A Self-Organizing Modular Neural Network for Nonlinear System Modeling

Xi Meng
Faculty of Information Technology, Beijing University of Technology
Beijing Key Laboratory of Computational Intelligence and Intelligence System
Beijing, China
mengxi@bjut.edu.cn

Limin Quan
Faculty of Information Technology, Beijing University of Technology
Beijing Key Laboratory of Computational Intelligence and Intelligence System
Beijing, China
quanlimin12@sina.com

Junfei Qiao
Faculty of Information Technology, Beijing University of Technology
Beijing Key Laboratory of Computational Intelligence and Intelligence System
Beijing, China
junfeiq@bjut.edu.cn

***Abstract*—In this paper, a self-organizing modular neural network (SO-MNN) is proposed for nonlinear system modeling, in which the modular structure and sub-networks are optimized to improve the modeling accuracy and efficiency. First, the modular structure is constructed by seeking the maximum modularity degree of the whole neural network and the optimal hub center in each sub-network. Simultaneously, the task is divided into several sub-tasks. Then, according to the assigned sub-tasks, sub-networks are constructed based on an incremental radial basis function (RBF) neural network, whose performance can be guaranteed by a structure growing mechanism and an adaptive second-order learning algorithm. Finally, during the testing or application processes, a winner-take-all strategy is used to integrated all the sub-networks. The effectiveness of the proposed methodology is verified by two benchmark problems and a real-world application.**

Keywords—Modular neural network, modularity optimization, radial basis function neural network, nonlinear system modeling

I. Introduction

Modeling of nonlinear systems has always been a hot topic; it is because a majority of systems are inherently nonlinear in industrial processes [1-3]. And it is first and foremost to establish accurate models for advanced control and optimization. Although mechanistic modeling methodology can provide the objective phenomenon explanation and physics sense, it usually requires detailed knowledge. Furthermore, it is generally difficult to guarantee the accuracy of the constructed mechanistic models [4,5]. In recent decades, the rapid growth of computation intelligence has offered a large quantity of data-driven modeling methods for nonlinear systems [4-9]. Due to the powerful learning ability and considerable generalization performance, artificial neural networks tend to be the most widely used approach for the state-of-the-art data-driven nonlinear system modeling [10-13].

In order to build useful computational models by mimicking the structure and functionality of the human brain, artificial neural networks have been widely and deeply investigated over the last decades. Studies on brain suggest that the structural brain networks exhibit highly modularity, which is the underlying of some important properties, such as efficient information passing, robustness, adaptability, and so on [14]. Thus, Jacobs and Jordan first introduced the modular structure to artificial neural networks in [15]. Thereafter, modular neural networks have been applied to handle complex problems due to their powerful “dividing-and-conquering” ability[16-21].

With the aim to predict the dynamic behavior of complex economic time series, Melin *et al.* proposed a hybrid modular neural network, wherein the sub-networks were integrated based on a fuzzy strategy[22]. In [23], a modular neural network was constructed based on radial basis function (RBF) neural networks, then it was verified by an exchange rates forecasting problem. In [24], modular neural networks were applied to identify dynamic nonlinear systems, and the computational complexity was greatly reduced by decomposing the whole system into several sub-systems. In order to model time-varying systems, Qiao *et al.* [25] developed an online self-adaptive modular neural network (OSAMNN), which obtained considerable modeling efficiency and accuracy on certain problems. In [26], a modular ridge randomized neural network was applied to estimate the sea-ice thickness, and experiment results presented its improved accuracy and robustness. Aiming to identify the nonlinear relationship between input and output variables which exhibited in most physical systems, Baldacchino proposed a mixture of experts model [27]. Modular neural networks are gradually showing their excellent performance on nonlinear system modeling.

The construction or application of modular neural networks mainly include three stages: task decomposition, sub-networks design, and the outputs integration. And the performance of modular neural networks are determined by those three stages. For the first task-decomposition stage, clustering algorithms are always utilized to divide a complex task into different sub-tasks, while the number of the modules are decided[26,28,29]. As to the sub-networks, feedforward neural networks tend to a

This work was supported by the National Science Foundation of China under Grants 61903012, 61533002 and 61890930-5, the Major Science and Technology Program for Water Pollution Control and Treatment of China under Grant 2018ZX07111005, and the Beijing Natural Science Foundation under Grant 4204088.

preferable choice due to their simple structures and good generalization ability. At last, the integration of different sub-networks depends on the decomposition methodology, generally using a competitive or cooperative strategy. Although modular neural networks can handle complex tasks well by implementing dividing-and-conquering, there are still two open questions:

1) How to decompose the whole task efficiently or construct the optimal modular structure?

2) How to design the sub-networks to reduce the computational complexity and guarantee the generalization performance?

Consequently, a self-organizing modular neural network (SO-MNN) is proposed to tackle the aforementioned two questions. First, a brain-inspired modularity index is developed to construct the modular structure, trying to find the optimal number of sub-networks. At this stage, the whole task is divided into several simple sub-tasks. Then, RBF neural networks are applied to construct the sub-networks. An incremental structure growing mechanism and a second-order algorithm work together to guarantee the parsimony, learning ability and generalization performance of both sub-networks and the whole modular neural network. Since the task is divided into different separate sub-tasks, a winner-take-all competition strategy is utilized to integrate all the sub-networks during the testing or application processes. The main contribution of this work are as follows.

1) A modularity index deriving from the brain network is developed to construct the modular structure, and the task can be adaptively decomposed under different conditions.

2) A self-organizing RBF network is utilized to handle separate sub-tasks and provides guarantee for the accuracy and efficiency of the designed model.

The remainder of this paper is organized as follows. Section 2 reviews modular neural networks briefly. Then the SO-MNN is introduced in details in Section 3. Section 4 demonstrates the effectiveness of the SO-MNN through a series of experiments. Finally, the paper is concluded in Section 5.

