Engineering Applications of Artificial Intelligence 62 (2017) 38–50

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

A new Self-Organizing Extreme Learning Machine soft sensor model and its applications in complicated chemical processes

Zhiqiang Geng[a,b], Jungen Dong[a,b], Jie Chen[a,b], Yongming Han[a,b,*]

[a] College of Information Science & Technology, Beijing University of Chemical Technology, Beijing, China
[b] Engineering Research Center of Intelligent PSE, Ministry of Education in China, Beijing, China

ARTICLE INFO

Keywords:
Self-Organizing
Extreme Learning Machine
Mutual Information
Hebbian learning rule
Soft sensor
Complicated chemical processes

ABSTRACT

The control of product quality of complex chemical processes strictly depends on the measure of the key process variables. However, the online measure device is extremely expensive, and these devices are hard to protect. Meanwhile, there is a delay for these online measure devices. Therefore, the soft sensor technology plays a vital role in measuring the key process variables. Extreme Learning Machine (ELM) is an efficient and simple single layer feed-forward neural networks (SLFNs) to building an exact soft sensor model. However, unsuitable selected hidden nodes and random parameters will greatly affect the performance of the ELM. Therefore, this paper proposes a novel Self-Organizing Extreme Learning Machine (SOELM) algorithm constructed by the biological neuron-glia interaction principle to solve the issue of the ELM. Firstly, the weights between input layer nodes and the CNS are tuned iteratively by the Hebbian learning rule. Then the network structure is adjusted self-organizing by Mutual Information (MI) among different structures of networks. Secondly, the weights between the CNS and output layer nodes are obtained by the ELM. The experimental results based on different UCI data sets prove that the SOELM has a better generalization capability and stability than that of the ELM. Moreover, our proposed method is developed as a soft sensor model for accurately predicting the key variables of the Purified Terephthalic Acid (PTA) process.

1. Introduction

Complex chemical processes make a difference in developing the industry development in most countries. At the same time, the PTA (purified terephthalic acid) production is the main process in chemical industries. Meanwhile, the demand of PTA has increased in recent years and has become a significant row material in chemical industries. Unfortunately, on account of the high cost of adding a new plant, the overall energy efficiency has decreased (Li and Sun, 2013). Meanwhile, some complicate chemical process are solved by soft sensor, as the synthesis of chemical product depends on soft chemical process (Rizwan and Farheen, 2015), and the automation of a reactor flow system (Edwin and Peter, 1998), and monitoring the process of curing of epoxy/graphite fiber composites (Hong et al., 1998). In summary, it is an effective way to improve the productivity and energy efficiency of the complex chemical process by building the soft sensor model.

Under some circumstances, accurately estimating the key process variables is extremely helpful to improve the control performance in the industrial processes (Ge and Song, 2010). In the process industry, measuring devices could be very expensive, could introduce significant delays, and the implemental technology could be extremely difficult, so we can't be easy to measure the key process variable online (Liu et al., 2013). Therefore, soft sensors have attracted more and more attention as a substitution of the hard device for either the monitoring purpose or the control purpose (Ge and Song, 2010), However, based on mechanism models, building an accurate soft sensor is more and more difficult and time consumed because many industries are more and more complicated especially the chemical and petrochemical industries. Some researchers have successfully adopted the Extreme Learning Machine (ELM) to develop soft sensors for complicated industrial processes (Zhang and Yin, 2016; Mortaza et al., 2016; Chakchai et al., 2016).

Although many self-organizing learning algorithms were proposed in the past, the study of self-organizing ELM has not been seen recently. This paper proposed a novel Self-Organizing Extreme Learning Machine (SOELM) algorithm constructed by neurons and glias according to biological neuron-glia interaction principles. The glia provide energy for the neuron, and the stability of the CNS depends on the entropy of all neurons. At the same time, the novel SOELM algorithm is also proposed to determine the self-organizing network

* Corresponding author at: College of Information Science & Technology, Beijing University of Chemical Technology, Beijing, China.
E-mail address: hanym@mail.buct.edu.cn (Y. Han).

http://dx.doi.org/10.1016/j.engappai.2017.03.011
Received 1 December 2016; Received in revised form 10 March 2017; Accepted 31 March 2017

structure adaptively including hidden nodes and linking weights. The CNS is analyzed and designed based on distributions of neurons and glias. The weights between the input layer and the CNS are obtained by Hebbian learning rule. The weights between the CNS and output layer nodes are derived by the ELM algorithm. The experimental results based on standard data sets prove that the SOELM has a better generalization capability and stability than that of conventional ELM. In order to accurately predict the key production of the PTA system in complex chemical processes, our proposed SOELM soft sensor model which is data-driven based method can remarkably improve the intelligent control capabilities for complex process because of its self-organized structure and robust generalization performance.

The organization of this paper is as follows: Section 1 presents the importance of soft sensor in complicated industrial processes and my related works. Section 2 presents the research status of ELM and some associated neural networks. Furthermore, the study status of neuron and glia in biology and neural network, Hebbian learning rule are also presented. The details of ELM method are described in Section 3. The structure and the learning algorithm of our proposed method are provided in Section 4. Comparisons and parameters are discussed by standard data sets in Section 5. Section 6 presents a case study: the prediction of key process variables in PTA chemical data based on SOELM soft sensor model, and compare with ELM method. Finally, the discussion and conclusions are given in Sections 7 and 8, respectively.

2. Literature review

The Extreme Learning Machine (ELM) algorithm of neural network was first proposed by Guang-bin Huang et al. in 2004, in which all the parameters were not to be tuned iteratively. The weights between hidden layer and output layer nodes were obtained by the solving of Moore-Penrose generalized inverse matrix (Huang et al., 2006). So the ELM algorithm has a faster training speed and has been applied in many fields, such as data mining, machine learning, pattern recognition etc. (Wang et al., 2015b; Fossaceca et al., 2015). Recently, many researchers have studied and improved the performance of ELM. Kefeng Ning et al. presented a novel robust ELM method based on a Bayesian framework which replaced the Gaussian distribution with a heavy-tailed distribution (Ning et al., 2015). Men and Wang (2015) proposed a randomized ELM speedup algorithm. In this method, the key matrix could be efficiently approximated by applying the randomized approximation method. Schaik and Tapson (2015) improved the conventional ELM to achieve online learning. The pseudo-inverse was solved by an incremental method, so the improved ELM could be applied in online learning to process large data sets. Although many ELM algorithms were proposed recently to improve the speed or extend applied fields of ELM, the traditional and improved ELMs still cannot adaptively obtain hidden nodes to determine the network structure based on training data sets.

Neurons and glias are two kinds of nervous cells in Central Neural System (CNS). The neurons have always been pivotal roles in a complex nervous system. In the past decades, the glia was known to support and nourish cells. However, some researchers have still thoroughly uncovered more and more mysteries about functions of glias in CNS in recent years. Nearly half of cells in human brain are made up of the glia, in which one kind of glias is astrocyte (Ullian et al., 2001). The neurons transmit electric signals to each other, and then the brain performs many complex activities according to the processing of signals. The synapse of neuron is responsible for receiving the electric signals transmitted by the dendrite of another neuron, so the number of the synapse affects activities of neurons. Although the synapses could be formed by neurons without glias, neurons may require glia-derived cholesterol to form numerous and efficient synapses (Pfrieger, 2002). The enough cholesterol was produced by neurons in the CNS to survive and grow, but the glia provided additional functions for the formation of numerous mature synapses (Mauch et al., 2001). The glia were involved in inducing the formation of new synapses, eliminating existing synapses, controlling the location of synaptic inputs, and stabilizing synaptic structure (Allen and Barres, 2009). Recently, some researchers discovered a mystery that certain ions were used for transmitting signals to glias (Haydon, 2001). The Ca2+, which is one of the most important ions, could change a membrane potential of the neuron (Ikuta et al., 2012b). As a result, the transmissions of the electric signals rely on the neurons and glias.

Glias outnumber neuronal cells by a ratio of 10 to 1. The complexity of a neuronal system is increased with an increasing in the ratio of glia versus neuronal cells (Granderath and Klambt, 1999). Meanwhile, the neuron migration, the axon growth and guidance depended on the neuron-glia interactions (Faissner, 2009). Annalisa Buffo et al. provided an extensive overview of the available literature and some novel insights about the origin and differentiation of variety of the glia (Buffo and Rossi, 2013). In a word, the transmission of electric signals in human brain not only depends on the neuron, but also the glia. Currently, the glia has been widely applied in artificial neural networks. Chihiro Ikuta et al. proposed a chaos glial network which connected to Multi-Layer Perceptron (MLP) for solving the Two-Spiral Problem (TSP), and some improved methods based on these references were proposed (Ikuta et al., 2010, 2011a, 2011b, 2012a, 2012c, 2013). According to the recent achievements in the neuroscience field, Ban Xiao-Juan et al. established a new artificial neuron model called Energy Artificial Neuron (EAN) based on the energy concept from the glia and realized a self-growing and self-organizing neural network based on the EAN model (Ban et al., 2011).

Hebbian learning rule was first proposed by Donald Hebb (Hebb, 2000). It is an unsupervised learning algorithm. It described how neurons excite each other and how this excitation subsequently changes with time based on biological principles (Kuriscak et al., 2015). In the 1970s, researchers began to study artificial neural networks which could simulate biological principles of the human brain (Willshaw and Von, 1976). It was applied in tuning weights between the input layer and the hidden layer nodes in artificial neural networks, which the variation of weights depends on input and output signals of every neuron.

The study of self-organizing structure of network is an important topic in neural network area. Many researchers proposed many effective self-organizing algorithms. Nasr and Chtourou (2011) proposed a novel hybrid algorithm for a feed-forward neural network based on a two stage learning approach. The structure of this network was adjusted in first stage and network parameters were adjusted by a fuzzy neighborhood-based hybrid learning algorithm in second stage. M.H. Ghaseminezhad et al. proposed a novel self-organizing map (SOM) neural network for discrete groups of data clustering because of the lack of solving for clustering discrete groups of data in the classic SOM (Ghaseminezhad and Karami, 2011; Kohonen, 1981). Han and Qiao (2012) proposed a self-organizing radial basis function (SORBF) neural network whose hidden nodes could be grown or pruned based on the node activity (NA) and mutual information (MI) to achieve the appropriate network complexity and maintain overall computational efficiency. Shen Furao et al. presented the incremental network and automated to adjust the number of hidden nodes (Chang et al., 2013; Han et al., 2012; Hsu et al., 2012; Qiao and Han, 2012; Li et al., 2004). Chow and Wu (2004) proposed the cell-splitting grid (CSG) algorithm, and proved that the CSG algorithm outperformed SOM and other SOM-related algorithms in vector quantization while maintaining relatively good topology preservation. In addition, Shen Furao and Osamu Hasegawa proposed an incremental network for online unsupervised classification and topology learning. This incremental network is just for unsupervised classification task, but not applied in solving regression problems (Furao and Hasegawa, 2006; Furao et al., 2007). Bernd Fritzke proposed a self-organizing network to model a given probability distribution. However, the distributions of most problems in real world are not known for us generally (Fritzke,

1992). Blackmore and Miikkulainen proposed an incremental grid growing neural network, which could understand high-dimensional real world data. However, this proposed method was not suitable for solving regression problems (Blackmore and Miikkulainen, 1995). Therefore, this paper proposed a novel SOELM algorithm.

