Neural Comput & Applic (2014) 24:1683–1695
DOI 10.1007/s00521-013-1398-7

ORIGINAL ARTICLE

Fast learning network: a novel artificial neural network with a fast learning speed

Guoqiang Li · Peifeng Niu · Xiaolong Duan · Xiangye Zhang

Received: 19 November 2012 / Accepted: 27 March 2013 / Published online: 13 April 2013

Abstract This paper proposes a novel artificial neural network called fast learning network (FLN). In FLN, input weights and hidden layer biases are randomly generated, and the weight values of the connection between the output layer and the input layer and the weight values connecting the output node and the input nodes are analytically determined based on least squares methods. In order to test the FLN validity, it is applied to nine regression applications, and experimental results show that, compared with support vector machine, back propagation, extreme learning machine, the FLN with much more compact networks can achieve very good generalization performance and stability at a very fast training speed and a quick reaction of the trained network to new observations. In addition, in order to further test the FLN validity, it is applied to model the thermal efficiency and NO_x emissions of a 330 WM coal-fired boiler and achieves very good prediction precision and generalization ability at a high learning speed.

Keywords Artificial neural network · Fast learning network · Extreme learning machine · Least squares

G. Li · P. Niu (✉) · X. Duan · X. Zhang
Key Lab of Industrial Computer Control Engineering of Hebei Province, Yanshan University, Qinhuangdao 066004, China
e-mail: niupeifeng2011@163.com

G. Li
e-mail: zhihuiyuang@163.com

G. Li · P. Niu · X. Duan · X. Zhang
National Engineering Research Center for Equipment and Technology of Cold Strip Rolling, Qinhuangdao 066004, China

1 Introduction

The artificial neural networks (ANNs) have been frequently used in a variety of applications with great success due to their ability to approximate complex nonlinear mappings directly from input patterns [1, 2]. Namely, ANNs do not require a user-specified problem solving algorithm, but they could learn from existing examples, much like human beings. In addition, ANNs have inherent generalization ability. This means that ANNs could identify and synchronously respond to the patterns that are similar with but not identical to the ones that are employed to train ANNs. However, the free parameters of ANNs would be defined by learning from the given training samples according to gradient descent algorithms, which makes the learning process relatively slow and brings some issues related to its local minima. Owing to these shortages, it could take much more time to train ANNs and have a suboptimal solution [3].

For these problems, a new artificial neural network, extreme learning machine (ELM), is proposed by Huang et al. [4]. Recently, ELM is a novel single hidden layer feedforward neural network (SLFN) where the input weights and the bias of hidden nodes are generated randomly without tuning and the output weights are determined analytically. ELM owns an extremely fast learning algorithm and good generalization capability. Up to now, the ELM has been successfully applied in various areas [5–7]. The ELM overcomes most issues encountered in traditional learning methods, such as the stopping criterion, number of epochs, learning rate and local minima. However, there are still some insufficiencies in ELM. ELM tends more hidden neurons than conventional tuning-based learning algorithms in many applications, which would make a trained ELM need longer time for responding to unknown testing samples [8, 9].

This paper proposes a novel fast learning network (FLN) based on the thought of ELM. The FLN is a double parallel forward neural network (DPFNN) [10, 11], which is a parallel connection of a multilayer feedforward neural network and a single layer feedforward neural network, and the DPFNN's output nodes not only receive the recodification of the external information through the hidden nodes, but also receive the external information itself directly through the input nodes. In FLN, the input weights and hidden layer biases are randomly generated, and the weight values of the connection between the output layer and the input layer and the weight values connecting the output node and the input nodes are analytically determined based on least squares methods. Compared with other methods, FLN with a smaller number of hidden units can achieve good generalization performance and stability at a very fast speed on most applications.

The paper is organized as follows: the proposed fast learning network is given in Sect. 2. Section 3 shows the performance evaluation of the FLN. Finally, Sect. 4 summarizes the conclusions of this paper.

2 Fast learning network

In this section, a novel artificial neural network called fast learning network, which is a double parallel forward neural network, is proposed based on the least squares methods. The fast learning network, as shown in Fig. 1, is described as follows in detail.

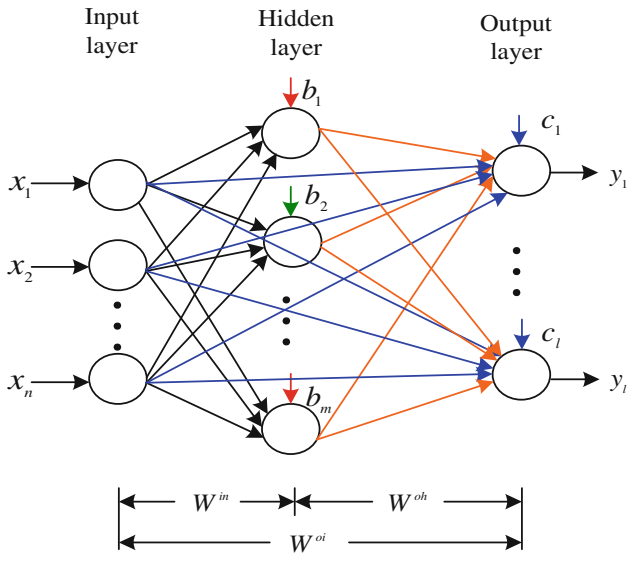

Fig. 1 Structure of the fast learning network

2.1 Approximation problem

The fast learning network called FLN is a parallel connection of a single layer feedforward neural network and a three layer feedforward neural network: input layer, hidden layer and output layer. Suppose, there are N arbitrary distinct samples $\{\mathbf{x}_i, \mathbf{y}_i\}$, in which $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ is the n-dimensional feather vector of the ith sample, and $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots y_{il}]^T \in R^l$ is the corresponding l-dimensional output vector. The FLN has m hidden layer nodes. $\mathbf{W}^{in}$ is the $m \times n$ input weight matrix, $\mathbf{b} = [b_1, b_2, \ldots, b_m]$ is the biases of hidden layer nodes, and $\mathbf{W}^{oh}$ is a $l \times m$ matrix which consists of the weight values of the connection between the output layer and the hidden layer. $\mathbf{W}^{oi}$ is a $l \times n$ weight matrix which contains weight values of the connection between the output layer and the input layer. $\mathbf{c} = [c_1, c_2, \cdots, c_l]^T$ is the biases of output layer nodes. $g(\cdot)$ and $f(\cdot)$ are the active functions of hidden nodes and output nodes.

Then, the FLN is mathematically modeled as:

$$\begin{cases} y_{j1} = f\left(\sum_{r=1}^{n} W_{1r}^{oi} x_{jr} + c_1 + \sum_{k=1}^{m} W_{1k}^{oh} g\left(b_k + \sum_{t=1}^{n} W_{kt}^{in} x_{jt}\right)\right) \\ y_{j2} = f\left(\sum_{r=1}^{n} W_{2r}^{oi} x_{jr} + c_2 + \sum_{k=1}^{m} W_{2k}^{oh} g\left(b_k + \sum_{t=1}^{n} W_{kt}^{in} x_{jt}\right)\right) \quad j = 1, 2, \cdots, N \\ \vdots \\ y_{jl} = f\left(\sum_{r=1}^{n} W_{lr}^{oi} x_{jr} + c_l + \sum_{k=1}^{m} W_{lk}^{oh} g\left(b_k + \sum_{t=1}^{n} W_{kt}^{in} x_{jt}\right)\right) \end{cases} \tag{1}$$

Equation (1) could be transformed into the following form:

$$\mathbf{y}_j = f\left(\mathbf{W}^{oi}\mathbf{x}_j + \mathbf{c} + \sum_{k=1}^{m} \mathbf{W}_k^{oh} g\left(\mathbf{W}_k^{in}\mathbf{x}_j + b_k\right)\right) \quad j = 1, 2, \cdots, N \tag{2}$$

where $\mathbf{W}^{oi} = \left[\mathbf{W}_1^{oi}, \mathbf{W}_2^{oi}, \ldots, \mathbf{W}_l^{oi}\right]$ is the weight vector connecting the jth output node and the input nodes, $\mathbf{W}_k^{oh} = \left[W_{1k}^{oh}, W_{2k}^{oh}, \ldots, W_{lk}^{oh}\right]^T$ is the weight vector connecting the kth hidden node and the output nodes, and $\mathbf{W}_k^{in} = \left[W_{k1}^{in}, W_{k2}^{in}, \ldots, W_{km}^{in}\right]^T$ is the weight vector connecting the kth hidden node and the input nodes.

