IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 February 2023, accepted 13 March 2023, date of publication 22 March 2023, date of current version 27 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3260251

RESEARCH ARTICLE

A Brand-New Simple, Fast, and Effective Residual-Based Method for Radial Basis Function Neural Networks Training

LIFEI SUN[1,2], SEN LI[1], HAILONG LIU[3,4], (Member, IEEE), CHANGKAI SUN[5], LIPING QI[6], ZHIXUN SU[7], (Member, IEEE), AND CHANGSEN SUN[8]

[1]Information Science and Technology College, Dalian Maritime University, Dalian, Liaoning 116024, China
[2]School of Intelligence and Electronic Engineering, Dalian Neusoft University of Information, Dalian, Liaoning 116024, China
[3]School of Biomedical Engineering, Faculty of Medicine, Dalian University of Technology, Dalian, Liaoning 116024, China
[4]Liaoning Key Laboratory of Integrated Circuit and Biomedical Electronic System, Faculty of Medicine, Dalian University of Technology, Dalian, Liaoning 116024, China
[5]School of Artificial Intelligence, Dalian University of Technology, Dalian, Liaoning 116024, China
[6]School of Kinesiology and Health Promotion, Dalian University of Technology, Dalian, Liaoning 116024, China
[7]School of Mathematical Sciences, Dalian University of Technology, Dalian, Liaoning 116024, China
[8]School of Optoelectronic Engineering and Instrumentation Science, Dalian University of Technology, Dalian, Liaoning 116024, China

Corresponding author: Hailong Liu (hlliu@dlut.edu.cn)

This work was supported by the National Key Research and Development Program of China under Grant 2018AAA0100300.

ABSTRACT The radial basis function (RBF) neural network is a type of universal approximator, and has been widely used in various fields. Improving the training speed and compactness of RBF networks are critical for promoting their applications. In the present study, we propose a simple, fast, and effective RBF networks training method, which is based on the residual extreme points and their neighborhoods (thus called the REN method for short in this paper). The REN method calculates RBF centers and widths through a two-level iterative process, and realizes two main functionalities, namely 1) adding multiple centers within one pass through the whole data set, and 2) calculating RBF widths specifically for each center. The use of this algorithm does not need any parameter adjustments, and the models for approximation or classification can be obtained by only one run. The performance of the proposed REN algorithm is compared with the classic and powerful orthogonal least squares (OLS) algorithm. By reaching the same accuracies, the REN algorithm trains RBF networks 50 and 320 times faster, in the chirp (0˜50 Hz, 2 s, 1 kHz, 2001 samples) and two-dimensional peaks (2401 samples) signal approximation tasks respectively, than the OLS algorithm does, and the number of centers obtained by the REN algorithm is reduced by half. When incorporating the same number of centers, the REN algorithm achieves accuracies up to 3 orders of magnitude higher than the best results obtained by the OLS algorithm. In the classification task of a real discrete breast cancer data, both methods result in accuracies comparable to many existent methods, but the REN algorithm has the advantages of fast training speeds and no requirements for parameter adjustments. The REN algorithm proposed in this study may potentially be used for tasks with large scale of data or applications that require high model performances.

INDEX TERMS Radial basis function (RBF) neural networks, residual, RBF center estimation, RBF width estimation, approximation, classification.

I. INTRODUCTION

Radial basis function (RBF) neural networks have been proved to be universal approximators [1], [2], [3], which means that they are capable of approximating any continuous functions with satisfied accuracies when an adequate network size and appropriate parameter settings are considered. The method of RBF network has attracted much attention in the past several decades due to its simple structure and excellent performance, and it has been widely used in many fields,

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Ali.

such as system identification and modeling [4], [5], [6], nonlinear system control [7], imaging processing [8], [9], [10], data generation [11], graph-based signal representation and processing [12], pattern recognition and classification [12], [13], [14], neuro-rehabilitation of tremor suppression [15], etc. Many currently popular methods, such as fuzzy [16] and particle swarm optimization (PSO) [6], [17], [18] algorithms, have been drawn to be used extensively in combination with RBF networks for efficient determinations of RBF model structures and parameters.

A typical RBF network consists of three layers, namely the input layer, the hidden layer, and the output layer. The input layer has as many nodes as the dimensions of input data, which is responsible for receiving input samples; The number of hidden layer nodes needs to be determined through certain training methods, and each of the hidden node, which has an activation function called an RBF, connects to all of the input layer nodes with the coefficients constituting the center vector of the RBF; During the calculation process of an RBF network, the distances between the inputs and centers will be calculated and fed to the RBFs; The output layer provides multi-dimensional outputs, with each of them being a linear combination of the hidden layer RBF outputs and, in most cases, an additional bias. In practical applications, appropriate values must be determined for the parameters of the number of centers (m_1), their locations ($\boldsymbol{C}$) and widths ($\boldsymbol{\sigma}$), and the connecting coefficients (ω) between the hidden and output layers, by using some training methods on a training data for the RBF networks to perform well in solving specific problems.

Compared to other types of neural networks, RBF networks have some special characteristics that can be employed for designing their training methods. Schwenker et al. [13] make a good systematic summary for RBF network training methods according to the number of training phases involved, and put forward the concepts of one-phase, two-phase and three-phase RBF network training methods. One-phase methods train all the parameters in a single learning process, such as the support vector learning (SVL) method [13], the orthogonal least squares (OLS) method [19], and the gradient descent algorithms [20], [21], [22], in which the target information is employed during training; Two-phase methods train RBF networks in two separate learning processes for the hidden layer and output layer parameters respectively. And the hidden layer parameters (RBFs centers and widths) can be obtained by using unsupervised clustering methods (such as k-means clustering) or supervised methods (such as LVQ or decision trees); The three-phase methods try to fine-tune the obtained results further to achieve a more optimal model through, e.g., the error back-propagation algorithms. This third phase treatment is reasonable and necessary, since continuous adjustments of centers and widths may result in better results, while adaptations of them totally from inappropriate initial values would usually cause very slow training speeds and high computational demands. In fact, this fine-tuning process is not only for combination with the two-phase methods, but can also be applied to RBF networks obtained with any methods. These concepts of phase-wise training is a good reference for selecting or designing appropriate training strategies for RBF networks.

For the output layer connecting coefficients, due to their linear characteristics, no matter what algorithms (direct inverse, least squares, or gradient descent methods, etc.) or process (calculated separately or together with the hidden layer parameters) are adopted, the determination of their values is relatively simple and the optimal values of them can always be obtained easily. On the contrary, it is commonly known that the number of centers, their positions, and RBF widths are essentially important and difficult to determine for achieving the optimal performance of RBF networks.

Previous research results have shown that the two-phase learning methods only realize the supervised or unsupervised learning of the center positions, and does not make full use of the label information of training data or the expression ability of the hidden layer, especially for the determination of RBF widths [13]. Therefore, it is generally hard for the method to obtain optimal results, and the performance obtained is generally worse than those of, e.g., multilayer perceptron neural networks. These can also be understood in that, if different set of RBFs are adopted comprising the target functions, then the optimal centers and widths might also be different accordingly; thus, only employing some clustering criterion arbitrarily without considering RBFs would be hard to get the best results. However, this kind of methods might also work well for some kind of data set, and have the advantage of having fast training speeds and being capable of providing very good initial conditions for other methods for further optimizations.

Training methods based on gradient information are one of the most popular RBF network training methods at present [14], [17], [21], [22], [23], [24]. They can not only be used as the third phase learning algorithm to optimize RBF networks further, but can also be used as the one-phase learning method to train all parameters in one process. Since the gradient based methods are implemented by using RBF network prediction errors in a supervised manner, which involves the evaluations of RBFs, they usually result in better performance [13], [21]. Recently, great progresses have been made in the gradient based training methods for RBF networks [17], [21], [22]. However, there still exist some problems and/or drawbacks that need to be solved for the gradient based methods to be more effective, such as tending to fall easily into local optima, demanding more computational resources, having slow converging speeds, and requiring repetitive adjustments of parameter values or thresholds (e.g. the learning rates). These make the performance of RBF networks trained by gradient based methods severely depend on the initial values of parameters and the pattern of input data. In Schwenker et al.'s

study [13], although fast training speeds and relatively higher accuracies are achieved, the gradient based back-propagation method does not alter center positions too much when it is used as the third phase training method, implicating that the RBF networks obtained might not be the global optimal one, but be deeply affected and constrained by the initial conditions provided by the prior two-phase training results.

To achieve better initial conditions, Zhang et al. [14] adopt the OLS-based forward subset selection method [19], [25] to specify centers in their study for further optimization that is implemented by a new Levenberg–Marquardt (LM) based method. The ErrCor algorithm proposed by Yu et al. [26] uses the location of the sample with the maximum error as the initial value for each newly added unit in each iteration, before learning the optimal results through the gradient method of modified Levenberg-Marquardt (LM) algorithm. With these methods of specifying initial center positions, faster training speeds and higher model performances are achieved.

In addition to the gradient based methods, some methods such as the orthogonal least squares (OLS) [19], [25] and support vector learning (SVL) [13] methods are considered as special one-phase RBF network training methods, since they achieve optimal results with respect to their respective criteria by incorporating the RBFs' outputs and label information. These methods are different from the two-phase methods, in which the outputs of RBFs are only used for determinations of the output layer weights. Both OLS and SVL select a portion of the input data as centers to produce compact RBF networks, and are quite effective in achieving satisfactory accuracies. However, SVL often leads to complex network structures, and OLS is computationally demanding. When dealing with large data sets, they usually need a large amount of calculations and memories, and a long training time. In addition, although RBFs' outputs have been involved during training, these methods do not consider the shape dependence of RBFs on the training data. Apparently, training methods which can update centers and widths simultaneously according to the training data would improve model accuracies and compactness.

To improve the performance of RBF networks, some researchers design training algorithms by using more accurate information, such as more appropriate basis functions and model prediction errors. Singla et al. [27] propose that using more appropriate basis functions with adapted shapes, scalings, and rotations according to the training data, rather than the predetermined ones, would help in achieving more compact RBF models while improving model accuracies. Lai et al. [28] propose a set of RBFs with adaptive inputs and composite trend representations for portfolio selection and achieve effective and robust asset price prediction performance. Reiner and Wilamowski [29] utilize the input data with the greatest error magnitude as centers for incrementally constructing RBF networks and determine the corresponding widths iteratively with the Nelder Mead's Simplex method, which greatly reduces the size of RBF networks while maintaining their approximating capabilities. Some researches have also tried other methods of RBF widths calculations. For examples, Yao et al. [30] propose a width optimization method, called the concurrent subspace width optimization (CSWO), to decompose the large-scale width optimization problem into several subspace optimization (SSO) problems, and Huan et al. [31] directly compute RBF widths by replacing the Euclidean norm by the Mahalanobis norm.