3. Extreme learning machine

Given n samples $X = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}_{n*p}$, and corresponding target outputs $\hat{Y} = \begin{pmatrix} \hat{Y}_1 \\ \vdots \\ \hat{Y}_n \end{pmatrix}_{n*m} = \begin{bmatrix} \hat{y}_{11} & \cdots & \hat{y}_{1m} \\ \vdots & \ddots & \vdots \\ \hat{y}_{n1} & \cdots & \hat{y}_{nm} \end{bmatrix}_{n*m}$, where p is the number of features for each sample and m is the number of outputs for each sample. For n arbitrary distinct samples $(X_i, \hat{Y}_i)$ where $X_i = (x_{i1}, x_{i2}, \ldots\ldots, x_{ip}) \in R^p$ and $\hat{Y}_i = (\hat{y}_{i1}, \hat{y}_{i2}, \ldots\ldots, \hat{y}_{im}) \in R^m$. Setting the hidden nodes to Q, and the standard SLFNs with activation function $\phi(x)$ are mathematically modeled as Eqs. (1) and (2), where $\rho = (\rho_1, \rho_2, \ldots\ldots, \rho_Q)_{p*Q}$ is the weight matrix between the input and hidden layer nodes, and $\rho_i = (\rho_{i1}, \rho_{i2}, \ldots\ldots, \rho_{ip})^T$, $(i=1, 2, \ldots\ldots, Q)$. Meanwhile, $B = (b_f, b_f, \ldots\ldots, b_f)^T$ is the threshold of neurons in hidden layer, $b_f = (bf_1, bf_2, \ldots\ldots, bf_Q)$, $(f = 1,2, \ldots\ldots, m)$. $\zeta = (\zeta_1, \zeta_2, \ldots\ldots, \zeta_Q)^T$ is the weight matrix between the hidden and output layer, and $\zeta_i = (\zeta_{i1}, \zeta_{i2}, \ldots\ldots, \zeta_{im})$, $(i=1, 2, \ldots\ldots, Q)$. Note that the operator '*' is the multiplication of two matrixes.

$$H = \phi(X*\rho+B) = \begin{bmatrix} h_{11} & \cdots & h_{1Q} \\ \vdots & \ddots & \vdots \\ h_{n1} & \cdots & h_{nQ} \end{bmatrix}_{n*Q} \quad (1)$$

$$Y = H*\zeta = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}_{n*m} = \begin{bmatrix} y_{11} & \cdots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{n1} & \cdots & y_{nm} \end{bmatrix}_{n*m} \quad (2)$$

In order to approximate the error of these n samples to zero error as $\sum_{i=1}^{n} \|(\hat{Y}_i - Y_i)\| = 0$. Therefore, the weight matrix between the hidden layer and output layer is shown in Eq. (3).

$$\zeta = H^{+}*\hat{Y} \quad (3)$$

4. The Self-Organizing Extreme Learning Machine Algorithm (SOELM)

4.1. A new structure of SLFNs

Because the distributions of neurons and glias are intricate in the brain (Fields and Beth, 2002), to simplify the design of neural network, we assume that the distribution of neurons or glias in current neural network is an approximative uniform, and the signal of one neuron or glia could only transmit to another neuron or glia, and glias will give energy to neurons. Meanwhile, to more realistically simulate the principle of the CNS, we put the glia into a traditional M-P neural model (Mcculloch, 1943). Assume that neurons and glias in the CNS are restricted into a unit square. The positions of neurons and glias will be generated based on certain rules.

Blood flow is regulated by astrocytes, and astrocytes provide much-needed energy to neurons (Cagla and Ben, 2010). The energy of glias will be reduced after supplying energy, the neighbor radius of neurons will be decreased. The glia will die when its energy is insufficient, and the neuron will die when it does not get enough energy supplying from glias. The entropy of network is calculated after each training process, and the network will add into a neuron if the entropy of network increases. A certain number of glias will be created along with the increasing of neurons. In this paper, we will abstract the utilities of glias in Neurobiology. There is a scope of roundness whose radius is one for every neuron, and if glias are in a range of the neuron, each neuron will be affected by glias. With the learning for samples and the growing of neurons and glias, the network structure tends to be a stable status by self-organized learning process.

Definition 1. the structure of the glia in a CNS is defined as $Glia = \{Pos_g, E_g\}$. Where $Pos_g = \{(x, y), x, y \in (0, 1)\}$ is the position of a glia in a two dimensional space, E_g=1 is the energy of a glia. A glia in the CNS is regarded as a point. The limitation of a glia $E_g^{(limit)}$=0. 5.

Definition 2. the structure of neuron in a CNS is defined as $Neuron = \{Pos_n, E_n, R, \theta, S, O, P\}$, all symbols are defined as follows:
Pos_n: The position of a neuron in the CNS, and $Pos_n = \{(x, y), x, y \in (0, 1)\}$
E_n: The energy of a neuron.
R: The radius of a neuron.
θ: The threshold of a neuron, and $\theta \in (-1, 1)$
S: The status of a neuron, and $neuron = \begin{cases} dead, & S=0 \\ active, & S=1 \end{cases}$
O: The output of a neuron.
P: The probability of a neuron, which shows the entropy a neuron contains.

Moreover, a neuron in the CNS is regarded as a point.

Based on above definitions, a new structure of neural network is proposed as shown in Fig. 1a, and the structure of neurons in CNS is described in Fig. 1b. Every neuron of the hidden layer is encircled by some glias, which is restricted at a unit square, called a CNS.

4.2. Learning Algorithm of SOELM

Before the algorithm is designed, it is necessary to initialize the CNS firstly, and the initialized CNS is defined in the following.

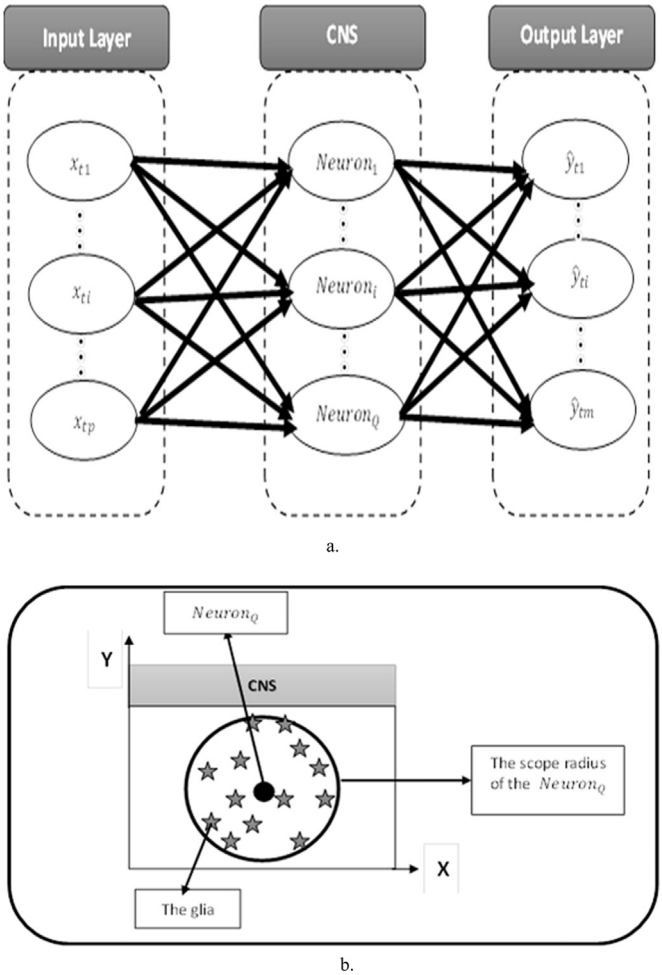

Fig. 1. a. The structure of SOELM network b. The structure of neuron in CNS.

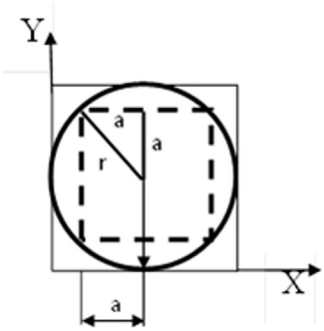

Fig. 2. A diagrammatic sketch of CNS.

Definition 3. A CNS only contains one neuron and *GtoN* ($inordertoreducethetrainingtime$, $wedecidethatGtoN\in[10, 20]andGtoN\in N^{+}$) glias firstly. Meanwhile, we initialize the neuron as follow:

$$Neuron^{(init)} = \left\{ Pos_n = (0.\ 5,\ 0.\ 5), \right.$$

$$\left. E_n = \frac{GtoN}{2},\ R^{(init)} = \frac{1}{\sqrt{\pi}},\ S = 1, P^{(init)} = 1, \quad and\theta isgeneratedrandomly \right\}$$

Where, $R^{(init)}$ is the initialized radius of a neuron, and $P^{(init)}$is the initialized probability of a neuron. Moreover, the positions of neurons are defined as $Pos_n = (0.\ 5,\ 0.\ 5)$. The positions of initialized glias are defined as $Pos_g^{(init)} = \{(x, y), \quad x, y\in(0.\ 2,\ 0.\ 8)\}$. How to define $Pos_g^{(init)}$ is proved as follows:

To guarantee that all initialized glias are located in neighbor range of an initialized neuron, we assume that the initialized CNS is shown in Fig. 2.

The circle is inscribed in a unit square. Because the glia is generated randomly, it may be not located in the square. However, we draw a square whose diameter are no more than the diameter of the circle. The diameter of the initial neuron is defined as $2*\frac{1}{\sqrt{\pi}}\approx 1.\ 128$, which can be obtained as follows:

The probability of each neuron is obtained by Eq. (4), where $S(neu_i)$ is the area of the i-th neuron, and S_{CNS} is the area of the CNS, which are described as Eq. (5) and Eq. (6).

$$P_i = \frac{S(neu_i)}{S_{CNS}} \tag{4}$$

$$S(neu_i) = \pi * R(neu_i)^2 \tag{5}$$

$$S_{CNS} = 1*1 = 1 \tag{6}$$

From the Eqs. (4) and (5) and (6), we can get Eq. (7).

$$P_i = \pi * R_i^2 \tag{7}$$

Because the probability of the initialized neuron in the CNS is one, so the radius of the initialized neuron is described as Eq. (8).

$$R^{(init)} = \sqrt{\frac{P^{(init)}}{\pi}} = \frac{1}{\sqrt{\pi}} \tag{8}$$

Then all glias in the dotted line square are located in the circle. From Fig. 2, we have

$$a^2 + a^2 = r^2$$

$$\Rightarrow \quad a = \frac{\sqrt{2}*r}{2} \approx 0.\ 707*r$$

For the initialized neuron, we know that the radius r of the neuron is about 0.5, so

$$a^{(initneuron)} = \frac{\sqrt{2}}{4} \approx 0.\ 3$$

Therefore, the position of all glias $P_g^{(init)} = \{(x, y), \quad x, y\in(0.\ 2,\ 0.\ 8)\}$.