Then, Eq. (2) can be rewritten compactly as Eq. (3)

$$\mathbf{Y} = f\left(\mathbf{W}^{oi}\mathbf{X} + \mathbf{W}^{oh}\mathbf{G} + \mathbf{c}\right) = f\left(\left[\mathbf{W}^{oi}\mathbf{W}^{oh}\mathbf{c}\right]\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}\right) = f\left(\mathbf{W}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}\right) \tag{3}$$

$$\mathbf{G}\left(\mathbf{W}_1^{in},\cdots,\mathbf{W}_m^{in},b_1,\cdots,b_m,\mathbf{x}_1,\cdots,\mathbf{x}_N\right) = \begin{bmatrix} g\left(\mathbf{W}_1^{in}\mathbf{x}_1+b_1\right) & \cdots & g\left(\mathbf{W}_1^{in}\mathbf{x}_N+b_1\right) \\ \vdots & \ddots & \vdots \\ g\left(\mathbf{W}_m^{in}\mathbf{x}_1+b_m\right) & \cdots & g\left(\mathbf{W}_m^{in}\mathbf{x}_N+b_m\right) \end{bmatrix}_{m\times N} \tag{4}$$

$$\mathbf{W} = \left[\mathbf{W}^{oi}\mathbf{W}^{oh}\mathbf{c}\right]_{l\times(n+m+1)} \tag{5}$$

$$\mathbf{I} = [11\cdots1]_{1\times N} \tag{6}$$

The matrix $\mathbf{W} = \left[\mathbf{W}^{oi}\mathbf{W}^{oh}\mathbf{c}\right]$ could be called as output weights. $\mathbf{G}$ is called the hidden layer output matrix of FLN; the ith row of $\mathbf{G}$ is the ith hidden neuron's output vector with respect to inputs $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N$.

2.2 Minimum norm least squares solution

For the input weights and biases of the hidden layer, many research results show that the input weights and hidden biases need not be adjusted at all and can be arbitrarily given [12–16]. Based on these researches, this proposed FLN could randomly generate the input weights $\mathbf{W}^{in}$ and biases $\mathbf{b} = [b_1,b_2,\ldots,b_m]$ of the hidden layer. After that, the FLN could be thought of a linear system, and the output weights $\mathbf{W}$ could be analytically determined through the following form:

$$\left\| f\left(\hat{\mathbf{W}}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}\right) - \mathbf{Y}\right\| = \min_{\mathbf{W}}\left\| f\left(\mathbf{W}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}\right) - \mathbf{Y}\right\| \tag{7}$$

For an invertible activation function $f(\cdot)$, the output weights are also analytically determined by Eq. (8)

$$\left\| \hat{\mathbf{W}}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix} - f^{-1}(\mathbf{Y})\right\| = \min_{\mathbf{W}}\left\| \mathbf{W}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix} - f^{-1}(\mathbf{Y})\right\| \tag{8}$$

where $f^{-1}(\cdot)$ is the invertible function of $f(\cdot)$.

According to the Moore–Penrose generalized inverse [17, 18], the minimum norm least-squares solution of the linear system could be written as:

$$\hat{\mathbf{W}} = f^{-1}(\mathbf{Y})\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}^{+} = f^{-1}(\mathbf{Y})\mathbf{H}^{+} \tag{9}$$

where $\mathbf{H} = [\mathbf{X}\ \mathbf{G}\ \mathbf{I}]^T = \left[\mathbf{X}^T\ \mathbf{G}^T\ \mathbf{I}^T\right]$.

Then

$$\begin{cases} \mathbf{W}^{oi} = \hat{\mathbf{W}}(1:l,\ 1:n) \\ \mathbf{W}^{oh} = \hat{\mathbf{W}}(1:l,\ n+1:(n+m)) \\ \mathbf{c} = \hat{\mathbf{W}}(1:l,\ n+m+1) \end{cases} \tag{10}$$

If the rank$(\mathbf{H}) = N$, Eq. (9) could be rewritten as

$$\begin{aligned} \hat{\mathbf{W}} &= f^{-1}(\mathbf{Y})\left(\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}^T\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}\right)^{-1}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}^T \\ &= f^{-1}(\mathbf{Y})\left(\left[\mathbf{X}^T\mathbf{G}^T\mathbf{I}^T\right]\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}\right)^{-1}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}^T \\ &= f^{-1}(\mathbf{Y})\left(\mathbf{X}^T\mathbf{X}+\mathbf{G}^T\mathbf{G}+\mathbf{I}^T\mathbf{I}\right)^{-1}\mathbf{H}^T \end{aligned} \tag{11}$$

If the rank$(\mathbf{H}) = m+n+1$, Eq. (9) could be rewritten as

$$\begin{aligned} \hat{\mathbf{W}} &= f^{-1}(\mathbf{Y})\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}^T\left(\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\\ \mathbf{I}\end{bmatrix}^T\right)^{-1} \\ &= f^{-1}(\mathbf{Y})\mathbf{H}^T\left(\mathbf{H}\mathbf{H}^T\right)^{-1} \end{aligned} \tag{12}$$

2.3 Simplifying model

In general, the output neurons' active function $f(\cdot)$ is often chosen linear, namely $f(x) = x$, and the output biases $\mathbf{c} = [c_1,c_2,\cdots,c_l]^T$ are often set as zeros. Then, the FLN's mathematical model (Eq. (1)) could be rewritten as

$$\begin{cases} y_{j1} = \sum_{r=1}^{n} W_{1r}^{oi}x_{jr} + \sum_{k=1}^{m} W_{1k}^{oh}g\left(b_k + \sum_{t=1}^{n} W_{kt}^{in}x_{jt}\right) \\ y_{j2} = \sum_{r=1}^{n} W_{2r}^{oi}x_{jr} + \sum_{k=1}^{m} W_{2k}^{oh}g\left(b_k + \sum_{t=1}^{n} W_{kt}^{in}x_{jt}\right) \\ \vdots \\ y_{jl} = \sum_{r=1}^{n} W_{lr}^{oi}x_{jr} + \sum_{k=1}^{m} W_{lk}^{oh}g\left(b_k + \sum_{t=1}^{n} W_{kt}^{in}x_{jt}\right) \end{cases}, \quad j = 1,2,\cdots,N \tag{13}$$

And simultaneously, Eqs. (3) and (5) could be, respectively, transformed into Eqs. (14) and (15)

$$\mathbf{Y} = \mathbf{W}^{oi}\mathbf{X} + \mathbf{W}^{oh}\mathbf{G} = \left[\mathbf{W}^{oi}\mathbf{W}^{oh}\right]\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix} = \mathbf{W}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix} \tag{14}$$

$$\mathbf{W} = \left[\mathbf{W}^{oi}\mathbf{W}^{oh}\right]_{l\times(n+m)} \tag{15}$$

The hidden layer's output matrix $\mathbf{G}$ is still calculated by Eq. (4). And Eq. (7) and Eq. (8) could be rewritten as Eq. (15).

$$\left\| \hat{\mathbf{W}}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix} - \mathbf{Y}\right\| = \min_{\mathbf{W}}\left\| \mathbf{W}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix} - \mathbf{Y}\right\| \tag{15}$$

The output weights $\mathbf{W} = \left[\mathbf{W}^{oi}\mathbf{W}^{oh}\right]$ could be analytically determined by the Moore–Penrose generalized inverse.

$$\hat{\mathbf{W}} = \mathbf{Y}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix}^{+} = \mathbf{Y}\mathbf{H}^{+} \tag{16}$$

$$\begin{cases}\mathbf{W}^{oi} = \hat{\mathbf{W}}(1:l, 1:n)\\ \mathbf{W}^{oh} = \hat{\mathbf{W}}(1:l, n+1:(n+m))\end{cases} \tag{17}$$

where $\mathbf{H} = \begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix}$.