Considering that each RBF actually expresses a partial aspect of the input-output mapping of training data, they, of course, also represent the residual information that RBF networks can improve or lack. Therefore, it can be speculated that determining RBF centers and widths according to model residuals, rather than just using the relatively constant or pre-determined RBFs to express the training data, will be of great help in improving RBF network performance.

The aim of this study is to construct a simple, fast and easy-to-use algorithm for RBF networks training. The algorithm automatically determines RBF centers and widths simultaneously based on model residual extreme points and their neighboring fields (so called the REN method in short in this paper), and involves significantly less distance computations. The hidden layer nodes are added recursively into the model in an incremental manner, with one or more centers determined in each learning iteration. With the proposed algorithm, the RBF network with accuracies of 3 orders of magnitude higher and decreased number of hidden nodes approximately by half could be obtained with less memory and computation requirements and a training speed about 50˜320 times faster, in contrast to the OLS method. In addition, no parameters or thresholds need to be preset for the use of this algorithm.

The paper is organized as follows. In section II. METHODS, first, the concepts and architectures of RBF networks are given; then the REN algorithm for RBF networks training is described in details, including the descriptions about the overall iterative and evaluation process, the inner iterative process, and the treatments of an internal parameter of multiplying factor. After summarization of the algorithm, an example of function approximation, meant to demonstrate the algorithm running process, is given. In section III. RESULTS, the influences of some critical internal parameters on RBF models are demonstrated first, and then the results of approximation of chirp signal with various frequency components are given to demonstrate the superiority of the REN algorithm, which is followed by a two-dimensional peaks function approximation. Finally, classifications on a real breast cancer data set are given to preliminarily demonstrate the feasibility of the REN algorithm in solving practical applications. At the end, discussion and conclusion are provided in section IV and V respectively.

II. METHODS

A. ARCHITECTURES OF RBF NETWORKS

As described above, an RBF network consists of three layers. The input layer feeds input data to the network; the hidden layer contains a number of nodes of RBFs to implement the mapping from the input data space to the feature space (RBF outputs); the output layer performs linear combinations of the RBF outputs, realizing a linear fitting or classification of the feature space data. If the Gaussian function $\phi(x)$ is used as the RBF, and the training data set D contains N input-output data pairs $\{(\mathbf{x}_i, \mathbf{y}_i)|i = 1, 2, \ldots, N\}$, where $\mathbf{x}_i$ is an m_0 dimensional input data and $\mathbf{y}_i$ is the corresponding m_2 dimensional output data, the output of the j^{th} RBF (m_1 RBFs in all, indicating that the feature space of the hidden layer is m_1 dimensional) can be expressed as:

$$\phi_{i,j} = \phi(\|\mathbf{x}_i - \mathbf{c}_j\|) = e^{-\frac{\|\mathbf{x}_i - \mathbf{c}_j\|^2}{2\sigma_j^2}} \quad 1 \leqslant i \leqslant N, 1 \leqslant j \leqslant m_1 \quad (1)$$

in which $\mathbf{c}_j$ and σ_j are the center and width of the j^{th} RBF, respectively, and $\|\cdot\|$ represents the l_2 norm calculating the Euclidean distance between the i^{th} input data and the j^{th} center point. The output layer can be multi-dimensional, and the k^{th} dimensional output for the i^{th} input sample is expressed as:

$$y_{ik} = \sum_{j=1}^{m_1} \omega_{jk}\phi_{i,j} = \sum_{j=1}^{m_1} \omega_{jk} e^{-\frac{\|\mathbf{x}_i - \mathbf{c}_j\|^2}{2\sigma_j^2}} \quad (2)$$

in which $\omega_{jk} (1 \leqslant j \leqslant m_1, 1 \leqslant k \leqslant m_2)$ is the connecting coefficient between the j^{th} RBF and k^{th} output layer unit.

If the following notations are defined such that

$$\begin{aligned} \boldsymbol{\omega}_k &= [\omega_{1k}, \omega_{2k}, \cdots, \omega_{m_1 k}]^T \\ \boldsymbol{\Omega} &= [\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \cdots, \boldsymbol{\omega}_{m_2}]^T \\ \mathbf{y}_i &= [y_{i1}, y_{i2}, \cdots, y_{im_2}]^T \\ \mathbf{Y} &= [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N]^T \\ \boldsymbol{\phi} &= \begin{bmatrix} \phi_{1,1} & \cdots & \phi_{1,m_1} \\ \vdots & \ddots & \vdots \\ \phi_{N,1} & \cdots & \phi_{N,m_1} \end{bmatrix} \end{aligned} \quad (3)$$

in which $\boldsymbol{\Omega}, \mathbf{Y}, \boldsymbol{\phi}$ are of m_2xm_1, Nxm_2, Nxm_1 dimensions respectively and $[\cdot]^T$ means the transpose of matrices or vectors, then we have

$$\boldsymbol{\phi}\boldsymbol{\Omega}^T = \mathbf{Y} \quad (4)$$

The training task of an RBF network is to determine all the RBF parameters ($\boldsymbol{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_{m_1}]^T$ and $\boldsymbol{\sigma} = [\sigma_1, \cdots, \sigma_{m_1}]^T$) and the connecting coefficients ($\boldsymbol{\Omega}$) by using the given training data set. After training is finished, the obtained model can be used to process new unlabeled input data to get the corresponding output in the task of, e.g., pattern classifications or function approximations.

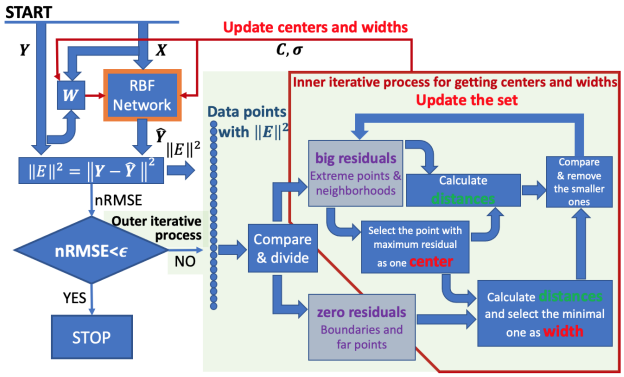


FIGURE 1. Flow chart for the implementation of the REN algorithm.

B. THE REN ALGORITHM FOR RBF NETWORKS TRAINING

1) THE ITERATIVE PROCESS OF THE REN ALGORITHM AND EVALUATION OF THE RBF NETWORK PERFORMANCE

The proposed REN algorithm (see Fig. 1 for the flow chart) is implemented in a two-level iterative process, namely the outer and inner iterative processes. In each outer iteration, two steps are involved. One is to update the RBF network parameters, namely the centers ($\mathbf{C}_n$) and widths ($\boldsymbol{\sigma}_n$) (hereafter, the iteration number is denoted by the subscript n), through the inner nested iterative process, and the other is to evaluate the performance of the updated RBF network. This two-step outer iterative training process will be continued until a stopping criterion is met. Usually, the stopping criteria could be that a satisfactory performance has been reached, or that a predetermined number of nodes have been added.

Within the n^{th} ($n \geqslant 1$) outer iteration, the nested inner iterative process firstly computes RBF centers ($\mathbf{C}_n = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_{m_{1,n}}]^T$) and widths ($\boldsymbol{\sigma}_n = [\sigma_1, \sigma_2, \cdots, \sigma_{m_{1,n}}]^T$) based on the squared residual norm vector $\|\mathbf{E}_{n-1}\|^2$ returned from the $(n-1)^{\text{th}}$ iteration, in which

$$\mathbf{E}_{n-1} = \mathbf{Y} - \hat{\mathbf{Y}}_{n-1} \quad (5)$$

is defined as the model residual (the detailed definition process will be described below). Given the input data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]^T$, the RBF layer output is then computed as $\boldsymbol{\phi}_n$. Through the formula $\boldsymbol{\phi}_n\boldsymbol{\Omega}_n^T = \mathbf{Y}$ (4) and the least squares (LS) or pseudo inverse algorithms, the estimate $\hat{\boldsymbol{\Omega}}_n$ can be obtained. These complete the construction of the whole RBF network in the n^{th} iteration (at this time point, the model has the parameters of $\mathbf{C}_n$, $\boldsymbol{\sigma}_n$, and $\hat{\boldsymbol{\Omega}}_n$). The obtained model has a prediction error of $e_{i,k,n} = y_{i,k} - \hat{y}_{i,k,n}$ at the k^{th} output for the i^{th} input data. The error for all dimensional outputs is expressed in a vector form as $\mathbf{e}_{i,n} = [e_{i,1,n}, e_{i,2,n}, \cdots, e_{i,m_2,n}]^T = \mathbf{y}_i - \hat{\mathbf{y}}_{i,n}$, and for all inputs as $\mathbf{E}_n = [\mathbf{e}_{1,n}, \mathbf{e}_{2,n}, \cdots, \mathbf{e}_{N,n}]^T = \mathbf{Y} - \hat{\mathbf{Y}}_n$ (in which $\hat{\mathbf{Y}}_n = \boldsymbol{\phi}_n\hat{\boldsymbol{\Omega}}_n^T$). The normalized root mean square error (nRMSE), i.e. the square root of the percentage of the model residual energy to the total energy of the targets, is used to evaluate the model

performance and expressed as

$$\text{nRMSE}_n = \sqrt{\frac{\|\mathbf{E}_n\|_F^2}{\|\mathbf{Y}\|_F^2}} = \sqrt{\frac{\sum_i \sum_k e_{i,k,n}^2}{\sum_i \sum_k y_{i,k}^2}} \quad 1 \leqslant i \leqslant N, 1 \leqslant k \leqslant m_2 \quad (6)$$

in which $\|\cdot\|_F$ denotes the Frobenius norm. The reason for the use of nRMSE is that even one single value of it could give a direct and intuitive index about the model performance. Meanwhile, the obtained residual norm is used as the input for the next iteration, which is computed as $\|\mathbf{E}_n\|^2 = [\|\mathbf{e}_{1,n}\|^2, \|\mathbf{e}_{2,n}\|^2, \cdots, \|\mathbf{e}_{N,n}\|^2]^T$, in which $\|\cdot\|$ denotes the Euclidean norm of column vectors $\mathbf{e}_{i,n}$ or rows in the matrix of $\mathbf{E}_n$. This outer iterative training process continues until the stopping condition is met, i.e. $\text{nRMSE}_n \leqslant \epsilon$ or $m_{1,n} \geqslant m_1$, in which ϵ is a preset model performance threshold value and $m_{1,n}$ is the number of centers at the end of the n^{th} iteration.

