Neural Process Lett (2011) 33:251–265
DOI 10.1007/s11063-011-9176-y

Composite Function Wavelet Neural Networks with Differential Evolution and Extreme Learning Machine

Jiuwen Cao · Zhiping Lin · Guang-Bin Huang

Published online: 24 April 2011

Abstract In this paper, we introduce a new learning method for composite function wavelet neural networks (CFWNN) by combining the differential evolution (DE) algorithm with extreme learning machine (ELM), in short, as CWN-E-ELM. The recently proposed CFWNN trained with ELM (CFWNN-ELM) has several promising features. But the CFWNN-ELM may have some redundant nodes due to the number of hidden nodes assigned a priori and the input weight matrix and the hidden node parameter vector randomly generated once and never changed during the learning phase. The introduction of DE into CFWNN-ELM is to search for the optimal network parameters and to reduce the number of hidden nodes used in the network. Simulations on several artificial function approximations, real-world data regressions and a chaotic signal prediction problem show some advantages of the proposed CWN-E-ELM. Compared with CFWNN-ELM, CWN-E-ELM has a much more compact network size and Compared with several relevant methods, CWN-E-ELM is able to achieve a better generalization performance.

Keywords Wavelet neural networks · Composite functions · Differential evolution · Extreme learning machine

1 Introduction

Over the past two decades, wavelet neural network (WNN) has been discussed thoroughly both on network constructions and real-world applications [1–13]. Current research

J. Cao (✉) · Z. Lin · G.-B. Huang
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore
e-mail: caoj0003@e.ntu.edu.sg

Z. Lin
e-mail: ezplin@ntu.edu.sg

G.-B. Huang
e-mail: egbhuang@ntu.edu.sg

activities of WNN are mainly on how to design computationally efficient parameter initialization methods and learning algorithms. A number of methods have been reported in the previous works [1,4,5,11–13] on these aspects. Among existing training methods, the back-propagation (BP) [14] algorithm and its variants have been the most popular method adopted for WNN. Although these gradient-based algorithms are able to ensure the networks to achieve reasonable performances, they are still relatively slow in training phase and faced with several limitations such as stopping criteria, learning rate and local minima. To handle these problems, a tuning-free algorithm which is much faster than traditional gradient-based algorithms named extreme learning machine (ELM) was recently proposed by Huang et al. [15] for single-hidden layer feedforward networks (SLFN-ELM). In ELM, a Moore–Penrose (MP) generalized inverse [16] is adopted to analytically determine the network output weights after randomly choosing the input learning parameters. Hence, the overall computational time costed in the training phase is sharply reduced and usually hundred times smaller than some classical methods. Moreover, ELM is not only able to have better generalization performances than some traditional algorithms such as BP in most cases, but also easier to implement.

Recently, a novel structure of WNN named composite function wavelet neural network adopting ELM as the training algorithm (CFWNN-ELM) was proposed by Cao et al. [17]. Utilizing composite functions at the hidden nodes constituted by a type of bounded nonconstant piecewise continuous activation functions $g : \mathbf{R} \rightarrow \mathbf{R}$ and different wavelets function generated from the same mother wavelet with different dilation values and translation values, the CFWNN-ELM has shown several salient features, such as a faster learning speed and a better generalization performance in many applications than several relevant conventional methods [17].

However, it is believed that some redundant hidden nodes may exist in ELM and that the network parameters may not be optimal because the number of hidden nodes are assigned a priori and the hidden node learning parameters are randomly generated once and they never changed during the learning phase. During the past few years, evolutionary algorithms (EA) are widely used heuristic methods for function optimizations because they have many excellent features, such as easy implementation, good performance for many applications, flexibility, etc. Differential evolution (DE), a variant of evolutionary computing techniques proposed by Storn and Price [18], has been demonstrated that it can achieve more accurate performance than most evolutionary algorithms for optimizations. Due to this reason, DE has been adopted as a tool for searching the optimal parameters by many researchers in the machine learning field. Zhu et al. [19] proposed a hybrid approach that combines both DE and ELM for SLFN (in short, as SLFN-E-ELM) and Subudhi and Jena [20] considered another combination method that takes advantages of both DE and Levenberg–Marquardt algorithm [21] (in short, as DE-LM). In [19], it has shown that adopting DE in training SLFN is able to achieve better generalization performance with much more compact networks than many relevant methods. Moreover, the simulation results presented in [19] manifested that SLFN-E-ELM not only learns much faster but also achieves much better performances than DE-LM in model regressions and signal classifications.

Although CFWNN-ELM performs better than SLFN-ELM in many applications, it is also limited by the same problem encountered with SLFN-ELM mentioned above, that is, some redundant nodes may exist in CFWNN-ELM and the hidden node learning parameters may not be optimal as the number of hidden nodes assigned a priori and the hidden node learning parameters randomly generated once and they never changed during the learning phase. With this motivation, a **c**omposite function wavelet neural network learning with differential evolution and extreme learning machine (CWN-E-ELM) is proposed in this paper.

In CWN-E-ELM, a parameter initialization method that takes the input information range and the generated wavelet function support into account is first utilized to initialize the dilation and translation parameters of the wavelet functions. The DE algorithm is then adopted to select the nearly optimal input weights and wavelet dilation and translation values and the ELM algorithm is applied to train the network and determine the output weights. Consequently, some redundant nodes that play a minor role in the network can be removed.

The rest of this paper is organized as follows. Section 2 presents the preliminaries, including the structure of the composite function wavelet neural networks (CFWNN), the ELM algorithm and the DE algorithm. The details of CWN-E-ELM is given in Sect. 3. In Sect. 4, Experimental evaluations on the simulations of several regression problems and a chaotic signal prediction problem conducted by CWN-E-ELM will be compared with the results of some relevant methods: SLFN-E-ELM, CFWNN-ELM and support vector machine for regression (SVR). Discussions and conclusions are drawn in Sect. 5.

2 Preliminaries

For convenience of exposition, we briefly review the CFWNN, ELM and DE in the following subsections correspondingly.