And if the rank$(\mathbf{H}) = N$, Eq. (16) could be rewritten as

$$\begin{aligned}\hat{\mathbf{W}} &= \mathbf{Y}\left(\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix}^{T}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix}\right)^{-1}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix}^{T}\\ &= \mathbf{Y}\left(\left[\mathbf{X}^{T}\mathbf{G}^{T}\right]\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix}\right)^{-1}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix}^{T}\\ &= \mathbf{Y}\left(\mathbf{X}^{T}\mathbf{X} + \mathbf{G}^{T}\mathbf{G}\right)^{-1}\mathbf{H}^{T}\end{aligned} \tag{18}$$

And if the rank$(\mathbf{H}) = m + n$, Eq. (16) could be rewritten as

$$\begin{aligned}\hat{\mathbf{W}} &= \mathbf{Y}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix}^{T}\left(\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix}\begin{bmatrix}\mathbf{X}\\ \mathbf{G}\end{bmatrix}^{T}\right)^{-1}\\ &= \mathbf{Y}\mathbf{H}^{T}\left(\mathbf{H}\mathbf{H}^{T}\right)^{-1}\end{aligned} \tag{19}$$

As seen from the above learning process, as the FLN is a parallel connection of a single layer feedforward neural network and a multilayer feedforward network, the output layer nodes not only get the recodification of the external information through the hidden layer nodes, but also get the external information itself directly through the input layer nodes. In addition, many literatures has shown that a single layer feedforward neural network in solving the linear problem with higher efficiency, a multilayer feedforward network can very well realize the complex non-linear mapping from the inputs to the outputs. Then, the FLN has the advantages of the two neural networks, but the ELM does not. So, the FLN with a same or a smaller number of hidden units can achieve much better generalization performance and stability than ELM. In addition, in FLN, the input weights and hidden layer biases are randomly assigned, and the other weights could be analytically determined by least squares methods. So, the FLN could overcome most issues encountered in traditional learning methods and simultaneously own very fast learning speed.

2.4 Learning algorithm for FLN

Suppose, given a training set $S = \{(\mathbf{x}_i, \mathbf{y}_i)|\mathbf{x}_i \in R^n, \mathbf{y}_i \in R^l\}$, hidden node number m and activation function $g(\cdot)$, then the FLN could be summarized as follows:

1. Randomly generate the input weight matrix $\mathbf{W}^{in}$ and the bias matrix $\mathbf{b}$.
2. Calculate the hidden output matrix $\mathbf{G}$ using Eq. (3).
3. Calculate the combination matrix $\mathbf{W}$ using Eq. (9) or Eq. (16).
4. Determine FLN's model parameters Eq. (10) or Eq. (17).

3 Experimental study and discussion

In order to evaluate the proposed FLN algorithm, the FLN is applied to the benchmark problems listed in Table 1, which include 9 regression applications (Auto MPG, Servo, California housing, Bank domains, Machine CPU, Abalone, Delta ailerons, Delta elevators, Boston housing), which are selected from the website: http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html and often adopted to test various learning algorithms [18, 19]. In addition, another three regression methods are employed for performance comparisons.

All evaluations for back propagation (Bp), support vector machine (SVM), ELM and FLN are carried out in Windows XP and Matlab 7.1 environment running on a desktop with AMD Phenom (tm) 9,650 processor 2.31 GHz and 2G RAM. For all data sets listed in Table 1, their observations are randomly split into two parts (training set and testing set) according to the number of samples listed in Table 1.

In order to state the superiority of the FLN, another three methods, SVM, Bp and ELM, are employed to compare the regression accuracy and generalization performance. Firstly, the number of hidden neurons is set as 30 for Bp, ELM and FLN, and the sigmoid function Eq. (20) is chosen as the activation function for the ELM and FLN algorithms.

$$g(x) = \frac{1}{1+\exp(-x)} \tag{20}$$

Table 1 Specification of real-world regression data sets

Data sets	Attributes	Observations	Training set	Testing set
Auto MPG	7	398	200	198
Servo	4	167	87	80
California housing	8	20,460	8,000	12,460
Bank domains	8	8,192	4,500	3,692
Machine CPU	6	209	100	109
Abalone	8	4,177	2,000	2,177
Delta ailerons	6	7,129	3,000	4,129
Delta elevators	6	9,517	4,000	5,517
Boston housing	13	506	250	256

Table 2 Comparison of the mean and its standard deviation of RMSE and the running time for SVM, BP, ELM and FLN using training samples

Data sets	SVM				Bp with 30 hidden nodes			
	RMSE		Time ($\times 10^{-3}$s)		RMSE		Time ($\times 10^{-3}$s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Auto MPG	6.1892×10^{-2}	1.9634×10^{-3}	4,534.687	792.01	8.2505×10^{-2}	8.5109×10^{-3}	3,539.218	12.020
Servo	7.3628×10^{-2}	2.4546×10^{-2}	2,515.625	993.98	9.6938×10^{-2}	1.3267×10^{-2}	2,400.937	12.548
California housing	1.2753×10^{-1}	3.2350×10^{-3}	3.74×10^{7}	4.9×10^{6}	1.5593×10^{-1}	1.4532×10^{-2}	1.09×10^{5}	1,340.8
Bank domains	6.3281×10^{-2}	7.6471×10^{-4}	5.38×10^{6}	7.8×10^{5}	9.7356×10^{-2}	1.0679×10^{-2}	58,029.53	231.53
Machine CPU	4.3571×10^{-2}	5.3401×10^{-3}	395.937	311.21	4.6485×10^{-2}	8.3280×10^{-3}	2,706.406	67.298
Abalone	6.5659×10^{-2}	6.2663×10^{-4}	5.28×10^{5}	3.2×10^{4}	9.3611×10^{-2}	8.0689×10^{-3}	22,319.22	673.91
Delta ailerons	4.0522×10^{-2}	5.5771×10^{-4}	1.19×10^{5}	1.1×10^{4}	7.3916×10^{-2}	1.1920×10^{-2}	37,391.41	22.237
Delta elevators	5.2337×10^{-2}	3.9344×10^{-4}	1.47×10^{6}	8.6×10^{4}	8.4651×10^{-2}	1.1434×10^{-2}	50,125.31	343.56
Boston housing	6.3899×10^{-2}	2.5548×10^{-3}	4,403.12	473.23	9.6166×10^{-2}	9.3308×10^{-3}	4,207.031	19.598
Data sets	ELM with 30 hidden nodes				FLN with 30 hidden nodes			
	RMSE		Time ($\times 10^{-3}$s)		RMSE		Time ($\times 10^{-3}$s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Auto MPG	6.9518×10^{-2}	4.5487×10^{-3}	3.1250	6.3568	6.0963×10^{-2}	4.3071×10^{-3}	4.5312	8.0976
Servo	6.8669×10^{-2}	1.3864×10^{-2}	2.0833	5.4023	6.2947×10^{-2}	1.0714×10^{-2}	2.0313	5.2812
California housing	1.3592×10^{-1}	2.1420×10^{-3}	104.1667	8.5418	1.3269×10^{-1}	2.0771×10^{-2}	145.7812	8.0242
Bank domains	5.7120×10^{-2}	3.0826×10^{-3}	53.1250	8.8006	4.7112×10^{-2}	9.4812×10^{-4}	77.1875	4.3405
Machine CPU	1.9435×10^{-2}	2.8173×10^{-3}	2.0833	5.4022	1.6634×10^{-2}	2.0574×10^{-3}	1.4062	5.0129
Abalone	7.4119×10^{-2}	1.1956×10^{-3}	19.2708	6.7216	7.4015×10^{-2}	1.5086×10^{-3}	31.4062	5.2059
Delta ailerons	3.8452×10^{-2}	8.2847×10^{-4}	32.2917	3.9642	3.8114×10^{-2}	7.6419×10^{-4}	42.6562	8.8430
Delta elevators	5.3109×10^{-2}	4.3674×10^{-4}	43.7500	6.3568	5.2822×10^{-2}	6.2815×10^{-4}	60.9375	5.6621
Boston housing	8.6356×10^{-2}	6.0737×10^{-3}	4.6875	7.2826	7.4124×10^{-2}	6.6907×10^{-3}	5.1562	8.0242