2) THE INNER ITERATIVE PROCESS - CALCULATIONS OF RBF CENTERS AND WIDTHS

During the nested inner iterative process of the n^{th} outer iteration, the squared residual norm ($\|\mathbf{E}_{n-1}\|^2$) obtained in the $(n-1)^{\text{th}}$ iteration are employed to estimate RBFs centers and widths, by regarding residual local extreme points as RBFs centers and utilizing a preset residual norm threshold (e_{th}^2) to locate RBFs boundaries and thus obtaining the RBFs widths. By doing these, the obtained several RBFs are deemed to be able to compensate the residuals, thus improve the model accuracies. The detailed process is described as follows.

The input training data set is firstly divided into two sets according to the residual norm vector $\|\mathbf{E}_{n-1}\|^2$ and the preset threshold value e_{th}^2. Here in this paper, we set $e_{\text{th}}^2 = 0.1(\max(\|\mathbf{E}_{n-1}\|^2) - \min(\|\mathbf{E}_{n-1}\|^2)) + \min(\|\mathbf{E}_{n-1}\|^2)$. The first set contains those input data points ($\mathbf{X}_{\text{nz}} = \{\mathbf{x}_{\text{nz},i} | i = 1, 2, \cdots\}$) with non-zero amplitude residual norms recognized by $\|\mathbf{E}_{n-1}\|^2 \geqslant e_{\text{th}}^2$, which correspond to points of residual extremes and their neighborhoods, and are used to locate RBF centers; The second set contains those input data points ($\mathbf{X}_{\text{z}} = \{\mathbf{x}_{\text{z},i} | i = 1, 2, \cdots\}$) with zero amplitude residual norms recognized by $\|\mathbf{E}_n\|^2 < e_{\text{th}}^2$, which correspond to points far from RBF centers beyond distances proportional to RBF widths, and are used to calculate RBF widths as the distances between centers and their nearest points in $\mathbf{X}_{\text{z}}$ multiplied by a constant factor λ (see section II-B3).

The point in $\mathbf{X}_{\text{nz}}$ with the maximum residual norm is recognized as one RBF center (denoted as $\boldsymbol{c}_{n,p} = \mathbf{x}_{\max}$, in which the subscript $p = 1, 2, \ldots P_n$ is the inner iteration number), and the shortest distance ($d_{\text{mz,min}} = \min(\mathbf{d}_{\text{mz}})$) between $\mathbf{x}_{\max}$ and $\mathbf{X}_{\text{z}}$ is used to calculate the corresponding RBF width as $\sigma_{n,p} = \lambda d_{\text{mz,min}}$. If we define the distance between $\mathbf{x}_{\max}$ and $\mathbf{X}_{\text{nz}}$ as $\mathbf{d}_{\text{mn}}$, then the neighboring points of $\mathbf{x}_{\max}$ correspond to those with $\mathbf{d}_{\text{mn}} < d_{\text{mz,min}}$. Following specifications of the center $\boldsymbol{c}_{n,p} = \mathbf{x}_{\max}$ and width $\sigma_{n,p} = \lambda d_{\text{mz,min}}$, the point $\mathbf{x}_{\max}$ and its neighbors within the radius of $d_{\text{mz,min}}$ can be removed from the set of $\mathbf{X}_{\text{nz}}$ (the corresponding residual norms in $\|\mathbf{E}_{n-1}\|^2$ are discarded too), and new updated $\mathbf{X}_{\text{nz}}$ and $\|\mathbf{E}_{n-1}\|^2$ can be obtained. In order to improve the model training speed, more centers and widths can be identified iteratively in this inner iterative process by repeating the steps described above, until all points in $\mathbf{X}_{\text{nz}}$ are processed and discarded, or a predetermined number of inner iterations are finished.

3) MULTIPLYING FACTOR (λ) OF RBF WIDTHS

If the error surface (in $\|\boldsymbol{E}\|^2$ in this paper) could be represented by an RBF in some ideal situation, then the width can be measured as the distance from the center to some peripheral point. In our REN algorithm, the boundaries are arbitrarily determined based on the predetermined residual norm threshold (some percentage of the maximum norm). The minimal distance between the center and boundaries is of course not equal to the true width value, but should be proportional to it and could be used to estimate the true value by multiplying a constant factor. In order to achieve more accurate RBF width values and thus improve the RBF model performance, the proposed REN training algorithm computes RBF widths $\sigma_{n,p}$ as the minimal distances ($d_{\text{mz,min}}$) between centers and boundaries multiplied by a constant factor λ, i.e. $\sigma_{n,p} = \lambda d_{\text{mz,min}}$, with $d_{\text{mz,min}}$ tracing the varied residual components to be compensated for and λ scaling $d_{\text{mz,min}}$ to some extents more suitable for residual compensations. The width multiplying factor (WMF) λ may be related to many aspects or parameters in optimizing model performance, and one of them is obviously the parameter of residual norm threshold. It can be inferred that, for a fixed residual norm threshold, λ would remain unchanged to maximize the model performance. The suitable value of λ is determined by performing a series of model training and testing experiments with different values of it and selecting the one with the best model performance. It is shown that the suitable values of λ are between 3~10 in our study, and that, once specified experimentally, it does not need to be changed any more in further training applications of the REN algorithm in different tasks.

C. SUMMARIZED STEPS OF THE REN ALGORITHM IMPLEMENTATION

The computing process in one outer iteration of the REN algorithm is summarized as follows:

1) Initialize $\|\mathbf{E}_0\|^2 = \|\mathbf{Y} - \hat{\mathbf{Y}}_0\|^2$ and $\text{nRMSE}_0 = \sqrt{\frac{\|\mathbf{E}_0\|_F^2}{\|\mathbf{Y}\|_F^2}} = \sqrt{\frac{\sum_i \sum_k e_{i,k,0}^2}{\sum_i \sum_k y_{i,k}^2}}$, in which $\mathbf{Y}_0$ can be simply set to 0, or the average or a linear fitting of the input data.
2) During the first step in the $n^{\text{th}}(n \geqslant 1)$ outer iteration, compute $\boldsymbol{c}_{n,p}$ and $\sigma_{n,p}$ through the inner iterative process as described in the next paragraph by using $\|\mathbf{E}_{n-1}\|^2$ returned from the $(n-1)^{\text{th}}$ outer iteration. All obtained centers and widths then constitute $\mathbf{C}_n$ and $\boldsymbol{\sigma}_n$.

3) During the second step in the $n^{\text{th}}(n \geqslant 1)$ outer iteration, compute the hidden layer outputs $\boldsymbol{\phi}_n$ based on centers ($\mathbf{C}_n$), widths ($\boldsymbol{\sigma}_n$), and network inputs ($\mathbf{X}$); compute the output layer weights ($\hat{\boldsymbol{\Omega}}_n$) through $\boldsymbol{\phi}_n \boldsymbol{\Omega}_n^T = \mathbf{Y}$ and the least squares algorithm; compute the model residual norm $\|\mathbf{E}_n\|^2 = \|\mathbf{Y} - \hat{\mathbf{Y}}_n\|^2$ and the network performance $\text{nRMSE}_n = \sqrt{\frac{\|\mathbf{E}_n\|_F^2}{\|\mathbf{Y}\|_F^2}} = \sqrt{\frac{\sum_i \sum_k e_{i,k,n}^2}{\sum_i \sum_k y_{i,k}^2}}$, in which $\hat{\mathbf{Y}}_n = \boldsymbol{\phi}_n \hat{\boldsymbol{\Omega}}_n^T$.
4) If $\text{nRMSE}_n > \epsilon$ and $m_{1,n} < m_1$, then repeat the above process from step 2; otherwise, the computations of centers $\mathbf{C}$ and widths $\boldsymbol{\sigma}$ are completed, and $\mathbf{C} = \mathbf{C}_n$, $\boldsymbol{\sigma} = \boldsymbol{\sigma}_n$, and $\boldsymbol{\Omega} = \hat{\boldsymbol{\Omega}}_n$ are returned.

The inner iterative process of updating RBF centers and widths of the REN algorithm, implemented in the second step in the paragraph above for the n^{th} outer iteration, is summarized as follows:

1) Divide the input data set $\mathbf{X}$ into two sets ($\mathbf{X}_{\text{nz}}$ and $\mathbf{X}_{\text{z}}$) based on the residual norm $\|\mathbf{E}_{n-1}\|^2 = [\|\mathbf{e}_{1,n-1}\|^2, \|\mathbf{e}_{2,n-1}\|^2, \cdots, \|\mathbf{e}_{N,n-1}\|^2]^T = \|\mathbf{Y} - \hat{\mathbf{Y}}_{n-1}\|^2$ and the preset threshold (e_{th}^2), with $\mathbf{X}_{\text{nz}} = \left\{ \mathbf{x}_{\text{nz},i} = \underset{\mathbf{x}_{\text{nz},i}}{\arg}(\|\mathbf{E}_{n-1}\|^2 \geqslant e_{\text{th}}^2) | i = 1, 2, \cdots \right\}$ containing points of non-zero amplitude residual norms and

$$\mathbf{X}_z = \left\{ \mathbf{x}_{z,i} = \underset{\mathbf{x}_{z,i}}{\arg}(\|\mathbf{E}_{n-1}\|^2 < e_{\text{th}}^2) | i = 1, 2, \cdots \right\}$$

containing points of zero amplitude residual norms.
2) Identify one RBF center as $\boldsymbol{c}_{n,p} = \mathbf{x}_{\max} = \underset{\mathbf{x}}{\arg\max} \left(\|\mathbf{E}_{n-1}\|^2 |_{\mathbf{x} \in \mathbf{X}_{\text{nz}}} \right)$ (with $p = 1, 2, \ldots P_n$ being the inner iteration number) if this $\mathbf{x}_{\max}$ point has not ever been identified as a center before, and the corresponding RBF width as $\sigma_{n,p} = \lambda d_{\text{mz,min}}$ with $d_{\text{mz,min}}$ being the shortest distance ($d_{\text{mz,min}} = \min(\mathbf{d}_{\text{mz}})$) between $\mathbf{x}_{\max}$ and $\mathbf{X}_{\text{z}}$ ($\mathbf{d}_{\text{mz}} = \{ d_{\text{mz},i} = \left\| \mathbf{x}_{\max} - \mathbf{x}_{\text{z},i} \right\| | i = 1, 2, \cdots \}$).
3) Compute distances

$$\mathbf{d}_{\text{mn}} = \left\{ d_{\text{mn},i} = \left\| \mathbf{x}_{\max} - \mathbf{x}_{\text{nz},i} \right\| | i = 1, 2, \cdots \right\}$$

between $\mathbf{x}_{\max}$ and points in $\mathbf{X}_{\text{nz}}$.
4) Update $\mathbf{X}_{\text{nz}} \leftarrow \left\{ \underset{\mathbf{x}_{\text{nz},i}}{\arg}(\mathbf{d}_{\text{mn}} \geqslant d_{\text{mz,min}}) \right\}$ and $\|\mathbf{E}_n\|^2 \leftarrow \|\mathbf{E}_n\|^2 |_{\mathbf{x} \in \mathbf{X}_{\text{nz}}}$.
5) Repeat the above steps, until $\mathbf{X}_{\text{nz}}$ becomes empty or P_n is equal to a preset value.

