net contribution, then repeatedly selects the nodes from the remainder of the set that best fits the data while combining with the previously selected terms [2], [42]. Based on (32), the net contribution of a node is defined as follows:

\[
[NC]_j = \frac{(w_T^j y)^2}{w^T_j w_j}.
\]  (38)

The SSE with \( M \) hidden nodes can also be expressed as follows:

\[
\Xi^T \Xi = y^T y - \sum_{j=1}^{M} [NC]_j.
\]  (39)

Given the SSE criterion, the subset selection procedure using the Gram–Schmidt orthogonal algorithm can be summarized as follows.

At the first step, for \( 1 \leq i \leq M \), calculate

\[
w^{(i)}_1 = p_i
\]

\[
[NC]_1^{(i)} = \frac{(w^{(i)}_1 y)^2}{w^T_1 w_1}.
\]  (40)

The node with the largest net contribution is then found according to

\[
[NC]^{(i)}_1 = \max([NC]_1^{(i)}, 1 \leq i \leq M)
\]  (41)

and the node associated with index \( i_1 \) is selected

\[
w_1 = w^{(i_1)}_1 = p_{i_1}.
\]  (42)

At the \( k \)-th step, for \( 1 \leq i \leq M, i \neq i_1, \ldots, i \neq i_{k-1} \), the following are calculated:

\[
\alpha^{(i)}_{jk} = \frac{w_j p_i}{w^T_j w_j}, 1 \leq j < k
\]

\[
w^{(i)}_k = p_i - \sum_{j=1}^{k-1} \alpha^{(i)}_{jk} w_j
\]  (43)

\[
[NC]_k^{(i)} = \frac{(w^{(i)}_k y)^2}{w^T_k w_k}.
\]  (44)

The node with largest net contribution from the remainder of the candidate node pool is again found using

\[
[NC]_k^{(i)} = \max([NC]_k^{(i)}, i \neq i_1, \ldots, i_{k-1})
\]  (45)

and the node associated with the index \( i_k \) is chosen

\[
\alpha_{jk} = \alpha^{(i_k)}_{jk}, 1 \leq j < k
\]

\[
w_k = w^{(i_k)}_k = p_{i_k} - \sum_{j=1}^{k-1} \alpha_{jk} w_j
\]  (46)

This procedure terminates at the \( m \)-th step when

\[
y^T y - \sum_{j=1}^{m} [NC]_j < \rho
\]  (47)

where \( \rho \) is some preset tolerance. Alternatively, the cross validation criterion [43] or the information-based statistical criterion, such as Akaike information criterion [44], Bayesian information criterion [45] and final prediction error [46], is satisfied. A subset of nodes \( \mathbf{P}_m = [p_1, \ldots, p_m] \) and its orthogonal basis \( \mathbf{W}_m = [w_1, \ldots, w_m] \) are selected. For simplicity, the indexes are reordered as follows:

\[
\mathbf{P}_m = [p_1, p_2, \ldots, p_m]
\]  (48)

and then the initial RBF model is constructed as follows:

\[
y = \sum_{i=1}^{m} p_i \theta_i
\]  (49)

where the weight \( \theta_i \) can be computed using backward elimination

\[
\mathbf{g} = [g_1, \ldots, g_m] = (\mathbf{W}_m^T \mathbf{W}_m)^{-1} \mathbf{W}_m^T \mathbf{y}
\]

\[
\mathbf{\theta}_m = \mathbf{g}_m
\]  (50)

\[
\mathbf{\theta}_i = g_i - \sum_{k=i+1}^{m} \alpha_{ik} \mathbf{\theta}_k, i = m-1, \ldots, 1
\]  (51)

where \( \mathbf{W}_m^T \mathbf{W}_m = \operatorname{diag}[w_1^T w_1, \ldots, w_M^T w_M] \) is a diagonal matrix and its inversion matrix is equal to \( (\mathbf{W}_m^T \mathbf{W}_m)^{-1} = \operatorname{diag}(1/w_1^T w_1, \ldots, 1/w_M^T w_M) \).

So far the forward stepwise selection method is used to build a RBF model with certain size and parameters (centers, widths, and output weights). In details, the hidden layer parameters (centers and weights) and output weights are given by the following:

\[
\mathbf{V}_m = \begin{bmatrix}
\mathbf{v}_1 \\
\mathbf{v}_2 \\
\vdots \\
\mathbf{v}_m \\
\end{bmatrix} = \begin{bmatrix}
d_1 & s_{11} & s_{12} & \ldots & s_{1q} \\
d_2 & s_{21} & s_{22} & \ldots & s_{2q} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{d}_m & s_{m1} & s_{m2} & \ldots & s_{mq} \\
\end{bmatrix}
\]  (52)

and

\[
\mathbf{\Theta}_m = [\theta_1, \theta_2, \ldots, \theta_m]
\]  (53)

respectively. To improve the model generalization ability, this paper further investigates the parameter optimization.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.
B. Parameter Optimization

The initial model shown in (48) can be expressed in the matrix form as follows:
\[ Y = P_m(V_m)\Theta_m. \]  
(52)