In addition, for each problem, we estimate the regression accuracy using different combination of cost parameters C and the kernel parameters γ, $C = [2^{-2}, 2^{-1}, \ldots, 2^{11}, 2^{12}]$ and $\gamma = [2^{-10}, 2^{-9}, \ldots, 2^{3}, 2^{4}]$, so the SVM would be tried $15 \times 15 = 225$ parameter combinations (C, γ) in order to find one to make SVM show best performance. Here, the search parameter combination process is thought of a part of the training process.

Every experiment is repeated 30 times. The mean and its standard deviations (SD) of root-mean-square error (RMSE) for the four methods are given in Table 2 for training samples and Table 3 for testing samples.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{21}$$

$$\mathrm{SD} = \sqrt{\frac{\sum_{i=1}^{n}\left(\hat{y}_i - \bar{y}\right)^2}{n-1}} \tag{22}$$

As seen from Table 2, for the training samples for data sets, compare with Bp and ELM, the FLN shows better regression accuracy on the all regression applications listed in Table 1. In addition, for the standard deviations, the FLN shows better performance on five applications (Auto MPG, Servo, Bank domains, Machine CPU and Delta ailerons), ELM on three applications (Abalone, Delta elevators and Boston housing) and Bp on one (California housing). Compared with SVM, for the regression accuracy, the FLN shows better superiority on five applications (Auto MPG, Servo, Bank domains, Machine CPU and Delta ailerons) and SVM on the other four applications. For the learning time, the ELM needs the least one; the next is FLN, then Bp, and the last SVM. In short, although the SVM shows better regression performance on some applications than FLN, it spends much more time to search better parameter combination. Although the ELM needs less learning time than ELM, the regression performance is not better than the FLN.

As seen from Table 3, for testing samples for data sets, compared with SVM, Bp and ELM, the FLN shows the best generalization ability on seven applications (Auto MPG, Servo, California housing, Bank domains, Abalone, Delta ailerons and Delta elevators), and for the other two data sets (Machine CPU and Boston housing), the SVM

Table 3 Comparison of the mean and standard deviation of RMSE and the running time for SVM, BP, ELM and FLN using testing samples

Data sets	SVM				Bp with 30 hidden nodes			
	RMSE		Time ($\times 10^{-3}$s)		RMSE		Time ($\times 10^{-3}$s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Auto MPG	8.5228×10^{-2}	9.1687×10^{-3}	1.7187	4.9135	1.0368×10^{-1}	1.5861×10^{-2}	4.2187	6.9718
Servo	1.2330×10^{-1}	1.6832×10^{-2}	0	0	1.8380×10^{-1}	3.8785×10^{-2}	3.4375	6.5052
California housing	1.4107×10^{-1}	4.6542×10^{-3}	871.246	62.184	1.6065×10^{-1}	2.8987×10^{-2}	93.2812	4.1280
Bank domains	6.8617×10^{-2}	5.7403×10^{-3}	542.732	49.561	9.9465×10^{-2}	1.1213×10^{-2}	28.1250	6.2814
Machine CPU	6.5646×10^{-2}	1.7430×10^{-2}	0.6250	3.0772	3.1303×10^{-1}	2.6623×10^{-1}	4.0625	6.8882
Abalone	9.3852×10^{-2}	2.7022×10^{-3}	223.081	16.526	9.8343×10^{-2}	1.3502×10^{-2}	17.9687	5.6073
Delta ailerons	4.4745×10^{-2}	2.5386×10^{-3}	51.5625	10.546	7.5887×10^{-2}	1.2511×10^{-2}	30.6250	3.0773
Delta elevators	5.6289×10^{-2}	1.0955×10^{-3}	221.875	19.207	8.6147×10^{-2}	1.1841×10^{-2}	40.3125	7.7507
Boston housing	9.6235×10^{-2}	9.4474×10^{-2}	3.1250	6.5880	1.3635×10^{-1}	3.3672×10^{-2}	5.1562	7.3840
Data sets	ELM with 30 hidden nodes				FLN with 30 hidden nodes			
	RMSE		Time ($\times 10^{-3}$s)		RMSE		Time ($\times 10^{-3}$s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Auto MPG	8.2210×10^{-2}	5.3998×10^{-3}	1.0417	3.9642	7.9443×10^{-2}	5.2732×10^{-3}	1.0937	4.0067
Servo	1.2592×10^{-1}	2.1971×10^{-2}	0	0	1.2113×10^{-1}	3.8030×10^{-2}	0.3125	2.1985
California housing	1.3635×10^{-1}	2.2256×10^{-3}	58.8541	6.7216	1.3445×10^{-1}	1.8493×10^{-3}	70.937	8.4333
Bank domains	5.7807×10^{-2}	3.3830×10^{-3}	16.6667	3.9642	4.7836×10^{-2}	9.5635×10^{-4}	18.5937	6.1605
Machine CPU	1.2887×10^{-1}	7.5468×10^{-2}	0.5208	2.8527	1.9366×10^{-1}	1.3955×10^{-2}	0.6250	3.0772
Abalone	7.7442×10^{-2}	1.7851×10^{-3}	10.4167	9.4762	7.7221×10^{-2}	1.9985×10^{-3}	10.3125	7.4389
Delta ailerons	3.9237×10^{-2}	5.868×10^{-4}	17.7083	5.4023	3.9314×10^{-2}	5.7717×10^{-4}	18.125	5.7571
Delta elevators	5.3443×10^{-2}	3.6202×10^{-4}	21.3542	7.6583	5.3551×10^{-2}	4.7900×10^{-4}	25.3125	7.6223
Boston housing	1.1078×10^{-1}	1.1522×10^{-2}	2.0833	6.7839	1.0209×10^{-1}	1.0096×10^{-2}	1.7188	5.3921

shows best. For the stability, the SVM has best performance on one application (Servo), ELM on two applications (Abalone and Boston housing) and FLN on the other six ones. In short, the FLN has the better generalization ability and stability on most applications.

In addition, in order to further state the FLN validity, the hidden neurons of ELM and FLN are reset as 20, and the sigmoid function is still adopted as the activation function. The related results are reported in Table 4.

As seen from Table 4, we could know that, although ELM needs a little less learning and responding time than the FLN on most data sets, the FLN shows much better regression accuracy on the all data sets and also has the better generalization ability on 8 data sets. The ELM shows better generalization ability on one application (Machine CPU).

As seen from Tables 2, 3 and 4, with the hidden neurons reducing, the regression accuracy of ELM and FLN would reduce on all data sets, and the learning and responding time is also reducing on most applications. However, the generalization ability is increasing on 2 applications (Auto MPG and Machine CPU) for ELM and on 4 applications (Auto MPG, Servo, Machine CPU and Abalone) for FLN. In addition, whatever the number of hidden neurons is 30 or 20, if only the number of hidden neurons is set as the same number, the FLN could show better regression accuracy and generalization ability on most applications than ELM.