D. AN EXAMPLE OF FUNCTION APPROXIMATION WITH THE REN ALGORITHM - DEMONSTRATION OF THE IMPLEMENTATION STEPS

When used for the function approximation task (the example function waveform is as shown in Fig. 2d-g), the data generated in each stage during the implementation of the REN algorithm is shown in Fig. 2. All

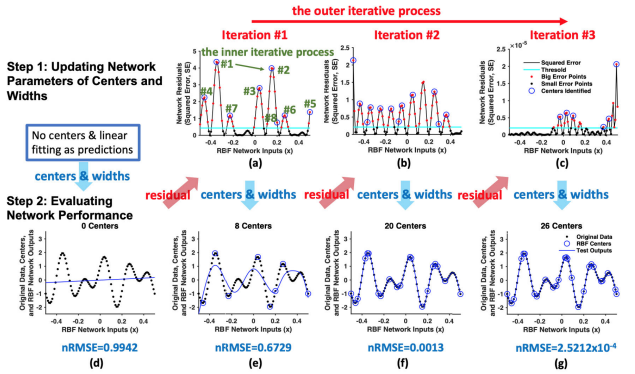


FIGURE 2. Results of three iterations when implementing the REN algorithm to train an RBF network to approximate the function as shown in d-g. Figures in each column are for one outer iteration, with the bottom left most figure providing the initial conditions for the learning procedure. The top row figures show the results of updating network parameters (step #1 in outer iterations), in each of which the centers are identified iteratively through the inner iterative process based on the provided residuals, and marked by blue circles. The bottom row figures show the results of evaluating network performance (step #2 in outer iterations) based on the obtained centers and widths, below each of which the computed nRMSE is displayed. The legends for the top and bottom row figures are only shown in the last figures respectively.

the programs, for this example and others in this paper, are implemented in the MATLAB environment on a macOS MacBook Pro (2.8 GHz Intel Core i7 processor and 16 GB RAM).

At start of the REN training procedure, no centers are selected and the model outputs are computed just based on the linear fitting of the training data (Fig. 2d). Model residuals are the squared errors between the expected and real model outputs (Fig. 2a). By comparing the obtained residuals with a preset threshold (the cyan line in Fig. 2a-c), the training data points are divided into two sets, namely the supra-threshold points (i.e. residual extreme points and their neighborhoods, indicated by red dots in Fig. 2a-c) and sub-threshold points (i.e. points beyond RBF widths, indicated by dark dots in Fig. 2a-c). The point with the highest residual amplitude is recognized as one center, and its nearest distance to sub-threshold points is used to compute the corresponding width. This center point and its neighborhoods are then removed for further recognition of centers by repeating the above process. Eight centers in all (Fig. 2a) are recognized in this inner iterative process. With the obtained centers and widths, the RBF model is updated and new outputs can be obtained (Fig. 2e). At this time point (the end of outer iteration), the model performance (nRMSE) is evaluated to determine whether a further iteration would be required. If nRMSE is greater than a predetermined threshold, and the number of centers has not reached its preset maximum number, another iteration will be initiated with the residuals as inputs. In this example, after three iterations, 26 centers in all are added into the RBF model, and the nRMSE is reduced from 0.9942 to $2.5212\text{x}10^{-4}$. In fact, after 2 iterations, the result of 20 centers and nRMSE=0.0013 has already given a satisfactory fitting waveform (Fig. 2f).

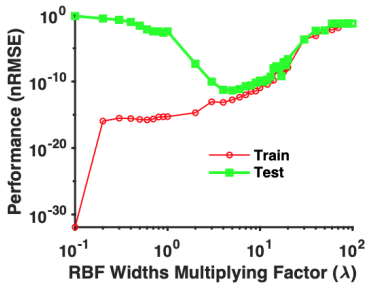


FIGURE 3. Influences of the REN algorithm width multiplying factor λ on the model performance. The train and test data are extracted from a sinusoidal signal of two cycles (41 and 401 points respectively).

III. RESULTS

Firstly, by taking function approximations as examples, the characteristics and advantages of the REN algorithm related to some inherent parameters are demonstrated; Then, in the example of chirp and peaks signal approximations, the effectiveness of the REN algorithm for approximating multi-frequency and multi-dimensional signals is shown; Finally, the task of breast cancer data classification is performed to demonstrate the feasibility of the REN algorithm in training RBF models to deal with practical classification problems. In addition, since the OLS method has displayed good performance in training RBF networks due to its ability of selecting relatively optimal centers [19], it is also used in this paper to train RBF networks in all approximation and classification tasks for comparison purposes. The OLS method for RBF network training has a function implementation (newrb.m) in MATLAB Neural Networks toolbox.

A. INFLUENCES OF SOME INHERENT PARAMETERS OF THE REN ALGORITHM ON RBF NETWORK PERFORMANCES

There are some inherent parameters involved in the REN algorithm that is closely related to the algorithm's training performance. These include the RBF widths multiplication factor (λ), RBF widths ($\boldsymbol{\sigma}$), and the number of centers added in each iteration (P_n). Once specified, their values do no need to be changed in further practical applications.

1) INFLUENCES AND DETERMINATIONS OF THE CONSTANT RBF WIDTH MULTIPLICATION FACTOR (λ)

As described in the Methods section, the REN algorithm automatically calculates RBF widths $\sigma_{n,p} = \lambda d_{\mathrm{mz,min}}$, where the multiplying factor λ is a preset constant relating the calculated distance $d_{\mathrm{mz,min}}$ to the actual width value. Thus, the value of λ directly affects the performance of the trained networks. To determine suitable values of λ, here the REN algorithm is applied to approximating a sinusoidal signal of two cycles with 41 points (the testing data is the 10 times sampled data points). The results (Fig. 3) show that small λ values ($< 2\text{x}10^0$) result in over fitted models (high training accuracies and low testing accuracies), and large λ values ($> 2\text{x}10^1$) the poorly trained models (low training and testing accuracies). With medium values of λ (about $2\text{x}10^0 < \lambda < 2\text{x}10^1$), the best results are obtained. Fortunately, once specified, the value of λ does not need to be changed for the REN algorithm to perform successfully in different approximation or classification tasks. In practical applications, λ can be set to some values in the range of 3˜10.

TABLE 1. Schemes of parameter combinations for demonstrating the two functionalities realized by the REN algorithm.

REN Schemes	Number of Centers Added in Each Outer Iteration ($\mathrm{P_n}$)	Widths ($\sigma_{\mathrm{n,p}}$)
REN 1	1	fixed
REN 2	1	computed
REN 3	computed	computed
REN 4	computed	fixed

2) TWO SUPERIOR PROPERTIES (P_n AND $\sigma_{n,p}$) REALIZED BY THE REN ALGORITHM AND THEIR INFLUENCES ON MODEL PERFORMANCES

The proposed REN algorithm realizes two major functionalities in determining centers and widths. One is that more than one centers ($P_n \geqslant 1$) can be added one time through the inner iterative process, and the other is that the RBF width ($\sigma_{n,p}$) for each center is determined automatically by computations. To demonstrate the important roles of these two aspects in improving RBF network performances and training speeds, the REN algorithm is applied to approximating the function as shown in Fig. 2d-g, by using different combination schemes of the parameter values (specifically, i.e. P_n is limited to 1 or not, and $\sigma_{n,p}$ is fixed or not). The four possible combination schemes are named REN 1, REN 2, REN 3, and REN 4 respectively here (see Table 1). The scheme REN 3 has complete inherent properties of the REN algorithm, while the other three schemes limit one or two of the listed aspects. The performance curves of five models trained by the REN and OLS algorithms are compared on the testing data (10 times sampled points of the function in Fig. 2d-g) and the results are shown in Fig. 4.

The experiment results for 15 different width values in the range of 0.01˜0.29 are obtained, but only 4 performance curves of them are shown in Fig. 4. When employing constant width values, the REN 1 (add one center per outer iteration) and 4 (add multiple centers per outer iteration) schemes result in approximately the same performance curves as the OLS method for all RBF widths and number of centers (Fig. 4a-d). This implicates that the REN algorithm has the same basic effectiveness as the OLS method, when the parameters are limited to using fixed RBF widths and $P_n = 1$. However, when employing center-dependent widths calculated automatically in the REN algorithm, it is found as expected, that the REN 2 (add one center per outer iteration) and REN 3 (add multiple centers per outer iteration) schemes result in significantly better performance curves in contrast to the REN 1 and 4 schemes and the OLS method (Fig. 4a-d), which becomes especially apparent as more centers are incorporated in the networks (Fig. 4a-d). When less centers are

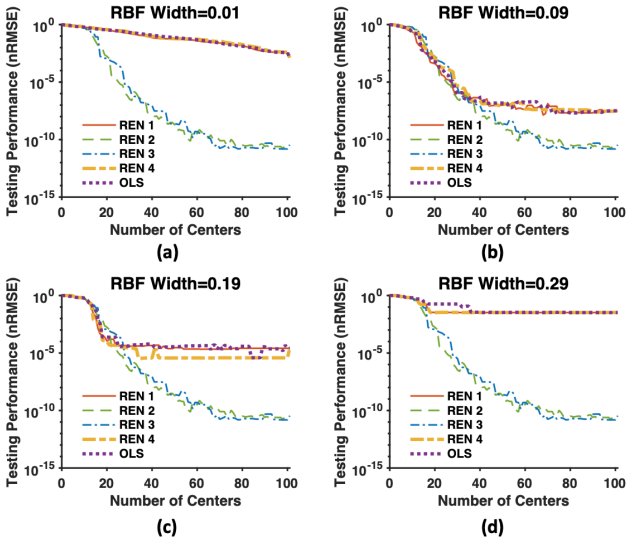


FIGURE 4. Comparison of the influences of the REN algorithm's two functional characteristics (i.e. adding multiple centers one time and determining width values for each center) on the model performance. The four subfigures show the results for different Gaussian RBF widths (RBF Widths=0.01, 0.09, 0.19, and 0.29). In each subfigure, the RBF width value used by the REN 1 and 4 and OLS algorithm is displayed above. It should be noted that the results of the REN 2 and 3 remain the same for different width values indicated (because the actual widths used are calculated automatically, not assigned manually), and they are drawn repeatedly in the four subfigures just for comparison purposes. The values of the relevant parameters, i.e. P_n (the number of centers added in each outer iteration) and $\sigma_{n,p}$ (the width value), are summarized in Table 1.