Meanwhile, we define positions of initialized glias in Eq. (9), where $Pos_g^{(init)}.x$ and $Pos_g^{(init)}.y$ are the horizontal and vertical ordinate, respectively. The k is the serial number of initialized glias from one to $GtoN$.

$$\begin{cases} Pos_g^{(init)}.x = \left(\frac{2}{1+e^{-k}} - 1\right) * 0.\ 6 + 0.\ 2 \\ Pos_g^{(init)}.y = \left(\frac{2}{1+e^{-2*k}} - 1\right) * 0.\ 6 + 0.\ 2 \end{cases} \tag{9}$$

The positions of the non-initialized glias are defined by Eq. (10), where k is the serial number of glias from one to $GtoN$, t is the serial number of training samples. $Pos_g^{(t|k)}$ shows the position of the k-th glia in the t-th iterative procedure. $Rel^{(t)}$ is the relevancy coefficient between the t-th training sample and (t+1)-th training sample, which is obtained by Eq. (11). $MI^{(t)}$ is the Mutual Information (MI) between the t-th training sample and (t+1)-th training sample, where p is the attributes of the t-th training sample $x^{(t)}$, and LEN is the sample size.

$$\begin{cases} \tau = -\frac{-1}{k * Rel^{(t)} * MI^{(t)}} \\ a = 0.\ 707*\frac{1}{\sqrt{\pi}} \\ Pos_g^{(t|k)}.x = Pos_g^{(t|k)}.y = \left(\frac{2}{1+e^{-\tau}} - 1\right) * 2*a + (0.\ 5 - a) \end{cases} \tag{10}$$

$$Rel^{(t)} = \begin{cases} \sqrt{\sum_{i=1}^{p} (x_i^{(t)} - x_i^{(t+1)})^2}, & if \quad (t < LEN) \\ 0, & else \end{cases} \tag{11}$$

In this algorithm, all the neurons and glias are restricted into a unit square. This algorithm is divided into two phases.

4.2.1. The first phase

In this phase, a goal is to determine the number of neurons and tune weights among input nodes and neurons in the CNS.

Given n samples $X = \begin{pmatrix} X_i \\ \vdots \\ X_n \end{pmatrix}_{n*p} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}_{n*p}$, $\hat{Y} = \begin{pmatrix} \hat{Y}_1 \\ \vdots \\ \hat{Y}_n \end{pmatrix}_{n*m} = \begin{bmatrix} \hat{y}_{11} & \cdots & \hat{y}_{1m} \\ \vdots & \ddots & \vdots \\ \hat{y}_{n1} & \cdots & \hat{y}_{nm} \end{bmatrix}_{n*m}$, where p is the number of features for each sample and m is the number of outputs for each sample. Every neuron or glia in the CNS has a 2D position. A glia is in the neighbor range of a neuron to supply its energy to the neuron. The total energy of the k-th neuron is calculated by Eq. (12), where u is the number of glias in the neighbor range of a neuron.

$$E_{nk} = \sum_{i=1}^{u} E_{g_i} \tag{12}$$

The weights among input neurons and neurons in the CNS are tuned iteratively by an improved Hebbian learning rule. The weights will be determined when all samples are input into the network. And there are Q neurons in the CNS.

For n arbitrary distinct samples $\{(X_i, \hat{Y}_i), i = 1, 2, \ldots\ldots, n\}$. The weights $\rho = (\rho_1, \rho_2, \ldots\ldots, \rho_Q)_{p*Q}$ between the input layer and CNS are updated by Eqs. (13) and (14), where $\rho_i = (\rho_{i1}, \rho_{i2}, \ldots\ldots, \rho_{ip})^T$, $(i=1, 2, \ldots\ldots, Q)$, t is the iterative time, γ is the energy adjustment index, β is the Hebbian learning index, E_{ni} is obtained by Eq. (12). $\phi(x)$ is activation function, e.g. the sigmoid function.

$$\Delta\rho_i = \{\beta * \phi(X_t * \rho_i^{(t-1)}) - \theta_i^{(t-1)} + \gamma * \varphi(E_{ni})) * X_t, \quad \beta, \gamma \in (0, 1)\} \tag{13}$$

$$\rho_i^{(t)} = \rho_i^{(t-1)} + (\Delta\rho_i)^T \tag{14}$$

Especially, the initialized weights $\rho_i^{(0)} = \{\varepsilon, \varepsilon > 0\}$, where ε is an arbitrary small positive number. $\varphi(x)$ is defined as Eq. (15).

$$\varphi(x) = \frac{1}{1+e^{-x}} \tag{15}$$

The weights will be limited between −1 and 1 after the training is

over, defined by the Eq. (16), where $MAX = max\{\rho_{ij}\}$, (i=1, 2, ……p *and* j=1, 2, ……Q) . MAX is the maximum among all weights ρ_{ij}.

$$\rho^{(norm)} = \frac{\rho}{MAX} \tag{16}$$

The distance between a glia and a neuron in CNS is defined by Euclidean distance. Glia.Pos_g = (x_g, y_g)and Neuron.Pos_n = (x_n, y_n) represent a glia and neuron position respectively. The Euclidean distance formula between a glia (G) and a neuron (N) is presented by Eq. (17).

$$Eu_{G\to N} = \sqrt{(x_g - x_n)^2 + (y_g - y_n)^2} \tag{17}$$

4.2.2. The second phase

The parameters between the input and the CNS are determined at the first phase, then the number of neurons and glias in CNS are determined as follows.

More energy is consumed when a glia is far away from the neuron in limitation scope of a neuron. The energy of a glia will be reduced according to the Eq. (18), where $\eta^{(t)}\in(0, 1)$ is the damped factor, and it is adjusted by Eq. (19), where n is the number of the samples, and $\mu\in(0, 1)$.

$$E_g^{(t+1)} = E_g^{(t)} - \eta^{(t)} * Eu_{G\to N} \tag{18}$$

$$\eta^{(t+1)} = \begin{cases} \eta^{(t)} * \sqrt{\Delta Entroy^{(post)}}, & if\ \Delta Entroy^{(post)} \le \Delta Entroy^{(pre)} \\ \mu * \Omega(\eta^{(t)} * n) & else \end{cases} \tag{19}$$

Meanwhile, $\Omega(x)$ and $\Delta Entroy^{(post)}$ are defined by Eq. (20) and Eq. (21), where $\Psi^{(pre)}$ and $\Psi^{(post)}$ are obtained by Eq. (22) and Eq. (23), where P_{k+1} is the new probability of a new neuron added into the CNS.

$$\Omega(x) = \frac{1}{1+e^x} \tag{20}$$

$$\Delta Entroy^{(post)} = \Psi^{(post)} - \Psi^{(pre)} \tag{21}$$

$$\Psi^{(pre)} = -\sum_{i=1}^{k} P_i log_2^{P_i} \tag{22}$$

$$\Psi^{(post)} = -\sum_{i=1}^{k} P_i log_2^{P_i} - P_{k+1} log_2^{P_{k+1}} \tag{23}$$

$\Delta Entroy^{(post)}$ will be turned into $\Delta Entroy^{(pre)}$ when η is adjusted. So MI between two different network states can be computed by Eq. (24), where $\Psi^{(t)}$ is obtained by Eq. (23), and the $\Psi^{(t)}$ turns into $\Psi^{(t-1)}$ with inputting next training sample.

$$MI^{(t)} = \Psi^{(t-1)} - \Psi^{(t)},\ (\Psi^{(0)}=0) \tag{24}$$

Meanwhile, the radius of a neuron will be reduced when a neuron absorbs energy from the glias, Eq. (25) shows the principle of a neuron radius reducing process, where $\lambda^{(t)}$ is the damped step, and $\lambda^{(t)}$ must meets Eq. (26).

$$R^{(t+1)} = R^{(t)} - \lambda^{(t)} * R^{(init)} \tag{25}$$

$$\lambda^{(t)} = \frac{1 - e^{-(Rel^{(t)}/gNum)}}{1 + e^{-(Rel^{(t)}/gNum)}} \tag{26}$$

Where $gNum$ is the number of glias in the current neighbor of a neuron, and $\lambda^{(t)}\in[0,1]$, which is proved as follows:

The radius of the neuron will be reduced more slowly when more glias are in range of the neuron. Because including more glias, the neuron obtains more energy. Therefore, there is Eq. (27).

$$\Delta R = (R^{(t)} - R^{(t+1)}) \propto \vartheta(gNum) * R^{(init)} \tag{27}$$

Where $gNum$ is the number of the glias in the range of the neuron, and $\vartheta(gNum)$ will increase follow with the decrease of $gNum$. $\vartheta(x)$ is a strictly decreasing function, and $0 \le \vartheta(x) < 1$.

From Eqs. (8) and (27), we get Eq. (28).

$$R^{(t+1)} = R^{(t)} - \vartheta(x) * R^{(init)} = R^{(t)} - \frac{\vartheta(x)}{\sqrt{\pi}} \tag{28}$$

Then we can derive the Eq. (29) from the Eq. (28).

$$\begin{aligned} (R^{(t+1)})^2 &= \left(R^{(t)} - \frac{\vartheta(x)}{\sqrt{\pi}}\right)^2 = (R^{(t)})^2 + \frac{\vartheta^2(x)}{\pi} - \frac{2*\vartheta(x)*R^{(t)}}{\sqrt{\pi}} \\ &= (R^{(t)})^2 + \frac{\vartheta^2(x)}{\pi} - 2*\vartheta(x)*R^{(t)}*R^{(init)} \\ &\ge (R^{(t)})^2 + \frac{\vartheta^2(x)}{\pi} - 2*\vartheta(x)*R^{(init)}*R^{(init)} \end{aligned} \tag{29}$$

Because there is

$$(R^{(t+1)})^2 < (R^{(t)})^2 \tag{30}$$

Therefore, from the Eqs. (29) and (30), we get Eq. (31).

$$\begin{aligned} &(R^{(t)})^2 + \frac{\vartheta^2(x)}{\pi} - 2*\vartheta(x)*(R^{(init)})^2 < (R^{(t)})^2 \\ &\Rightarrow 2*\vartheta(x)*(R^{(init)})^2 > \frac{\vartheta^2(x)}{\pi} \Rightarrow \vartheta(x) < 2 \end{aligned} \tag{31}$$

Because $0 \le \vartheta(x) < 1$, so we obtain $0 \le \vartheta(x) < 1$.