Compared with ELM with 30 hidden neurons, for the training samples of all data sets, although the ELM shows better regression accuracy than the FLN with 20 hidden neurons on six regression applications (Auto MPG, Servo, California housing, Machine CPU, Abalone and Delta ailerons), the FLN with 20 hidden neurons could show better superiority on only three applications (Bank domains, Delta elevators and Boston housing). For testing set for all applications, the FLN with 20 hidden neurons could show better generalization ability than ELM with 30 hidden neurons on seven applications (Auto MPG, Servo, Bank domains, Machine CPU, Abalone, Delta ailerons and Boston housing). However, the ELM shows better performance than the FLN on only two applications (California

Table 4 Comparison of the mean and standard deviation of RMSE and the running time for ELM and FLN using training and testing samples

Data sets	ELM (training samples) with 20 hidden nodes				ELM (testing samples) with 20 hidden nodes			
	RMSE		Time ($\times 10^{-3}$s)		RMSE		Time ($\times 10^{-3}$s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Auto MPG	6.9665×10^{-2}	4.5038×10^{-3}	1.7188	4.9135	8.0270×10^{-2}	5.7331×10^{-3}	1.0938	4.0068
Servo	9.2938×10^{-2}	1.4568×10^{-2}	0.7813	3.4225	1.2798×10^{-1}	1.8046×10^{-2}	0.6250	3.0773
California housing	1.4015×10^{-1}	2.0989×10^{-3}	57.6563	7.2628	1.4070×10^{-1}	1.9726×10^{-3}	39.218	7.8502
Bank domains	6.6323×10^{-2}	7.9992×10^{-3}	30.625	3.7950	6.6763×10^{-2}	7.8527×10^{-3}	9.8437	7.9004
Machine CPU	2.2979×10^{-2}	3.1875×10^{-3}	2.0312	5.2812	8.3653×10^{-2}	4.1265×10^{-2}	0.4687	2.6788
Abalone	7.5865×10^{-2}	1.5442×10^{-3}	12.0312	6.6086	7.7460×10^{-2}	1.6104×10^{-3}	6.0937	7.6595
Delta ailerons	3.8851×10^{-2}	6.5961×10^{-4}	18.1250	6.9238	3.9520×10^{-2}	5.3632×10^{-4}	10.937	7.1963
Delta elevators	5.3281×10^{-2}	6.3806×10^{-4}	25.9375	7.4389	5.3605×10^{-2}	4.7367×10^{-4}	15.3125	4.4305
Boston housing	9.9456×10^{-2}	7.5832×10^{-3}	2.0312	6.1445	1.1565×10^{-1}	7.9467×10^{-3}	1.40625	5.0129
Data sets	FLN (training samples) with 20 hidden nodes				FLN (testing samples) with 20 hidden nodes			
	RMSE		Time ($\times 10^{-3}$s)		RMSE		Time ($\times 10^{-3}$s)	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Auto MPG	6.6315×10^{-2}	4.1711×10^{-3}	2.6042	5.9226	7.8921×10^{-2}	4.8938×10^{-3}	1.5625	4.7676
Servo	7.8692×10^{-2}	1.3279×10^{-2}	1.0416	3.9641	1.1972×10^{-1}	1.9759×10^{-2}	0	0
California housing	1.3630×10^{-1}	1.9502×10^{-3}	83.8541	7.6583	1.3696×10^{-1}	1.7131×10^{-3}	46.3541	2.8527
Bank domains	4.7622×10^{-2}	7.7918×10^{-4}	45.3125	4.7676	4.8183×10^{-2}	9.3479×10^{-4}	10.9375	7.2826
Machine CPU	2.0317×10^{-2}	2.2557×10^{-3}	1.5625	4.7676	1.1859×10^{-1}	6.5693×10^{-2}	0	0
Abalone	7.4823×10^{-2}	1.7943×10^{-3}	17.1875	4.7676	7.6933×10^{-2}	2.1051×10^{-3}	6.2500	7.7855
Delta ailerons	3.8723×10^{-2}	6.6724×10^{-4}	23.4375	7.9461	3.9323×10^{-2}	4.7892×10^{-4}	11.9791	6.7216
Delta elevators	5.3097×10^{-2}	6.8177×10^{-4}	34.8958	8.8800	5.3560×10^{-2}	5.4503×10^{-4}	17.7083	5.4022
Boston housing	8.2361×10^{-2}	8.0322×10^{-3}	3.1250	7.4916	1.0489×10^{-1}	9.1439×10^{-3}	1.0416	3.9641

housing and Delta elevators). In addition, the FLN could have better stability than ELM on most regression applications according to the standard deviations, and a faster speed for both learning and testing than ELM under the assigned number of hidden neurons. Especially, the more the data set contains samples, the less time FLN spends in learning and testing than ELM. Therefore, the proposed FLN with less hidden neurons could achieve good generalization performance and stability with much more compact networks and fast speed.

In addition, in order to state the validity of the sigmoid function, another two activation functions: 'Hardlim' and 'Sine,' are employed to compare the performance of the FLN. Here, the number of hidden neurons is still set as 20. The related results are recorded in Table 5.

$$\text{'Hardlim'} : g(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{23}$$

$$\text{'Sine'} : g(x) = \sin(x) \tag{24}$$

As seen from Table 5, for the training sample of all data sets, the FLN with 'Sigmoid' has a better regression accuracy on six applications (Auto MPG, Servo, Bank domains Machine CPU, Delta ailerons and Delta elevators) and the FLN with 'Sine' on three applications (California housing, Abalone and Boston housing). For the testing samples of data sets, the FLN with 'Sigmoid' has the better generalization ability than the FLN with the other two activation functions on seven applications, the FLN with 'Hardlim' on one application (Machine CPU) and the FLN with 'Sine' on one application (Boston housing). According to the standard deviation of RMSE, the FLN with 'Sigmoid' has better stability for the training and testing sample on most applications. So, the 'Sigmoid' as the activation function of FLN makes the regression accuracy, generalization ability and stability of FLN better.

As observed from Fig. 2, it shows the relationship between the generalization performance of FLN and its network size for the Bank domains. The testing RMSE decreases with the increase in the number of hidden layer neurons, so the generalization performance of FLN is more and more stable with the increase in the number of hidden layer neurons.

Table 5 The mean and standard deviation of RMSE for FLN with 20 hidden nodes and different activation functions

Data sets	FLN with 'Hardlim'				FLN with 'Sine'				FLN with 'Sigmoid'			
	Training samples		Testing samples		Training samples		Testing samples		Training samples		Testing samples	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Auto MPG	7.5536×10^{-2}	4.8492×10^{-3}	9.0295×10^{-2}	7.6180×10^{-3}	6.6671×10^{-2}	4.2359×10^{-3}	8.0528×10^{-2}	5.7606×10^{-3}	6.6315×10^{-2}	4.1711×10^{-3}	7.8921×10^{-2}	4.8938×10^{-3}
Servo	1.2112×10^{-1}	1.4948×10^{-2}	1.7234×10^{-1}	2.1745×10^{-2}	8.0899×10^{-2}	1.2716×10^{-2}	1.2557×10^{-1}	1.9924×10^{-2}	7.8692×10^{-2}	1.3279×10^{-2}	1.1972×10^{-1}	1.9759×10^{-2}
California housing	1.4041×10^{-1}	1.8097×10^{-3}	1.4191×10^{-1}	1.6822×10^{-3}	1.3583×10^{-1}	2.0416×10^{-3}	1.3740×10^{-1}	1.9233×10^{-3}	1.3630×10^{-1}	1.9502×10^{-3}	1.3696×10^{-1}	1.7131×10^{-3}
Bank domains	7.9208×10^{-2}	1.7588×10^{-2}	8.0569×10^{-2}	1.7421×10^{-2}	1.0342×10^{-1}	2.1341×10^{-2}	1.0441×10^{-1}	2.1168×10^{-2}	4.7622×10^{-2}	7.7918×10^{-4}	4.8183×10^{-2}	9.3479×10^{-4}
Machine CPU	2.5690×10^{-2}	4.4279×10^{-3}	7.8655×10^{-2}	2.7201×10^{-2}	2.0395×10^{-2}	2.7610×10^{-3}	1.4423×10^{-1}	9.9251×10^{-2}	2.0317×10^{-2}	2.2557×10^{-3}	1.1859×10^{-1}	6.5693×10^{-2}
Abalone	7.6376×10^{-2}	1.3885×10^{-3}	7.8202×10^{-2}	1.3363×10^{-3}	7.462910^{-2}	1.5609×10^{-3}	7.712510^{-2}	1.6453×10^{-3}	7.4823×10^{-2}	1.7943×10^{-3}	7.6933×10^{-2}	2.1051×10^{-3}
Delta ailerons	3.9728×10^{-2}	1.6118×10^{-3}	4.0809×10^{-2}	1.6474×10^{-3}	3.8940×10^{-2}	8.0971×10^{-4}	3.9693×10^{-2}	6.8044×10^{-4}	3.8723×10^{-2}	6.6724×10^{-4}	3.9323×10^{-2}	4.7892×10^{-4}
Delta elevators	5.5228×10^{-2}	2.3429×10^{-3}	5.5970×10^{-2}	2.3349×10^{-3}	5.4423×10^{-2}	1.0724×10^{-3}	5.4943×10^{-2}	1.1266×10^{-3}	5.3097×10^{-2}	6.8177×10^{-4}	5.3560×10^{-2}	5.4503×10^{-4}
Boston housing	9.2175×10^{-2}	6.5267×10^{-3}	1.1075×10^{-1}	7.3173×10^{-3}	8.1495×10^{-2}	6.2829×10^{-3}	1.0334×10^{-1}	8.7550×10^{-3}	8.2361×10^{-2}	8.0322×10^{-3}	1.0489×10^{-1}	9.1439×10^{-3}