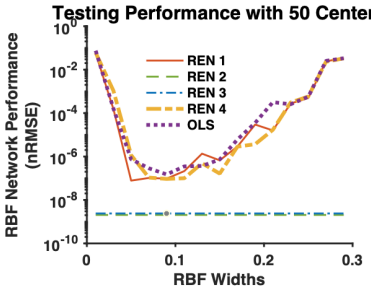


FIGURE 5. Testing performances (nRMSE) for the 50-center RBF networks obtained with different RBF widths. See Table 1 for explanations on the parameter specifications.

used, all schemes or methods here tend to result in similar performances (Fig. 4a-d).

In order to obtain the optimal width values for the REN 1 and 4 and OLS methods, and to see how their corresponding optimal performances are compared to those for the REN 2 and 3 schemes, the results for the 50-center models at 15 RBF width values between 0.01˜0.29 are shown in Fig. 5. It is seen that the results for the REN 1 and 4 and OLS methods are approximately the same at all widths, and reach their optimums only at medium width values (around 0.1 in this example) (Fig. 5). This leads to the problem of having to search for the optimal values of widths in a trial-by-trial manner, if the best performance needs to be achieved. In contrast, the schemes of REN 2 and 3, which compute RBF widths

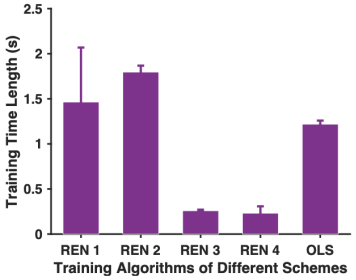


FIGURE 6. Comparison of training time lengths (mean±standard deviation) for the four REN schemes and OLS method. The statistics is made upon the 15 results corresponding to different RBF widths as described above. Note that the REN 3 and 4 schemes employ widths calculated automatically. See Table 1 for explanations on the parameter specifications.

automatically, do not have this problem and achieve the final results only in one training run. The REN 2 and 3 schemes result in performances two orders of magnitude better than the optimal ones obtained by the REN 1 and 4 and OLS methods (Fig. 5). These results clearly demonstrate that, by computing RBF widths that fit the training data better, the REN algorithm can significantly improve the RBF network performance.

The training time lengths consumed by the REN and OLS algorithms are shown in Fig. 6. It is seen that the REN 3 and 4 schemes (adding multiple centers per outer iteration) need significantly less time than the others. Moreover, from the above results (Figs. 4 and 5), we can see that the REN 3 and 4 schemes result in performance results similar to REN 2 and 1 respectively, implicating that accelerating the training speed by identifying multiple centers at one time has no negative effects on the model performance. In addition, by automatically calculating suitable RBF widths from the training data, the REN algorithm avoids performing multiple training runs to select the most appropriate width, thus further reducing the time lengths required for the construction of RBF networks.

3) THE INFLUENCE OF THE UPPER LIMIT (P') OF P_n ON THE RBF MODEL PERFORMANCE AND TRAINING SPEED

In order to get a further understand of how P_n impacts the model performance and training speed, an upper limit (P') is applied to it in the REN algorithm, such that the number of centers added in each outer iteration will not exceed that limit. The same function approximation process is conducted as above, and 28 different values of P' between 1 and 100 are used for RBF networks training. The results are shown in Fig. 7. It can be seen that the standard deviations of the 28 performance curves are relatively small (Fig. 7a). Thus, P' has little impacts on both training and testing accuracies of RBF networks. On the other hand, the training speed displays great sensitivities to P' (Fig. 7b). Too small P' values (i.e. adding fewer centers each time) result in apparently longer training time lengths (Fig. 7b), which are significantly and rapidly decreased as larger P' values are used (Fig. 7b). Relaxing P' to some certain values can reduce the training

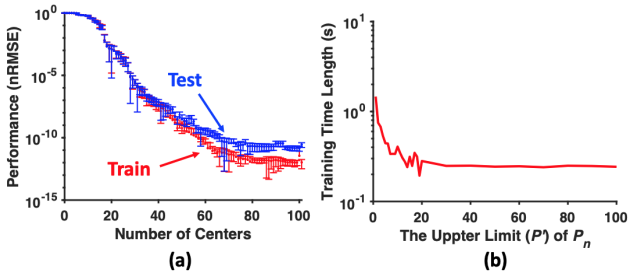


FIGURE 7. Influences of the upper limit (P') of P_n on the model performance and training speed. Model training and testing are conducted for 28 different values of P' between 1 and 100. The results in (a) are expressed as mean±(standard deviation) of the 28 results. (b) shows the training time lengths, when using different P' to train a 100-center RBF network.

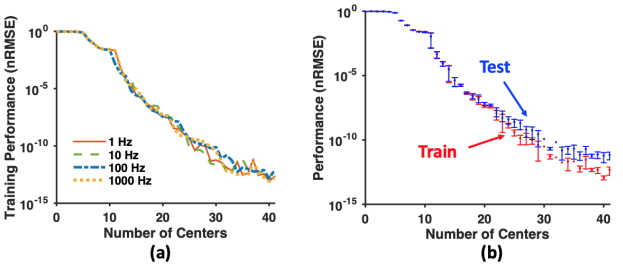


FIGURE 8. Approximation performances for the sinusoidal signals of 1, 10, 100, and 1000 Hz with the REN algorithm. (a) Training performances. (b) Mean±(standard deviation) of the four model results. All sinusoidal signals have two cycles of 41 samples. The testing data are the 10 times sampled points. The REN algorithm are applied without any adjustments during training the four models.

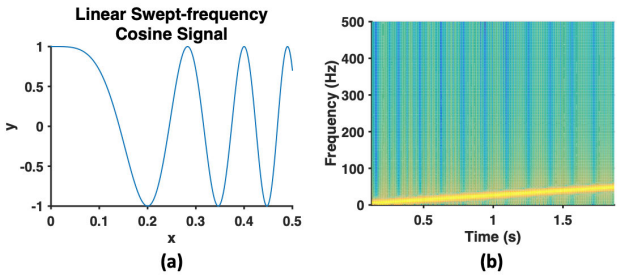


FIGURE 9. The cosine (chirp) signal of linear swept-frequencies. (a) time varying signal shape (only 0.5 of 2 s is shown here for the clarity purpose). (b) the corresponding time-frequency spectrum. The chirp signal frequency increases linearly from 0 Hz (at 0 s) to 50 Hz (at 2 s). Sampling rate: 1 kHz. The testing data are the randomly sampled 2001 data points in the range of 0~2 s.

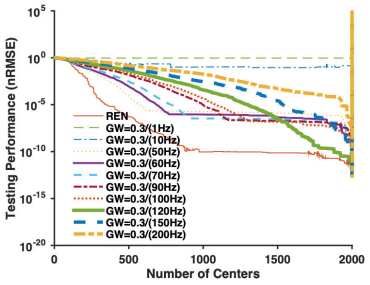


FIGURE 10. The testing performance curves for the chirp signal (0~50 Hz, 2 s, 1 kHz, as shown in Fig. 9) approximations with the REN and OLS algorithms. The OLS algorithm is implemented with various Gaussian RBF width values (GW=0.3/(1~200 Hz)=0.0015~0.3).

time lengths to a minimum value while maintaining the same model accuracies. It should be noted here that, in each iteration of the REN algorithm, the value of P_n ($\leqslant P'$) will be determined automatically according to residuals, and is iteration specific and probably smaller than P'. In practical applications of the REN algorithm, P' can be simply set to infinity, or to the maximum number of centers allowed by the trained RBF network model.

B. APPROXIMATION OF SIGNALS OF MULTIPLE FREQUENCY COMPONENTS

1) INSENSITIVITIES OF THE REN ALGORITHM TO SIGNAL FREQUENCIES

Since the REN algorithm is designed to automatically calculate RBF widths, the obtained model performance is supposed to be insensitive to the shapes or frequencies of signals. In order to verify this, the approximation of sinusoidal signals with different frequencies (1, 10, 100, and 1000 Hz) are studied here. All the sinusoidal signals consist of two cycles with a total of 41 points. The testing points are those 10 times sampled (thus 401 points in all). The results show that, although used without any adjustments, the REN algorithm results in almost the same training performance curves for all signals of different frequencies (Fig. 8a). Further statistical results (mean±standard deviation of the four model performances) display small variances for both the training and testing performance curves (Fig. 8b). The testing performance is a little poorer than the training performance when more centers are incorporated, but still smaller than 10^{-10} (Fig. 8b). When less centers are used, the two curves are almost the same. This result show that the REN algorithm is insensitive to frequencies and may well be used for approximations of signals with multi-frequency components. This will be further demonstrated and verified by the approximations of chirp and peaks signals next.

2) FAST CONVERGENCE AND BETTER PERFORMANCE OF THE REN ALGORITHM IN APPROXIMATING CHIRP SIGNALS

In order to determine the performance differences between the REN (automatically calculating center specific RBF widths σ) and OLS algorithms (using the fixed RBF width value) in dealing with multiple-frequency signals, the approximations of a chirp signal (0-50 Hz, 2 s) by using the two algorithms are carried out. For the OLS algorithm, various Gaussian RBF widths (GW) are tested. The chirp signal and its time-frequency spectrum are shown in Fig. 9.

The obtained testing performance curves (Fig. 10) show that the REN algorithm results in a faster and better (low nRMSE values) RBF network training, while the results obtained by the OLS algorithm with various width values exhibit either slow converging speeds or poor performances

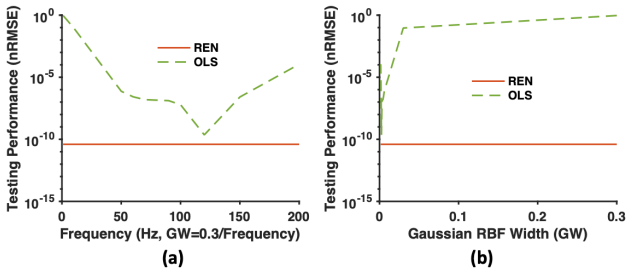


FIGURE 11. The testing performances for the 1800-center networks obtained by using the REN and OLS algorithms. The REN algorithm automatically calculates the center-specific RBF widths, and the OLS algorithm is implemented with various fixed RBF width values. These results correspond to those for 1800 centers in Fig. 10.

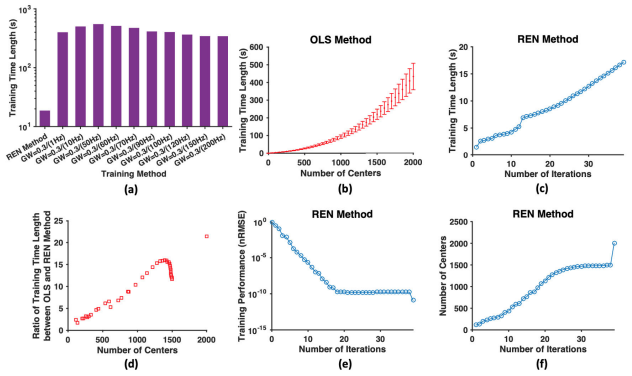


FIGURE 12. Training time lengths required by the REN and OLS algorithms in the chirp signal (0~50 Hz, 2 s, 1 kHz, Fig. 9) approximation task. (a) Training time lengths for constructing the 2001-center models. (b) Training time lengths (mean±standard deviation) averaged over those obtained by the OLS algorithm with various Gaussian RBF width values. (c) Training time lengths for different number of iterations of the REN algorithm. (d) Ratios between the training time lengths for the OLS and REN algorithms versus the number of centers of the constructed models. (e) Training performance (nRMSE) versus the number of iterations for the REN algorithm. (f) Number of centers acquired versus the number of iterations for the REN algorithm. The corresponding performance curves are shown in Fig. 10.