II. Preliminary

A. *Brief introduction of modular neural networks*

Since the modular neural network was first proposed in 1989, it has been applied to many areas over the last three decades. The core of modular neural networks is to divide a complex task into several sub-tasks and then tackle each sub-task with a separate sub-network. The typical structure of modular neural networks is given in Fig. 1, which is composed of a task-decomposition stage, a sub-networks stage, and an outputs-integration stage.

Consider a task D contains a training data set $D=\{(\boldsymbol{x}_i,y_i)|\boldsymbol{x}_i\in\mathbb{R}^M,|y_i\in\mathbb{R}^1,i=1,2,...,N\}$. In the task-decomposition stage, the whole task D is divided into a series of sub-tasks $\{D_1,D_2,...,D_s,...,D_P\},(D_s\subset D)$ based on a decomposition method:

$$D_s=\{(\boldsymbol{x}_i,y_i)|\boldsymbol{x}_i\in\mathbb{R}^M,|y_i\in\mathbb{R}^1,i=1,2,...,N_s\} \quad (1)$$

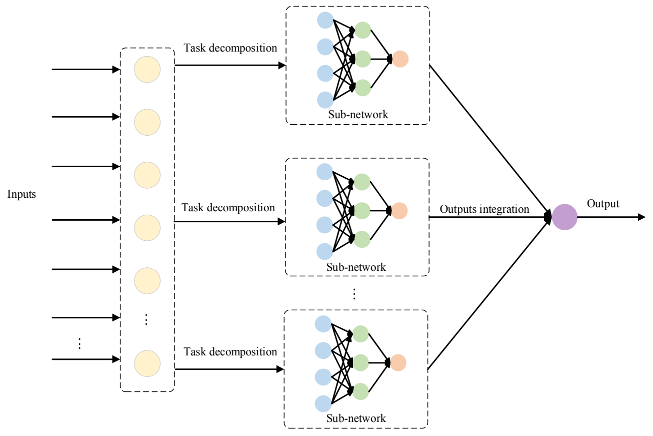

Fig. 1. The typical structure of modular neural networks.

where D_s is the sub-set that is assigned to the s-th sub-network, and N_s denotes the number of data samples in this sub-network. In general, clustering methods are the commonly used decomposition method.

Then, sub-networks would be constructed in parallel based on those assigned sub-tasks. Due to the simple structure and achievable approximation ability, forward neural networks such as the multilayer perceptron (MLP) and RBF neural network are the prior choices for the sub-networks:

$$\tilde{y}_s(\boldsymbol{x}_i)=\sum_{j=1}^{J_s}w_jf_j(\boldsymbol{x}_i) \quad (2)$$

where J_s is the number of hidden nodes of the s-th sub-network, $f_j(\boldsymbol{x}_i)$ denotes the activation function of this sub-network, and $\tilde{y}_s(\boldsymbol{x}_i)$ is the output calculated by the sub-network.

Finally, during the testing or application processes, when a new sample ($\boldsymbol{x}_t$, y_t) is fed into the modular neural network, an integration strategy should be used to obtain the final output:

$$\hat{y}_t=g\left(\tilde{y}_1(\boldsymbol{x}_t),...,\tilde{y}_s(\boldsymbol{x}_t),...,\tilde{y}_p(\boldsymbol{x}_t)\right) \quad (3)$$

where $\hat{y}_t$ denotes the actual output of the modular neural network, and $g(.)$ is the selective competition or cooperative integration methodology.

III. The self-organizing modular neural network

One of the most important organizational principles in brain networks is the modular structure[14]. A module is topologically defined as a region with several densely inter-connected nodes which are relatively sparsely connected to other regions. Moreover, there is a node that is highly connected within each module, and it is called the hub[30]. Taking inspiration from these properties of brain networks, a self-organizing modular neural network (SO-MNN) is proposed in this section.

A. *Modularity index*

In order to evaluate the modularity degree of the whole network quantificationally, a modularity index is developed in this work, including compactness and separation. A compactness index J_C is developed to described the density in each module, while a separation index J_S is proposed to denote the sparsity between different modules. Consequently, these two indices can be utilized to seek the maximum modularity degree of the network.

First, the compactness is calculated by the average density of the whole SO-MNN, which is expressed as follows:

$$J_C = \frac{1}{P}\sum_{l=1}^{P}\left(\frac{1}{N_l}\sum_{i=1}^{N_l}\exp\frac{-\|\boldsymbol{h}_l - \boldsymbol{x}_i\|}{r_l}\right) \tag{4}$$

where P is the number of sub-networks, N_l is the number of data samples which are assigned to the l-th sub-network, $\boldsymbol{h}_l$ represents the center vector of the hub in the l-th sub-network, r_l is the effect range, and $\boldsymbol{x}_i$ denotes the input.

As for the sparsity between different modules, the Euclidean distance between different sub-networks is utilized to design the separation index:

$$J_S = \frac{1}{P}\sum_{l=1}^{P}\exp\left[-\min_{s\in P, s\neq l}\left\{d\left(\boldsymbol{h}_l, \boldsymbol{h}_s\right)\right\}\right] \tag{5}$$

where $d(\boldsymbol{h}_l, \boldsymbol{h}_s)$ is the distance metric between the l-th hub and the s-th hub. Then, taking account of the compactness index and the separation index, the modularity degree of the whole network can be quantitatively measured by:

$$MQ = \frac{J_C}{J_S} = \frac{\frac{1}{P}\sum_{l=1}^{P}\left(\frac{1}{N_l}\sum_{i=1}^{N_l}\exp\frac{-\|\boldsymbol{h}_l - \boldsymbol{x}_i\|}{r_l}\right)}{\frac{1}{P}\sum_{l=1}^{P}\exp\left[-\min_{s\in P, s\neq l}\left\{d\left(\boldsymbol{h}_l, \boldsymbol{h}_s\right)\right\}\right]} = \frac{\sum_{l=1}^{P}\left(\frac{1}{N_l}\sum_{i=1}^{N_l}\exp\frac{-\|\boldsymbol{h}_l - \boldsymbol{x}_i\|}{r_l}\right)}{\sum_{l=1}^{P}\exp\left[-\min_{s\in P, s\neq l}\left\{d\left(\boldsymbol{h}_l, \boldsymbol{h}_s\right)\right\}\right]} \tag{6}$$

Thus, the design of modular structure in SO-MNN can be achieved by maximizing the modularity index, which will be described in details in the following section.