2.1 Composite Function Wavelet Neural Network

As proposed in [17], the structure modifications of CFWNN compared with traditional ones are in two aspects. One is using a nonzero bias term x_0 in the input layer to ensure that the network will never stop learning in case all the input values are zero. The other modification is in the hidden layer, where the composite functions constituted by a type of bounded nonconstant piecewise continuous activation functions $g : \mathbf{R} \to \mathbf{R}$ and different wavelets function generated from the same mother wavelet with different dilation values a_h and translation values b_h are adopted as the activation functions.

For N arbitrary training samples $\left(\boldsymbol{x}^k, \boldsymbol{d}^k\right) \in \mathbf{R}^I \times \mathbf{R}^O$ and H hidden nodes, where $\boldsymbol{x}^k$ is the input vector and $\boldsymbol{d}^k = \left(d_1^k, d_2^k, \ldots, d_O^k\right)^T$ $(k = 1, 2, \ldots, N)$ is the O-dimensional corresponding desired output vector, we directly add the bias term x_0 into the input vector and the new formed kth input vector becomes $\hat{\boldsymbol{x}}^k = \left(x_0, x_1^k, x_2^k, \ldots, x_I^k\right)^T$. Then, the jth network output of $\boldsymbol{y}^k = \left(y_1^k, y_2^k, \ldots, y_O^k\right)^T$ after feeding forward the training sample $\hat{\boldsymbol{x}}^k$ can be summed as

$$y_j^k = \sum_{i=1}^{H} w_{ji}\, g\left(\psi_{a_i,b_i}\left(\mathbf{v}_i^T \hat{\mathbf{x}}^k\right)\right), \quad j = 1, 2, \ldots, O. \tag{1}$$

where $\boldsymbol{v}_i = (v_{i0}, v_{i1}, \ldots, v_{iI})^T$ $(i = 1, 2, \ldots, H)$ is the i th input weight vector and the input weight matrix $V \in \mathbf{R}^{H\times(I+1)}$ can be described in vector form as $V = (\boldsymbol{v}_1\, \boldsymbol{v}_2 \ldots \boldsymbol{v}_H)$, and $\boldsymbol{w}_j = \left(w_{j1}, w_{j2}, \ldots, w_{jH}\right)^T$ $(j = 1, 2, \ldots, O)$ is the jth output weight vector and the output weight matrix $W = (\boldsymbol{w}_1\, \boldsymbol{w}_2 \ldots \boldsymbol{w}_O)$. The functions ψ_{a_i,b_i}, called wavelets, are dilated and translated versions of a single mother wavelet function ψ

$$\psi_{a_i,b_i}(x) = |a_i|^{-1/2}\, \psi\left(\frac{x - b_i}{a_i}\right), \quad i = 1, 2, \ldots, H, \tag{2}$$

where a_i and b_i are the corresponding dilation and translation parameters respectively.

2.2 Extreme Learning Algorithm

Extreme learning machine was recently proposed for SLFNs by Huang et al. [15] and has been widely used in many real-world applications in [19,22]. Different from traditional theories that all the parameters of the feedforward neural networks need to be tuned to minimize the cost function, ELM theories claim that the hidden node learning parameters can be randomly assigned a priori. The SLFNs becomes a linear system when the hidden node parameters are randomly assigned and the network output weights can then be derived analytically by a simple generalized inverse of the hidden layer output matrix.

For N arbitrary distinct samples $(\boldsymbol{x}_k, \boldsymbol{t}_k) \in \mathbf{R}^n \times \mathbf{R}^m$, the standard SLFNs with L hidden nodes and an activation function $G(\cdot)$ are mathematically described as

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i G\left(\boldsymbol{x}_k; c_i, \boldsymbol{a}_i\right) = \boldsymbol{t}_k, \quad k = 1, 2, \ldots, N \tag{3}$$

where $c_i \in \mathbf{R}$ is the randomly assigned bias of the ith hidden node and $\boldsymbol{a}_i \in \mathbf{R}^n$ is the randomly assigned input weight vector connecting the ith hidden node and the input nodes. $\boldsymbol{\beta}_i$ is the weight vector connecting the ith hidden node to the output node. $G\left(\boldsymbol{x}_k; c_i, \boldsymbol{a}_i\right)$ is the output of the ith hidden node with respect to the input sample $\boldsymbol{x}_k$. Then (3) can be written as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{4}$$

where $\mathbf{H} = \{h_{ki}\} (k = 1, 2, \ldots, N \text{ and } i = 1, 2, \ldots, L)$ is the hidden-layer output matrix, $h_{ki} = G\left(\boldsymbol{a}_i^T \boldsymbol{x}_k + c_i\right)$ denotes the output of ith hidden node with respect to $\boldsymbol{x}_k$. $\boldsymbol{\beta} = \left(\boldsymbol{\beta}_1 \boldsymbol{\beta}_2 \ldots \boldsymbol{\beta}_L\right)_{m \times L}$ is the output weight matrix and $\mathbf{T} = (\boldsymbol{t}_1 \boldsymbol{t}_2 \ldots \boldsymbol{t}_L)_{m \times N}$ is the target output. To this end, the output weights can be analytically derived by finding the least-square solutions to the above linear system which is given as

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T} \tag{5}$$

where $\mathbf{H}^{\dagger}$ is the MP generalized inverse of the hidden layer output matrix $\mathbf{H}$.

2.3 Differential Evolution

A heuristic approach for minimizing possibly nonlinear and non-differentiable continuous space functions named differential evolution (DE) is proposed by Storn and Price [18], which is known as one of the most efficient evolutionary algorithms (EAs). The basic strategy of DE can be described in the following steps

– *Initialization*
  Randomly generate a set of D-dimensional parameter vectors

$$\left\{ \Theta_{i,G} \middle| \, i = 1, 2, \ldots, NP \right\} \tag{6}$$

  as a population at each generation G, where the initial vector populations should cover the entire parameter space. NP denotes the number of populations and dose not change during the minimization process.
– *Mutation*
  For each target vector $\Theta_{i,G}, i = 1, 2, \ldots, NP$, a mutant vector is created according to

$$\boldsymbol{v}_{i,G+1} = \Theta_{r_1,G} + F \cdot \left(\Theta_{r_2,G} - \Theta_{r_3,G}\right) \tag{7}$$

with random and mutually different indices $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$ and an amplification constant factor $F \in [0, 2]$, which controls the amplification of the differential variation $\left(\Theta_{r_2,G} - \Theta_{r_3,G}\right)$.

