Neural Computing and Applications (2020) 32:899–911
https://doi.org/10.1007/s00521-018-3763-z

ORIGINAL ARTICLE

A novel modular RBF neural network based on a brain-like partition method

Jun-Fei Qiao[1,2] · Xi Meng[1,2] · Wen-Jing Li[1,2] · Bogdan M. Wilamowski[3]

Received: 25 September 2017 / Accepted: 28 September 2018 / Published online: 6 October 2018

Abstract
In this study, a modular design methodology inherited from cognitive neuroscience and neurophysiology is proposed to develop artificial neural networks, aiming to realize the powerful capability of brain—divide and conquer—when tackling complex problems. First, a density-based brain-like partition method is developed to construct the modular architecture, with a highly connected center in each sub-network as the human brain. The whole task is also divided into different sub-tasks at this stage. Then, a compact radial basis function (RBF) network with fast learning speed and desirable generalization performance is applied as the sub-network to solve the corresponding task. On the one hand, the modular structure helps to improve the ability of neural networks on complex problems by implementing divide and conquer. On the other hand, sub-networks with considerable ability could guarantee the parsimonious and generalization of the entire neural network. Finally, the novel modular RBF (NM-RBF) network is evaluated through multiple benchmark numerical experiments, and results demonstrate that the NM-RBF network is capable of constructing a relative compact architecture during a short learning process with achievable satisfactory generalization performance, showing its effectiveness and outperformance.

Keywords Modular neural network · Brain-like partition · Radial basis function (RBF) network · Second-order algorithm

1 Introduction

Artificial neural networks (ANNs), a kind of biologically motivated computing paradigm, have been extensively applied to a variety of areas due to their representational capability and generalization performance [1–4]. Although various techniques have been put forward to ANNs, from structure constructing methods to parameters tuning algorithms [5], they still suffer from large structure, computation complexity and slow convergence when solving complex problems [6]. Going backwards to the human brain, one essential and substantial feature is the modular and parallel structural connection and functional connectivity [7, 8]. Thus, the brain can handle different sub-tasks simultaneously after dividing and assigning a task to different brain regions. Inherited from the brain, the modular and parallel characteristics have been introduced to ANNs, aiming to solve problems quickly and effectively. Deriving from the modularity and parallelism, Jacobs et al. first introduced the "divide-and-conquer" idea to design modular and parallel ANNs in [9]. Inspired by their success on complex problems, lots of methods have been proposed to design and apply modular neural networks (MNNs) or parallel neural networks (PNNs) over the past 20 years, and a brief literature study is given below.

To improve the generalization performance of neural networks, Islam et al. [10] presented a constructive algorithm for training cooperative neural network ensembles (CNNEs). In CNNEs, different individual neural networks were encouraged to learn different aspects of training data so that the ensemble could learn the whole training data better. When applying MNNs to pattern profiling problems [11], the given problem was decomposed into smaller