The growing or dead of neurons and glias obeys the principles of the nature. Therefore, we give a definition about the survival of a neuron or a glia in the CNS.

Definition 4. (1) The glia will die when the energy of glia is lower than a limitation $E_g^{(limit)}$. **(2)** *GtoN* $\in$[10, 20] glias will be created when the CNS grows a neuron. **(3)** The neuron will die when the energy it contains is lower than a limitation E_n. **(4)** A neuron will be created when the MI of network is lower than zero.

The growing of a neuron depends on the MI of network. Given one sample, we can know the MI of network according to Eq. (24) when putting a neuron into this network. A neuron will be put into the CNS when the current entropy $\Psi^{(t)}$ is greater than precious entropy $\Psi^{(t-1)}$, the relationship between $\Psi^{(t-1)}$ and $\Psi^{(t)}$ is denoted as Eq. (32). Therefore, the neurons $N^{(t)}$ will increase.

$$N^{(t)} = \begin{cases} N^{(t-1)} + 1, & if \Psi^{(t)} > \Psi^{(t-1)} \\ N^{(t-1)}, & otherwise \end{cases} \tag{32}$$

Then, the next task is getting parameters between the CNS and output layer by one step according to the ELM algorithm at the second phase. The weights between the CNS and output layer can be obtained by Eq. (3).

Finally, the SOELM algorithm is described as follows:

Step 1: Input sample $\mathfrak{X} = \{(X_i, \hat{Y}_i) X_i \in R^p, \hat{Y}_i \in R^m, i = 1,2, \ldots\ldots, n\}$; activation function $\phi(x)$; parameter μ.

Step 2: Initialize $Neuron^{(init)}$ by Definition 3; initialize $Glia^{(init)}$ by Eq. (9); obtain the distances from glia to neuron by Eq. (17); initialize weights $\rho^{(0)} = \{\varepsilon, \varepsilon > 0\}$, ε is an arbitrary small positive number.

Step 3: Self-organize the structure of network and adjust weights of all links.

for t=1 → n

If $t < n$

Obtain the relevancy coefficient $Rel^{(t)}$ by Eq. (11)

for i=1 → $SizeOfNeu$ ($SizeOfNeu$ is number of neurons)

Find glia $Glia^{(k)}$ which is in the range of $Neuron^{(i)}$

Obtain energy E_{ni} by Eq. (12) and update the glia $Glia^{(k)}$ by Eq. (18)

If $E_{ni} < E_n Neuron^{(i)}$. S=0

Else

Update radius of $Neuron^{(i)}$ by Eq. (25)

If $Neuron^{(i)}.\ R \le 0 Neuron^{(i)}.\ S=0$
for $r=1 \rightarrow SizeOfGlia$ ($SizeOfGlia$ is number of glias)
If $Glia^{(r)}.\ E_g \le E_g^{(limit)}$
Clear up $Glia^{(r)}$
If $Neuron^{(i)}.\ S \neq 0$
Update weights $\rho^{(t)}$ by Eq. (13) and Eq. (14)
for $r=1 \rightarrow SizeOfNeu$
If $Neuron^{(r)}.\ S==0$
Clear up $Neuron^{(r)}$
Obtain current network entropy $\Psi^{(pre)}$ by Eq. (22)
After pre-adding a neuron, obtain network entropy $\Psi^{(post)}$ or $\Psi^{(t)}$ by Eq. (23)
If $t < n$
Obtain $\Delta Entroy^{(post)}$ by Eq. (21)
Adjust η by Eq. (19).
$\Delta Entroy^{(pre)} \leftarrow \Delta Entroy^{(post)}$
Obtain $MI^{(t)}$ by Eq. (24). $\Psi^{(t-1)} \leftarrow \Psi^{(t)}$
If $MI^{(t)}<0$
Add a neuron and $GtoN$ glias and the positions of glias obtained by Eq. (10)
Obtain the distances from glia to neuron by Eq. (17)
Initialize weights $\rho^{(t)} = \{\varepsilon, \varepsilon>0\}$, ε is an arbitrary small positive number
Adjust weights by Eq. (15)

5. Experimental results

ELM algorithm owns a better performance than Back-Propagation (BP), Radial Basis Function (RBF) and Support Vector Machine (SVM) (Huang et al., 2006). In order to test the performance of SOELM algorithm, Benchmark datasets of regression, classification problems from UCI repository (Benchmarks,) will be used to compare the SOELM with the ELM algorithm. In this paper, the accuracy of classification is used as the quantity assessing for all classification tasks, as shown in Eq. (33).

$$Accuracy = \frac{\sum_{i=1}^{n} Num_i^{ture}}{Num^{total}} * 100 \tag{33}$$

Num_i^{ture} is the number of correct classified test vectors for all samples (i=1, 2, ……,n). Num^{total} is the total number of vectors to be tested. Given the inverse normalized network output $NetOut_i^{inver}$ (i=1, 2, ……,n) for all samples, and the expected output $ExpectOut_i^{inver}$, we can obtain the average relative error (ARE). Given the normalized network output $NetOut_i^{norm}$ for all samples, and the expected output $ExpectOut_i^{norm}$, we will obtain the Root Mean Square Error (RMSE) and the standard derivation (Dev.). The computing process is described in the following, where Abs is the absolute value function.

$$\begin{cases} \begin{cases} RE_i = Abs\left(\frac{NetOut_i^{inver} - ExpectOut_i^{inver}}{ExpectOut_i^{inver}}\right), & if\ ExpectOut_i^{inver} \neq 0 \\ RE_i = Abs\,(NetOut_i^{inver} - ExpectOut_i^{inver}), & else \end{cases} \\ ARE = \frac{\sum_{i=1}^{n} RE_i}{n} * 100 \end{cases} \tag{34}$$

$$\begin{cases} MSE_i = (NetOut_i^{norm} - ExpectOut_i^{norm})^2 \\ RMSE_i = \sqrt{MSE_i} \\ RMSE = \sqrt{\frac{\sum_{i=1}^{n} MSE_i}{n}} \end{cases} \tag{35}$$

$$\begin{cases} \overline{RMSE} = \frac{\sum_{i=1}^{n} RMSE_i}{n} \\ Dev. = \sqrt{\frac{\sum_{i=1}^{n} (RMSE_i - \overline{RMSE})^2}{n}} \end{cases} \tag{36}$$

Table 1
Specification of real-world regression cases.

Data sets	#Samples		#Attributes	
	Training	Testing	Inputs	Outputs
Housing	338	168	13	1
Airfoil	1000	503	5	1
Wine	1000	599	11	1
Abalone	2000	2177	7	1
Servo	80	87	4	1

Table 2
Choice of parameter μ for regression data sets.

Order	Linear		Non-Linear T (x)		Housing	
	μ	Q	μ	Q	μ	Q
1	0.25	42	0.26	17	0.22	26
2	0.10	43	0.15	22	0.11	42
3	0.18	44	0.81	14	0.18	39
4	0.06	43	0.19	12	0.43	44
5	0.40	27	0.01	16	**0.57**	**15**
6	0.95	28	0.05	16	0.08	35
7	0.67	21	**0.92**	**11**	0.12	15
8	0.63	26	0.08	13	0.74	13
9	**0.65**	**17**	0.23	16	0.34	34
10	0.09	50	0.10	22	0.56	14

Order	Airfoil		Wine	
	μ	Q	μ	Q
1	0.11	18	0.35	6
2	0.27	17	**0.43**	**4**
3	0.09	15	0.73	4
4	0.13	16	0.69	8
5	0.23	14	0.62	4
6	0.75	7	0.39	6
7	0.20	10	0.70	7
8	0.22	9	0.28	5
9	**0.77**	**5**	0.54	5
10	0.55	6	0.14	6

Order	Non-Linear S (x)		Servo		Abalone	
	μ	Q	μ	Q	μ	Q
1	0.13	17	0.30	18	0.02	14
2	0.16	16	0.71	17	0.07	15
3	**0.10**	**14**	0.13	14	0.42	3
4	0.56	17	0.16	16	**0.39**	**2**
5	0.36	17	0.06	30	0.10	3
6	0.30	19	**0.81**	**13**	0.29	2
7	0.99	19	0.83	13	0.14	2
8	0.24	19	0.57	13	0.09	2
9	0.40	18	0.98	21	0.33	5
10	0.52	18	0.12	15	0.03	2

NOTE: the Q expresses nodes in the CNS.

We change one parameter and fix other hyper parameters, and perform some experiments. Then we know that parameters β and γ should be set to 0.4, and $GtoN$ should be set to 10, and the initialized energy damped factor $\eta^{(0)}$ should be set to 0.4.

5.1. Testing with regression problems

In order to prove the effectiveness of the SOELM algorithm, three artificial cases (a linear function and two non-linear functions) and five datasets derived from real-world will be used to validate the ability for processing the regression problems. The specification of data sets are listed in Table 1. To choose the parameter μ, we perform 100 times