– *Crossover*
The crossover is introduced to increase the diversity of the perturbed parameter vectors. In this step, the D-dimensional trial vector

$$\boldsymbol{\mu}_{i,G+1} = \left(\boldsymbol{\mu}_{1i,G+1}, \boldsymbol{\mu}_{2i,G+1}, \ldots, \boldsymbol{\mu}_{Di,G+1}\right) \tag{8}$$

is formed, where

$$\boldsymbol{\mu}_{ji,G+1} = \begin{cases} \boldsymbol{v}_{ji,G+1} & \text{if } (\text{randb}(j) \leqslant \text{CR}) \\ & \text{or} j = \text{rnbr}(i) \\ \Theta_{ji,G} & \text{if } (\text{randb}(j) > \text{CR}) \\ & \text{and} j \neq \text{rnbr}(i) \end{cases} \tag{9}$$

$j = 1, 2, \ldots, D$ and $i = 1, 2, \ldots, NP$. In the above equation, randb(j) is the jth evaluation of a uniform random number generator with outcome in [0, 1], CR is the crossover constant in [0, 1] which has to be determined by the user. rnbr(i) is a randomly chosen index $\in \{1, 2, \ldots, D\}$ which ensures that $\boldsymbol{\mu}_{ji,G+1}$ gets at least one parameter from $\boldsymbol{v}_{ji,G+1}$.

– *Selection*
If vector $\boldsymbol{\mu}_{ji,G+1}$ yields a smaller cost function value than $\Theta_{i,G}$, then $\Theta_{i,G+1}$ is set to $\boldsymbol{\mu}_{ji,G+1}$. Otherwise, the old value $\Theta_{i,G}$ is retained as $\boldsymbol{\mu}_{ji,G+1}$.

3 CWN-E-ELM

In this section, we will describe in details how to combine DE and ELM to train the CFWNN.

CFWNN forms a single hidden-layer feedforward neural network as described in (1) and can be expressed using the following system model

$$\mathbf{H}_W \mathbf{W} = \mathbf{D} \tag{10}$$

where $\mathbf{H}_W = \{g_{kh}\}\,(k = 1, 2, \ldots, N \quad and \quad h = 1, 2, \ldots, H)$ is the hidden-layer output matrix, $g_{kh} = g\left(\psi_{a_h,b_h}\left(\boldsymbol{v}_h^T \hat{\boldsymbol{x}}^k\right)\right)$ denotes the output of kth hidden node with respect to $\hat{\boldsymbol{x}}^k$. The output weight matrix is $\mathbf{W} = (\boldsymbol{w}_1 \boldsymbol{w}_2 \ldots \boldsymbol{w}_O)$ and the desired output matrix is $\mathbf{D} = (\boldsymbol{d}_1 \boldsymbol{d}_2 \ldots \boldsymbol{d}_N)_{N \times O}$ is the desired output. Inspired by the similarity to SLFNs-ELM, the proposed CFWNN can be trained with ELM directly if the input weight matrix V and the dilation and translation values are properly assigned in the beginning. The output weights can be then derived by the MP generalized inverse of the hidden node output matrix as

$$\hat{\mathbf{W}} = \mathbf{H}_W^{\dagger} \mathbf{D}. \tag{11}$$

where $\mathbf{H}_W^{\dagger}$ is the MP generalized inverse of the hidden layer output matrix $\mathbf{H}_W$.

However, as the hidden node parameters are randomly assigned and they remain unchange during the training phase, there may exist a few redundant or nonoptimal nodes and input weights. As a result, more hidden nodes may be required by the ELM algorithm than traditional gradient-based learning algorithms in some applications. Therefore, more compact networks are desirable.

To ensure a compact network with good generalization performance and a fast learning speed, DE and ELM are combined to train the proposed composite function wavelet neural

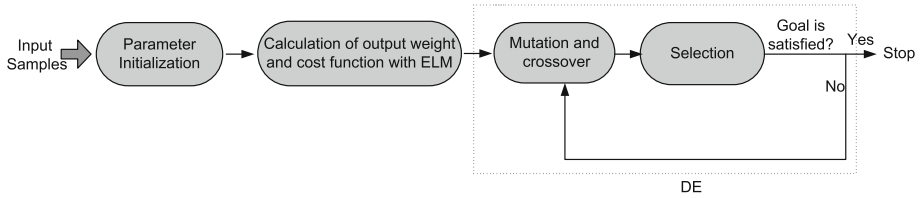

Fig. 1 Flow chart of the proposed method for CWN-E-ELM

network, where the differential evolutionary algorithm is used to select the network parameters including the input weights, the dilation and translation values and the MP generalized inverse is adopted to derived the output weights. A parameter initialization method that takes the input information into account is used to initialize the dilation and translation parameters in the proposed CWN-E-ELM. The flow chart of combining the DE algorithm and the ELM algorithm to train the network is illustrated in Fig. 1. The detailed learning steps are given as follows

– *Initialization*
A set of populations is created initially at generation G, where each population is composed of input weights and the dilation and translation parameters as

$$\Theta_{i,G} = \left[v_{10}^i, v_{11}^i, \ldots, v_{1I}^i, v_{20}^i, v_{21}^i, \ldots, v_{2I}^i, \ldots, v_{H0}^i, v_{H1}^i, \ldots, v_{HI}^i, a_1^i, a_2^i, \ldots, a_H^i, b_1^i, b_2^i, \ldots, b_H^i \right], \tag{12}$$

where $i = 1, 2, \ldots, NP$. The input weight v_{hn}^i $(h = 1, 2, \ldots, H)$ $(n = 0, 1, \ldots, I)$ is randomly initialized within the range $[-1, 1]$. To ensure that the hidden node activation functions cover the input domain evenly, the following dilation and translation parameter initialization method used by many researchers, see e.g. Zhou et al. [23], is adopted in our work. Let $\left[x_{n,\min}, x_{n,\max}\right]$ $(n = 1, 2, \ldots, I)$ represent the domain containing the input item x_n in all the observed samples (the bias term x_0 is not taken into account here) and t^*, ψ^* denote the time-domain center and radius of the mother wavelet ψ, respectively. Then, the domain of the generated hth hidden node wavelet function $\psi_{a_h^i, b_h^i}$ is given as $\left[b_h^i + a_h^i\left(t^* - \psi^*\right), b_h^i + a_h^i\left(t^* + \psi^*\right)\right]$. The input information range of the hth hidden layer can be calculated as