✉ Xi Meng
mengxi@bjut.edu.cn

1 Faculty of Information Technology, Beijing University of Technology, Beijing, China

2 Beijing Key Laboratory of Computational Intelligence and Intelligence System, Beijing, China

3 Department of Electrical and Computer Engineering, Auburn University, Auburn, AL, USA

modules and comparable performance could be achieved with improvement in computation and model complexity. Aiming to improve the design flexibility and reliability to self-organizing map (SOM) users, Tokunaga and Furukawa [12] developed a modular network SOM (MNSOM) which was constitutive of parallel multilayer perceptrons (MLPs). In [13], a modular network with Hebbian learning rule was proposed, improving the recognition rate on separate handwritten Arabic digits of the Modified National Institute of Standards and Technology (MNIST) database. Considering the inability of single neural networks to cope with highly nonlinear relationships, Cimino et al. [14] proposed a novel neural network—referential multilayer perceptrons (RMLPs), whose modeling capabilities were compared with those of standard RBF networks. Focusing on the powerful concept of modularity, Ding et al. [15] introduced a two-layer modular neural system, which was useful in solving problems with high-dimensional inputs. In [16], unlike other methodologies, a group of well-trained MLPs were developed at first; then, some individual MLPs were selected based on sensitivity to process information. Qiao et al. [17] proposed an online self-adaptive modular neural network (OSAMN) for time-varying system modeling, in which the sub-networks could be added or merged to maintain suitable model complexity. With the aim to meet the challenge of big data issues, Wang et al. [18] proposed a parallelized extreme learning machine (ELM) ensemble, in which ELM was utilized to solve each sub-problem. To find the optimal number of sub-networks, Valdez et al. combined the advantages of particle swarm optimization (PSO) algorithm and genetic algorithm (GA) algorithm in [19]. When solving face recognition tasks in [20], three improved RBF neural networks were utilized to realize different phases, trying to improve the recognition rate. In order to tackle the large-scale Euclidean traveling salesman problems, Wang et al. [21] proposed a parallel computation model that was based on the SOM neural network. In [6], a parallelized structure ANN—multi-column RBF network (MCRBF)—was proposed to improve the classification ability of RBF networks on complex problem.

As above, although numerous algorithms have been proposed for MNNs, there are two major issues when designing or applying them: (1) dividing a whole task into different sub-tasks and (2) selecting or constructing ANNs to solve the corresponding sub-task. For the first question, the common method is to adopt an "appropriate" clustering algorithm to achieve "dividing" [6, 15, 18, 21]. Though all these techniques could obtain required performance on classification problems, most of these algorithms needed to set the number of categories initially. Additionally, heuristic algorithms have also been applied to design modular structure. When using the MNNs to solve human recognition problems, Sanchez and Melin [22] chose a hierarchical genetic algorithm to search for the optimal number of sub-modules. In [23], the differential evolution algorithm was utilized to divide the studied region into a set of subregions. Nevertheless, heuristic algorithms always encounter the computation complexity problem. How to construct the modular structure autonomously and quickly is still an open question.

As for the sub-network, MLP and RBF networks tend to be the popular choices [6, 10, 14–21]. It is due to their simple structures and proved generalization ability. Moreover, RBF networks, which are inherited from the concept of biological respective fields and with local mapping feature, should be a better choice for MNNs. Generally, the primary questions for applying RBF networks involve two issues: (1) the construction of the network structure and (2) the optimization of the network parameters. One conventional approach to determine the network structure size is by experience or by performing trials. However, such an approach cannot obtain a parsimonious neural network with desirable generalization performance. To overcome this deficiency, RBF networks with dynamic structure have been utilized to construct sub-networks. Recently, Yu et al. [24] proposed an error correction (ErrCor) algorithm to construct RBF networks which was tuned by an improved second-order (ISO) learning algorithm [25]. Results have demonstrated that this method can show significant advantages over some other popular algorithms, including support vector regression (SVR) [26], growing and pruning RBF (GAP-RBF) [27], generalized growing and pruning RBF (GGAP-RBF) [28] and extreme learning machine (ELM) [29]. Hence, in [6], ErrCor RBF network has been utilized to construct sub-networks in MNNs. However, there are still two deficiencies to overcome. On the one hand, since ISO algorithm is a gradient-based learning algorithm, random initial parameters may affect the convergence performance. On the other hand, computation complexity will increase with the data size, which may affect the computational efficiency. How to construct a compact neural network architecture with desirable generalization performance during fast learning process is still a challenge [30].

Considering the aforementioned problems, in this study, a novel modular RBF (NM-RBF) neural network is developed based on a brain-like partition method, aiming to solve complex problems quickly and efficiently. At first, a density-based brain-like partition method is proposed to realize the modular structure and perform "dividing" function. After the whole task is divided into one or several small sub-tasks individually, corresponding RBF networks are constructed to handle the assigned sub-task.