Parameter optimization involves tuning the hidden layer parameters \( V_m \) in (50) and output weights \( \Theta_m \) in (51) to further minimize the SSE cost function. Generally, the parameters \( V_m \) and \( \Theta_m \) are interconnected. If parameters in \( V_m \) are changed, the \( \Theta_m \) should be adjusted accordingly. The conventional LM method neglects the correlation elements, and treats \( V_m \) and \( \Theta_m \) separately when computing derivatives. This can simplify the computation, however, may also lead to slow convergence of the training algorithm. The CFA [28] considers the correlations between \( V_m \) and \( \Theta_m \) and converts \( \Theta_m \) and the estimation error sequence \( \Xi \) into \( (P_m^TP_m)^{-1}P_m^T \Xi \) and \( Y - P_m(P_m^TP_m)^{-1}P_m^Ty \) using least squares method, respectively, and then computes the derivatives of the estimated error sequence with respect to \( V_m \). Once \( V_m \) is updated making use of the derivative information, the output weights \( \Theta_m \) can be easily updated by least squares methods. However, the disadvantage of the CFA is that it only optimizes one vector parameter \( v_i, i = 1, \ldots, m \) at a time while all the other parameters \( v_j, j = 1, \ldots, m, j \neq i \) are fixed, hence the solution may be not optimal.

The proposed optimization algorithm not only considers the correlations between \( V_m \) and \( \Theta_m \) but also computes all the hidden layer parameters \( V_m \) simultaneously. The proposed algorithm first transforms the output weights \( \Theta_m \) into the dependent parameters on hidden layer parameters \( V_m \) using the OLS method, hence the task of optimizing both \( V_m \) and \( \Theta_m \) is reduced to only optimizing the hidden layer parameters \( V_m \) using the LM method. After all the hidden layer parameters are determined, the output weights are updated accordingly using the OLS solutions.

The LM algorithm optimizes the parameters by [47]
\[ V_m' = V_m + \Delta V_m \]  
(53)

where
\[ \Delta V_m = -H^{-1}J^T \Xi. \]  
(54)

The Jacobian matrix \( J \) is the first-order derivative of the error sequence \( \Xi \) with respect to \( V_m \), given by
\[ \frac{\partial \Xi}{\partial V_m} = \begin{bmatrix} \frac{\partial R_m}{\partial v_1}, \ldots, \frac{\partial R_m}{\partial v_m} \end{bmatrix} \]
\[ \frac{\partial R_m}{\partial v_k} = \begin{bmatrix} \frac{\partial R_m}{\partial v_{ki}}, \ldots, \frac{\partial R_m}{\partial v_{kr}} \end{bmatrix}, \quad k = 1, \ldots, m \]  
(55)
\[ \frac{\partial R_m}{\partial v_{ki}} = m \sum_{j=1}^{m} \frac{\partial w_j}{\partial v_{ki}}, \quad i = 1, \ldots, r \]

where \( v_k = [v_{k1}, \ldots, v_{kr}] \) and the Hessian matrix \( H \) is then calculated by the following:
\[ H = J^TJ + \mu I \]  
(56)

where \( I \) is the identity matrix and the scalar \( \mu \) controls the size of the confidence region. Notice that when \( \mu \) is large the algorithm becomes steepest descent, while for small \( \mu \) the algorithm becomes Gauss–Newton [47].

The key step for the LM algorithm is the computation of the Jacobian matrix. For each element in the Jacobian matrix shown in (55), the derivative of \( w_j \) with respect to \( v_{ki} \) needs to be calculated, \( j = 1, \ldots, m, i = 1, \ldots, r \). The mapping between \( w_j \) and \( v_{ki} \) can be obtained from the relations among \( v_{ki}, p_j, \) and \( w_j \). Each RBF term \( p_j \) is determined by a vector \( v_j = [v_{j1}, \ldots, v_{jr}] \)
\[ p_j = f(v_j) \]  
(57)

where \( f \) is the RBF function. Furthermore, (45) presents the relationship between variable \( v_j \) and the orthogonal basis \( w_j \), which is given by the following:
\[ w_j = g(w_1, w_2, \ldots, w_{j-1}, p_j) = q(p_1, p_2, \ldots, p_j) = z(v_1, v_2, \ldots, v_j) \]  
(58)

where \( g, q, \) and \( z \) are supposed to be some vector mapping functions. Equation (58) implies that \( w_j \) is determined by \( v_1, \ldots, v_j \) and it is independent of \( v_{j+1}, \ldots, v_m \), hence
\[ \partial w_j / \partial v_{ki} = \begin{cases} 0, & j < k, i = 1, \ldots, r \\ \partial w_j / \partial v_{ki}, & j \geq k, i = 1, \ldots, r \end{cases} \]  
(59)

and the element in (55) can be further simplified as follows:
\[ \frac{\partial R_m}{\partial v_{ki}} = \frac{\partial \left\{ w_1w'_1, \ldots, w_jw'_j, \ldots, w_tw'_{m} \right\}}{\partial v_{ki}}, \quad i = 1, \ldots, r \]  
(60)

while \( k = m \), (60) becomes
\[ \frac{\partial R_m}{\partial v_{mi}} = \frac{\partial \left\{ w_1w'_1, \ldots, w_{m-1}w'_{m-1} \right\}}{\partial v_{mi}}, \quad i = 1, \ldots, r. \]  
(61)