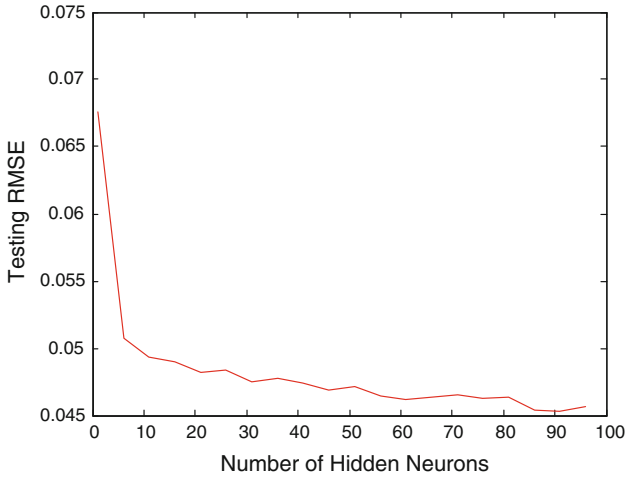

Fig. 2 The relationship between the generalization performance and number of hidden neurons

In summation, the FLN with much more compact networks and fast speed could achieve good generalization performance and stability on most functions. So, the FLN could be thought of an effective machine learning tool.

4 Real-world design problem

For studying the influence of adjustable boiler operation parameters for the thermal efficiency and NO_x emissions of a 330 WM double-furnace coal-fired boiler, 20 cases were carried out under various operating conditions. The boiler equips a coal pulverizing system with intermediate silo, 4 coal pulverizers, 4 mill exhausters and 32 pulverized fuel feeders. In order to set up a function between the thermal efficiency/NO_x emissions and operating conditions, the proposed FLN is employed.

Firstly, as seen from the Table 1 of the Ref. [20], Cases 1–3, Case 4, Cases 5–6, Cases 7–14 and Cases 15–20, respectively, work in different properties of coal-fired. According to the coal characteristics, the 20 cases are divided into two parts: training samples (17 cases: Cases 1–2, Cases 4–11, Cases 13–16 and Cases 18–20) and testing samples (3 cases: Case 3, Case 12 and Case 17), in order to set up the function relation between the thermal efficiency/NO_x emissions and operating conditions, and test its validity. The thermal efficiency/NO_x emissions mainly depend on 26 operational conditions given as follows.

- The boiler load (MW).
- The coal feeder rotation speed (CFRS, r min^{-1}), including A, B, C, D levels.
- The primary air velocity, including A, B, C, D levels.

Fig. 3 Simplified boiler model

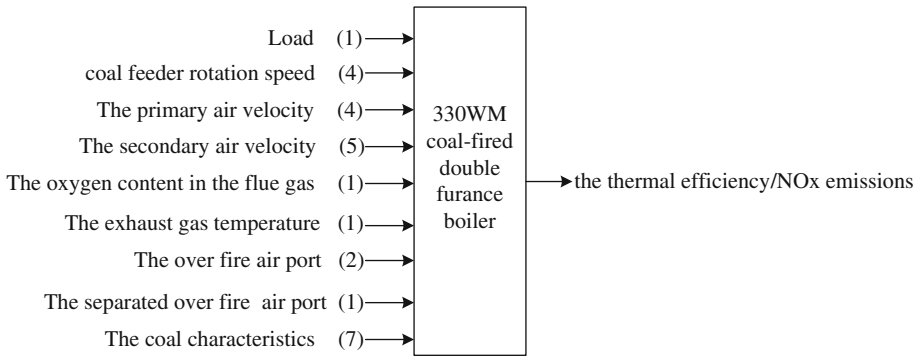

Table 6 Training time and related values of RMSE for the thermal efficiency

Method	RMSE				Time (s)
	Min	Max	Med	Mean	
Bp	7.063×10^{-1}	7.063×10^{-1}	7.063×10^{-1}	7.063×10^{-1}	0.9761
SVM	9.4612×10^{-2}	9.4612×10^{-2}	9.4612×10^{-2}	9.4612×10^{-2}	1.4035
LSSVM	1.6190×10^{-4}	1.6190×10^{-4}	1.6190×10^{-4}	1.6190×10^{-4}	3.7975
ELM	4.5983×10^{-1}	7.3271×10^{-1}	6.1009×10^{-1}	6.0817×10^{-1}	$\mathbf{6.2500 \times 10^{-4}}$
FLN	$\mathbf{1.2673 \times 10^{-13}}$	$\mathbf{1.2454 \times 10^{-12}}$	$\mathbf{3.5409 \times 10^{-13}}$	$\mathbf{4.3855 \times 10^{-13}}$	$\mathbf{9.3750 \times 10^{-4}}$

Values given in bold are the best one

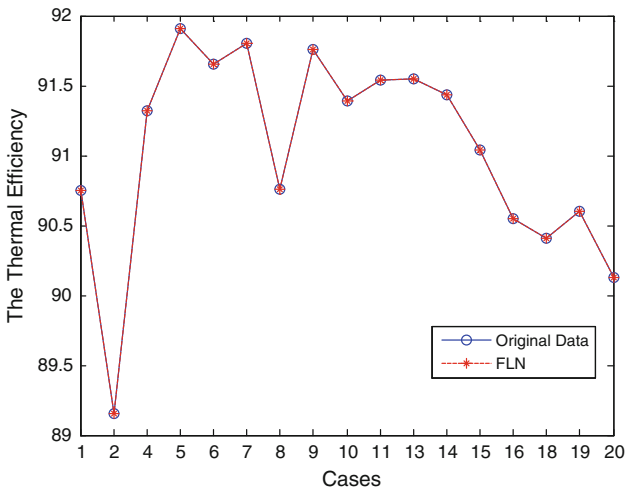

Fig. 4 The prediction values of the thermal efficiency for training samples

- The secondary air velocity (m s^{-1}), including AA, AB, BC, CD, DE levels.
- The oxygen content in the flue gas (OC, %).
- The exhaust gas temperature (EGT, °C).
- The over fire air port (OFA, %)
- The separated over fire air port (SOFA, %).
- The coal characteristics including content of carbon (Car, %), hydrogen (Har, %), oxygen (Oar, %), nitrogen (Nar, %), water (War, %), volatile (Var, %), heat value (Qdw.ar, kJ/kg).

The proposed FLN is adopted to set up mathematical models of the thermal efficiency/NO_x emissions, which could state the mapping relation between the thermal efficiency/NO_x emissions and operational conditions. The complex mapping function relation could be simplified as a model which is shown in Fig. 3.