The main contributions of this paper are threefold:

1. A novel modular RBF network is developed to simulate the structural and functional feature of the brain. By the principle of divide and conquer, there are more possibility and feasibility for ANNs to deal with large and complex problems. The NM-RBF neural network can yield better performance than the single ANN.
2. A density-based brain-like partition method is proposed to divide a large complex task into different subtasks. The whole data can be partitioned to different data sets autonomously in this study, and each set is more isolated from others while with denser inner relations, which reflect the feature of brain intuitively and effectively.
3. A compact RBF neural network with optimal initial parameters which are tuned by an improved second-order algorithm is utilized as the sub-network. Based on the characteristic of RBF node, some modifications are proposed to seek optimal initial parameter for RBF networks in this paper, which can enhance and improve the convergence performance and stability of networks. Thus, sub-networks with desirable performance guarantee the computational efficiency, generalization performance and stability of the whole NM-RBF network.

The remainder of this paper is organized as follows. Section 2 introduces RBF network briefly. In Sect. 3, the proposed methodology is presented in detail. Section 4 evaluates the proposed NM-RBF neural network through a series of benchmark problems. Finally, this study is concluded in Sect. 5.

2 Related work

The novel modular RBF network is constructed based on RBF networks, which emulate the local response characteristics of certain biological networks. The typical structure of RBF networks is given in Fig. 1.

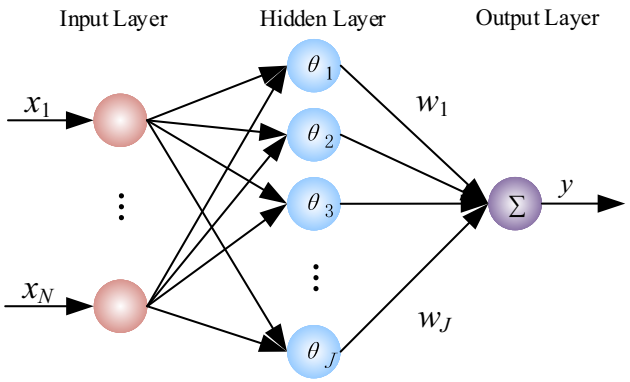

Fig. 1 Structure of RBF networks

RBF network is comprised of three layers: the input layer, the hidden layer and the output layer. The input layer receives information from the environment and transmits it into the hidden layer directly. In the hidden layer, the information is processed by each locally sensitive RBF node, and the activation function is always Gaussian function:

$$\theta_j(\boldsymbol{x}) = \mathrm{e}^{-\|\boldsymbol{x}-\boldsymbol{c}_j\|^2/\sigma_j^2}, \tag{1}$$

where $\boldsymbol{c}_j$ is the "center" of the receptive field of the jth RBF node and σ_j denotes the diameter of its effective area.

The output layer, which is most often composed of simple linear nodes, processes the information from the hidden layer as follows:

$$y(\boldsymbol{x}) = \sum_{j=1}^{J} w_j \theta_j(\boldsymbol{x}). \tag{2}$$

where w_j is the connection weight from hidden node j to output node and $y(\boldsymbol{x})$ is the network output.

Generally, the design of RBF networks involves two processes: the determination of the network size and the training of the network parameters. Those are the key and crucial factors affecting the learning ability and generalization performance of RBF networks. In this study, for the performance modular neural networks, it is necessary and crucial to construct RBF networks with compact structures and considerable generalization ability.

3 The novel modular RBF (NM-RBF) neural network

This section introduces the novel modular RBF neural network, which is designed to solve complex problems by performing "divide and conquer." The topological structure of NM-RBF neural network is shown in Fig. 2, which consists of three main functional layers: task dividing layer, sub-network layer and output combining layer. In the following, the function of each layer is described in detail, respectively.