(relatively big nRMSE values). To achieve a specific level performance, the REN algorithm leads to a network with only approximately half number of centers of those obtained by the OLS algorithm (Fig. 10), implicating that more compact models can be obtained by using the REN algorithm. In addition, as shown in Fig. 10 and 11, only in a very narrow range of width values (0.0021-0.0030), the OLS algorithm gets its best performance (by taking the training results of the 1800-center networks for examples in this approximation task for the linear swept-frequency chirp signal). This implicates that more runs of training will be required to find the best width value when adopting the fixed RBF width training strategy. While, in contrast, the REN algorithm obtains much better results in one training run without any specific parameter settings or adjustments.

3) LESS TRAINING TIME AND NUMBER OF CENTERS ACHIEVED BY THE REN ALGORITHM

The training time lengths for approximating the chirp signal are shown in Fig. 12. It is seen that the REN algorithm completes the training procedure in about 17 s ending with the 2001-center model (Fig. 12a, c, and f). In fact, the REN algorithm identifies all significant centers (i.e. about 1000 out of the 2001 training points) and achieves a stationary performance of nearly the smallest nRMSE within much less number of iterations (18 iterations out of 39 in all) (Fig. 12e), and this takes only about 8 s training time (Fig. 12e and c). In contrast, the OLS algorithm requires about 430 s to complete one single training process (Fig. 12a and b, see Fig. 10 for the corresponding performance curves). The REN algorithm trains the chirp signal approximation model up to 53.75 times faster than the OLS algorithm while maintaining a better performance. Even though the same number of centers are employed for the comparison purpose, the REN algorithm still exhibits up to 22 times at most (in this task) faster training speeds than the OLS algorithm (Fig. 12.d). Moreover, considering that the training strategy like the OLS algorithm using fixed width values usually needs more time by repeating model training several times to acquire the optimal result, the REN algorithm proposed in this paper is more effective in reducing the training time due to its capability of training RBF models successfully in one run.

Furthermore, the efficiency of the REN algorithm is also at its ability of constructing more compact networks with less number of centers. Take the chirp signal approximation task here for an example, by employing half (1000) of the training data as centers (Figs. 10, 12e, and f), the REN algorithm achieves the same performance results as the OLS algorithm that uses all of the training data (2001) as centers. This means that the models obtained by the REN algorithm will be computationally efficient and fast in later prediction applications.

One commonly used criterion of stopping an RBF network training is to see if the model error is already small enough compared with a preset threshold. Different problems may have different tolerance. The results in Fig. 10 and 12 are redrawn in Fig. 13 to display the dependancies of training time lengths and number of centers on the expected performance for the REN and OLS methods. It is seen that, as the training performance increases (i.e., nRMSE values decrease), the training time lengths and number of centers increase rapidly for the OLS method, while remain relatively plat for the REN method. This implicates that the REN algorithm may work efficiently with larger data sets or higher performance requirements.

C. TWO DIMENSIONAL PEAKS FUNCTION APPROXIMATION

The peaks function (Fig. 14a) generates several peaks and valleys in the three-dimensional space and is commonly used to evaluate algorithms of approximations. Here, the performance of the REN and OLS algorithms are compared by approximating the peaks function. The peaks function is spatially regularly sampled (49×49 points) as the training data, and randomly sampled (also 49×49=2401 points) as the test data.

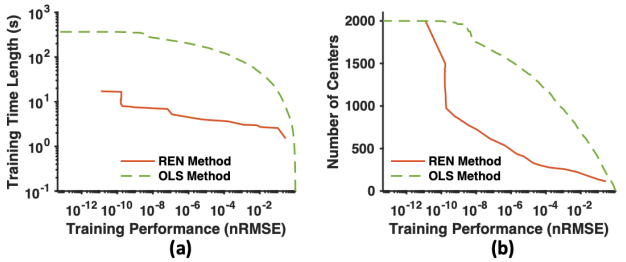


FIGURE 13. Training time lengths and the number of centers versus the training performance (nRMSE) for the REN and OLS methods.

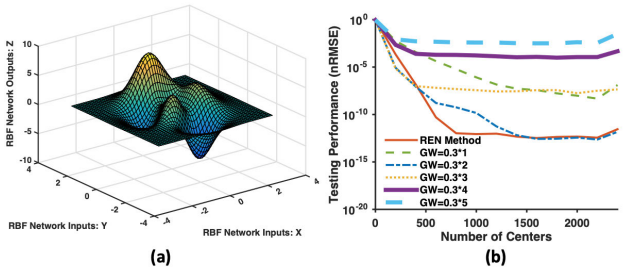


FIGURE 14. Approximation of the peaks function. (a) The peaks function shape. (b) The approximation performance by the REN and OLS algorithms. Peaks function: $z = 3(1-x)^2 e^{(-x^2)} - (y+1)^2 - 10(\frac{1}{5}x - x^3 - y^5)e^{(-x^2-y^2)} - \frac{1}{3}e^{(-(x+1)^2-y^2)}$. Train data: 49 × 49 spatially equally sampled points; Test data: 49 × 49 randomly sampled points.

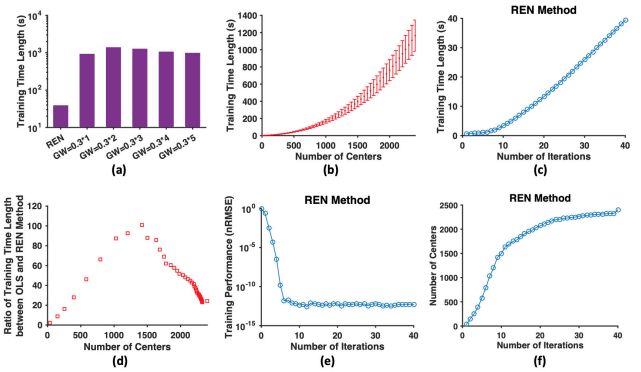


FIGURE 15. Training time lengths of the peaks function (Fig. 14a) approximation models by using the REN and OLS algorithms. (a) Training time lengths for constructing the 2401-center model. The OLS algorithm is implemented with 5 different Gaussian RBF widths. (b) Training time lengths (mean±standard deviation) averaged over those obtained by the OLS algorithm with 5 Gaussian RBF width values. (c) Training time lengths for different number of iterations of the REN algorithm. (d) Ratios between the training time lengths for the OLS and REN algorithms versus the number of centers of the constructed models. (e) Training performance (nRMSE) versus the number of iterations for the REN algorithm. (f) Number of centers acquired versus the number of iterations for the REN algorithm. The corresponding performance curves are shown in Fig. 14.

The results show that the OLS algorithm gives its best performance at the Gaussian RBF width of about GW=0.3 × 2=0.6 (Fig. 14b). In contrast, the REN algorithm, although achieving the same level of best results as the OLS algorithm in this task, results in faster convergence, thus exhibiting superior performance results for models of about 500-1500 centers (Fig. 14b).

Different from the approximation of the chirp signal for which the OLS method needs to incorporate all of the training data points as centers to achieve the best approximation results (Fig. 10), this time for the peaks function approximation, the OLS method reaches its best result by using only 1500 out of 2401 points as centers. This is probably because the variations of the peaks function (Fig. 14a) do not occur at all positions as the chirp signal (Fig. 9a). The corresponding training time at 1500 centers is about 400 s for the OLS method (Fig. 15b). The REN method achieves the same performance with approximately 794 centers (only about half of that for the OLS method) (Fig. 14b) at the 6th iteration (Fig. 15e and f), and the corresponding training time is 1.2358 s. Thus, the REN method is about 324 (400/1.2358) times faster than the OLS method for achieving the best performance in this peaks function approximation task.

D. CLASSIFICATION OF THE DISCRETE BREAST CANCER DATA

The above approximation tasks are conducted on the constructed, relatively more densely distributed, and noise free data to demonstrate the characteristics and superior properties of the REN algorithm in RBF network training. Real problems commonly encountered in the real world also include other situations, such as the classification of discrete patterns, in which many different patterns may correspond to one same target (class), and the same patterns acquired at different time points may be of different targets with certain probabilities. In order to test the feasibility and ability of the REN algorithm in dealing with the real classification problems of noisy and uncertain data, RBF networks are trained with the REN algorithm on the Breast Cancer Wisconsin (Original) Data Set obtained from the UCI Machine Learning Repository [32], and compared with the OLS algorithm.

The data set contains 683 samples (with 16 other unavailable ones discarded in our study) of 9 morphological features of breast tumor cells, and provides the labels of 2 (benign) or 4 (malignant). Each feature of a sample is an integer between 1 and 10. The 5-fold cross validation method is implemented, with all the samples randomly divided into 5 disjoint groups while maintaining constant ratios of sample numbers among categories in each group. When one of them is selected as the test set, the other four are used as the training set. Note that some other-fold cross validations are also tested for comparisons (see Table 2). The classification results obtained on the five test sets are averaged (mean±standard deviation) as indications of the model performance. The category of an input pattern is determined by the label of 2 or 4 which is nearest to the one-dimensional output of the RBF network. This method, instead of the commonly used m_2-output method, is used in this paper due to the purpose of simplicity and the fact that the RBF model implemented here is inherently an approximation model.

TABLE 2. Comparisons of classification results A_o of the REN algorithm with some other methods on the Breast Cancer data.

Method	Classification Accuracy A_o	Train Data Percentage (%)	Test Data Percentage (%)
RBFN [26], [33]	91.4	80	20
MCRN [33]	97.1	80	20
SVM 1 [33]	95.7	80	20
KNN (k=3) [33]	96.4	80	20
CRBF [34]	99 ± 0	50	25
ERBF [34]	97 ± 1	50	25
MLP [34]	98 ± 1	50	25
SVM 2 [34]	97 ± 0	50	25
NB [34]	98 ± 0	50	25
CART [34]	97 ± 0	50	25
RFT [34]	99 ± 0	50	25
SLIRBF [35]	96.6 ± 2.5	90	10
LR	95.9 ± 1.3	50	25
LR	96.1 ± 1.1	80	20
LR	96.1 ± 1.8	90	10
OLS	96.8 ± 1.2	50	25
OLS	97.2 ± 1.4	80	20
OLS	97.2 ± 2.0	90	10
REN	96.9 ± 0.7	50	25
REN	97.4 ± 1.4	80	20
REN	97.5 ± 1.5	90	10

The classification accuracies are displayed with different precision digits in this table, as they are extracted from different published papers.