$$\left[v_{h0}^i x_0 + \sum_{n=1}^{I} v_{hn}^i x_{n,\min}, v_{h0}^i x_0 + \sum_{i=n}^{I} v_{hn}^i x_{n,\max} \right]$$

where $h = 1, 2, \ldots, H$ and $n = 1, 2, \ldots, I$. To ensure the hidden node wavelet function $\psi_{a_h^i, b_h^i}$ can cover the input information domain, we make the following equations to hold

$$b_h^i + a_h^i\left(t^* - \psi^*\right) = v_{h0}^i x_0 + \sum_{n=1}^{I} v_{hn}^i x_{n,\min} \tag{13}$$

$$b_h^i + a_h^i\left(t^* + \psi^*\right) = v_{h0}^i x_0 + \sum_{n=1}^{I} v_{hn}^i x_{n,\max} \tag{14}$$

Since the input weights are already assigned randomly, based on equations (13) and (14), the dilation parameter a_h and translation parameter b_h in the hth hidden layer can be derived as

$$a_h^i = \frac{\sum_{n=1}^{I} v_{hn}^i \left(x_{n,\max} - x_{n,\min}\right)}{2\psi^*} \tag{15}$$

$$b_h^i = v_{h0}x_0 + \frac{1}{2\psi^*}\left[\sum_{n=1}^{I} v_{hn}^i x_{n,\max}\left(\psi^* - t^*\right) + \sum_{n=1}^{I} v_{hn}^i x_{n,\min}\left(\psi^* + t^*\right)\right] \tag{16}$$

where $h = 1, 2, \ldots, H$, the time-domain center and radius are calculated by the following formulas

$$t^* = \frac{1}{\|\psi\|_2^2}\int t\,|\psi(t)|^2\,dt \tag{17}$$

$$\psi^* = \frac{1}{\|\psi\|_2}\left\{\int \left(t - t^*\right)^2 |\psi(t)|^2\,dt\right\}^{\frac{1}{2}} \tag{18}$$

where $\|\cdot\|$ denotes the norm in $L^2(\boldsymbol{x})$ space.

– *Calculations of the output weights and the cost function*
For each population $\Theta_{i,G}$ $(i = 1, 2, \ldots, NP)$ initialized above, the corresponding output weight $W^{\Theta_{i,G}}$ and the root mean square error $(\text{RMSE}^{\Theta_{i,G}})$ can be calculated according to (11) and the following equation, respectively,

$$\text{RMSE}^{\Theta_{i,G}} = \sqrt{\frac{\sum_{k=1}^{N}\left(\sum_{j=1}^{O}\left(\sum_{h=1}^{H} w_{jh} g\left(\psi_{a_h^i, b_h^i}\left((\boldsymbol{v}_h^i)^T \cdot \hat{\boldsymbol{x}}_k\right)\right) - d_{kj}\right)^2\right)}{N \times O \times H}}. \tag{19}$$

Similar to BP and E-ELM, the RMSE is calculated on a validating data set, which normally has no overlap with the training data to avoid overfitting.

– *Mutation and crossover*
The mutation and crossover steps of the DE algorithm are applied here to generate new populations $\boldsymbol{\mu}_{i,G+1}$ $(i = 1, 2, \ldots, NP)$ for the candidates of next generation.

– *Selection*
For each new candidate $\boldsymbol{\mu}_{i,G+1}$ $(i = 1, 2, \ldots, NP)$, the network output weight matrix $W^{\boldsymbol{\mu}_{i,G+1}}$ and the root mean square error $(\text{RMSE}^{\boldsymbol{\mu}_{i,G+1}})$ can be derived according to (11) and (19), respectively. Generally, in the selection step of the DE algorithm, the RMSE of the new candidate $\boldsymbol{\mu}_{i,G+1}$ is compared with the RMSE of the original one $\Theta_{i,G}$, and the vector which has the smaller RMSE will be retained to the next generation. However, as analyzed in E-ELM [19], a small validation error does not guarantee a small testing error. Therefore, the norm of the output weight $\|W\|$ is added as one more criteria for selection as mentioned in E-ELM [19], which was on account of the analysis of Bartlett [24] that the neural networks tend to have better generalization performance with smaller weights. Consequently, the criterions of selecting the new generation $\Theta_{i,G+1}$ can be described as follows

$$\Theta_{i,G+1} = \begin{cases} \boldsymbol{\mu}_{i,G+1} & \text{if} \quad \text{RMSE}^{\Theta_{i,G}} - \text{RMSE}^{\boldsymbol{\mu}_{i,G+1}} > \varepsilon \cdot \text{RMSE}^{\Theta_{i,G}}, \\ \boldsymbol{\mu}_{i,G+1} & \text{if} \quad \left|\text{RMSE}^{\Theta_{i,G}} - \text{RMSE}^{\boldsymbol{\mu}_{i,G+1}}\right| < \varepsilon \cdot \text{RMSE}^{\Theta_{i,G}} \\ & \text{and} \quad \|W^{\boldsymbol{\mu}_{i,G+1}}\| < \left\|W^{\Theta_{i,G}}\right\|, \\ \Theta_{i,G} & \text{else}, \end{cases} \tag{20}$$

where ε is the preset positive tolerance rate and $i = 1, 2, \ldots, NP$.
Once the populations of the new generation are generated, repeat steps (3) and (4) until the preset goal is reached or the maximum learning iterations is completed.