B. Modular structure constructing stage

Initially, there is no sub-network in the second stage. When the first training sample ($\boldsymbol{x}_1$, y_1) is added into the network, this sample is placed as the hub of the first sub-network:

$$\boldsymbol{h}_1 = \boldsymbol{x}_1 \tag{7}$$

where $\boldsymbol{h}_1$ denotes the center vector of the first hub. Then, the effect range is set based on the distribution of the samples. After calculating the Euclidean distances between data samples, the effect range can be obtained according to the largest distance:

$$r_1 = \xi d_{\max},\ \ \xi \in \left[\tfrac{1}{5}, \tfrac{1}{3}\right] \tag{8}$$

$$d_{\max} = \max_{i\neq j}\left\{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\right\} \tag{9}$$

At time t, suppose there are l sub-networks in the second stage. When a new sample $\boldsymbol{x}_t$ is added into the task-decomposition stage, find the hub $l_{\min}$ that is closest to the newly adding sample:

$$l_{\min} = \arg\min_{s\in(1,..,l)}\left\{\|\boldsymbol{x}_t - \boldsymbol{h}_s\|\right\} \tag{10}$$

If the sample is in the effect range of this hub, it would be assigned to the $l_{\min}$-th sub-network. Then, to guarantee the modularity degree of the whole network, using eq (6) to calculate the modularity index under different conditions. And $MQ_{l\min}$ refers to the modularity degree when keeping the original hub, while MQ_t denotes the modularity while selecting the t-th sample as the hub.

If $MQ_{l_{\min}} \geq MQ_t$, it means the $l_{\min}$-th hub is more appropriate to be as the hub. Thus, only the effect range should be adjusted to the largest distance between the $l_{\min}$-th and other samples:

$$r_{l_{\min}} = \max\left\{\|\boldsymbol{h}_{l_{\min}} - \boldsymbol{x}_i\|\right\}\ \ i = 1,2,..,N_{l_{\min}} \tag{11}$$

Contrariwise, the t-th sample would replace the original $l_{\min}$-th hub:

$$\boldsymbol{h}_{l_{\min}} = \boldsymbol{x}_t \tag{12}$$

$$r_{l_{\min}} = \max\left\{\|\boldsymbol{h}_{l_{\min}} - \boldsymbol{x}_i\|\right\}\ \ i = 1,2,..,N_{l_{\min}} \tag{13}$$

On the other hand, if the sample is not at the effect range of this hub, a new sub-network should be constructed:

$$\boldsymbol{h}_{l+1} = \boldsymbol{x}_t \tag{14}$$

$$r_{l+1} = \xi d_{\max},\ \ \xi \in \left[\tfrac{1}{5}, \tfrac{1}{3}\right] \tag{15}$$

After all the training samples are sent into the network one-by-one, the modular structure is evolved with the whole task being decomposing into different sub-tasks.

C. Sub-network designing stage

During the modular structure constructing stage, the data set $D = \left\{\left(\boldsymbol{x}_i, y_i\right) \middle| \boldsymbol{x}_i \in \mathbb{R}^M, \middle| y_i \in \mathbb{R}^1, i = 1,2,...,N\right\}$ has been divided into P disjoint sub-sets. To tackle the sub-tasks fast and effectively, an incremental RBF network is applied to design the corresponding sub-network, which is based on an improved error correction (IErrCor) algorithm [11]. Take the s-th sub-network for example, its output can be calculated by:

$$\theta_j(\boldsymbol{x}) = \exp(-\|\boldsymbol{x} - \boldsymbol{c}_j\|^2 / \sigma_j^2) \tag{16}$$

$$\tilde{y} = \sum_{j=1}^{J_s} w_j \theta_j \tag{17}$$

where θ_j is the activation function of the jth RBF node, $\boldsymbol{c}_j$ is its center vector, σ_j denotes the radius, and w_j is the connection weight between the RBF node and the output.

For RBF networks, sound learning performance is the precondition of their applications, which can be quantitatively evaluated by the root mean square error (RMSE):

$$E = \sqrt{\frac{1}{N_s}\sum_{n=1}^{N_s}\left(\tilde{y}(n) - y_n\right)^2} \tag{18}$$

where N_s is the number of samples in the s-th subnetwork, while $\tilde{y}(n)$ and y_n denote the actual output and desired output of the n-th sample. During the training process, if $E(t)>\varepsilon$, then a new RBF node should be added to improve the learning performance:

$$k = \arg \max_{n \in 1,2..,N_s} \{\|\tilde{y}(n) - y_n\|\} \tag{19}$$

$$\boldsymbol{c}_{new} = \boldsymbol{x}_k \tag{20}$$

$$w_{new} = \tilde{y}(k) - y_k \tag{21}$$

$$\sigma_{new} = 0.5 \min\left\{\text{dist}\left(\boldsymbol{c}_{new}, \boldsymbol{c}_{j \neq new}\right)\right\} \tag{22}$$

where k refers to the sample with the instantaneous largest residual error.