3.1 A brain-like partition method for task division

The topological organization of brain network is the foundation of its function, performance and behavior. Functional networks are constrained by structural connections [8]. Developments in the complex networks give access to study brain network organizations. Research results show that a characteristic feature of the arrangement of brain networks is the modular organization, which enables their action, perception and cognition by

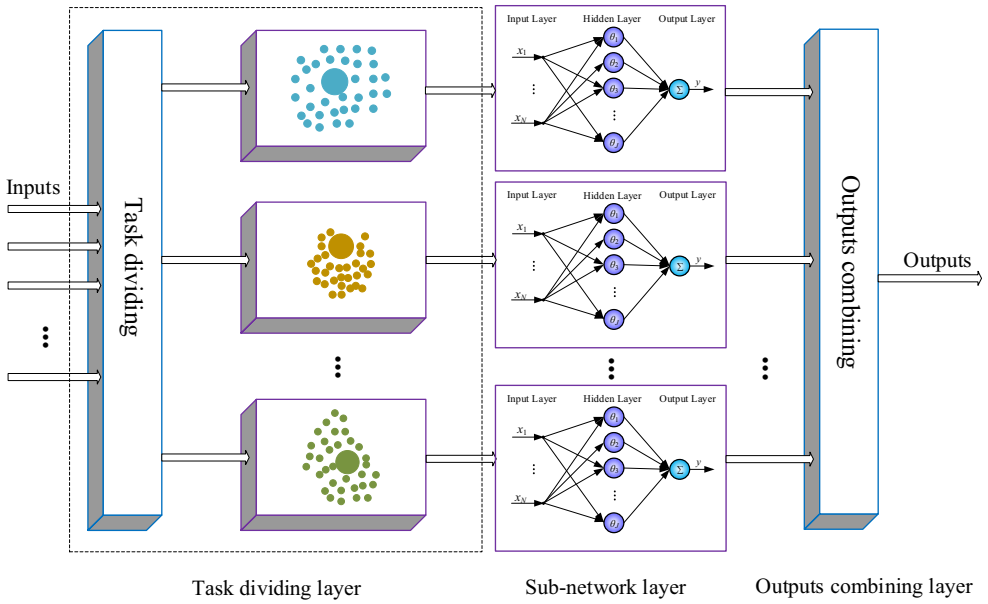

Fig. 2 Structure of modular neural network

establishing local interactions to cope with diverse environmental demands [31]. Modularity denotes that each region contains several densely interconnected nodes, and there are sparse and weak connections between nodes in different regions. Additionally, there is a highly connected hub which owns high centrality in each module.

In this study, a brain-like partition method is proposed to realize modular feature in ANNs, and it is derived from a fast density-based clustering approach proposed by Rodriguez and Laio [32]. In this clustering algorithm, cluster centers are those that have a high local density and are relatively far from any other point which has a higher local density. Likewise, hubs in each module are with high activity and centrality. Thus, by searching hubs in each module, the modular structure can be constructed and the whole training data set D can be divided to individual sub-data sets $\{D_1, D_2, \ldots, D_p, \ldots, D_p\}, (D_p \subset D)$ at the same time. Implementation of the proposed partition method is described in detail as follows.

Initially, there is neither node nor sub-network in the hidden layer. Then, when the first training sample $(\boldsymbol{x}_1, y_1)$ is added to the network, the first input sample is placed as the hub of the first sub-network:

$$\boldsymbol{h}_1 = \boldsymbol{x}_1 \tag{3}$$

$$r_1 = 1 \tag{4}$$

where $\boldsymbol{h}_1$ and r_1 are the location and effect range of the first hub, respectively; $\boldsymbol{x}_1$ is the input vector of the first training sample.

Thus, at time t, when a new training sample is added into the task dividing layer, activity of each hub is measured to determine the assignment of the current training sample. The activity vector of existing hubs is calculated as follows:

$$\mathbf{V}(t) = \left[V_1(t), \ldots, V_p(t), \ldots, V_P(t)\right] \tag{5}$$

$$V_p(t) = \mathrm{e}^{-\left\|\boldsymbol{x}_t - \boldsymbol{h}_p\right\|^2 / r_p^2} \tag{6}$$

where $V_p(t)$ represents the activity of the pth hub that is activated by training sample $\boldsymbol{x}_t$ and P is the number of sub-networks, while $\boldsymbol{h}_p$ and r_p are the location and effect range of the pth hub, respectively.