In order to test the model repeatability and validity, every experiment is repeated 30 times. The minimum, maximum, median and mean of the root-mean-square error (RMSE) have been recorded for the training samples. And simultaneously, the minimum, maximum, median and mean of the prediction values are also recorded for testing samples. For the thermal efficiency, the training time and related values of RMSE are given in Table 6, the prediction values of training samples are shown in Fig. 4, its error comparisons are shown in Fig. 5, and the prediction values of testing samples are given in Table 7. For NO_x emissions, the training time and related values of RMSE are given in Table 8, the prediction values of training samples are shown in Fig. 6, its error comparisons are shown in Fig. 7, and the prediction values of testing samples are given in Table 9.

As to the comparison of the thermal efficiency's model, just as we can see from Tables 6, 7 and Figs. 4, 5, the minimum, maximum, median and mean of the RMSE and the prediction values of testing samples remain unchanged for Bp, SVM and LSSVM in 30 training models. Although these values have some variations for FLN, these variations are too small. But for ELM, the minimum, maximum, median and mean of RMSE are different, and the

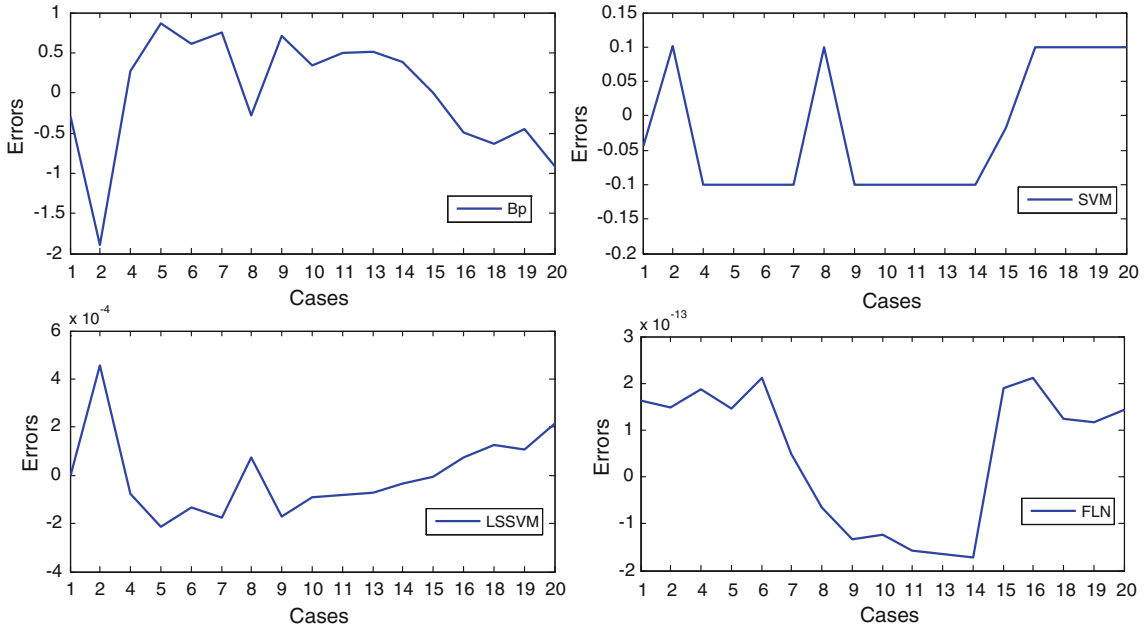

Fig. 5 The prediction errors of the thermal efficiency for training samples

Table 7 Prediction values of testing samples for the thermal efficiency

Case	Original data	Bp	SVM	ELM	LSSVM	FLN
3	89.46					
	Min	91.0435	90.9746	-7.27×10^{-4}	90.3505	90.2680
	Max	91.0435	90.9746	362.53	90.3505	90.6963
	Med	91.0435	90.9746	90.9738	90.3505	90.3705
	Mean	91.0435	90.9746	-6.34×10^{-2}	90.3505	90.3799
12	91.57					
	Min	91.0435	91.0229	88.1218	91.1729	91.9602
	Max	91.0435	91.0229	1.34×10^{6}	91.1729	92.7832
	Med	91.0435	91.0229	91.2796	91.1729	92.2220
	Mean	91.0435	91.0229	1.37×10^{4}	91.1729	92.2689
17	90.36					
	Min	91.0435	91.0177	89.3654	90.4588	90.2555
	Max	91.0435	91.0177	91.7367	90.4588	90.6306
	Med	91.0435	91.0177	90.7685	90.4588	90.4804
	Mean	91.0435	91.0177	90.7735	90.4588	90.4702

difference is very large. So, ELM's prediction errors of training samples are not shown in Fig. 5. For the prediction precision of training samples, the FLN's RMSE is the least among all the 5 methods, whose values achieve 10^{-13}. The runtime of training process is also the least, except for ELM. As seen from the prediction values of testing samples, the LSSVM's prediction errors are the least one, secondly FLN. The Bp's prediction values are same for different cases, and ELM's some prediction values of testing samples overstep the limitation of the thermal efficiency. Therefore, we could think the ELM and BP are out of work for the prediction of the thermal efficiency. In a word, although LSSVM has better generalization ability than FLN, FLN's generalization ability is also very well and better than those of Bp, SVM and ELM; in addition, FLN's learning time and RMSE of training samples are much better than any one of Bp, SVM and LSSVM.

As to the comparison of NO_x emissions' model, just as we can see from Tables 8, 9 and Figs. 6, 7, the minimum, maximum, median and mean of the RMSE and the prediction values of testing samples keep unchanged as the thermal efficiency's model for Bp, SVM and LSSVM.

Table 8 Training time and related values of RMSE for NO_x emissions

Method	RMSE				Time (s)
	Min	Max	Med	Mean	
Bp	93.0858	93.0858	93.0858	93.0858	1.0268
SVM	9.9957×10^{-2}	9.9957×10^{-2}	9.9957×10^{-2}	9.9957×10^{-2}	1.0739
LSSVM	1.9083×10^{-2}	1.9083×10^{-2}	1.9083×10^{-2}	1.9083×10^{-2}	3.7678
ELM	33.7875	92.7139	71.9961	69.6539	$\mathbf{7.8125 \times 10^{-4}}$
FLN	$\mathbf{9.5794 \times 10^{-13}}$	$\mathbf{7.5443 \times 10^{-12}}$	$\mathbf{2.5816 \times 10^{-12}}$	$\mathbf{2.8531 \times 10^{-12}}$	$\mathbf{4.6875 \times 10^{-4}}$

Values given in bold are the best one

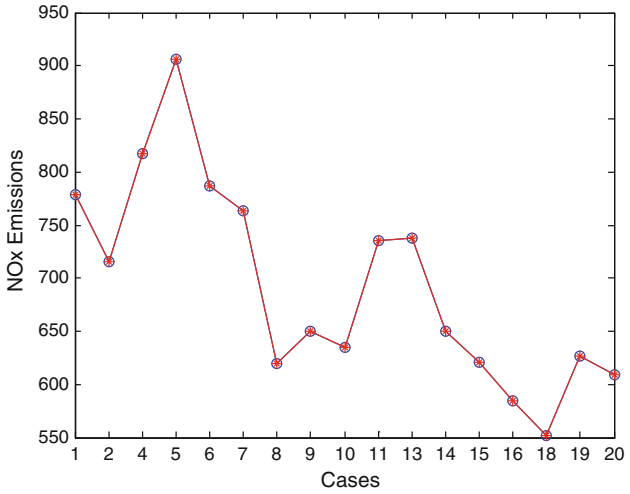

Fig. 6 The prediction values of the thermal efficiency for training samples

Although there are tiny variations in these values of FLN, they are negligible. But for ELM, there are very large variations in these values, so ELM's prediction errors of training samples are not shown in Fig. 6. According to the RMSE, the training precision of FLN is the best among the five methods, and simultaneously, the FLN needs the least training time. For the predicted NO_x emission of testing samples, the prediction values of Bp and SVM are same for different cases, and some ELM's prediction values overstep the limitation of NO_x emissions; therefore, we may think the ELM, BP and SVM are out of work for the prediction of NO_x emissions. Although the predicted NO_x emissions of FLN are not better than those of LSSVM for Case 12, majority of predicted NO_x emissions of FLN outperform those of LSSVM for Case 3 and Case 17 in 30 prediction models. So, the FLN has very good prediction precision and generalization ability with little runtime for training process.