4 Performance Evaluation

In this section, the proposed CWN-E-ELM is compared with some relevant methods, namely, SLFN-E-ELM, CFWNN-ELM and SVR on several problems in function approximation, real-world data regressions and chaotic signal prediction. All these experiments are carried out in the MATLAB 7.4 environment running on a 2.66 GHZ CPU. The simulations for SVR are carried out using compiled C-code SVR package: Libsvm[1] running in the same PC. The kernel function used in SVR is radial basis function. For CWN-E-ELM and CFWNN-ELM, the popular Morlet wavelet function $\psi(t) = \cos(1.75t)e^{(-0.5t^2)}$ is used as the mother wavelet and the sigmoid function $g(x) = 1/(1 + \exp(-x))$ is adopted to form the composite function used at the hidden nodes. For SLFN-E-ELM, the sigmoid function is used at the hidden nodes. To make a fair comparison, the same parameter initialization method utilized by CWN-E-ELM is also applied to initialize the dilation and translation values of CFWNN-ELM. For SVR, the cost parameter C and the kernel parameter γ are searched in a grid formed by $C = \left[2^{12}, 2^{11}, \ldots, 2^{-2}\right]$ and $\gamma = \left[2^{4}, 2^{3}, \ldots, 2^{-10}\right]$ as proposed in [25] and the best parameters are then obtained in terms of the generalization performance. For all the simulation results shown in the following tables, we will use the bold values to denote the best results.

4.1 Evaluation on function approximations

In this subsection, CWN-E-ELM and the other three relevant methods, SLFN-E-ELM, CFWNN-ELM and SVR, are evaluated on three nonlinear function approximation problems, whose mathematical representations are described as follows:

$$f_1 = \begin{cases} \frac{\sin(x)}{x}, & x \neq 0, \\ 1, & x = 0. \end{cases} \tag{21}$$

$$f_2 = \begin{cases} 43.72x - 8.996, & x \in [0, 0.4] \\ -84.62x + 42.46, & x \in (0.4, 0.5] \\ 10e^{-x}\sin(12x^2 + 2x - 4), & x \in (0.5, 1] \end{cases} \tag{22}$$

$$f_3 = \left(x_1^2 - x_2^2\right)\sin\left(0.5x_1\right), x_1, x_2 \in [-10, 10] \tag{23}$$

Function f_1 is the famous 'SinC' function and the last two are a one-dimensional piecewise function and a two-dimensional nonlinear function, respectively. For function approximation problems, the parameters of the DE are set as follows: the amplification factor F and the crossover constant CR are set to 1 and 0.8, respectively. The number of populations NP is determined manually according to the number of training samples, where the more the training samples is, the larger the number of populations NP will be used.

For function f_1, a training set $(x_i, f_1(x_i))$ and a testing set $(x_i, f_1(x_i))$ with 400 points, respectively, are generated where x_i's are uniformly distributed in the input region $[-10, 10]$ for all these four algorithms. For CWN-E-ELM and SLFN-E-ELM, the training dataset is separated into two non-overlapped subsets with a partition ratio 2:1, where one is for training and the other is for validation in the differentials evolution phase. The number of populations NP is set to be 30 in this simulation.

Similar to function f_1, a training set $(x_i, f_2(x_i))$ and a testing set $(x_i, f_2(x_i))$ with 400 points, respectively, are generated where x_i's are uniformly distributed in the input region $[0, 1]$. For CWN-E-ELM and SLFN-E-ELM, the training dataset is separated into the training

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Table 1 Comparisons for function approximations

Function	NNs	Testing		Training time (s)	Nodes/SVs	$NP/(C, \gamma)$
		RMSE	Dev			
	CWN-E-ELM	**0.0112**	**0.0029**	5.4409	**10**	30
f_1	SLFN-E-ELM	0.0147	0.0041	1.1247	**10**	30
	CFWNN-ELM	0.0187	0.0040	**0.0036**	20	–
	SVR	0.0258	0.0053	19.6114	199.85	$\left(2^{-3}, 2^{-3}\right)$
	CWN-E-ELM	**0.0956**	**0.0200**	5.3844	**25**	40
f_2	SLFN-E-ELM	0.1138	0.0262	1.5781	**25**	40
	CFWNN-ELM	0.2205	0.0480	**0.0062**	30	–
	SVR	0.4545	0.0476	12.6745	264.06	$\left(2^5 2^5\right)$
	CWN-E-ELM	**0.0534**	**0.0109**	38.6711	**70**	80
f_3	SLFN-E-ELM	0.5138	0.0667	65.7883	140	80
	CFWNN-ELM	0.3938	0.2005	**0.1944**	180	–
	SVR	0.4527	0.1880	397.6044	912.94	$\left(2^5, 2^{-3}\right)$

The bold values indicate the smallest RMSE, Dev, training time and nodes corresponding to each column

subset and the validation subset with the same partition ratio used in function f_1 and the number of populations NP is set to be 40.

For function f_3, a training dataset $((x_{1i}, x_{2i}), f_3(i))$ and a testing dataset $((x_{1i}, x_{2i}), f_3(i))$ are collected where (x_{1i}, x_{2i}) are uniformly randomly distributed over the input domain $D = [-10, 10] \times [-10, 10]$, which forms a 40×40 grid. Similarly, a validation dataset is separated from the training dataset for CWN-E-ELM and SLFN-E-ELM and the partition ratio is the same to the one used in function f_1 and f_2. The numbers of populations NP for CWN-E-ELM and SLFN-E-ELM are set to be 80 in this experiment.

To make these approximation problems 'real', uniformly distributed noise in $[-0.2, 0.2]$ is added to all the training samples of these three functions, while the testing data remain noise-free. In our simulations, the number of hidden nodes is incremented by 5 for CWN-E-ELM, SLFN-E-ELM and CFWNN-ELM. The nearly optimal numbers of nodes for these three NNs are then decided based on cross-validation method. Table 1 shows the performance comparisons using these four methods on the three function approximations. All these results are collected by averaging the results of 50 trials, and for each trial, all the datasets are randomly generated and validation datasets are randomly collected from the training datasets with the partition ratio mentioned above.

From Table 1, one can easily find that CWN-E-ELM wins the lowest testing RMSEs among these four methods for all three function approximations while CFWNN-ELM has the shortest training time. Although the training time by CWN-E-ELM is longer than the one by CFWNN-ELM as the DE algorithm is adopted for searching the optimal network parameters, the testing RMSE obtained by CWN-E-ELM is much lower than the one by CFWNN-ELM, especially for the piecewise function f_2 and the two-dimensional function f_3. In our simulations, the training time costed by SVR includes the searching time of the optimal combination of C and γ. The training time by SVR is the longest one among all these methods and the average support vectors used by SVR are larger than the number of hidden nodes adopted by other three methods.