The results of overall classification accuracies defined as

$$A_o = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

where TP=True Positive, TN=True Negative, FP=False Positive, FN=False Negative, and obtained by the REN and OLS algorithms and linear regression (LR) models in this paper are shown in Table 2 for comparisons with some other methods having been published. Many of the methods listed in the table are quite powerful for pattern classifications, but usually need to find the optimal and problem specific parameter values by using the exhaustive search or some specially designed methods [33], [34], [35]. The REN algorithm achieves classification accuracies comparable to those methods, and has the advantages of simple and fast one-run training. In addition to the overall classification accuracies, the results of classification accuracies (precisions) and recall rates on each class are also given as shown in Table 3. The classification accuracies on each class are defined as $p_c = \frac{TP}{TP+FP}$, and the recall rates as $r_c = \frac{TP}{TP+FN}$. It can be seen from Table 3 that the REN algorithm achieves results comparable to LR and OLS methods also in terms of p_c and r_c for each class.

The breast cancer data is actually highly linearly separable, which is implicated by the high classification accuracies (an average of 96.1%) for 0 number of centers in Table 2 and Fig. 16a that is obtained by using linear regression models. The REN algorithm results in classification accuracies comparable to the linear models (Fig. 16a), implicating that, for one or more center models, RBF width values suitable for this classification task have been identified and computed by the REN algorithm successfully (Fig. 16a). In contrast, the OLS algorithm leads to results depending upon RBF widths as before (Fig. 16b), which shows that, in this experiment, larger RBF width values result in higher classification accuracies (Fig. 16b), while smaller RBF widths result in complex

TABLE 3. Mean and standard deviation of the classification results on each class with LR, REN and OLS methods. In all, 50 classification results from all 5-fold testing data sets and corresponding models of 1~10 centers are used for REN and OLS methods respectively. The LR model gives rise to 10 classification results in all.

Method	Classification Precision		Recall Rates	
	class 2	classs 4	class 2	class 4
LR	95.5±3.2	96.5±5.3	98.2±2.7	91.3±6.1
OLS	96.8±2.4	95.7±4.4	97.7±2.4	94.0±4.6
REN	96.7±2.2	95.8±4.3	97.7±2.4	93.7±4.5

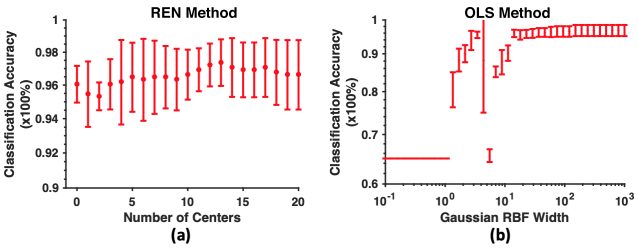


FIGURE 16. Classification accuracies of RBF networks on the breast cancer data obtained with the REN and OLS methods. (a) classification accuracies expressed as mean±(standard deviation) of the five test set results for 0~20 center models trained by the REN algorithm. (b) classification accuracies expressed as mean±(standard deviation) of the five test set results for 1-center models trained by the OLS algorithm by using different Gaussian RBF width values. The 5-fold cross validation method is used for achieving this result.

results with lower accuracies of about 65% occurring at two discontinuous ranges of width values for the one-center models (Fig. 16b). This implicates that, even for the linearly separable problems, the data may also exhibit complex RBF width dependent structures, and the REN algorithm can automatically calculate the width values effectively. Simply adopting large width values in algorithms employing fixed RBF widths, of course, could achieve good classification results for the linearly separable problems. However, for the nonlinearly separable problems, large width values would result in poor approximations at class boundaries and thus poor classification accuracies, and of course, would also lead to models of more centers, similar to those shown in Fig. 10 and 14b. These, however, need to be verified and explored extensively on more data sets with various distribution characteristics by further intensively designed experiments, although the results here tentatively demonstrate the feasibility and ability of the REN algorithm in classifying the breast cancer data.

IV. DISCUSSION

In this paper, a brand-new REN algorithm for training RBF neural network models is designed and implemented, and the corresponding characteristics and performance results are provided based on approximations of some artificially synthetic signals; also, a preliminary attempt of classification with the REN algorithm is made on a real breast cancer data and results comparable to some other methods are obtained, indicating that the REN training algorithm may be used in practical applications. Since this paper focuses mainly on the implementation and characteristics analysis of the REN algorithm and demonstration of its applicability, synthetic

one- and two-dimensional data with desired characteristics are employed to demonstrate and explain the use of the algorithm. At the same time, considering that different data sets may have different distribution characteristics and modeling demands, no models, generally speaking, are optimally applicable to all data sets. Thus, this paper employed only one real data set just to preliminarily demonstrate the feasibility of REN algorithm in solving practical classification tasks. We hope that in the future work, we can further analyze and determine the characteristics and performance of REN algorithm on more data sets with different distribution characteristics by comparing with other methods.

Application schemes of the REN algorithm need further consideration for problems of, for example, how to deal with noise and how to use it for classifications. Generally speaking, in the application of traditional machine learning methods, noise removal is completed in the data preprocessing stage, which requires a deep understand of the characteristics of the input data, so as to effectively remove noise and retain the useful information; the subsequent learning algorithms following noise removal, such as approximation or classification, can be considered to be implemented based on clean data samples, and only need to carry out effective and fast fitting or classification at the cost of some information loss. In this sense, the implementation of REN algorithm may not consider the effects of noise. But in specific applications, the REN algorithm capable of dealing with noise will of course give great benefits, and this kind of improvement and investigation will be one of our main works in the future. For the classification task, it is clear that the classifier, as obtained in this paper by simply approximating the input data, should not be the optimal one. In the further, we will intensely design a classifier based on the proposed REN algorithm more suitable for pattern recognition and classification applications.

The implemented REN training algorithm for RBF models in this paper has several important internal parameters, including the number of center points in each iteration, width, width factor and data division threshold. Section III-A explores the impacts of some parameters on the performance of the algorithm, and gives suggestions about the corresponding parameter values. Among them, the results about data set division thresholds are not given in the paper, which however, is shown to have a certain impact on the REN algorithm performance. Selecting a threshold value of 0.1 as adopted in this paper or dividing the dataset evenly in each iteration can usually give rise to a stable and good model performance. Other data set segmentation strategies in each iteration and their impacts on the model performance need further thorough investigations.

By means of automatically calculating the center point position and corresponding width, the proposed REN algorithm trains RBF models adaptively, thus without having to adjust parameter values specifically for different data sets. There are actually more ways to get the widths and center positions from the residual space. For example, the widths may also be calculated for each orthogonal basis, instead of each center point as done in this paper. Whether this would give improvements to the algorithm needs further investigation; for the calculation of center points, the information of their neighborhood including the number of neighbor points and the average value of the residual errors, etc, can be utilized. These treatments may eliminate the impacts of various types of noise, and have center positions not necessarily at the given data set points. The statistics made on the residual extreme neighborhood may provide more functions to the REN training algorithm. We would like to do more work on this topic in the near future.

The results in this paper show that the REN algorithm can train RBF models faster by identifying multiple center points in each iteration. Thus, determining center points as many as possible in one cycle may be a useful working direction to make full use of residual information to further improve the REN algorithm training speed. Additionally, the width factor corrects the computed distances in a certain scale, so that the width values obtained automatically by the REN algorithm can correspond to the true widths of the data set. However, in order to meet certain requirements, such as improving the anti-noise capability, eliminating high-frequency interference or components, or being inclined to model signal components in certain frequency ranges, adjusting width factor values appropriately may be necessary and reasonable. These also require deep detailed exploration and verification.

Finally, the proposed REN algorithm trains RBF models in a batch manner, which has the advantage of fast training, but will also require a large amount of memory and thus limit the ability of the algorithm to process larger data sets. In order for the REN algorithm to be used for larger data sets, and even online, one of our further work is to improve the REN algorithm to process data in a mini-batch or sequential manner.

V. CONCLUSION

The brand-new REN algorithm proposed in this paper is implemented based on the residual extreme points and their neighborhoods, and realized by a two-level iterative process. The outer iterations calculate the model performance and control whether to update the model. The inner iterative process, implemented within each outer iteration, calculates RBF centers and widths by locating the residual extreme points and estimating the size of their neighborhoods. The experimental results show that the REN algorithm is simple to implement and use, and fast and effective for training RBF networks. By reaching the same accuracies, the REN algorithm trains RBF networks 50 and 320 times faster, in the chirp and two-dimensional peaks signal approximation tasks respectively, than the OLS algorithm does, and the number of centers obtained by the REN algorithm is reduced by half. When incorporating the same number of centers, the REN algorithm achieves accuracies up to 3 orders of magnitude higher than the best results obtained by the OLS algorithm.

Two important characteristics are realized by the REN algorithm, including 1) adding multiple centers within each

cycle of data set processing in each outer iteration, and 2) calculating RBF widths separately for each center. When limited to adding only one center in each iteration and employing a constant RBF width, the REN algorithm achieves the same training speed and model performance as the OLS algorithm. When no limitation is applied, the two characteristics of the REN algorithm make it capable of obtaining faster training speeds and higher model performances. The RBF widths calculated by the REN algorithm make the basis functions adapt better to the changing signals, so that the obtained models are insensitive to data changes or frequencies of components. This gives the REN algorithm the ability of modeling those data that contains many distinct components.

The REN algorithm not only results in fast training speeds and high accuracies, but also tends to generate a more compact network. The number of centers is only half of that obtained by the OLS algorithm, implicating the effectiveness of the obtained centers and RBF widths. This will greatly promote the applications of the obtained models, since the more compactness means the less computation and faster application speed.

The implementation of the REN algorithm is simple. Less distance calculations are involved in the determinations of centers and RBF widths. This makes it capable of dealing with tasks that contain more data. In addition, the REN algorithm is easy to use, and the inherent parameters, that can be determined experimentally prior to applications, do not need any adjustments even for solving different problems. In addition to intentionally modifying parameter values to verify their effects, all tasks in this paper implement the REN algorithm with the same experimentally predetermined parameter values.

Due to the characteristics of being simple to implement and use, fast training speed, and being able to result in more accurate and compact RBF networks, the REN algorithm proposed in this paper may be used to solve problems with large amount of data or high performance requirements.

ACKNOWLEDGMENT

The authors would like to thank Dr. William H. Wolberg from the University of Wisconsin Hospitals, Madison, for providing the breast cancer database. They also wish to thank the handling editors and anonymous reviewers for their helpful comments and suggestions to improve the quality of this article.

REFERENCES

[1] M. Hou and X. Han, "Constructive approximation to multivariate function by decay RBF neural network," *IEEE Trans. Neural Netw.*, vol. 21, no. 9, pp. 1517–1523, Sep. 2010.