Each time an RBF node is added, an adaptive second-order learning algorithm is used to adjust all the parameters [31]:

$$\boldsymbol{\Psi}(t+1) = \boldsymbol{\Psi}(t) - (\boldsymbol{Q}(t) + \lambda(t)\boldsymbol{I}(t))^{-1}\boldsymbol{g}(t) \tag{23}$$

where $\boldsymbol{Q}$ is the Quasi-Hessian matrix, λ is the learning coefficient; $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{g}$ denotes the gradient vector, and $\boldsymbol{\Psi}$ refers to all the parameters needed to be tuned:

$$\begin{aligned}\boldsymbol{\Psi}(t) = [&\boldsymbol{c}_1(t), \boldsymbol{c}_2(t), \ldots, \boldsymbol{c}_J(t), \sigma_1(t), \sigma_2(t), \\ &\ldots, \sigma_J(t), w_1(t), w_2(t), \ldots, w_J(t)]\end{aligned} \tag{24}$$

To reduce the computation complexity and memory requirements, quasi Hessian matrix and gradient vector are transformed to the sum of sub Quasi-Hessian matrixes and the sum of sub gradient vectors:

$$\boldsymbol{Q}(t) = \sum_{n=1}^{N_s} \boldsymbol{q}_n(t) \tag{25}$$

$$\boldsymbol{g}(t) = \sum_{n=1}^{N_s} \boldsymbol{\eta}_n(t) \tag{26}$$

And the sub Quasi-Hessian matrix $\boldsymbol{q}_n(t)$ and sub gradient vector $\boldsymbol{\eta}_n(t)$ can be obtained by:

$$\boldsymbol{q}_n(t) = \boldsymbol{j}_n^T(t)\boldsymbol{j}_n(t) \tag{27}$$

$$\boldsymbol{\eta}_n(t) = \boldsymbol{j}_n^T(t)e_n(t) \tag{28}$$

where e_n and $\boldsymbol{j}_n$ are given as follows:

$$e_n = y_n - \tilde{y}(n) \tag{29}$$

$$\boldsymbol{j}_n = \left[\frac{\partial e_n}{\partial c_j}, \ldots, \frac{\partial e_n}{\partial \sigma_j}, \ldots, \frac{\partial e_n}{\partial w_j}\right] \tag{30}$$

With differential chain rule, the Jacobian row elements in eq (30) can be rewritten as:

$$\frac{\partial e_n}{\partial c_j} = -\frac{\partial \tilde{y}(n)}{\partial c_j} = -\frac{2w_j\theta_j(\boldsymbol{x}_n)\|\boldsymbol{x}_n - \boldsymbol{c}_j\|}{\sigma_j} \tag{31}$$

$$\frac{\partial e_P}{\partial \sigma_j} = -\frac{\partial \tilde{y}(n)}{\partial \sigma_j} = -\frac{2w_j\theta_j(\boldsymbol{x}_n)\|\boldsymbol{x}_n - \boldsymbol{c}_j\|^2}{\sigma_j^3} \tag{32}$$

$$\frac{\partial e_n}{\partial w_j} = -\frac{\partial \tilde{y}(n)}{\partial w_j} = -\theta_j(\boldsymbol{x}_n) \tag{33}$$

In each sub-network, RBF nodes are located successively to achieve the desired learning accuracy according to the assigned sub-task.

D. The winner-take-all integration stage

Since the sub-networks have been constructed in parallel, a winner-take-all decision strategy is applied in the integration stage. When a testing sample $\boldsymbol{x}_t$ is presented, only the sub-network whose hub is next to the current input (in Euclidean distance metric) will be allowed to take the final integration:

$$l_{\text{act}} = \arg \min_{s \in (1,..,P)} \{\|\boldsymbol{x}_t - \boldsymbol{h}_s\|\} \tag{34}$$

where the l_{act}-th sub-network is the winner. Thus, the actual output of the SO-MNN can be obtained by

$$\hat{y}_t = \tilde{y}_{l_{\text{act}}} \tag{35}$$

It should be noted that the integration only works during the testing or application processes of the SO-MNN.

IV. Experimental studies

In this section, the nonlinear modeling ability of the proposed method is evaluated by two benchmark problems initially. Then, the tested SO-MNN is applied to a wastewater treatment process. For simplicity and consistency, the parameter ξ is set to $\frac{1}{3}$ in the experiments. All the simulations were programmed with MATLAB R2016a and were run on a PC with a clock speed 3.6 GHz and 8 GB RAM, under a Microsoft Windows 10.0 environment.

A. Rapidly changing function approximation

In this experiment, sin$E$ function is utilized to verify the nonlinear modeling performance of the proposed SO-MNN:

$$y = 0.8\exp(-0.2x)\sin(10x) \tag{36}$$

There is a total of 4500 data points for this problem, with the input x randomly distributed in the interval [0, 10]. Among them, 3000 data points are chosen as the training samples, while the other 1500 data points are taken as the testing samples.

During the modular structure constructing stage, the training samples are decomposed into 3 sets, and each set contains 1443, 211, 1346 samples, respectively. Then, sub-networks based on the incremental RBF networks are designed correspondingly. The structures of the sub-networks are 1-10-1, 1-6-1, and 1-10-1. Finally, the constructed SO-MNN is evaluate by the 1500 testing samples. The fitting curve and error curve are illustrated in Fig.2 and Fig.3, demonstrating the satisfactory approximation ability of the proposed SO-MNN on this function approximation problem.