We can also find from Table 1 that DE helps to reduce the number of hidden nodes used by CWN-E-ELM and SLFNs-E-ELM. For these three cases, only 10, 25 and 70 nodes are enough

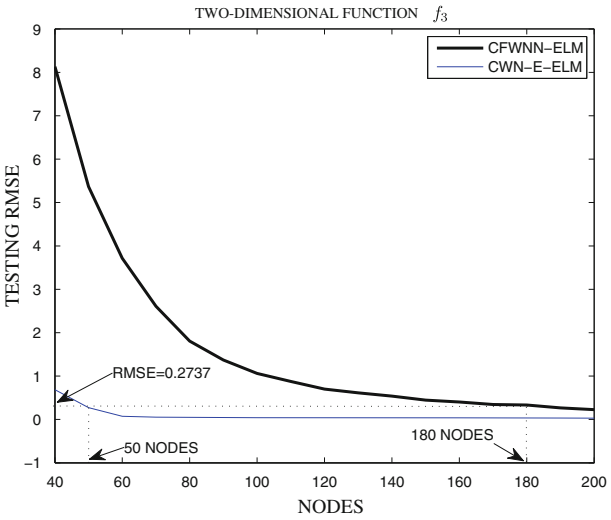

Fig. 2 Testing RMSEs of the two-dimensional function f_3 w.r.t different number of nodes

to ensure a reasonable generalization performance for CWN-E-ELM for f_1, f_2, and f_3, respectively. Figure 2 shows the testing RMSEs of the two-dimensional function f_3 obtained by CWN-E-ELM and CFWNN-ELM using 40×40 training samples with respect to different number of nodes. It is easy to find that to achieve the testing RMSE 0.2737, CFWNN-ELM requires 180 hidden nodes, while for CWN-E-ELM, only 50 hidden nodes are needed. Moreover, we can also see that the differences of the testing RMSEs between different numbers of hidden nodes obtained by CWN-E-ELM are smaller than the one derived by CFWNN-ELM. When the number of nodes used by CWN-E-ELM is 40, the testing RMSE is 0.6840, and when increasing the number of nodes to 200, the corresponding testing RMSE derived by CWN-E-ELM is 0.0310. The difference between the RMSEs obtained by CWN-E-ELM with the hidden nodes 40 and 200 is only 0.6530. However, the difference between the RMSEs obtained by CFWNN-ELM with the hidden nodes 40 and 200 is higher to 7.9038.

Different sample sizes, from 20×20 pairs to 60×60 pairs for function f_3 are also used to train and test by all these four methods to validate the effectiveness of the proposed method. Figure 3 depicts the curve of the averaging testing RMSEs collected by 50 trials. From this figure, one can easily find that CWN-E-ELM can achieve the best generalization performance among the four methods. For each data pair, the testing RMSE derived by CWN-E-ELM is many times lower than the results obtained by the other three methods. Figure 4 depicts the error surfaces between the target two-dimensional function and the approximation curves by all these five methods using 40×40 training sample pairs. It has shown that the error surface obtained by CWN-E-ELM is much more close to zero surface than other methods.

4.2 Evaluation on real-world data regressions

In this subsection, the performances of CWN-E-ELM, SLFN-E-ELM, CFWNN-ELM and SVR are compared on the regressions of several real-world data sets covering various domains from the UCI database.[2] The data distributions are unknown and most of them contain noises.

[2] http://www.ics.uci.edu/~mlearn/MLRepository.html.

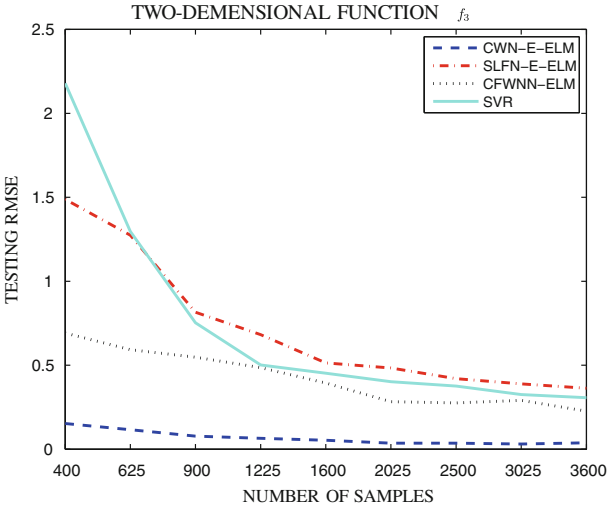

Fig. 3 Testing RMSEs of the two-dimensional function f_3 w.r.t different sizes of samples