Then, find the hub with the highest activity:

$$V_{t\max}(t) = \operatorname{argmax}[V(t)] \tag{7}$$

To ensure the activity of each hub, we assume that the hub can only be activated when the activity value attains certain level, namely the activity threshold. Hence, when the highest value meets the following situation:

$$V_{t\max}(t) < \varepsilon \tag{8}$$

where $\varepsilon \in (0, 1)$ is the activity threshold and should be set initially. The relationship in (8) elucidates that the current training sample has less influence on the existing hubs. In other words, it is unable to activate any existing sub-network. Hence, a new sub-network $P + 1$ should be constructed to process this information, and the hub of the new sub-network is initialized by:

$$\boldsymbol{h}_{P+1} = \boldsymbol{x}_t \tag{9}$$

$$r_{P+1} = 1 \tag{10}$$

Contrariwise, if $V_{t\max}(t) \geq \varepsilon$, then the new input sample is assigned to the $t$max-th sub-network. Moreover, in order

to guarantee the centrality of hub, a density-based index is proposed to quantify the centrality of the new sample and the activated hub:

$$CI_t = \sum_{i=1}^{I_t} \exp(-\|\boldsymbol{h}_t - \boldsymbol{x}_i\|/r_t).d_t \tag{11}$$

$$CI_{t\max} = \sum_{i=1}^{I_{t\max}} \exp(-\|\boldsymbol{h}_{t\max} - \boldsymbol{x}_i\|/r_{t\max}).d_{t\max} \tag{12}$$

where $\boldsymbol{x}_i$ refers to the ith input sample in the $t$max-th sub-network, I_t and $I_{t\max}$ are the numbers of samples stored in each sub-network and d_t ($d_{t\max}$) is measured by the minimum distance between the tth input sample ($t$max-th hub) and any other hub.

If $CI_t \geq CI_{t\max}$, then the tth input sample has a higher centrality than the current $t$max-th hub. Consequently, the current $t$max-th hub should be replaced by the tth training sample:

$$\boldsymbol{h}_{t\max} = \boldsymbol{x}_t \tag{13}$$

$$r_{t\max} = \max\{\|\boldsymbol{h}_t - \boldsymbol{x}_i\|\} \quad i = 1, 2, \ldots, I_t \tag{14}$$

If $CI_t < CI_{t\max}$, which indicates that the $t$max-th hub is still superior, only the effect range should be adjusted as follows:

$$r_{t\max} = \max\{\|\boldsymbol{c}_{t\max} - \boldsymbol{x}_i\|\} \quad i = 1, 2, \ldots, I_{t\max} \tag{15}$$

After all the training samples are sent into the network one by one, the task is divided into different sub-tasks and each of the sub-data is stored for the following stage. It can be concluded that this brain-like partition method covers two significant characteristics: (1) autonomous partition can be achieved according to different data sets and (2) sub-networks are more isolated from each other while with denser connections within their own nodes.

Note that once a training sample comes into the network, it will be kept and stored for the following construction and training of each sub-network. Next, each sub-network is constructed according to its assigned data set.

3.2 Design of sub-networks

After the whole data are divided, corresponding sub-networks would be constructed parallelly. To guarantee the structure complexity, computing efficiency and generalization ability of the whole NM-RBF neural network, the performance of each sub-network is particularly essential. In this section, an incremental RBF network tuned by an improved second-order (ISO) algorithm [25] is utilized to design the sub-network.

Like the original ErrCor algorithm [24], the RBF node growing mechanism is based on the instantaneous learning performance. First, calculate the error vector for all the training samples at time t_1:

$$\mathbf{e}(t_1) = [e_1(t_1), e_2(t_1), \ldots, e_m(t_1), \ldots, e_M(t_1)]^{\mathrm{T}} \tag{16}$$

where M is the number of the training samples. The error between the desired output and actual output of the mth sample is calculated by:

$$e_m = y_{dm} - y_m \tag{17}$$

where y_{dm} is the desired output and y_m is the actual output under current network.