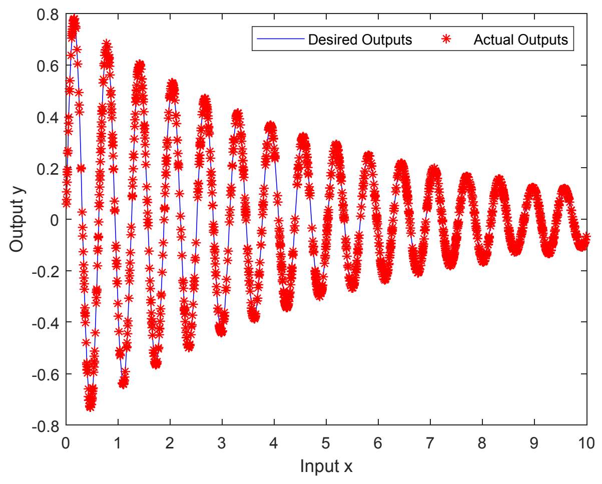

Fig. 2. Testing results of the SO-MNN on sin$E$ function approximation

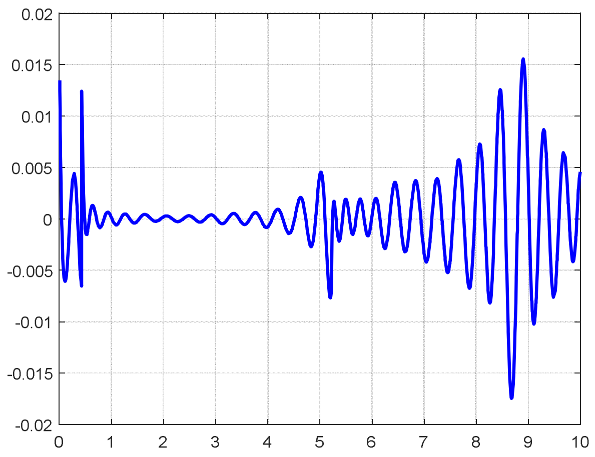

Fig.3. Testing error values of the SO-MNN on sin$E$ function approximation

TABLE I. PERFORMANCE COMPARISON ON SINE FUNCTION APPROXIMATION

Algorithms	***Training time(s)***	***Number of modules***	***Number of hidden nodes***	***Training RMSE***	***Testing RMSE***
GAP-RBF	24.808*	1*	45*	0.0261*	0.0269*
ELM	0.590	1	100	0.1315	0.1352
ErrCor	22.120	1	20	0.0057	0.0056
OSAMNN	13.579	4	28	0.0217	0.0240
SO-MNN	11.246	3	26	0.0037	0.0039

*: the results are presented in the original literature[20].

To further evaluate the performance of the proposed SO-MNN, the results are compared with other popular algorithms and modular neural network from the training time, the number of modules, the number of hidden nodes, training RMSE, and testing RMSE, which are listed in Table I. As can be seen, when comparing with the single feedforward neural networks such as GAP-RBF[32], ELM[33], and ErrCor[34], the SO-MNN shows distinct advantages on the modeling accuracy. Although there are 3 sub-networks in the SO-MNN, the number of hidden nodes is still less than most neural networks. Besides, in comparison with the OSAMNN, the SO-MNN presents significant superiority on both the model structure and accuracy, demonstrating the effectiveness of the proposed algorithm.

B. Nonlinear plant identification

This simulation evaluates the proposed method by a nonlinear plant[25,35], which is described by the following second-order difference equation:

$$\begin{aligned} y(t+1) &= g(y(t), y(t-1)) + u(t) \\ &= \frac{y(t)y(t-1)[y(t)+2.5]}{1+y^2(t)+y^2(t-1)} + u(t) \end{aligned} \tag{37}$$

The goal is to generate a nonlinear mapping between the input vector ($u(t)$, $y(t$-1), $y(t)$) and the output vector $y(t$+1). The observations for $1 \le t \le 5000$ are used as the training samples, and the input $u(t)$ is a random signal from the interval [0, 10]. Then the designed SO-MNN are evaluated on the observations for $5001 \le t \le 5200$, while the input is a sinusoidal signal given by $u(t) = \sin(2\pi t/25)$.

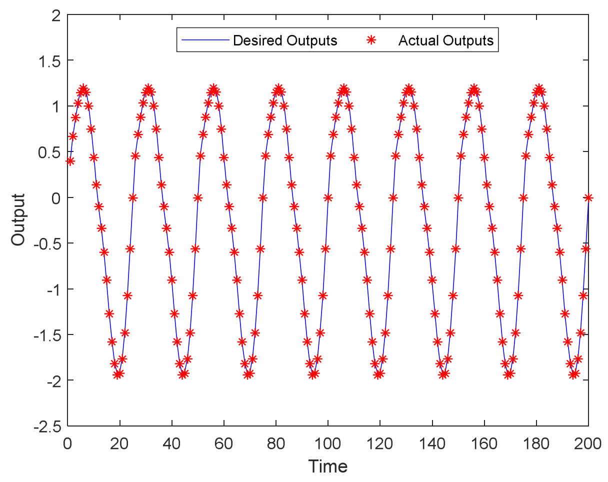

Fig. 4. Identification results of the SO-MNN during the testing process

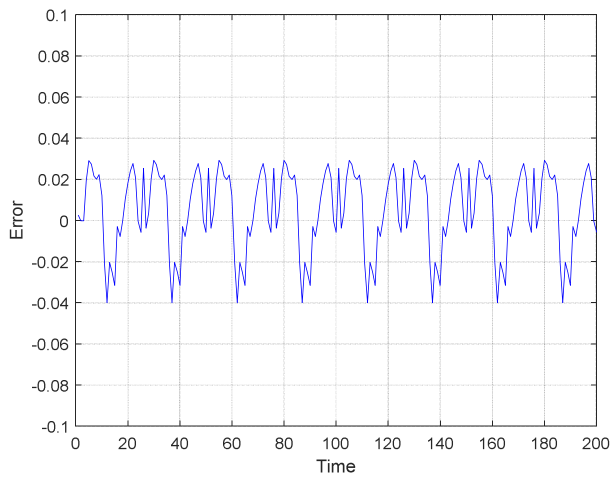

Fig. 5. Identification errors of the SO-MNN during the testing process

TABLE II. PERFORMANCE COMPARISON ON NONLINEAR PLANT IDENTIFICATION

Algorithms	***Training time(s)***	***Number of modules***	***Number of hidden nodes***	***Training RMSE***	***Testing RMSE***
MRAN[36]	/	1*	22*	0.0371*	0.0271*
SeroFAM [35]	/	1*	27.8*	/	0.0432*
ErrCor	57.593	1	20	0.0270	0.0208
OSAMNN	/	10*	40*	0.0336*	0.0206*
SO-MNN	19.485	4	28	0.0239	0.0204

*: the results are presented in the original literature[25];
/: the results are not listed in the original literature[25].