Compare (60) and (61), and it is clear that the derivative with respect to the last term variables \( v_m = [v_{m1}, \ldots, v_{mrt}] \) only needs compute \( \partial w_m / \partial v_{mi} \), \( i = 1, \ldots, r \) while for other derivatives, say \( v_k = [v_{k1}, \ldots, v_{krr}] \), \( \partial w_k / \partial v_{ki}, \partial w_m / \partial v_{ki}, i = 1, \ldots, r \), have to be computed. To reduce the computation load for derivatives, each term except the last one is moved to the last position and its derivatives are then computed. If any term position is changed, the orthogonal decomposition procedure needs being implemented again. Suppose \( p_k, k = 1, \ldots, m - 1 \) is to be moved to the last position. This leads to shifting the columns \( p_{k+1}, \ldots, p_m \), left by one place as shown in (62)
\[ \{p_1, \ldots, p_k, \ldots, p_m\} \rightarrow \{p_1, \ldots, p_{k-1}, p_{k+1}, \ldots, p_m, p_k\} \]  
(62)

and therefore their corresponding changed orthogonal basis is given as follows:
\[ \{w_1, \ldots, w_k, \ldots, w_m\} \rightarrow \{w_1, \ldots, w_{k-1}, w'_k, \ldots, w'_m\} \]  
(63)

If \( k = 1 \), then all the orthogonal terms \( w'_1, \ldots, w'_m \) have to be computed using (23). If \( k > 1 \), the terms \( w'_1, \ldots, w'_{k-1} \)
are left unchanged except for \( \{w'_k, \ldots, w'_m\} \). In details, shift the position by the following:

\[
\begin{align*}
p'_j &= p_{j+1}, \quad k \leq j < m \\
p'_m &= p_k
\end{align*}
\]

and then redecompose \( p'_k, \ldots, p'_m \) by the following:

\[
\begin{align*}
\alpha'_{nj} &= \min_{<w_n, p_k>} \frac{w_n}{w_n}, \quad 1 \leq n < k \\
\alpha'_{nj} &= \min_{<w_n, p_k>} \frac{w_n}{w_n}, \quad k \leq n < j, \\
w_j &= p_j - \sum_{n=1}^{k-1} \alpha'_{nj} w_n - \sum_{n=k}^{j-1} \alpha'_{nj} w_n 
\end{align*}
\]

The derivatives with respect to the last term \( p_m \) in Jacobian matrix are calculated by the following:

\[
\begin{align*}
\frac{\partial p_m}{\partial v_{mi}} &= \left[ \frac{\partial p_m(x(t), v_m)}{\partial v_{mi}}, \quad t = 1, \ldots, N \right] \\
\frac{\partial w_m}{\partial v_{mi}} &= \frac{\partial p_m}{\partial v_{mi}} - \sum_{j=1}^{m-1} \frac{w_j^T}{w_j} \frac{\partial p_m}{\partial v_{j}} \\
J_{mi} &= -\frac{\partial R_m}{\partial v_{mi}} y + \frac{\partial w_{mi}}{\partial v_{mi}} y \\
&= \frac{\partial w_m}{\partial v_{mi}} y + \frac{\partial w_m}{\partial v_{mi}} y + 2 \frac{\partial w_m}{\partial v_{mi}} \frac{\partial w_m}{\partial v_{mi}} \\
&\quad \left( w_m \frac{\partial w_m}{\partial v_{mi}} \right)
\end{align*}
\]

and for any other term \( p_k \) that is moved to the last position the same formula (66) is used, where \( \partial p_m/\partial v_{mi} \) equals \( \partial p_k/\partial v_{ki} \). To compute \( \partial p_k/\partial v_{ki} \), the basis function \( p_k(x(t), v_k) \) should be given. In this paper, the Gaussian function is considered

\[
p_k(x(t), v_k) = p_k(x(t), d_k, s_k) = \exp(-\eta(x(t)))
\]

where \( \eta(x(t)) = \sum_{t=1}^{N} (x(t) - s_k_i/d_k)^2, \quad t = 1, \ldots, N \) and \( k = 1, \ldots, m \). The first-order partial derivatives with respect to the widths and centers are given as follows:

\[
\begin{align*}
\frac{\partial p_k(x(t), d_k, s_k)}{\partial d_k} &= \frac{2}{d_k^2} \eta(x(t)) \exp(-\eta(x(t))) \\
\frac{\partial p_k(x(t), d_k, s_k)}{\partial s_k_i} &= \frac{2}{d_k^2} (x_i - s_k_i) \exp(-\eta(x(t)))
\end{align*}
\]

respectively.

The procedure for parameter optimization is summarized as follows.

**Step 1:** Initial model construction. The initial model with parameters \( V_m = [v_1, \ldots, v_m] \) is built by forward stepwise selection method given in (40)-(46).