[2] J. Park and I. W. Sandberg, "Approximation and radial-basis-function networks," *Neural Comput.*, vol. 5, no. 2, pp. 305–316, 1993.

[3] C. Arteaga and I. Marrero, "Universal approximation by radial basis function networks of Delsarte translates," *Neural Netw.*, vol. 46, pp. 299–305, Oct. 2013.

[4] R. Zhang, J. Tao, R. Lu, and Q. Jin, "Decoupled ARX and RBF neural network modeling using PCA and GA optimization for nonlinear distributed parameter systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 2, pp. 457–469, Feb. 2018.

[5] H.-G. Han and J.-F. Qiao, "Adaptive computation algorithm for RBF neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 342–347, Feb. 2012.

[6] H. G. Han, W. Lu, Y. Hou, and J. F. Qiao, "An adaptive-PSO-based self-organizing RBF neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 104–117, Jan. 2018.

[7] H. A. Kingravi, G. Chowdhary, P. A. Vela, and E. N. Johnson, "Reproducing kernel Hilbert space approach for the online update of radial bases in neuro-adaptive control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1130–1141, Jul. 2012.

[8] T.-Y. Ho, C. S. Leung, P.-M. Lam, and T.-T. Wong, "Efficient relighting of RBF-based illumination adjustable images," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1987–1993, Dec. 2009.

[9] S. Ferrari, F. Bellocchio, V. Piuri, and N. A. Borghese, "A hierarchical RBF online learning algorithm for real-time 3-D scanner," *IEEE Trans. Neural Netw.*, vol. 21, no. 2, pp. 275–285, Feb. 2010.

[10] B.-H. Chen, S.-C. Huang, C.-Y. Li, and S.-Y. Kuo, "Haze removal using radial basis function networks for visibility restoration applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3828–3838, Aug. 2018.

[11] M. Robnik-Sikonja, "Data generators for learning systems based on RBF networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 5, pp. 926–938, May 2016.

[12] I. J. Sledge and J. C. Principe, "An exact reformulation of feature-vector-based radial-basis-function networks for graph-based observations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4990–4998, Nov. 2020.

[13] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Netw.*, vol. 14, nos. 4–5, pp. 439–458, 2001.

[14] L. Zhang, K. Li, H. He, and G. W. Irwin, "A new discrete-continuous algorithm for radial basis function networks construction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 11, pp. 1785–1798, Nov. 2013.

[15] F. Gianfelici, "RBF-based technique for statistical demodulation of pathological tremor," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 10, pp. 1565–1574, Oct. 2013.

[16] N. Wang, M. J. Er, and M. Han, "Large tanker motion model identification using generalized ellipsoidal basis function-based fuzzy neural networks," *IEEE Trans. Cybern.*, vol. 45, no. 12, pp. 2732–2743, Dec. 2015.

[17] H. Han, X. Wu, L. Zhang, Y. Tian, and J. Qiao, "Self-organizing RBF neural network using an adaptive gradient multiobjective particle swarm optimization," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 69–82, Jan. 2019.

[18] A. Alexandridis, E. Chondrodima, and H. Sarimveis, "Radial basis function network training using a nonsymmetric partition of the input space and particle swarm optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 219–230, Feb. 2013.

[19] 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.

[20] N. B. Karayiannis, "Reformulated radial basis neural networks trained by gradient descent," *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 657–671, May 1999.

[21] H.-G. Han, M.-L. Ma, and J.-F. Qiao, "Accelerated gradient algorithm for RBF neural network," *Neurocomputing*, vol. 441, pp. 237–247, Jun. 2021.

[22] J. Zhao, H. Wei, C. Zhang, W. Li, W. Guo, and K. Zhang, "Natural gradient learning algorithms for RBF networks," *Neural Comput.*, vol. 27, no. 2, pp. 481–505, Feb. 2015.

[23] W. Yao, X. Chen, and W. Luo, "A gradient-based sequential radial basis function neural network modeling method," *Neural Comput. Appl.*, vol. 18, no. 5, pp. 477–484, Jun. 2009.

[24] T. Xie, H. Yu, J. Hewlett, P. Rozycki, and B. Wilamowski, "Fast and efficient second-order method for training radial basis function networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 609–619, Apr. 2012.

[25] J. B. Gomm and D. L. Yu, "Selecting radial basis function network centers with recursive orthogonal least squares training," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 306–314, Mar. 2000.

[26] H. Yu, P. D. Reiner, T. Xie, T. Bartczak, and B. M. Wilamowski, "An incremental design of radial basis function networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 10, pp. 1793–1803, Oct. 2014.

[27] P. Singla, K. Subbarao, and J. L. Junkins, "Direction-dependent learning approach for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 203–222, Jan. 2007.

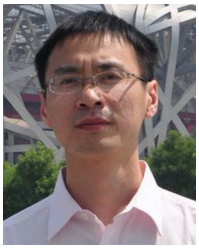
[28] Z.-R. Lai, D.-Q. Dai, C.-X. Ren, and K.-K. Huang, "Radial basis functions with adaptive input and composite trend representation for portfolio selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6214–6226, Dec. 2018.
[29] P. Reiner and B. M. Wilamowski, "Efficient incremental construction of RBF networks using quasi-gradient method," *Neurocomputing*, vol. 150, pp. 349–356, Feb. 2015.
[30] W. Yao, X. Chen, Y. Zhao, and M. van Tooren, "Concurrent subspace width optimization method for RBF neural network modeling," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 247–259, Feb. 2012.
[31] H. X. Huan, D. T. T. Hien, and H. H. Tue, "Efficient algorithm for training interpolation RBF networks with equally spaced nodes," *IEEE Trans. Neural Netw.*, vol. 22, no. 6, pp. 982–988, Jun. 2011.
[32] D. Dua and C. Graff, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2019. [Online]. Available: https://archive.ics.uci.edu/ml/citation_policy.html and http://archive.ics.uci.edu/ml
[33] A. O. Hoori and Y. Motai, "Multicolumn RBF network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 4, pp. 766–778, Apr. 2018.
[34] A. Alexandridis, E. Chondrodima, N. Giannopoulos, and H. Sarimveis, "A fast and efficient method for training categorical radial basis function networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2831–2836, Nov. 2017.
[35] P. A. Gutiérrez, C. Hervás-Martínez, and F. J. Martínez-Estudillo, "Logistic regression by means of evolutionary radial basis function neural networks," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 246–263, Feb. 2011.

LIFEI SUN received the B.S. and M.S. degrees in biomedical engineering from the Department of Electrical Engineering, Jilin University, Changchun, China, in 2000 and 2003, respectively. She is currently pursuing the Ph.D. degree with the Information Science and Technology College, Dalian Maritime University, Dalian, China.

She is currently an Associate Professor with the School of Intelligence and Electronic Engineering, Dalian Neusoft University of Information, Dalian. Her research interests include artificial intelligence and the development of embedding systems.

SEN LI received the B.S. degree from Liaoning University, Shenyang, China, in 1996, the M.S. degree in communication and information system from Dalian Maritime University, Dalian, China, in 1999, and the Ph.D. degree in signal and information processing from the Dalian University of Technology, Dalian, in 2010.

She is currently a Full Professor with the Information Science and Technology College, Dalian Maritime University. Her current research interest includes communication theory and systems.



HAILONG LIU (Member, IEEE) received the B.S. degree in biomedical engineering from the Department of Electrical Engineering, Jilin University, Changchun, China, in 2000, and the Ph.D. degree in biomedical engineering from the Institute of Biomedical Engineering, Xi'an Jiaotong University, Xi'an, China, in 2006.

He was a Postdoctoral Fellow with the Department of Pharmacology and the Department of Urology, University of Pittsburgh, Pittsburgh, PA, USA, from 2007 to 2009. He is currently an Associate Professor with the School of Biomedical Engineering, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian, China. His current research interests include the application and modeling of electrical nerve stimulation, biomedical signal processing, modeling and optimization, and artificial neural networks.

CHANGKAI SUN received the B.S. degree in medicine from Second Military Medical University, Shanghai, China, in 1986, and the master's degree in clinical neurology and the M.D./Ph.D. degree in human anatomy from Fourth Military Medical University, Xi'an, China, in 1995 and 1997, respectively.

He was a Postdoctoral Fellow with the Institute of Basic Medical Sciences, Academy of Military Medical Sciences, Beijing, China, from 2000 to 2003; and a Visiting Scholar with the Yue Laboratory of Neurophysiology, Institute of Biomedical Engineering, and the Najm Epilepsy Center, Institute of Neurology, Cleveland Clinic, Cleveland, OH, USA, from 2005 to 2007. He is currently the Chairperson and a Professor with the Research and Educational Center for the Control Engineering of Translational Precision Medicine, Faculty of Electronic Information and Electrical Engineering, School of Artificial Intelligence, Dalian University of Technology, Dalian, China. His current research interest includes the multidisciplinary innovation of rBNN-rBNN+-iANN about real and artificial neural networks.

LIPING QI received the M.S. degree in zoology from the Institute of Zoology, Chinese Academy of Sciences, Beijing, China, in 2003, and the Ph.D. degree in surgery and interventional science from University College London, London, U.K., in 2010.

From 2010 to 2013, she was a Postdoctoral Research Fellow with the University of Alberta, Edmonton, Canada. Since 2014, she has been an Associate Professor with the IC Technology Key Laboratory of Liaoning, Department of Biomedical Engineering, Dalian University of Technology, China. Her main research interests include the evaluation of muscle function during locomotion, biomechanics, and sports medicine. She is a member of the International Functional Electrical Stimulation Society. She was a recipient of the Dorothy Hodgkin Postgraduate Award, U.K., in 2006.

ZHIXUN SU (Member, IEEE) received the B.S. degree in mathematics from Jilin University, Changchun, China, in 1987, the M.S. degree in computer science from Nankai University, Tianjin, China, in 1990, and the Ph.D. degree in computational mathematics from the Dalian University of Technology, Dalian, China, in 1993. He has been a Professor with the School of Mathematical Sciences, Dalian University of Technology, since 1999. He is currently the Director of the Key Laboratory for Computational Mathematics and Data Intelligence of Liaoning Province and the Vice Director of the Liaoning Center for Applied Mathematics. His current research interests include computer graphics, computer vision, computational geometry, and machine learning.

CHANGSEN SUN received the B.S. and M.S. degrees from the Department of Electrical Engineering, Jilin University of Technology, in 1989 and 1992, respectively, and the Ph.D. degree in optical engineering from the Dalian University of Technology, Dalian, China, in 1999.

He is currently a Full Professor with the School of Optoelectronic Engineering and Instrumentation Science, Dalian University of Technology. His research interests include optical fiber sensor and its applications in engineering and medicine.