In this experiment, the SO-MNN is constructed with 4 sub-networks. The testing results are shown in Fig.4 and Fig.5, and both the fitting curve and the error curve demonstrate the considerable identification accuracy of the proposed methodology. Additionally, detailed comparisons between the SO-MNN and other algorithms are listed in Table II. It is clear to see that the SO-MNN can obtain the highest identification accuracy among all the algorithms. On the one hand, the comparison between the SO-MNN and the ErrCor demonstrates the advantage of modular neural networks. On the other hand, in compared with the OSAMNN, the proposed SO-MNN achieves higher accuracy with a more compact structure. All the results reveal that the SO-MNN can establish an accurate model fast for this nonlinear plant.

C. Applications to the wastewater treatment process

In the wastewater treatment processes, some key water quality parameters such as BOD and TN present challenging difficulties in their fast and accurate measurements. Data-driven soft-sensing techniques tend to an efficient way to obtain the real-time measurements. Thus, after evaluating its nonlinear modeling ability on the aforementioned benchmark problems, the SO-MNN is applied to construct data-driven soft-sensing models to realize the online measurement of the effluent BOD and the effluent TN.

The simulation data was from a real wastewater treatment plant in Beijing, China. After data preprocessing, 270 samples

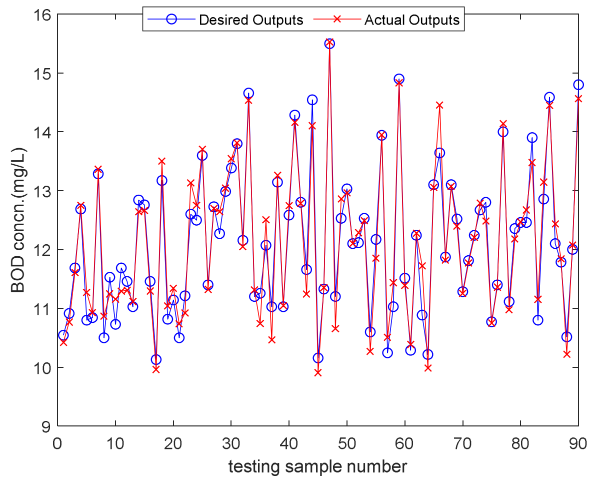

Fig. 6. Prediction results of effluent BOD concentration by SO-MNN

are utilized as training data, while the other 90 samples are taken as testing data. Before constructing the soft-sensing models, to reduce the computational complexity and improve the model accuracy, the mutual information is applied to measure the correlation between other variables and the effluent BOD or the effluent TN. The mutual information between different variables can be calculated by

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)} \tag{38}$$

where $p(x,y)$ represents the joint probability density function of X and Y, and $p(x)$ and $p(y)$ are the probability density functions of X and Y respectively. After the mutual information analysis, the influent TP, the effluent NH4-N, the settling velocity (SV), the effluent oil, the mixed liquor suspended solids (MLSS), and the influent oil are selected as the input variables for the effluent BOD model. As for the effluent TN model, the input variables are the influent water temperature, the effluent water temperature, the effluent NH4-N, the influent TN, the influent TP, and the MLSS.

Based on the training samples, the effluent BOD soft-sensing model is constructed by 4 sub-networks with 18 RBF nodes. As can be seen from Fig.6, the prediction curve can match the real curve well. Additionally, the prediction errors are illustrated in Fig.7. To realize the online measurement of the effluent TN, the SO-MNN needs 2 sub-networks and 14 RBF nodes. The prediction curve and error curve in Fig.8 and Fig.9 demonstrate the considerable results of the proposed algorithm on this application.

Furthermore, other algorithms are also used to design the soft sensors, including the GAP, ELM, ErrCor, APSO-RBF[37], and the OSAMNN, which are listed in Table III and Table IV. In comparison with those single feedforward neural networks, the SO-MNN obtains better performance by implementing the dividing-and-conquering. Besides, when comparing with the OSAMNN, the proposed algorithm also shows outperformance, demonstrating the validity of the task-decomposing and sub-network designing methods.