**Step 2:** Compute Jacobian matrix using (68), (69), (64), (65), and (66).

**Step 3:** Compute the SSE with the optimized \( V_m \). Set the value \( \mu \) and solve (56) and (54) to obtain \( V_m = V_m + \Delta V_m \). Construct the basic functions terms \( P_m \) according to (67) using the new variables \( V'_m \), and then compute the SSE using (23) and (32). One issue in step 3 is to set the value of the regularization parameter \( \mu \in [0, \infty] \) as there is no theoretical framework to initialize it. An exhaustive search with small steps can lead to too much computation efforts while a coarse search with large steps may suffer from dissatisfactory optimization accuracy. Unlike the previous scheme used in [47] where the initial regularization parameter was manually given and step size was tuned by try-and-error, a simple way combining coarse and fine search was proposed here. A coarse set of \( 10^{-8}, \ldots, 10^1 \) with ten times difference between two adjacent numbers is first used to find an initial value, say \( \mu_1 \), which gives the minimal SSE. A fine set \( \mu_1 \times [0.1, 0.2, \ldots, 0.9, 1, 2, \ldots, 9] \) is then used to further optimize the SSE and identify the best value \( \mu_{\text{min}} \). In a word, only 35 values in total are involved and this is an effective and efficient tuning scheme. This paper uses this scheme on all the test examples. However, it should be noted that a larger coarse set with less than ten times differences between two adjacent numbers and a larger fine set can be considered if the optimization results are not satisfactory.

**Step 4:** Update \( V_m = V_m \) with \( \mu_{\text{min}} \) and repeat steps 2 and 3. This procedure is terminated when the SSE is reduced to a certain predefined error goal or the iteration number reaches a given number. After the hidden layer parameters \( V_m \) are determined, the output weights are calculated by back elimination (49).

**C. Regularization and Generalization**

The ridge and LASSO regularization methods can be easily incorporated into the parameter optimization process to avoid over-fitting. The optimization goal is therefore to minimize the SSE under a constraint. If the new updated output weights \( \Theta \) satisfy the ridge or LASSO constraint, the new parameters will be used. Otherwise, the new parameters will be discarded. The difficulty is to determine the tuning parameter \( t_r \) or \( t_l \). The loose constraint with large tolerance values are considered first, and then a series of tighter constraints are used. In this paper, the two basic initial tolerance values are given by the following:

\[
\begin{align*}
t_r &= \sum_{i=1}^{M} \hat{\theta}_i^2 \\
t_l &= \sum_{i=1}^{M} \hat{\theta}_i
\end{align*}
\]

for ridge and LASSO, respectively, where \( \hat{\theta}_i \) is the weights of the initial model built by forward stepwise selection. A serial of constraint values \( t_r \times \{100, 1, 0.1\} \) or \( t_l \times \{100, 10, 0.1, 0.01\} \) are used and the parameters are tuned under these constraints.

It should be noted that the proposed method can not only be used for constructing Gaussian RBF networks, it can also be directly used for the construction of other basis function networks, such as thin-plate-spline, multiqadratic, and inverse multiquadratic RBF networks. The only difference in
the implementation is that the first derivatives for centers and widths shown in (68) and (69) need to be changed accordingly. Furthermore, the new method can be extended to kernel-based models in future paper.

V. Computational Complexity

The computational complexity for the new method includes initial model construction and parameter optimization. For initial model construction, given \( N \) is the size of the training dataset and \( m \) nodes are selected from \( M \) nodes, then the main computation effort includes calculating inner products of \( \langle \mathbf{w}, \mathbf{p} \rangle \), \( \langle \mathbf{w}, \mathbf{y} \rangle \) and \( \langle \mathbf{w}, \mathbf{w} \rangle \) and corresponding division. The total number of initial model construction operations is thus given by the following:

\[
C_1 = 4MNm + 3Mm - 2Nm^2 - \frac{3}{2}m^2 + \frac{3}{2}m - M. \tag{72}
\]

It is noted that the reused variables are stored, and hence, it is not necessary to recompute them, leading to a reduced computation load [18]. If they are not stored, then the total operation number is around \( MNm^2 \) [41].

For parameter optimization, the computation load includes computing the Jacobian matrix and searching for the best adjustment parameters. The computation complexity for computing the Jacobian matrix and searching the best adjustment parameters involved in one iteration is given as follows:

\[
C_J = \frac{1}{3}Nm^3 + \frac{3}{2}Nm^2 - \frac{11}{6}Nm + \frac{1}{2}m^2 + \frac{1}{2}m + 6Nm \tag{73}
\]

and

\[
C_s = Nmr + Nm^2 + m3 + \frac{1}{2}m^2 + \frac{1}{2}m + 6Nm \tag{74}
\]

respectively. Given are \( h \) iterations in step 3 and \( n \) iterations in step 4, the total operation number for parameter optimization is given by the following:

\[
C_2 = n(C_J + hC_s). \tag{75}
\]

For the parameter optimization, the comparison of the proposed method with the CFA is now discussed. The CFA optimizes one hidden node at a time. For each time optimization, it carries out the same steps 3 and 4 with the new method, hence it needs

\[
C_3 = n(C_J' + hC_s) \tag{76}
\]

where \( C_J' = Nm^2r + 8Nmr \) is less than \( C_J \) as it does not need shift each term to the last position. However, it needs \( m \) times to optimize the whole model with \( m \) nodes. Therefore, its total operation is as follows:

\[
C_4 = mn(C_J' + hC_s). \tag{77}
\]

The additional computation by the CFA method is as follows:

\[
C_4 - C_2 = (m - 1)n(hC_s - C_r) \tag{78}
\]

where

\[
C_r = C_J - C_J' = \frac{1}{3}Nm^3 + \frac{3}{2}Nm^2 - \frac{11}{6}Nm. \tag{79}
\]

It is noted that as the model size \( m \) increases, the CFA needs much more computation time than the new method.