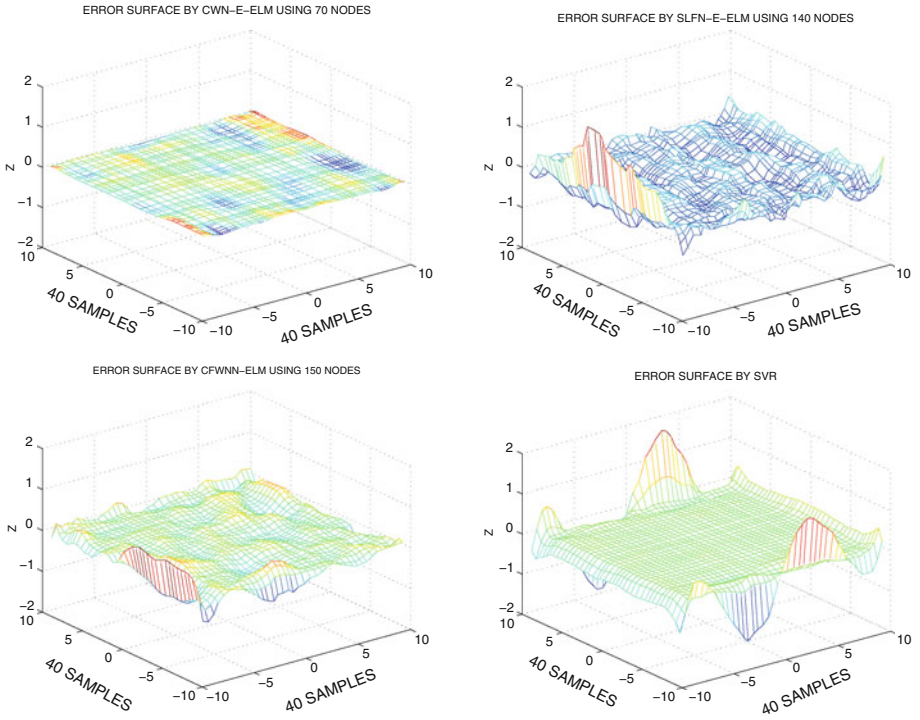

Fig. 4 Approximation error surfaces of the two-dimensional function

Table 2 Specification of real-world datasets

Datasets	Data		Attributes	
	Training	Testing	Continuous	Nominal
Abalone	1,000	2, 177	7	1
Autompg	220	178	8	0
BosHousing	250	256	13	0
CalHousing	8,000	12,640	8	0
Auto Price	40	119	14	1
Parkinsons	2,000	3,875	25	0
COIL-2000	5,822	4,000	85	0

Table 2 lists the specifications of these real-world datasets. All the attributes (inputs) have been normalized to the range of $[-1, 1]$ while the outputs have been normalized into $[0, 1]$ in our simulations. The number of populations NP for all these real-world datasets is set to be 80. For the first 4 datasets, the training data and the testing data are randomly generated from its whole dataset before each trial of simulation while the training and testing data of the last dataset are fixed for all trials of simulations. For CWN-E-ELM and SLFN-E-ELM, the validation set is obtained from the training set where the partition ratio between the training set and the validation set is 2:1.

In our simulations, the number of hidden nodes is incremented by 5 for CWN-E-ELM, SLFN-E-ELM and CFWNN-ELM. The nearly optimal numbers of nodes for these three NNs are then decided based on cross-validation method. For SVR, the cost parameter C and kernel parameter γ are searched in a grid and the best parameter pairs are collected in terms of the generalization performance. Average results of 50 trials for all these four methods are reported. Table 3 shows the testing RMSEs, the standard derivations of testing RMSEs of these datasets, the numbers of hidden nodes and the training times used for these methods. As observed from this table, CWN-E-ELM obtains the best generalization performances in most cases among all these methods. The numbers of hidden nodes needed for CWN-E-ELM are much smaller than the one of CFWNN-ELM and the average support vectors of SVR. The training time used by CFWNN-ELM is the smallest among all these methods. Although the training times by CWN-E-ELM and SLFN-E-ELM are longer than CFWNN-ELM as the DE algorithm is applied to optimize the parameters of these two NNs, they still learn much faster than SVR.

4.3 Evaluation on chaotic signal prediction

CWN-E-ELM and other three methods are also evaluated on predicting a chaotic signal in this subsection, which is an one-step-ahead classical time series prediction problem described in [13]. The logistic function used to generate the chaotic signal is described as

$$x\left(k+1\right) = \alpha x\left(k\right)\left(1 - x\left(k\right)\right). \tag{24}$$

The value of the parameter α critically determines the behavior of the time series generated by the above equation. If $\alpha < 1$, the time series collapses to a constant value from a random initial value in the region $[0, 1]$. For $\alpha > 3$, the system generates a periodic attractor. If the value of α goes beyond 3.6, the system shows the chaotic behavior. In our study,

Table 3 Comparisons for regressions of real-world datasets

Datasets	NNs	Testing		Training time (s)	Nodes/SVs	$NP/(C,\gamma)$
		RMSE	Dev			
Abalone	CWN-E-ELM	**0.0767**	0.0021	28.432	24	80
	SLFN-E-ELM	0.0774	0.0022	4.8721	**22**	80
	CFWNN-ELM	0.0775	0.0016	**0.0450**	50	–
	SVR	0.0821	**0.0011**	290.21	332.22	$\left(2^{0}, 2^{-3}\right)$
Autompg	CWN-E-ELM	**0.0712**	0.0105	3.9783	30	80
	SLFN-E-ELM	0.0770	0.096	0.8967	**20**	80
	CFWNN-ELM	0.0743	0.0115	**0.0084**	50	–
	SVR	0.0987	**0.0026**	22.336	80.54	$\left(2^{-2}, 2^{-5}\right)$
BosHousing	CWN-E-ELM	**0.0922**	0.0111	3.4312	26	80
	SLFN-E-ELM	0.1055	0.0124	0.8675	**24**	80
	CFWNN-ELM	0.1067	0.0114	**0.0109**	60	–
	SVR	0.1102	**0.0069**	13.399	57.24	$\left(2^{1}, 2^{-6}\right)$
CalHousing	CWN-E-ELM	0.1311	0.0020	188.43	**40**	80
	SLFN-E-ELM	0.1313	0.0025	65.78	**40**	80
	CFWNN-ELM	0.1290	**0.0018**	**0.3318**	50	–
	SVR	**0.1193**	0.0022	1836.7	2176.2	$\left(2^{3}, 2^{1}\right)$
Auto Price	CWN-E-ELM	0.0512	**0.0076**	1.0785	12	80
	SLFN-E-ELM	0.0611	0.0096	0.3521	**10**	80
	CFWNN-ELM	0.0912	0.0209	**0.0001**	30	–
	SVR	0.0945	0.0098	5.2988	22.75	$\left(2^{8}, 2^{-5}\right)$
Parkinsons	CWN-E-ELM	**0.2146**	0.0023	132.7914	**25**	80
	SLFN-E-ELM	0.2224	0.0033	26.1141	**25**	80
	CFWNN-ELM	0.2201	0.0059	**0.0462**	50	–
	SVR	0.2245	**0.0016**	313.5116	1207.4	$\left(2^{4}, 2^{0}\right)$
COIL-2000	CWN-E-ELM	**0.2287**	0.0007	383.23	**50**	80
	SLFN-E-ELM	0.2336	0.0005	234.35	**50**	80
	CFWNN-ELM	0.2334	0.0005	**0.3491**	100	–
	SVR	0.2397	0	14131	4117	$\left(2^{-4}, 2^{-4}\right)$

The bold values indicate the smallest RMSE, Dev, training time and nodes corresponding to each column

α is set to 3.8. With initial value $x(1) = 0.001$, the 60 pair samples (the pair is formed as $(x(n), x(n+1)), n = 1, 2, \ldots, 60$) are generated as training data. For CWN-E-ELM and SLFN-E-ELM, the first 40 pair samples are used as training data and the last 20 pair samples are used as validation data. With initial value $x(1) = 0.9$, another 100 pair samples from $x(1)$ to $x(100)$ are collected as the prediction data.