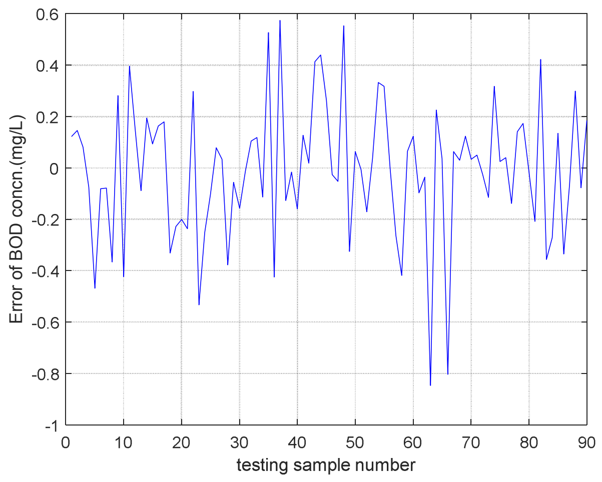

Fig. 7. Prediction errors of the effluent BOD concentration by SO-MNN

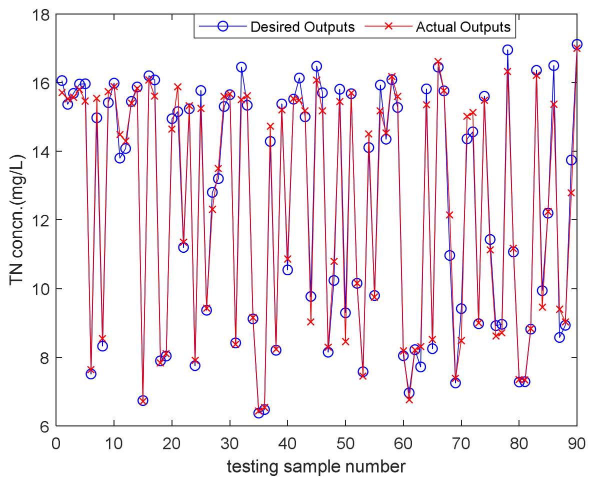

Fig. 8. Prediction results of the effluent TN concentration by SO-MNN

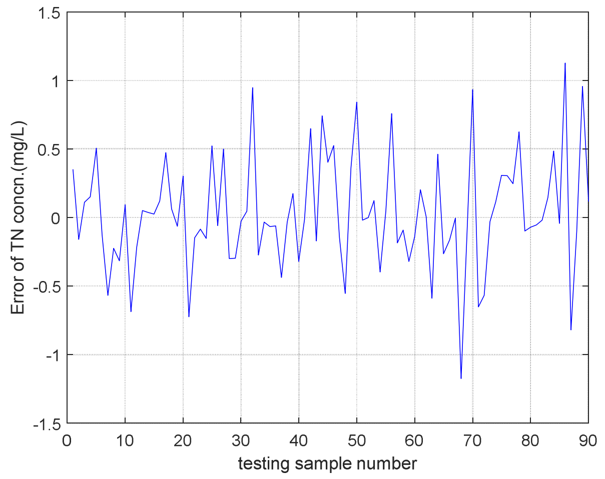

Fig. 9. Prediction results of the effluent TN concentration by SO-MNN

TABLE III. PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS ON PREDICTION OF THE EFFLUENT BOD

Algorithms	*Training time(s)*	*Number of modules*	*Number of hidden nodes*	*Testing RMSE*	*Testing MAPE(%)*
GAP	17.528	1	48	0.5034	6.3787
ELM	0.362	1	100	0.8464	4.4621
ErrCor	6.612	1	19	0.2865	1.6276
APSO-RBF	64.625	1	25	0.4898	3.4130
OSAMNN	1.471	4	24	0.3292	1.5392
SO-MNN	1.011	4	18	0.2665	1.7046

TABLE IV. PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS ON PREDICTION OF THE EFFLUENT TN

Algorithms	*Training time(s)*	*Number of modules*	*Number of hidden nodes*	*Testing RMSE*	*Testing MAPE(%)*
GAP	27.031	1	60	0.6509	3.8732
ELM	0.274	1	100	0.9993	4.2524
ErrCor	5.721	1	16	0.4418	2.4910
APSO-RBF	36.480	1	20	0.8292	5.6000
OSAMNN	1.539	4	28	0.6089	2.9386
SO-MNN	1.116	2	14	0.4139	2.4600

V. CONCLUSION

In this paper, a self-organizing modular neural network was proposed to construct efficient models for nonlinear systems. With the aim to determine the optimal module number of the SO-MNN, a modularity index was developed and worked with the hub to construct the modular structure, which was inspired by the structure feature in the brain networks. Simultaneously, the task was decomposed into several separate sub-tasks. Then, an incremental RBF neural network was applied to design the sub-networks in parallel, guaranteeing the compactness and generalization performance of the whole SO-MNN. Since the sub-tasks were independent from each other, a winner-take-all strategy was used to integrate them during the testing or application processes. Finally, the efficiency and superiority of the proposed method were demonstrated by a benchmark simulation and a real-word application. In the future, to cope with the changing working conditions during the complex industrial processes, a dynamic integration strategy could be proposed for modular neural networks to improve the modeling performance.

REFERENCES

[1] X. Yuan, Y. Wan, C. Yan, Z. Ge, Z. Song, and W. Gui, "Weighted linear dynamic system for feature representation and soft sensor application in nonlinear dynamic industrial processes," IEEE Trans. Ind. Electron., vol. 65, no. 2, pp. 1508–1517, 2017.

[2] W. Shao, L. Yao, Z. Ge, and Z. Song, "Parallel computing and SGD-based DPMM for soft sensor development with large-scale semisupervised data," IEEE Trans. Ind. Electron., vol. 66, no. 8, pp. 6362–6373, 2019.

[3] J. Qiao, X. Meng, and W. Li, "An incremental neuronal-activity-based RBF neural network for nonlinear system modeling," Neurocomputing, vol. 302, 2018.

[4] T. Wang, Z. Han, J. Zhao, and W. Wang, "Adaptive granulation-based prediction for energy system of steel industry," IEEE Trans. Cybern., vol. 48, no. 1, pp. 127–138, 2018.

[5] Y. Xie, J. Yu, S. Xie, T. Huang, and W. Gui, "On-line prediction of ferrous ion concentration in goethite process based on self-adjusting structure RBF neural network," Neural Networks, vol. 116, pp. 1–10, 2019.

[6] J. Zhao, W. Pedrycz, C. Sheng, Q. Liu, and W. Wang, "Data-Based Predictive Optimization for Byproduct Gas System in Steel Industry," IEEE Trans. Autom. Sci. Eng., vol. 14, no. 4, pp. 1761–1770, 2016.

[7] W. Shao, L. Yao, Z. Song, and Z. Ge, "Parallel Computing and SGD Based DPMM For Soft Sensor Development With Large-Scale Semisupervised Data," IEEE Trans. Ind. Electron., vol. PP, no. c, p. 1, 2018.

[8] H. Haimi, M. Mulas, F. Corona, and R. Vahala, "Data-derived soft-sensors for biological wastewater treatment plants: An overview," Environ. Model. Softw., vol. 47, pp. 88–107, 2013.

[9] H. G. Han, Z. Y. Chen, H. X. Liu, and J. F. Qiao, "A self-organizing interval Type-2 fuzzy-neural-network for modeling nonlinear systems," Neurocomputing, vol. 290, pp. 196–207, 2018.