VI. Simulation Examples

A. Regression Modeling

To illustrate the performance of the new method, two examples are used to test its approximation ability and running time. The comparison with the OLS method and the CFA is also discussed.

Example 1: The following nonlinear system model is taken from [13]:

\[
y(t) = -0.6377y(t - 1) + 0.07298y(t - 2) + 0.03597u(t - 1) + 0.06622u(t - 2) + 0.06568u(t - 1)y(t - 1) + 0.02357u^2(t - 1) + 0.05939 + \xi(t) \tag{80}
\]

where \( t \) is the time series, \( u \) and \( y \) are the system input and output, respectively. A data sequence of length 1000 is generated from this nonlinear system, with input \( u \) being randomly distributed within \([-1, 1]\). \( \xi(t) \) is a Gaussian sequence with zero mean and signal-to-noise ratio (SNR) 25 dB. The average of equivalent noise variance for SNR 25 dB is \( 1.1 \times 10^{-5} \). The first 500 points are used for training and the remaining 500 data points are used for testing.

Following [13], five variables \( [u(t - 1), u(t - 2), u(t - 3), y(t - 1), y(t - 2)] \) are selected as the RBF input \( \mathbf{x}(t) \). All the widths are chosen as 1 by trial-and-error, while the centers are

\begin{table}[h]
\centering
\caption{Monte Carlo Simulation-Based Model Performance With Different Model Size in Example 1}
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{Size} & \textbf{Meth} & \textbf{Training SSE} & \textbf{Test SSE} & \textbf{Time (s)} \\
\hline
OLS & 0.356 ± 0.036 & 0.356 ± 0.036 & 0.027 ± 0.005 & \\
2 & CFA & 0.240 ± 0.026 & 0.240 ± 0.028 & 5.485 ± 0.135 & \\
& New & 0.187 ± 0.023 & 0.187 ± 0.024 & 0.053 ± 0.012 & \\
& OLS & 0.294 ± 0.030 & 0.294 ± 0.031 & 0.048 ± 0.003 & \\
3 & CFA & 0.214 ± 0.023 & 0.214 ± 0.023 & 8.229 ± 0.157 & \\
& New & 0.141 ± 0.023 & 0.141 ± 0.022 & 0.081 ± 0.002 & \\
& OLS & 0.237 ± 0.032 & 0.237 ± 0.034 & 0.072 ± 0.005 & \\
4 & CFA & 0.196 ± 0.025 & 0.196 ± 0.028 & 11.198 ± 0.661 & \\
& New & 0.110 ± 0.027 & 0.110 ± 0.028 & 0.118 ± 0.010 & \\
& OLS & 0.204 ± 0.021 & 0.203 ± 0.021 & 0.098 ± 0.004 & \\
5 & CFA & 0.165 ± 0.025 & 0.163 ± 0.025 & 13.903 ± 0.330 & \\
& New & 0.059 ± 0.031 & 0.059 ± 0.031 & 0.154 ± 0.011 & \\
& OLS & 0.184 ± 0.020 & 0.185 ± 0.021 & 0.126 ± 0.005 & \\
6 & CFA & 0.140 ± 0.024 & 0.141 ± 0.024 & 16.772 ± 0.431 & \\
& New & 0.040 ± 0.020 & 0.041 ± 0.020 & 0.192 ± 0.017 & \\
& OLS & 0.168 ± 0.018 & 0.171 ± 0.018 & 0.156 ± 0.005 & \\
7 & CFA & 0.125 ± 0.025 & 0.127 ± 0.026 & 19.645 ± 0.604 & \\
& New & 0.032 ± 0.014 & 0.034 ± 0.015 & 0.231 ± 0.008 & \\
& OLS & 0.160 ± 0.020 & 0.161 ± 0.020 & 0.187 ± 0.005 & \\
8 & CFA & 0.120 ± 0.026 & 0.122 ± 0.026 & 22.307 ± 0.444 & \\
& New & 0.028 ± 0.014 & 0.029 ± 0.015 & 0.272 ± 0.003 & \\
& OLS & 0.149 ± 0.018 & 0.149 ± 0.019 & 0.223 ± 0.005 & \\
9 & CFA & 0.104 ± 0.028 & 0.104 ± 0.028 & 25.255 ± 0.649 & \\
& New & 0.024 ± 0.011 & 0.024 ± 0.011 & 0.317 ± 0.001 & \\
& OLS & 0.136 ± 0.019 & 0.137 ± 0.020 & 0.261 ± 0.006 & \\
10 & CFA & 0.093 ± 0.023 & 0.094 ± 0.023 & 28.048 ± 0.587 & \\
& New & 0.023 ± 0.011 & 0.024 ± 0.011 & 0.364 ± 0.001 & \\
\hline
\end{tabular}
\end{table}
initialized on the training data points, to produce a total of 500 candidate nodes. The SSE over the training data and testing datasets are used to evaluate the model performance. To make a fair comparison, the CFA and new method used the same initialization procedure. Models with different model sizes (the number of hidden layer nodes) are used to evaluate the performances of the three methods. The averages and standard deviations of Monte Carlo simulation results with 100 trials are listed in Table I. Furthermore, Figs. 2 and 3 present the comparison results on the variation of the training and testing errors, respectively. These results show that, as the model size increases, the proposed method produces better models than the OLS method and the CFA method. Fig. 4 compares the running time and it is shown that the new method is much more efficient in terms of training time as the CFA method takes about 80–100 times than the new method.