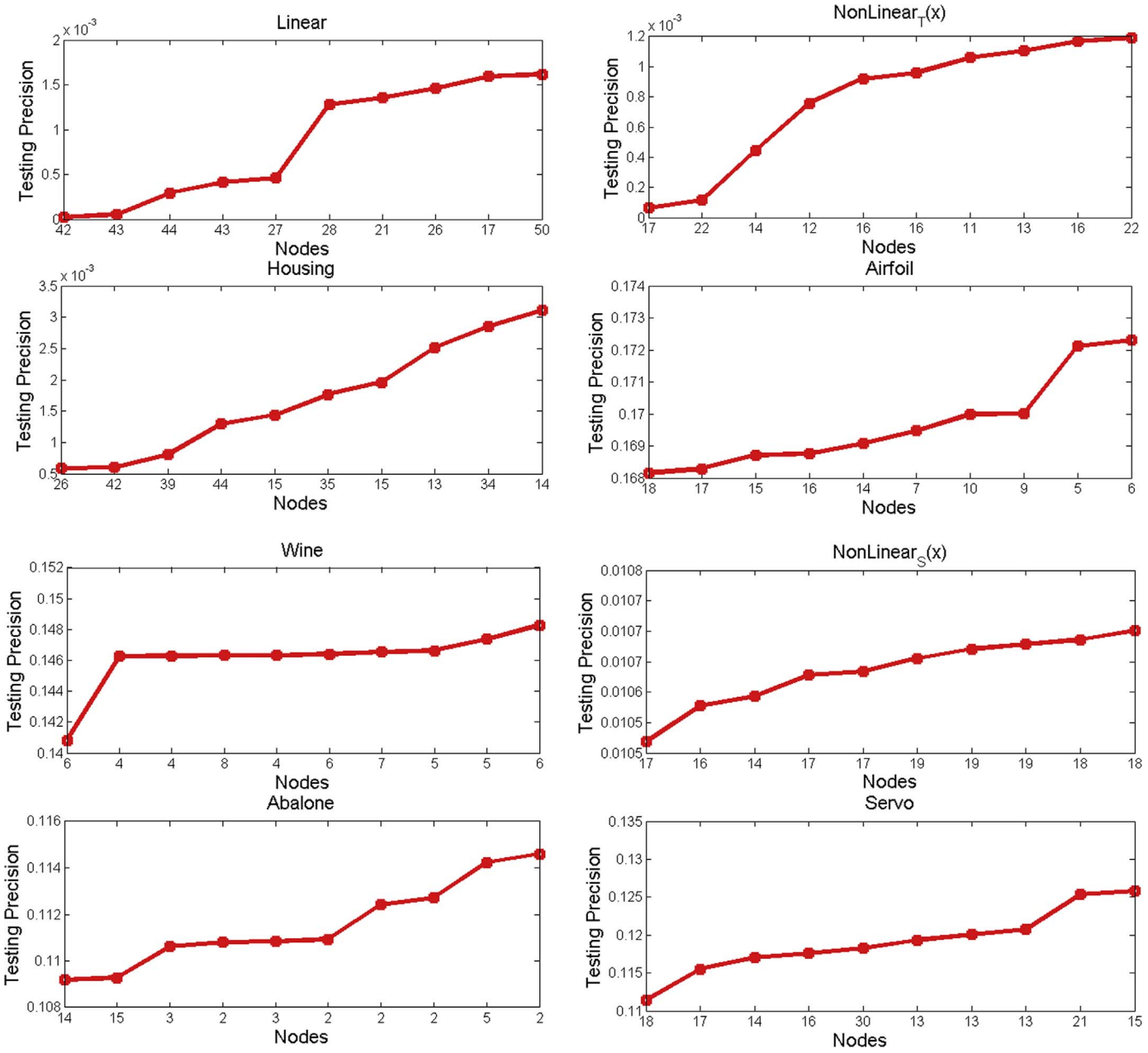

Fig. 3. The relationship between testing precisions and nodes for regression data sets.

Table 3
Comparison of training and testing RMSE of ELM and SOELM.

Function	Node	ELM (%)		SOELM (%)		
		Training	Testing	Training	Testing	μ
Linear	**17**	7.865	8.208	0.091	0.102	0.65
	18	5.681	6.114	N	N	N
	20	4.617	5.069	N	N	N
Non-Linear T(x)	**11**	1.973	2.073	0.097	0.099	0.92
	15	0.372	0.380	N	N	N
	19	0.149	0.157	N	N	N
Non-Linear S(x)	**14**	11.53	1.062	11.31	0.989	0.10
	20	11.45	1.048	N	N	N
	25	11.41	1.122	N	N	N

NOTE: N expresses None.

experiments repeatedly for each regression data set. Meanwhile, the parameter μ is changed from 0.01 to 1 with step 0.01 (Sun, 2012). Finally we choose the first ten best results, which are presented at Table 2. The results in Table 2 are ascending sorted by the testing precision. The relationship between the testing precision and nodes for each regression data set is presented at Fig. 3.

For regression problems, the testing precision is set to 2×10^{-3}. In order to obtain the simplest network structure, we choose the smallest nodes when the testing precision meets the required testing precision. For Linear, NonLinear T(x) and Housing regression data sets, testing precisions are set to1.59×10^{-3}, 1.06×10^{-3} and 1.43×10^{-3}respectively when owning the smallest nodes according to the Table 2 and Fig. 3. For other regression data sets, because all testing precisions are not up to the required testing precision, their testing precisions are very approximately, we choose the smallest node as a final result. Therefore, the choices of all regression data sets are shown in boldface in Table 2.

Then we start training the ELM and SOELM network using the artificial cases and real-world data sets. And the first nodes of each data

Table 4
Comparison of training and testing RMSE and ARE of ELM and SOELM.

D. S	Node	ELM			SOELM			
		RMSE (%)		ARE (%)	RMSE (%)		ARE (%)	μ
		Test	Dev.	Test	Test	Dev.	Test	
Housing	**15**	2.07	0.016	16.36	0.14	0.001	0.65	0.57
	22	1.58	0.090	12.22	N	N	N	N
	26	1.50	0.007	10.76	N	N	N	N
Airfoil	**5**	22.1	0.018	4.97	17.2	0.002	4.05	0.77
	14	18.5	0.034	4.23	N	N	N	N
	21	17.0	0.029	4.00	N	N	N	N
Abalone	**2**	10.7	0.008	20.72	9.73	0.001	19.40	0.39
	25	9.87	0.006	19.52	N	N	N	N
	30	9.79	0.012	19.42	N	N	N	N
Servo	**13**	13.6	0.008	79.27	11.7	0.001	20.55	0.81
	30	12.8	0.011	60.54	N	N	N	N
	35	13.1	0.013	77.02	N	N	N	N
Wine	**4**	0.82	0.006	12.53	0.73	0.002	10.45	0.43
	14	0.84	0.004	12.94	N	N	N	N
	18	0.85	0.005	13.11	N	N	N	N

NOTE: N expresses None.

set are obtained by the SOELM, which are shown in boldface in Tables 3 and 4 and 7.

5.1.1. Approximation of three artificial cases

(1) Approximation of Linear Function

$$f(x) = 0.2*x1 + 0.8*x2 - 1.1*x3 + 0.9*x4 - 0.5*x5$$

(2) Approximation of Non-Linear Function

a) $T(x) = tanh(x)$

b) $S(x) = \begin{cases} sin(x)/x, x \neq 0 \\ 1, x = 0 \end{cases}$

Firstly, we use a linear function, a non-linear function T(x) (500 Training Samples and 500 Testing Samples) and a non-linear function S(x) (Huang et al., 2006) (2500 Training Samples and 2500 Testing Samples) to validate the abilities of the recall and generalization of SOELM algorithm. In order to more convincingly illustrate the performance of SOELM, we randomly choose more hidden nodes in ELM. The experimental results are shown in Table 3.

It can be seen from the experiments that both ELM and SOELM algorithm are good at simulating the linear and non-linear function. However the approximate capacity of SOELM is much better than that of ELM. For these three functions, the training and testing RMSEs are improved with the growth of hidden nodes, at the same time, the changing rate of generalization performance decreases (Huang et al., 2006). Therefore, our proposed model can find more proper structure adaptively with less number of nodes than ELM.

5.1.2. Real-world regression problems

Five data sets are chosen to validate the performance of SOELM in Table 1. The comparison results between ELM and SOELM algorithm are shown in Table 4. It can be seen from Table 4 that the generalization ability of SOELM is better than that of ELM algorithm.

According to above results from Tables 3 and 4, the testing RMSEs of SOELM are generally smaller than that of ELM algorithm when both SOELM and ELM use the same number of hidden nodes. Furthermore, the modeling precision of SOELM is much better than that of ELM, at the same time ELM has more hidden nodes. It takes Housing data set as an example, the testing RMSE is 2.07% for ELM with choosing 15 hidden nodes, but the testing RMSE of SOELM is 0.14% with the same nodes. However, ELM algorithm will up to the precision of SOELM with growing more nodes. It shows that SOELM can not only determine hidden nodes self-organizing but also obtain more abbreviated network structure. Furthermore, we compare our proposed SOELM method with recent variants of the ELM algorithm (Wang et al., 2015a), which are presented at Fig. 4.

As seen from Fig. 4, we know that the RMSE of some variants of ELM algorithm is higher than our proposed method, such as ELM-RBF, ELM-Bernstein and ELM-Sigmoid, which RMSE are 14.69, 13.68 and 10.7 for Abalone data set, and are 13.78, 12.11 and 13.6 for Servo data set, but our proposed method is 9.73 and 11.7, respectively. Meanwhile, SOELM can achieve faster convergence rate than that of ELM algorithm which is shown in Fig. 5. The average testing error of SOELM is much smaller than that of ELM when training 15 times, which shows that SOELM can obtain much better convergence than that of ELM. Next, the stability of SOELM algorithm is validated using above five datasets. We repeat experiments for 15 times with different hidden nodes for ELM. Namely, hidden nodes are 5, 14 and 21 for Airfoil, 15, 22 and 26 for Housing, 4, 14 and 18 for Wine, 2, 25 and 30 for Abalone, 13, 30 and 35 for Servo. For SOELM, there are 5 neurons in the CNS for Airfoil, 15 neurons for Housing, 4 neurons for Wine, 2 neurons for Abalone, 13 neurons for Servo. Moreover, we get the testing error curves of SOELM and ELM algorithm under different hidden neurons in Figs. 6–10. Seen from Fig. 6–10, we can easily know that the error curves obtained by SOELM are much smoother than that of ELM. It shows that the testing error of SOELM is more stable than that of ELM. Therefore, we prove that the stability of SOELM is much better than that of ELM algorithm.

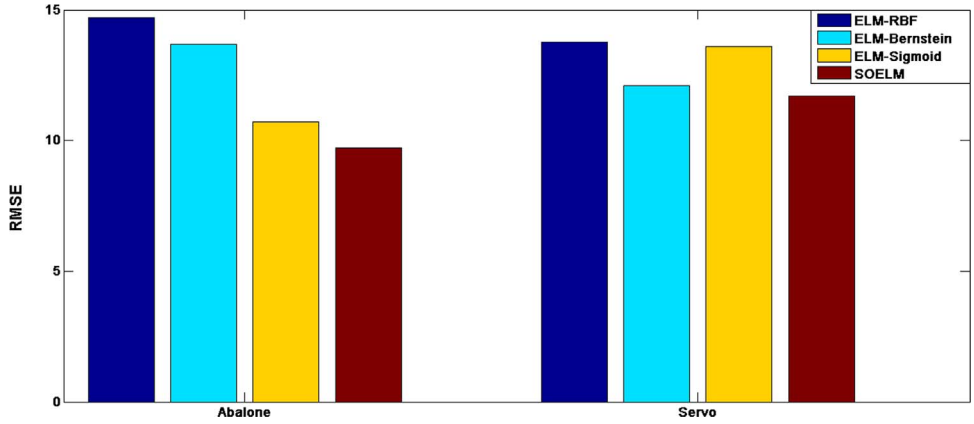

Fig. 4. The comparison of RMSE among SOELM, ELM-RBF, ELM-Bernstein and ELM-Sigmoid.

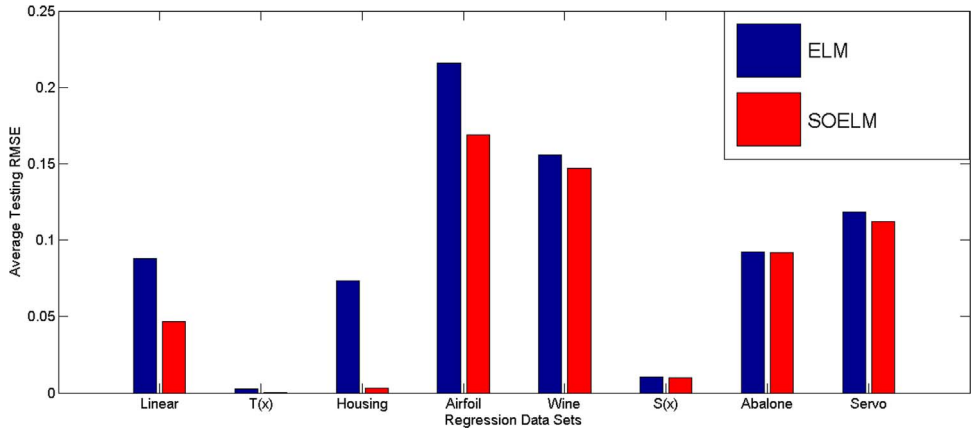

Fig. 5. The comparison of average testing error between SOELM and ELM.