[10] L. Wang, S. Mao, and B. Wilamowski, "Short-Term load forecasting with LSTM based ensemble learning," Proc. - 2019 IEEE Int. Congr. Cybermatics 12th IEEE Int. Conf. Internet Things, 15th IEEE Int. Conf. Green Comput. Commun. 12th IEEE Int. Conf. Cyber, Phys. So, pp. 793–800, 2019.

[11] X. Meng, P. Rozycki, J.-F. Qiao, and B. M. Wilamowski, "Nonlinear system modeling using RBF networks for industrial application," IEEE Trans. Ind. Informatics, vol. 14, no. 3, 2018.

[12] N. Vuković and Z. Miljković, "A growing and pruning sequential learning algorithm of hyper basis function neural network for function approximation," Neural Networks, vol. 46, pp. 210–226, 2013.

[13] C. Cecati, J. Kolbusz, P. Rózycki, P. Siano, and B. M. Wilamowski, "A Novel RBF Training Algorithm for Short-Term Electric Load Forecasting and Comparative Studies," IEEE Trans. Ind. Electron., vol. 62, no. 10, pp. 6519–6529, 2015.

[14] H. J. Park and K. Friston, "Structural and functional brain networks: From connections to cognition," Science, vol. 342, no. 6158, 2013.

[15] R. Jacobs and M. Jordan, "A competitive modular connectionist architecture," Adv. Neural Inf. Process. Syst., pp. 1–7, 1991.

[16] P. Melin *et al.*, "A Hybrid Approach for Modular Neural Network Design Using Intercriteria Analysis and Intuitionistic Fuzzy Logic," Complexity, vol. 2018, pp. 1–11, 2018.

[17] P. Melin, I. Miramontes, and G. Prado-Arechiga, “A hybrid model based on modular neural networks and fuzzy systems for classification of blood pressure and hypertension risk diagnosis,” Expert Syst. Appl., vol. 107, pp. 146–164, 2018.

[18] D. Sánchez, P. Melin, and O. Castillo, “A grey Wolf optimizer for modular granular neural networks for human recognition,” Comput. Intell. Neurosci., vol. 2017, 2017.

[19] D. Sánchez, P. Melin, and O. Castillo, “Optimization of modular granular neural networks using a firefly algorithm for human recognition,” Eng. Appl. Artif. Intell., vol. 64, no. 6, pp. 172–186, 2017.

[20] J. F. Qiao, X. Meng, W. J. Li, and B. M. Wilamowski, “A novel modular RBF neural network based on a brain-like partition method,” Neural Comput. Appl., vol. 32, pp. 172–186,2020.

[21] M. S. Zaghloul, R. A. Hamza, O. T. Iorhemen, J. H Tay, “Performance prediction of an aerobic granular SBR using modular multilayer artificial neural networks,” Science of The Total Environment, vol. 645, pp. 449–459, 2018.

[22] P. Melin, A. Mancilla, M. Lopez, and O. Mendoza, “A hybrid modular neural network architecture with fuzzy Sugeno integration for time series forecasting,” Appl. Soft Comput. J., vol. 7, no. 4, pp. 1217–1226, 2007.

[23] L. Yu, K. K. Lai, and S. Wang, “Multistage RBF neural network ensemble learning for exchange rates forecasting,” Neurocomputing, vol. 71, no. 16–18, pp. 3295–3302, 2008.

[24] G. Puscasu and B. Codres, “Nonlinear system identification and control based on modular neural networks,” Int. J. Neural Syst., vol. 21, no. 04, pp. 319–334, 2011.

[25] J. F.Qiao, Z. Z. Zhang, and Y. C. Bo, “An online self-adaptive modular neural network for time-varying systems,” Neurocomputing, vol. 125, pp. 7–16, 2014.

[26] A. Mozaffari, K. A. Scott, S. Chenouri, and N. L. Azad, “A modular ridge randomized neural network with differential evolutionary distributor applied to the estimation of sea ice thickness,” Soft Comput., vol. 21, no. 16, pp. 4635–4659, 2017.

[27] T. Baldacchino, E. J. Cross, K. Worden, and J. Rowson, “Variational bayesian mixture of experts models and sensitivity analysis for nonlinear dynamical systems,” Mech. Syst. Signal Process., vol. 66–67, pp. 178–200, 2016.

[28] Y. Tian, X. Ju, and Y. Shi, “A divide-and-combine method for large scale nonparallel support vector machines,” Neural Networks, vol. 75, pp. 12–21, 2016.

[29] A. O. Hoori and Y. Motai, “Multicolumn RBF Network,” IEEE Trans. Neural Networks Learn. Syst., vol. 29, no. 4, pp. 766–778, 2018.

[30] E. Bullmore and O. Sporns, “Complex brain networks: Graph theoretical analysis of structural and functional systems,” Nat. Rev. Neurosci., vol. 10, no. 3, pp. 186–198, 2009.

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

[32] G. B. Huang, P. Saratchandran, and N. Sundararajan, “An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks,” IEEE Trans. Syst. Man, Cybern. Part B Cybern., vol. 34, no. 6, pp. 2284–2292, 2004.

[33] G. B. Huang, Q. Y. Zhu, and C. K. Siew, “Extreme learning machine: Theory and applications,” Neurocomputing, vol. 70, no. 1–3, pp. 489–501, 2006.

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

[35] J. Tan and C. Quek, “A BCM theory of meta-plasticity for online self-reorganizing fuzzy-associative learning,” *IEEE Trans. Neural Networks*, vol. 21, no. 6, pp. 985–1003, 2010.

[36] Y. Lu, N. Sundararajan, and P. Saratchandran, “A sequential learning scheme for function approximation using minimal radial basis function neural networks,” Neural Computation, vol. 9, no. 2, pp. 461-478, 1997.

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