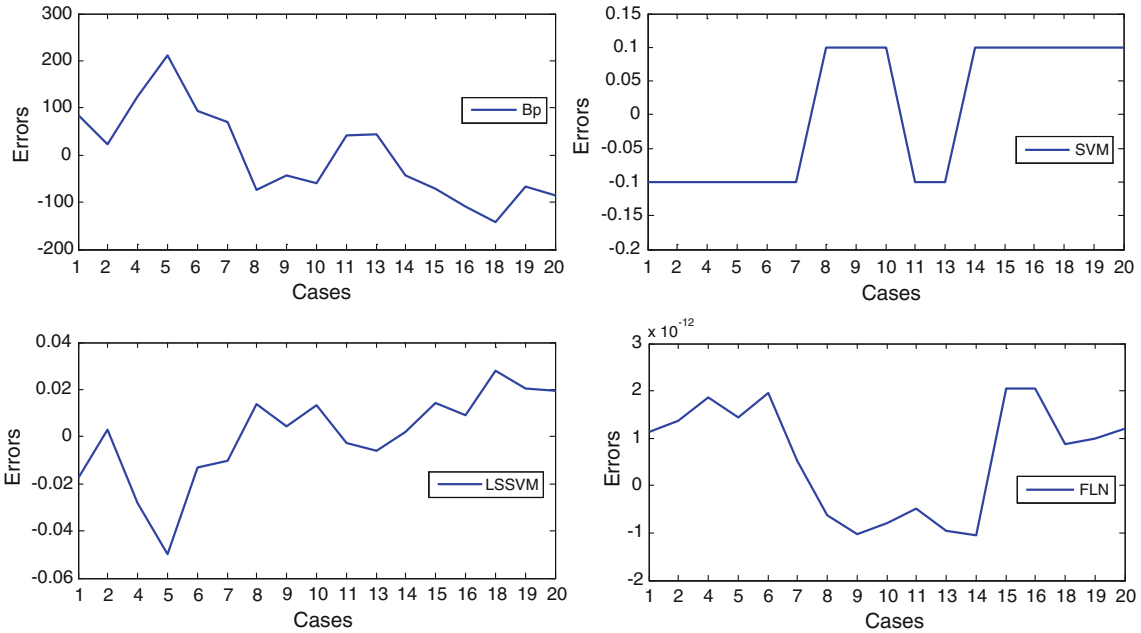

Fig. 7 The prediction errors of NO_x emissions for training samples

Table 9 Prediction values of testing samples for NO_x emissions

Case	Original data	Bp	SVM	ELM	LSSVM	FLN
3	753.5					
	Min	693.47	693.48	586.01	741.86	751.36
	Max	693.47	693.48	6.72×10^4	741.86	785.63
	Med	693.47	693.48	727.82	741.86	762.46
	Mean	693.47	693.48	1.39×10^4	741.86	763.42
12	754.35					
	Min	693.47	693.48	-3.2×10^7	725.17	788.35
	Max	693.47	693.48	7.39×10^9	725.17	851.24
	Med	693.47	693.48	714.42	725.17	806.98
	Mean	693.47	693.48	9.41×10^7	725.17	811.08
17	577					
	Min	693.47	693.48	501.95	557.85	547.65
	Max	693.47	693.48	749.14	557.85	583.75
	Med	693.47	693.48	647.36	557.85	574.40
	Mean	693.47	693.48	640.81	557.85	571.89

In summary, the FLN has very good prediction precision and generalization ability and simultaneously needs little learning time in the training process. So, the FLN may set up very good models of the thermal efficiency and NO_x emissions and proves to be a good effective learning method.

5 Conclusions

This paper proposes a novel artificial neural network, fast learning network (FLN), which is a double parallel forward neural network. In FLN, the input weights and hidden layer biases are randomly generated like ELM, and the other weights and biases are would be analytically determined based on least squares methods. In order to test FLN's validity, it is applied to 9 regression problems. Experimental results show that compared with SVM, Bp and ELM, the FLN with the same hidden neurons can achieve better regression accuracy, generalization performance and stability at a very fast speed. Although the hidden neuron's number of FLN is less than that of ELM, the FLN still has better generalization performance and stability than ELM on most applications. In addition, the FLN is applied to a 330 WM coal-fired boiler. Compared with other learning methods (such as Bp, ELM, SVR and LSSVR), the FLN can achieve better regression accuracy, generalization performance and stability at a high speed. So, the FLN could be a useful machine learning tool and be applied to various applications like other intelligent learning machines.

Acknowledgments This project is supported by the National Natural Science Foundation of China (Grant No. 60774028) and Natural Science Foundation of Hebei Province, China (Grant No.F2010001318).

References

1. Green M, Ekelund U, Edenbrandt L, Björk J, Forberg JL, Ohlsson M (2009) Exploring new possibilities for case-based explanation of artificial neural network ensembles. Neural Netw 22:75–81
2. May RJ, Maier HR, Dandy GC (2010) Data splitting for artificial neural networks using SOM-based stratified sampling. Neural Netw 23:283–294
3. Kiranyaz S, Ince T, Yildirim A, Gabbouj M (2009) Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. Neural Netw 22:1448–1462
4. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
5. Suresh S, Venkatesh Babu R, Kim HJ (2009) No-reference image quality assessment using modified extreme learning machine classifier. Appl Soft Comput 9:541–552
6. Li G, Niu P (2011) An enhanced extreme learning machine based on ridge regression for regression. Neural Comput Appl. doi: 10.1007/s00521-011-0771-7
7. Romero E, Alquézar R (2012) Comparing error minimized extreme learning machines and support vector sequential feed-forward neural networks. Neural Netw 25:122–129
8. Zhu Q-Y, Qin AK, Suganthan PN, Huang G-B (2005) Evolutionary extreme learning machine. Pattern Recogn 38:1759–1763
9. Huynh HT, Won Y (2008) Small number of hidden units for ELM with two-stage linear model. IEICE Trans Inform Syst 91-D:1042–1049
10. He M (1993) Theory, application and related problems of double parallel feedforward neural networks. Ph.D. thesis, Xidian University, Xi'an
11. Wang J, Wu W, Li Z, Li L (2011) Convergence of gradient method for double parallel feedforward neural network. Int J Numer Anal Model 8:484–495

12. Tamura S, Tateishi M (1997) Capabilities of a four-layered feedforward neural network: four layers versus three. IEEE Trans Neural Netw 8:251–255
13. Huang G-B (1998) Learning capability of neural networks. Ph.D. thesis, Nanyang Technological University, Singapore
14. Huang G-B (2003) Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Trans Neural Netw 14:274–281
15. Huang G-B, Chen L, Siew C-K (2006) Universal approximation using incremental networks with random hidden computation nodes. IEEE Trans Neural Netw 17:879–892
16. Rao CR, Mitra SK (1971) Generalized inverse of matrices and its applications. Wiley, New York
17. Serre D (2002) Matrices: theory and applications. Springer, New York
18. Liang N-Y, Huang G-B, Saratchandran P, Sundararajan N (2006) A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw 17:1411–1423
19. Lan Y, Soh YC, Huang G-B (2010) Two-stage extreme learning machine for regression. Neurocomputing 73:3028–3038
20. Xu C, Lu J, Zheng Y (2006) An experiment and analysis for a boiler combustion optimization on efficiency and NO_x emissions. Boil Technol 37:69–74