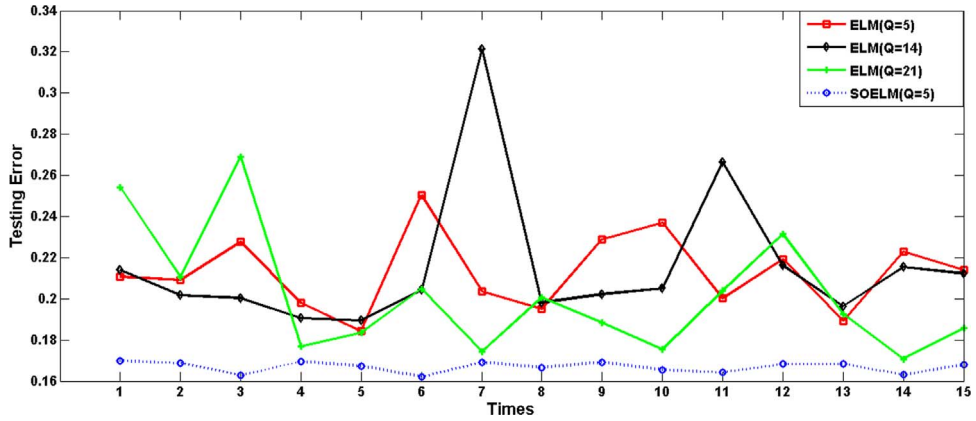

Fig. 6. The comparison of stability between SOELM and ELM (Airfoil).

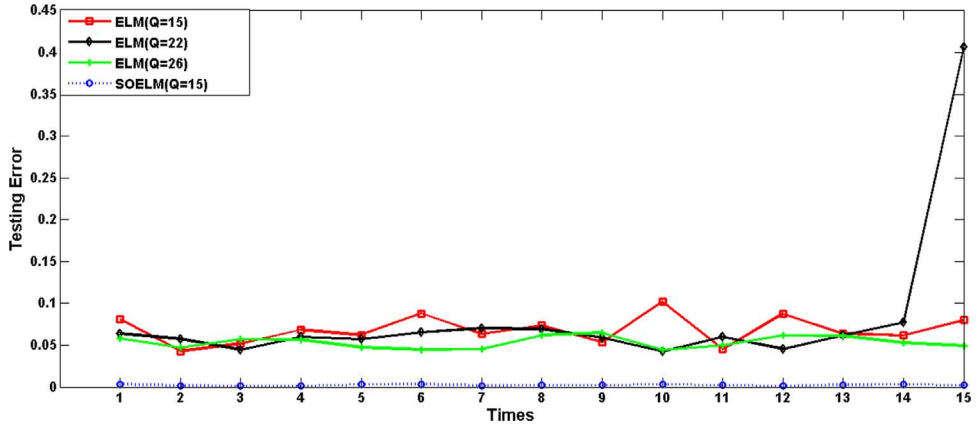

Fig. 7. The comparison of stability between SOELM and ELM (Housing).

5.2. Testing with classification problems

The experimental classification data sets are listed in Table 5. To choose the parameter μ, we perform 100 times experiments repeatedly for each classification data set. The parameter $\mu$is changed from 0.01 to 1 with step 0.01 (Sun, 2012). Finally we choose the first ten best results presented in Table 6. The results are descending sorted by testing accuracy. The relationship between the testing accuracy and nodes for each classification data set is presented at Fig. 11.

For classification problems, we set the required testing accuracy threshold as 90%. In order to obtain the simplest network structure, we choose the smallest nodes when the testing accuracy satisfies the setting threshold. For Ionosphere, Zoo and Iris, the testing accuracy are all more than 90%, so we select the smallest nodes, which are 10, 14 and 8 respectively. For CMC, because all testing accuracy are not up to the threshold, from Nodes 15 to 7, their testing errors are very approximately which are 49.04%, 47.78% and 47.56%, and Nodes 7 is the smallest nodes, we choose the smallest nodes 7 for CMC. Therefore, the nodes of each classification data set are chosen with boldface in Table 7.

In order to more convincingly illustrating the performance of SOELM, we randomly choose more hidden nodes for ELM to compare

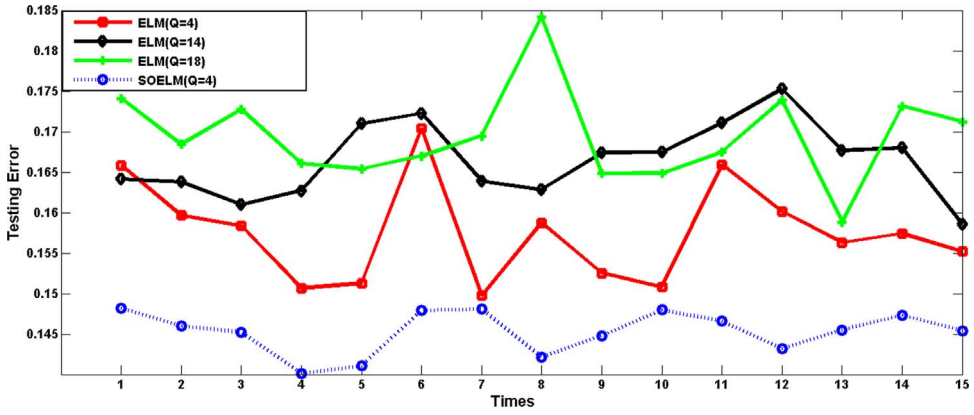

Fig. 8. The comparison of stability between SOELM and ELM (Wine).

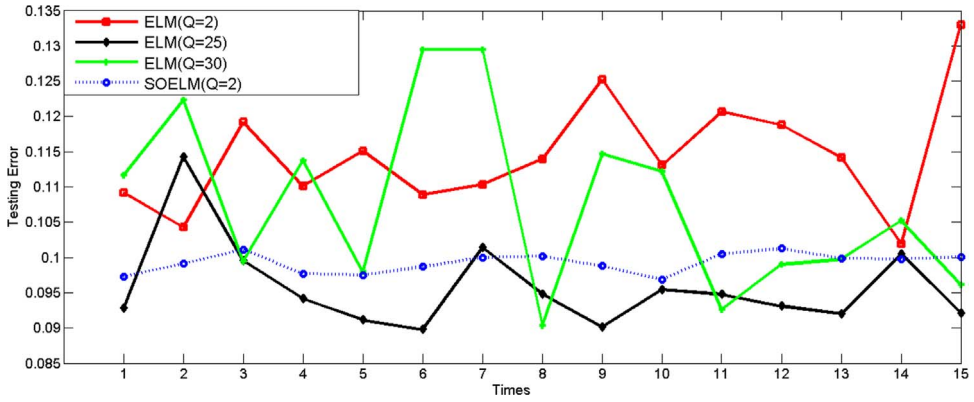

Fig. 9. The comparison of stability between SOELM and ELM (Abalone).

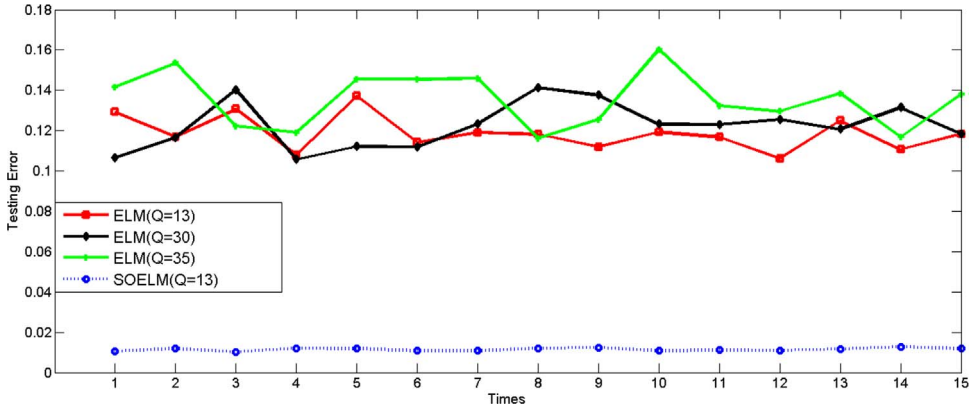

Fig. 10. The comparison of stability between SOELM and ELM (Servo).

Table 5
Specification of real-world classification cases.

Data sets	#Samples		#Attributes	
	Training	Testing	Inputs	Classes
Ionosphere	224	127	34	2
Zoo	68	33	16	7
CMC	1000	473	9	3
Iris	147	6	4	3

Table 6
Choice of parameter μ for classification data sets.

Order	Ionosphere		CMC		Zoo		Iris	
	μ	Q	μ	Q	μ	Q	μ	Q
1	**0.82**	**10**	0.16	15	**0.52**	**14**	**0.02**	**8**
2	0.15	27	0.04	14	0.63	14	0.04	8
3	0.06	37	**0.25**	**7**	0.65	14	0.05	8
4	0.30	21	0.37	12	0.23	17	0.07	40
5	0.51	15	0.26	13	0.26	14	0.09	12
6	0.01	77	0.40	15	0.27	14	0.12	15
7	0.18	21	0.51	10	0.35	15	0.13	24
8	0.26	21	0.58	12	0.36	17	0.14	25
9	0.60	23	0.61	13	0.40	14	0.16	13
10	0.70	12	0.86	10	0.42	18	0.19	18

NOTE: the Q expresses nodes in the CNS.

with SOELM. The testing results are shown in Table 7.