Table 4 shows the evaluation performances of all these four methods. From this table, one can easily find that CWN-E-ELM reaches the lowest prediction RMSE 8.2155e−006 with only three hidden nodes, which is much more compact than CFWNN-ELM. Although the used number of hidden nodes of CWN-E-ELM is the same as SLFN-E-ELM, the prediction performance of CWN-E-ELM is better than SLFN-E-ELM.

Table 4 Comparisons for chaotic signal prediction

NNs	Predicting		Training	Nodes/SVs	$NP/(C, \gamma)$
	RMSE	Dev	time (s)		
CWN-E-ELM	**8.2155e-006**	2.1105e-005	1.2413	**3**	20
SLFN-E-ELM	5.4421e-004	2.1032e-004	0.3251	**3**	20
CFWNN-ELM	6.5225e-005	6.8571e-004	**0.0018**	10	–
SVR	0.0687	**3.7890e-015**	11.5349	4	$\left(2^{12}, 2^{-5}\right)$

The bold values indicate the smallest RMSE, Dev, training time and nodes corresponding to each column

5 Discussions and Conclusion

From the simulation results presented in the previous section, we have the following observations.

1. Compared with the CFWNN-ELM which uses the ELM algorithm alone, the adoption of the DE algorithm enables the CWN-E-ELM to efficiently reduce the network size by removing a large number of redundant hidden nodes. The generalization performance of CWN-E-ELM is also better because the DE algorithm is incorporated into searching the optimal network parameters. The only disadvantage of CWN-E-ELM is that the training time is longer due to the introduction of DE.
2. Although the number of hidden nodes and the training time for both CWN-E-ELM and SLFN-E-ELM are about the same, CWN-E-ELM performs much better than SLFN-E-ELM in general.
3. Compared with SVR, the proposed CWN-E-ELM is able to achieve a much better generalization performance with a faster learning speed for most of the simulations.

In conclusion, we have shown in this paper that the proposed CWN-E-ELM, which incorporates the DE algorithm into the composite function wavelet neural network trained with ELM, has better performances in function approximations, real-world dataset regressions and the chaotic signal prediction problem than some relevant methods and the network size is much more compact than CFWNN-ELM due to the introduction of the DE algorithm which helps remove some redundant nodes. Future investigation will be concentrated on how to reduce the learning time caused by employing the DE algorithm in CWN-E-ELM.

Acknowledgments The authors would like to thank the anonymous reviewers whose insightful and helpful comments greatly improved this paper.

References

1. Zhang QH, Benveniste A (1992) Wavelet networks. IEEE Trans Neural Netw 3(6):889–898
2. Szu H, Telfer B, Kadambe S (1992) Neural network adaptive wavelets for signal representation and classification. Opt Eng 31(9):1907–1916
3. Szu H, Telfer B, Garcia J (1996) Wavelet transforms and neural networks for compression and recognition. Neural Netw 9(4):659–708
4. Zhang J, Walter GG, Miao YB, Lee WNW (1995) Wavelet neural networks for function learning. IEEE Trans Signal Process 43(6):1485–1497
5. Zhang QH (1997) Using wavelet network in nonparametric estimation. IEEE Trans Neural Netw 8(2):227–236

6. Liu Q, Yu X, Feng Q (2003) Fault diagnosis using wavelet neural networks. Neural Process Lett 18: 115–123
7. Subasi A, Yilmaz M, Ozcalik HR (2006) Classification of EMG signals using wavelet neural network. J Neurosci Methods 156(1-2):360–367
8. Xu JX, Tan Y (2007) Nonlinear adaptive wavelet control using constructive wavelet networks. IEEE Trans Neural Networks 18(1):115–127
9. Pan H, Xia LZ (2008) Efficient object recognition using boundary representation and wavelet neural network. IEEE Trans Neural Networks 19(12):2132–2149
10. Yoo SJ, Park JB, Choi YH (2008) Adaptive output feedback control of flexible-joint robots using neural networks: dynamic surface design approach. IEEE Trans Neural Networks 19(10):1712–1726
11. Jin N, Liu DR (2008) Wavelet basis function neural network for sequential learning. IEEE Trans Neural Networks 19(3):523–528
12. Oussar Y, Dreyfus G (2000) Initialization by selection for wavelet network training. Neurocomputing 34(1-4):131–143
13. Lin C-J (2006) Wavelet neural networks with a hybrid learning approach. J Inf Sci Eng 22(6):1367–1387
14. Haykin S (2001) Neural networks: a comprehensive foundation, 2nd edn. Pearson Education Press, Asia
15. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1-3):489–501
16. Serre D (2002) Matrices: theory and applications. Springer, New York
17. Cao JW, Lin ZP, Huang G-B (2010) Composite function wavelet neural networks with extreme learning machine. Neurocomputing 73(7–9):1405–1416
18. Storn R, Price K (2004) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
19. Zhu Q-Y, Qin A-K, Suganthan P-N, Huang G-B (2005) Evolutionary extreme learning machine. Pattern Recognit 38(10):1759–1763
20. Subudhi B, Jena D (2008) Differential evolution and Levenberg Marquardt trained neural network scheme for nonlinear system identification. Neural Process Lett 27:285–296
21. Levenberg K (1944) A method for the solution of certain non-linear problems in least squares. Q Appl Math 2:164–168
22. Huang G-B, Zhu Q-Y, Mao K-Z, Siew C-K, Saratchandran P, Sundararajan N (2008) Can threshold networks be trained directly?. IEEE Trans Circuits Syst II 53(3):187–191
23. Zhou B, Shi AG, Cai F, Zhang YS (2004) Wavelet neural networks for nonlinear time series analysis. Lecture notes in Computer Science, vol 3174. Springer, Berlin, pp 430–435
24. Bartlett PL (2008) The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans Inf Theory 44(2):525–536
25. Hsu C-W, Lin C-J (2002) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Netw 13(2):415–425