It is clear that the new method produces most training error reduction and over-fitting is avoided. The reason why the new method significantly improves the model performance lies in the fact that all the reduction errors are compared with that of OLS. OLS uses the discrete candidate term pool where the optimal values for the candidate terms may not be included, and it is a local optimization method that may be trapped at a local minima with undesirable performance. The new method uses the OLS results as initial model and then tunes all the parameters in the continuous space, which provides chances to find better model parameters. The reason why over-fitting does not happen lies in the facts these model sizes are not too big to make the resulting models too flexible and the 25 dB SNR is not too large. If the over-fitting occurs, the new method with LASSO or ridge regularization needs to be considered.

Example 1: The second example is taken from [48] and concerns approximating the following function:

\[ y(t) = 0.1t + \frac{\sin(t)}{t} + \sin(0.5t) + \xi(t), \quad -10 \leq t \leq 10 \quad (81) \]
where $\xi(t)$ is a Gaussian white noise with zero mean and variance $0.01$. A total of 400 data samples are generated, the first 200 data being used for training and the rest for testing.

To illustrate the model generalization ability and running time, models of different size are built using the same three proposed methods as in example 1 to produce the average results of Monte Carlo simulations with 100 repetitions, which are listed in Table II. Further, Figs. 5 and 6 compare the training and testing errors with increasing model size, while Fig. 7 compares the variation in running time. These results show that, as the model size increases, the new method produces much better models compared with OLS and converges about 100 times faster than the CFA.

### Table II

**Monte Carlo Simulation-Based Model Performance With the Different Model Size for Example 2**

<table>
<thead>
<tr>
<th>Size</th>
<th>Meth</th>
<th>Training SSE</th>
<th>Test SSE</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>OLS</td>
<td>3.869 ± 0.258</td>
<td>4.025 ± 0.350</td>
<td>0.021 ± 0.000</td>
</tr>
<tr>
<td></td>
<td>CFA</td>
<td>3.582 ± 0.455</td>
<td>3.766 ± 0.453</td>
<td>4.176 ± 0.118</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>2.508 ± 0.159</td>
<td>2.676 ± 0.182</td>
<td>0.041 ± 0.001</td>
</tr>
<tr>
<td>5</td>
<td>OLS</td>
<td>3.368 ± 0.259</td>
<td>3.448 ± 0.294</td>
<td>0.030 ± 0.005</td>
</tr>
<tr>
<td></td>
<td>CFA</td>
<td>2.938 ± 0.404</td>
<td>3.077 ± 0.474</td>
<td>5.262 ± 0.198</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>2.479 ± 0.169</td>
<td>2.607 ± 0.177</td>
<td>0.052 ± 0.001</td>
</tr>
<tr>
<td>6</td>
<td>OLS</td>
<td>2.671 ± 0.180</td>
<td>2.828 ± 0.228</td>
<td>0.038 ± 0.005</td>
</tr>
<tr>
<td></td>
<td>CFA</td>
<td>2.546 ± 0.179</td>
<td>2.760 ± 0.245</td>
<td>6.407 ± 0.354</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>2.377 ± 0.149</td>
<td>2.568 ± 0.160</td>
<td>0.066 ± 0.003</td>
</tr>
<tr>
<td>7</td>
<td>OLS</td>
<td>2.472 ± 0.152</td>
<td>2.667 ± 0.194</td>
<td>0.047 ± 0.001</td>
</tr>
<tr>
<td></td>
<td>CFA</td>
<td>2.316 ± 0.157</td>
<td>2.567 ± 0.196</td>
<td>7.513 ± 0.333</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>2.166 ± 0.203</td>
<td>2.392 ± 0.193</td>
<td>0.78 ± 0.003</td>
</tr>
<tr>
<td>8</td>
<td>OLS</td>
<td>2.231 ± 0.129</td>
<td>2.406 ± 0.190</td>
<td>0.057 ± 0.001</td>
</tr>
<tr>
<td></td>
<td>CFA</td>
<td>2.071 ± 0.090</td>
<td>2.361 ± 0.165</td>
<td>8.611 ± 0.397</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>1.997 ± 0.103</td>
<td>2.246 ± 0.141</td>
<td>0.092 ± 0.003</td>
</tr>
<tr>
<td>9</td>
<td>OLS</td>
<td>2.076 ± 0.106</td>
<td>2.325 ± 0.144</td>
<td>0.068 ± 0.001</td>
</tr>
<tr>
<td></td>
<td>CFA</td>
<td>2.019 ± 0.101</td>
<td>2.341 ± 0.143</td>
<td>9.693 ± 0.367</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>1.956 ± 0.102</td>
<td>2.248 ± 0.119</td>
<td>0.106 ± 0.001</td>
</tr>
</tbody>
</table>