It can be seen from Table 7, the testing RMSEs of SOELM are generally smaller than that of ELM algorithm when both SOELM and ELM have the same number of hidden nodes. Furthermore, the classification precisions of different data sets obtained by SOELM are

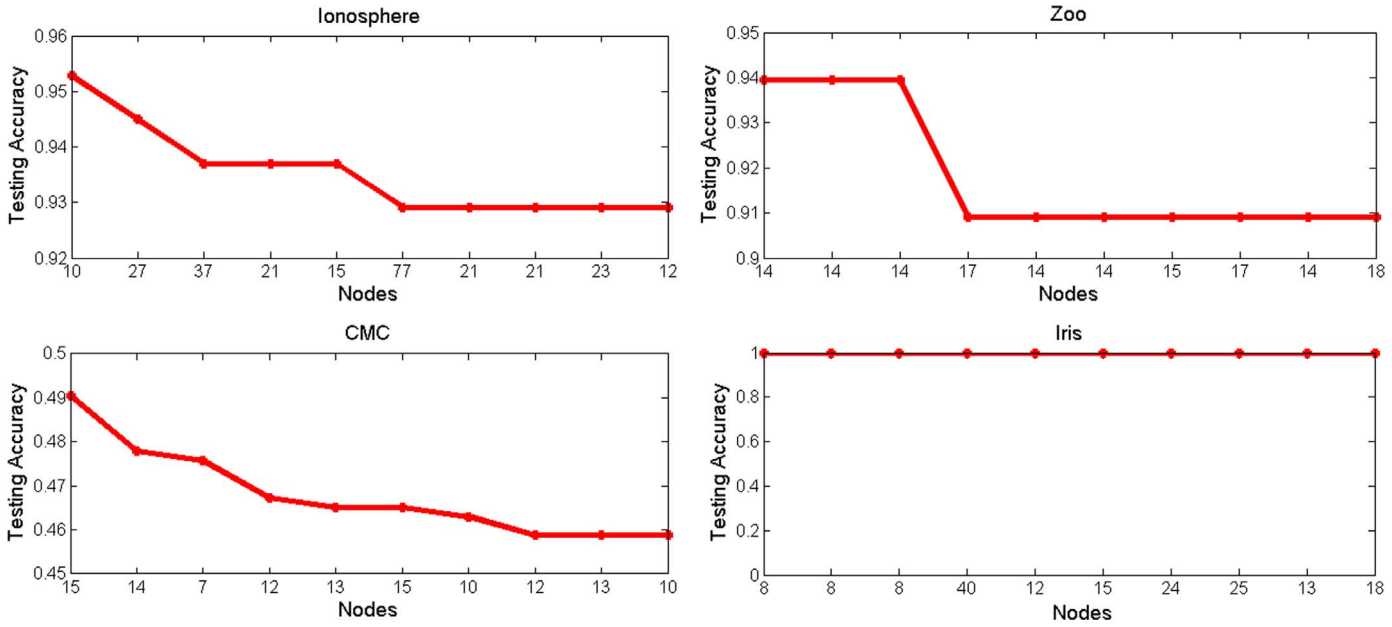

Fig. 11. The relationship between testing accuracy and nodes for classification data set.

Table 7
Comparison of training and testing accuracy of ELM and SOELM.

Data Sets	Nodes	ELM		SOELM		
		Testing	Dev.	Testing	Dev.	μ
Ionosphere	**10**	85%	0.082	93%	0.021	0.82
	17	90%	0.079	N	N	N
	18	91%	0.028	N	N	N
Zoo	**14**	78%	0.042	93%	0.025	0.52
	17	72%	0.028	N	N	N
	20	79%	0.033	N	N	N
CMC	**7**	45%	0.020	48%	0.020	0.25
	17	46%	0.020	N	N	N
	20	48%	0.020	N	N	N
Iris	**8**	83%	0.073	100%	0.001	0.02
	14	100%	0.073	N	N	N
	15	100%	0.001	N	N	N

NOTE: N expresses None.

much better than that of ELM. However, ELM needs more hidden nodes to get the same precision. Taken Zoo data sets as an example, the hidden nodes of ELM are set as 14, 17 or 20 respectively, the testing classification successful rates are 78%, 72% and 79% respectively. But SOELM can decide hidden nodes automatically, and the testing classification successful rate is 93%. At the same time, SOELM is more practical than ELM algorithm because of the self-organization feature in network structure.

6. Case study: prediction of key process variables in PTA solvent systems

PTA is one of the important raw materials for the production of polyester chemical companies. Solvent system can be divided into three parts: the solvent dehydration tower, the re-boiler and the reflux tank. We focus on the analysis of the solvent dehydration tower. And the schematic flow diagram of a solvent dehydration tower is shown in Fig. 12.

An important indicator to measure the advancement and effectiveness of PTA technology is the consumption of acetic acid. And the

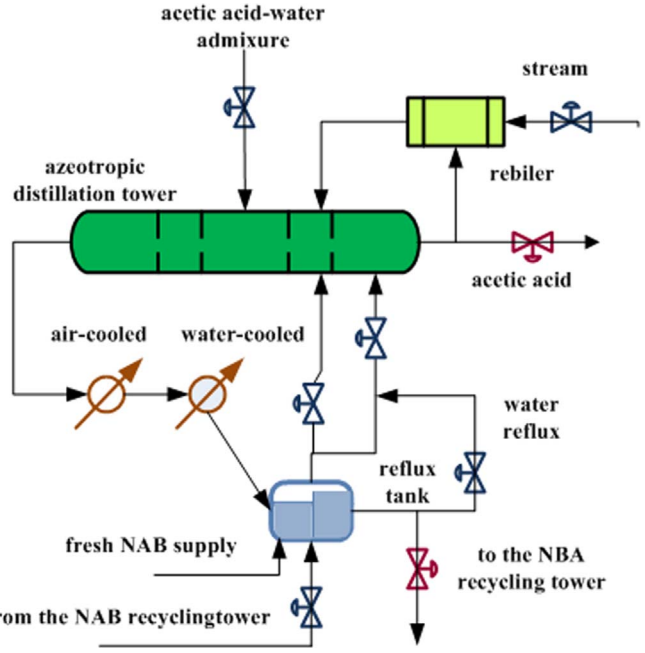

Fig. 12. Schematic flow diagram of a solvent dehydration tower.

Table 8
Specification of PTA data.

#Samples		#Attributes	
Training	Testing	Inputs	Output
174	86	17	1

optimal control of the solvent system is the most important way to reduce the consumption of acetic acid. The development of PTA technology depends on the stability of produce plant, which is an important precondition of the operation of plant. Therefore, a stable PTA produce prediction model is extremely important to reduce materials consumption and energy consumption.

Table 9
Performance comparison between SOELM and ELM.

ELM				SOELM			
ARE (%)		RMSE		ARE (%)		RMSE	
Training	Testing	Training	Testing	Training	Testing	Training	Testing
0.799	0.817	0.459	0.468	0.742	0.759	0.435	0.454

In order to validate the effectiveness of our proposed SOELM method, this paper choose PTA solvent system data, whose input variables consisted of the water reflux, the feed composition (acetic acid content), the feed quantity, the NBA main reflux, the NBA side reflux, the temperature point between the 44th tray and the 50th tray, the tray temperature near the sensitive plate, the steam flow, the reflux temperature, the temperature of the top tower, the temperature point above the 35th tray, the temperature point between the 35th tray and the 40th tray, the tray temperature near the sensitive plate, the produced quantity of the top tower, the feed temperature, the controllable temperature point between the 53rd tray and the 58th tray and the reflux tank level, and whose output variable is the conductivity of the top tower. And the specification of PTA data is presented at Table 8.

Then we compare SOELM with ELM method using PTA data. Firstly, we choose parameters as same as aforementioned strategies. We set hidden nodes as 12 via self-organizing and we set μ as 0.39. Other parameters are the same as above parameters. Then we obtain ARE and RMSE shown in Table 9.

As seen from Table 9, we know that the performance of SOELM is superior to ELM in chemical data. And SOELM performs 0.057%, 0.058% better than ELM for training and testing ARE respectively, 0.024, 0.014 better than ELM for training and testing RMSE. Furthermore, the outputs are presented at Fig. 13. From aforementioned results, our proposed SOELM model is more stable than ELM model. Therefore, SOELM is easy to analyze the electric conductivity of acetic acid in energy efficiency analysis and more suitable to guide chemical production than ELM. Furthermore, production activities have a minimum impact on the production intensity, and the intensity of production is at the peak in the 11th sample of PTA, whose actual electric conductivity of acetic acid is about 47.55, and prediction of ELM and SOELM are 48.78, 48.45 respectively. The production activities have a largest impact on the production intensity, the intensity of production is minimum, and reaches the highest acetic acid conductivity in the 85th sample, whose actual acetic acid conductivity is about 49.99, and prediction of ELM and SOLEM are 48.89, 49.53 respectively. Therefore, the performance of SOELM is better than that of ELM. And the SOELM soft sensor model could accurately predict the key process variable in PTA solvent system. Therefore, the precise SOELM soft sensor could guide PTA production in the complicated chemical processes.

7. Discussion

First, the paper proposes a novel SOELM algorithm based on the CNS to solve the issue of the ELM algorithm. The weights between input layer nodes and the CNS are tuned iteratively by the Hebbian learning rule. And the network structure is adjusted self-organizing by Mutual Information (MI) among different structures of networks. Moreover, the weights between the CNS and output layer nodes are obtained by the ELM algorithm. The SOELM algorithm can overcome shortcomings of the traditional ELM algorithm and provide better generalization performance than that of ELM.

Second, the robustness and effectiveness of the SOELM model is validated through the standard data source from the UCI repository. Meanwhile, compared with the ELM from the standard testing and PTA production data, the objectivity and robustness of the SOELM model have been proved. However, the important hyper parameter μ need been determined by performing experiments more times. Finally, we would consume much time to obtain the hyper parameter μ, which affect the online measure. Therefore, how to obtain the parameter μ automatically according to different data sets is our further work.

Third, the model is applied to predict the key process variable in PTA solvent systems effectively, and the parameters of model are adjusted by experiments. Therefore, we will improve our model that parameters can be adjusted automatically, such as designing a parameters self-adjusting SOELM soft sensor model, which is more suitable to the real world applications. Furthermore, we will explore a heuristic algorithm to find an optimal hyper parameter in future. Moreover, we will try to build a chaotic CNS and use the transfer entropy instead of the weight.

8. Conclusion

This paper studies a new structure of SOELM with which glias play a great role in transmitting electric signals among neurons. A novel learning algorithm is proposed to determine the hidden nodes by self-organized training. The weights between the input layer and CNS are obtained by the Hebbian learning rule, and weights between CNS and output layer are obtained by the ELM algorithm. The parameters of network are adjusted using the energy provided to neurons by glias, and neurons in CNS are self-organizing by comparing the mutual information under two different network status. Then we build a new soft sensor model base on SOELM algorithm. The effectiveness and

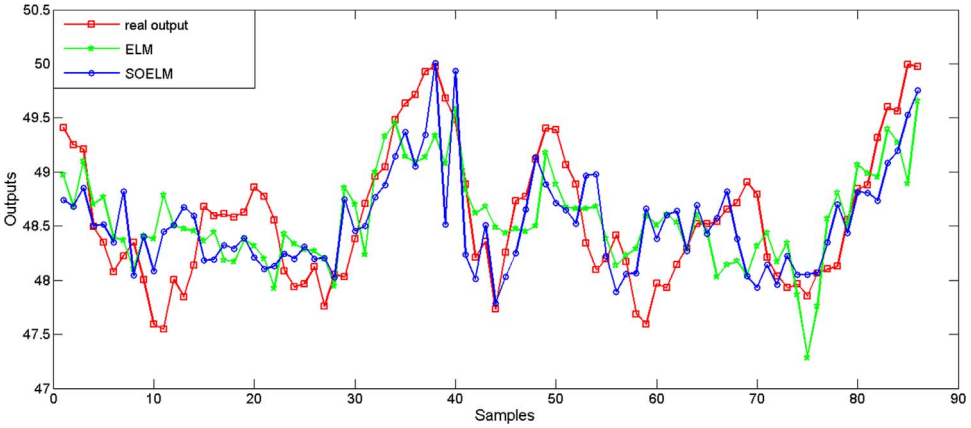

Fig. 13. The predict results of ELM and SOELM for PTA data.

practicality of the SOELM algorithm is validated by some UCI data sets and PTA chemical data comparing with that of ELM algorithm.

In our future studies, we will improve the selecting method of SOELM parameters self-organizing, and explore a heuristic algorithm to find an optimal hyper parameter. Furthermore, we will compare our proposed method with more recent variants of ELM, and we also use more complex data to validate our method. Moreover, we will try to build a chaotic CNS and use transfer entropy instead of weight. Furthermore, we will improve the prune of glias in CNS and apply the soft sensor in other complicated chemical processes.

Acknowledgment

This research was partly funded by National Natural Science Foundation of China (61374166 and 61673046), Natural Science Foundation of Beijing (4162045).

References

Allen, N.J., Barres, B.A., 2009. Glia and synapse formation: an overview. Ref. Modul. Biomed. Sci. Encycl. Neurosci., 731–736.
Ban, X.J., Liu, H., Xu, Z.R., 2011. An energy artificial neuron model based self-growing and self-organizing neural network. Acta Autom. Sin..
Benchmarks. Available Online: ⟨http://archive.ics.uci.edu/ml/datasets⟩.
Blackmore, J., Miikkulainen, R., 1995. Visualizing high-dimensional structure with the incremental grid growing neural Network. In: Machine Learning Proceedings, pp. 55–63.
Buffo, A., Rossi, F., 2013. Origin, lineage and function of cerebellar glia. Progress. Neurobiol. 109, 42–63.
Cagla, E., Ben, A.B., 2010. Regulation of synaptic connectivity by glia. Nature 468 (7321), 223–231.
Chakchai, S.I., Songyut, P., Kanokmon, R., 2016. Soft computing-based localizations in wireless sensor networks. Pervasive Mob. Comput. 29, 17–37.
Chang, F.J., Tsai, W.P., Chen, H.K., Yam, R.S.W., Herricks, E.E., 2013. A self-organizing radial basis network for estimating riverine fish diversity. J. Hydrol. 476, 280–289.
Chow, T.W.S., Wu, S., 2004. Cell-splitting grid: a self-creating and self-organizing neural network. Neurocomputing 57, 373–387.
Edwin, D., Peter, M., 1998. The automation of a reactor flow system with online GC analysis. J. Anal. Appl. Pyrolysis 44 (2), 167–179.
Faissner, A., 2009. Axon guidance by glia. Ref. Modul. Biomed. Sci. Encycl. Neurosci., 1063–1072.
Fields, R.D., Beth, S.G., 2002. New insights into neuron-glia communication. Science, 556–562.
Fossaceca, J.M., Mazzuchi, T.A., Sarkani, S., 2015. MARK-ELM: application of a novel Multiple Kernel Learning framework for improving the robustness of network intrusion detection. Expert Syst. Appl. 42 (8), 4062–4080.
Fritzke, B., 1992. Growing cell structures – a self-organizing network in k dimensions. Artif. Neural Netw., 1051–1056.
Furao, S., Hasegawa, O., 2006. An incremental network for on-line unsupervised classification and topology learning. Neural Netw. 19 (1), 90–106.
Furao, S., Ogura, T., Hasegawa, O., 2007. An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Netw. 20 (8), 893–903.
Ge, Z.Q., Song, Z.H., 2010. A comparative study of just-in-time-learning based methods for online soft sensor modeling. Chemom. Intell. Lab. Syst. 104 (2), 306–317.
Ghaseminezhad, M.H., Karami, A., 2011. A novel self-organizing map (SOM) neural network for discrete groups of data clustering. Appl. Soft Comput. 11 (4), 3771–3778.
Granderath, S., Klambt, C., 1999. Glia development in the embryonic cns of drosophila. Curr. Opin. Neurobiol. 9 (6), 531–536, (6).
Han, H.G., Qiao, J.F., 2012. Prediction of activated sludge bulking based on a self-organizing RBF neural network. J. Process Control 22 (6), 1103–1112.
Han, H.G., Qiao, J.F., Chen, Q.L., 2012. Model predictive control of dissolved oxygen concentration based on a self-organizing RBF neural network. Control Eng. Pract. 20 (4), 465–476.
Haydon, P.G., 2001. Glia: listening and talking to the synapse. Nat. Rev. Neurosci. 2, 844–847.
Hebb, D.O., 1949. "The Organization of Behavior," of the CNS, Coding of Sensory Information. In: Principles of Neural Science, 2000, McGraw-Hill Companies, vol. 9(3), pp. 213–218.
Hong, B.S., Fan, L.T., John. S, R.S., 1998. Monitoring the process of curing of epoxy/graphite fiber composites with a recurrent neural network as a soft sensor. Eng. Appl. Artif. Intell. 11 (2), 293–306.
Hsu, C.F., Chiu, C.J., Tsai, J.Z., 2012. Indirect adaptive self-organizing RBF neural controller design with a dynamical training approach. Expert Syst. Appl. 39 (1), 564–573.
Huang, G.B., Zhu, Q.Y., Siew, C.K., 2006. Extreme learning machine: theory and applications. Neurocomputing 70 (1–3), 489–501.
Ikuta, C., Uwate, Y., Nishio, Y., 2010. Chaos Glial Network Connected to multi-layer perceptron for solving two-spiral problem. In: Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS).
Ikuta, C., Uwate, Y., Nishio, Y., 2011a. Performance and features of multi-layer perceptron with impulse glial network. In: Proceedings of the 2011 International Joint Conference on IEEE, Neural Networks (IJCNN), vol. 42, pp. 2536–2541.
Ikuta, C., Uwate, Y., Nishio, Y., 2011. Multi-layer perceptron with glial network for solving two-spiral problem. Ieice Trans. Fundam. Electron. Commun. Comput. Sci. 94 (9), 1864–1867.
Ikuta, C., Uwate, Y., Nishio, Y., 2012a. Multi-Layer Perceptron decided learning neurons by regular output glias. In: Proceedings of the Nolta.
Ikuta, C., Uwate, Y., Nishio, Y., 2012b. Investigation of Multi-layer perceptron with propagation of glial pulse to two directions. In: Proceedings of the 2012 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2099–2102.
Ikuta, C., Uwate, Y., Nishio, Y., 2012c. Improvement of Learning performance of multi-layer perceptron by two different pulse glial networks. In: Proceedings of the IEEE Asia Pacific Conference on Circuits & Systems.
Ikuta, C., Uwate, Y., Nishio, Y., Yang, G., 2013. Multi-Layer Perceptron including glial pulse and switching between learning and non-learning. In: Proceedings of the 2013 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2107–2110.
Kohonen, T., 1981. Self-organized formation of topologically correct feature maps. Biol. Cybern. 43 (1), 59–69.
Kuriscak, E., Marsalek, P., Stroffek, J., Toth, P.G., 2015. Biological context of Hebb learning in artificial neural networks, a review. Neurocomputing 152, 27–35.
Li, P., Farkas, I., Whinney, B.M., 2004. Early lexical development in a self-organizing neural network. Neural Netw. 17 (8–9), 1345–1362.
Li, Y.L., Sun, X.T., 2013. Development status and trend of China PTA industry. China Pet. Chem. Ind. Anal. 8, 46–49.
Liu, Y., Huang, D., Li, Z., 2013. A SEVA soft sensor method based on self-calibration model and uncertainty description algorithm. Chemom. Intell. Lab. Syst. 126, 38–49.
Mauch, D.H., Nägler, K., Schumacher, S., Göritz, C., Müller, E.C., Otto, A., Pfrieger, F.W., 2001. CNS synaptogenesis promoted by glia-derived cholesterol. Science, 1354–1357.
Mcculloch, W.S., 1943. A logical calculus of ideas immanent in nervous activity. Bull. Math. Biophys. 52 (4), 99–115.
Men, C.Q., Wang, W.J., 2015. A randomized ELM speedup algorithm. Neurocomputing 159, 78–83.
Mortaza, A., Shahaboddin, S., Meisam, T., Por, L.Y., Yaser, N.L., 2016. The use of ELM-WT (extreme learning machine with wavelet transform algorithm) to predict exergetic performance of a DI diesel engine running on diesel/biodiesel blends containing polymer waste. Energy 94 (1), 443–456.
Nasr, M.B., Chtourou, M., 2011. A self-organizing map-based initialization for hybrid training of feedforward neural networks. Appl. Soft Comput. 11 (8), 4458–4464.
Ning, K.F., Ning, M., Dong, M.Y., 2015. A new robust ELM method based on a Bayesian framework with heavy-tailed distribution and weighted likelihood function. Neurocomputing 149, 891–903.
Pfrieger, F.W., 2002. Role of glia in synapse development. Curr. Opin. Neurobiol. 12 (5), 486–490.
Qiao, J.F., Han, H.G., 2012. Identification and modeling of nonlinear dynamical systems using a novel self-organizing RBF-based approach. Automatica 48 (8), 1729–1734.
Rizwan, W., Farheen, K., 2015. Soft chemically synthesized zinc oxide micro-flowers for the enhanced photocatalytic properties and their analytical determination. J. Ind. Eng. Chem. 22, 192–198.
Schaik, A.V., Tapson, J., 2015. Online and adaptive pseudoinverse solutions for ELM weights. Neurocomputing 149, 233–238.
Sun, J., 2012. Learning algorithm and hidden node selection scheme for local coupled feedforward neural network classifier. Neurocomputing 79, 158–163.
Ullian, E.M., Sapperstein, S.K., Christopherson, K.S., Barres, B.A., 2001. Control of synapse number by glia. Science, 657–661.
Wang, D.G., Song, W.Y., Li, H.X., 2015. Approximation properties of ELM-fuzzy systems for smooth functions and their derivatives. Neurocomputing 149, 265–274, (Part A).
Wang, R., He, Y.L., Chow, C.Y., Ou, F.F., Zhang, J., 2015. Learning ELM-Tree from big data based on uncertainty reduction. Fuzzy Sets Syst. 258, 79–100.
Willshaw, D.J., Von, d.M.C., 1976. How patterned neural connections can be set up by self-organization. Proc. R. Soc. Lond. 194 (1117), 431–445.
Zhang, H.G., Yin, Y.X., 2016. An improved ELM algorithm for the measurement of hot metal temperature in blast furnace. Neurocomputing 174, 232–237.