**Fig. 6.** Variation in the test SSE with different model size for OLS and the new method in example 2 Monte Carlo simulations.

**Fig. 7.** Running time comparison with different model sizes between CFA and the new method in example 2 Monte Carlo simulations.

### Table III

**UCI Benchmarks Information**

<table>
<thead>
<tr>
<th>Data set</th>
<th>Dim</th>
<th>Train No.</th>
<th>Test No.</th>
<th>Repeat</th>
<th>Size</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banana</td>
<td>2</td>
<td>400</td>
<td>4900</td>
<td>100</td>
<td>20</td>
<td>0.8</td>
</tr>
<tr>
<td>B. cancer</td>
<td>9</td>
<td>200</td>
<td>77</td>
<td>100</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Diabetes</td>
<td>8</td>
<td>468</td>
<td>300</td>
<td>100</td>
<td>6</td>
<td>128</td>
</tr>
<tr>
<td>German</td>
<td>20</td>
<td>700</td>
<td>300</td>
<td>100</td>
<td>9</td>
<td>64</td>
</tr>
<tr>
<td>Heart</td>
<td>13</td>
<td>170</td>
<td>100</td>
<td>100</td>
<td>7</td>
<td>115</td>
</tr>
<tr>
<td>Image</td>
<td>18</td>
<td>1300</td>
<td>1010</td>
<td>20</td>
<td>140</td>
<td>12</td>
</tr>
<tr>
<td>Ringnorm</td>
<td>20</td>
<td>400</td>
<td>7000</td>
<td>100</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>F.sonar</td>
<td>9</td>
<td>666</td>
<td>400</td>
<td>100</td>
<td>3</td>
<td>32</td>
</tr>
<tr>
<td>Splice</td>
<td>60</td>
<td>1000</td>
<td>2175</td>
<td>20</td>
<td>180</td>
<td>11</td>
</tr>
<tr>
<td>Thyroid</td>
<td>5</td>
<td>140</td>
<td>75</td>
<td>100</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>Titanic</td>
<td>3</td>
<td>150</td>
<td>2051</td>
<td>100</td>
<td>3</td>
<td>48.2</td>
</tr>
<tr>
<td>Twnorm</td>
<td>20</td>
<td>400</td>
<td>7000</td>
<td>100</td>
<td>12</td>
<td>36</td>
</tr>
<tr>
<td>Waveform</td>
<td>21</td>
<td>400</td>
<td>4600</td>
<td>100</td>
<td>14</td>
<td>32.2</td>
</tr>
</tbody>
</table>

### B. Classification

To evaluate the performance of the new method on classification problems, an extensive number of experiments are conducted on 13 well-known real world classification problems taken from the UCI repository [49]: banana, breast cancer, diabetes, german, heart, image segment, ringnorm, flare sonar, splice, thyroid, Titanic, two norms, and waveform. Comparison with other state-of-the-art classifiers [50], [51], including AB$_R$, SVM, and KFD, is also given. The input dimension, training number, test number, repeating number, model size (the number of hidden layer nodes), and initial width of 13 datasets are listed in Table III. 100 predefined splits for each dataset (except splice and image with 20 splits) into training and test samples make sure that all the test methods use the same training and test data points. The new method incorporating the LASSO regularization is used for comparison. Table IV shows average results with deviations and the new method produced the best results with six times out of 13 cases.
To draw a convincing conclusion, the Friedman test with corresponding post-hoc tests is recommended as a robust nonparametric test by Demšar [52] for comparison of the multiple classifiers over multiple datasets. The Friedman test is to decide whether all the algorithms are equal. If they are not equal, the post-hoc tests can be used to decide which algorithms are significantly different from each other. The Friedman test first ranks the compared algorithms for each dataset separately according to the model performance (average misclassification rate). Then it utilizes the number of algorithms $K$, the number of datasets $N$ and each rank, say $r_{ij}$ denotes the $j$th algorithm on the $i$th dataset, to calculate the Friedman statistic

$$S = \frac{12N}{K(K+1)} \left[ \sum_{j=1}^{K} R_{j}^{2} - \frac{K(K+1)^2}{4} \right]$$

(82)

where

$$R_{j} = \frac{1}{N} \sum_{i=1}^{N} r_{ij}$$

(83)

is the average rank of the $j$th algorithm on all the $N$ datasets. An improved Friedman test version adopts the $F$-distribution rather than the original Friedman statistic. It is given by the following:

$$F = \frac{(N-1)S}{N(K-1) - S}.$$  

(84)

Using the ranks list in Table V and for $K = 4, N = 13$, the $F$ in this paper is $1.152$. Look up the critical value at significance level $0.10$ from the $F$-distribution table with $(K-1)$ and $(K-1)(N-1)$ degrees of freedom, it is found that $F(0.10, 3, 36) = 2.226 > 1.152$ which implies that the null-hypothesis that all the methods are equal is rejected. In other words, the Friedman test confirms that these methods are different and the Bonferroni–Dunn post-hoc can then be carried out to compare the new method with others. The performance of pairwise classifier is significantly different if the difference between two average ranks is larger than the critical difference value

$$C = q_{\alpha} \sqrt{\frac{K(K+1)}{6N}}$$

(85)

where the $q_{\alpha} = 2.128$ when $K = 4$ and significant level $\alpha = 0.10$ [52] and therefore $C = 1.0775$. The differences between the new method and $AB_{R}$, SVM and KFD are 0.84, 0.69, 0.31, which are all smaller than $C$. This means the new method is not significantly better than others but it is a very competitive method.

VII. CONCLUSION

A new discrete-continuous algorithm for constructing RBF models was proposed in this paper. In the first stage (initial model construction stage), an initial model was generated using a forward selection procedure by selecting one term at a time from a large pool of candidates in the discrete space. In the second stage (parameter optimization), the parameters of the initial model were further optimized using LM and OLS algorithms in the continuous space. The computational complexity analysis was given to confirm that it was much more computationally efficient than the CFA. Two regression modeling examples showed that the proposed method was able to produce a better model than the conventional OLS method, and needed much less computational time than the CFA approach. Furthermore, the applications to 13 classification problems showed the new method can perform favorably in comparison with other state-of-the-art classification techniques.

ACKNOWLEDGMENT

The authors would like to thank the handling editor and anonymous reviewers for their constructive comments to improve the quality of this paper.

REFERENCES


Kang Li (M’05–SM’11) received the B.Sc. degree from Xiangtan University, Hunan, China, in 1989, the M.Sc. degree from the Harbin Institute of Technology, Harbin, China, in 1992, and the Ph.D. degree from Shanghai Jiaotong University, Shanghai, China, in 1995. He is currently a Professor of intelligent systems and control with the School of Electronics, Electrical Engineering and Computer Science, Queen’s University Belfast, Belfast, U.K. He is a Visiting Professor with the Harbin Institute of Technology, Harbin, China, Shanghai University, Shanghai, China, and Ningbo Institute of Technology, Zhejiang University, Zhejiang, China. He held visiting fellowship or professorship with the National University of Singapore, Singapore, University of Iowa, Iowa City, IO, USA, New Jersey Institute of Technology, Tsinghua University, Beijing, China, and Technical University of Bari, Toranto, ON, Canada. He has published over 180 papers and edited ten conference proceedings (Springer). His current research interests include nonlinear system modeling, identification and control, bio-inspired computational intelligence, and fault-diagnosis and detection, with recent applications in power systems and renewable energy, polymer extrusion processes, and bioinformatics with applications on food safety and biomedical engineering.

Dr. Li serves in the editorial boards of Neurocomputing, the IEEE TRANSACTIONS OF THE INSTITUTE OF MEASUREMENT AND CONTROL, International Journal of Modeling, Identification and Control, and Cognitive Computation.

Haibo He (SM’11) received the B.S. and M.S. degrees in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1999 and 2002, respectively, and the Ph.D. degree in electrical engineering from Ohio University, Athens, OH, USA, in 2006. He was an Assistant Professor with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, USA, from 2006 to 2009. He is currently an Associate Professor with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, USA. His current research interests include computational intelligence, machine learning, hardware design for machine intelligence, smart grid, and cybersecurity. He has published one research book (Wiley), edited one book (Wiley-IEEE), and six conference proceedings (Springer). He has authored and co-authored over 120 peer-reviewed journal and conference papers. His researches have numerous media, such as the IEEE Smart Grid Newsletter, The Wall Street Journal, and Providence Business News.

Dr. He is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and IEEE TRANSACTIONS ON SMART GRID. He was a recipient of the National Science Foundation CAREER Award in 2011 and the Providence Business News Rising Star Innovator Award in 2011.

George W. Irwin (F’04) received the first class Honours degree in electrical and electronic engineering, the Ph.D. degree in control theory, and the D.Sc. degree from Queens University Belfast, Belfast, U.K., in 1972, 1976, and 1998, respectively. He has been a Personal Chair in control engineering with Queens University Belfast since 1989. His current research interests include identification, monitoring, and control, including neural networks, fuzzy neural systems, and multivariate statistics. He has published 350 research publications, including 125 peer-reviewed journal papers.

Dr. Irwin is a fellow of the U.K. Royal Academy of Engineering in 2002 and the IFAC in 2009. He is a member of the Royal Irish Academy in 2002. He received the four IEE Premiums and two medals from the U.K. Institute of Measurement and Control. He is a former Editor-in-Chief of Control Engineering Practice and he currently serves on the Technical Board and the Publications Management Board of IFAC.