Find the kth sample with the maximum absolute value of error vector:

$$k = \operatorname{argmax}[\|\mathbf{e}(t_1)\|] \tag{18}$$

Thus, a new RBF node should be added to compensate for this current largest absolute error:

$$\boldsymbol{c}_{t_1} = \boldsymbol{x}_k \tag{19}$$

For gradient-based algorithms, it is extremely necessary to set the initial conditions as close as possible to the solutions [24]. Therefore, for sub-networks tuned by a second-order algorithm in the NM-RBF neural network, the desired output of the kth sample is assigned to the new RBF node as its output weight:

$$w_{t_1} = y_{dk} \tag{20}$$

Furthermore, radius represents the action scope of RBF node. If two RBF nodes are very close, especially that both even have a large radius, these two nodes may play the same role in this RBF network. Consequently, to avoid structure redundancy, the initial radius setting of each node is also crucial.

Thus, every time after locating a RBF node, calculate the Euclid distances between all the existing RBF nodes and the newly added one to find the minimum distance:

$$D_{\min} = \min\{dist(\boldsymbol{c}_{t_1}, \boldsymbol{c}_{j\neq t_1})\} \tag{21}$$

Deriving from the minimum distance, the initial radius parameter is initialized by:

$$\sigma_{t_1} = 0.5 D_{\min} \tag{22}$$

After adding one RBF node, the current largest error has been compensated. Then, an improved second-order algorithm is applied to tune all the network parameters, and the update rule is as follows:

$$\Delta_{t_2+1} = \Delta_{t_2} - (\mathbf{Q}_{t_2} + \mu_{t_2}\mathbf{I})^{-1}\mathbf{g}_{t_2} \tag{23}$$

where Δ refers to all the parameters that should be adjusted, $\mathbf{Q}$ denotes the quasi-Hessian matrix, μ is the learning coefficient, $\mathbf{I}$ is the identity matrix and $\mathbf{g}$ is the gradient vector.

To reduce the memory space and improve the convergence speed, the calculation of quasi-Hessian matrix is transformed to the sum of sub-matrixes $\mathbf{q}_m$:

$$\mathbf{Q} = \sum_{m=1}^{M} \mathbf{q}_m \tag{24}$$

$$\mathbf{q}_m = \mathbf{j}_m^{\mathrm{T}} \mathbf{j}_m \tag{25}$$

where $\mathbf{j}_m$ is the Jacobian row.

Similarly, the gradient vector is calculated through the sum of sub-vectors:

$$\mathbf{g} = \sum_{m=1}^{M} \boldsymbol{\eta}_m \tag{26}$$

$$\boldsymbol{\eta}_m = \mathbf{j}_m^{\mathrm{T}} \mathbf{e}_m \tag{27}$$

Thus, the calculations of the quasi-Hessian matrix and gradient vector are simplified into the calculations of the Jacobian row. Since there are three sorts of parameters in this RBF network, the Jacobian row is calculated by:

$$\mathbf{j}_m = \left[\frac{\partial e_m}{\partial c_{1j}}, \frac{\partial e_m}{\partial c_{2j}}, \ldots, \frac{\partial e_m}{\partial c_{nj}}, \frac{\partial e_m}{\partial \sigma_1}, \ldots, \frac{\partial e_m}{\partial \sigma_j}, \ldots, \frac{\partial e_m}{\partial w_1}, \ldots, \frac{\partial e_m}{\partial w_j} \right] \tag{28}$$

where n is the dimension of the input vector and j is the number of hidden nodes of the current network.

Integrating Eqs. (1), (2) and (23), the Jacobian row elements for the mth input sample in (28) can be rewritten as:

$$\begin{aligned} \frac{\partial e_m}{\partial c_{nj}} &= -\frac{\partial y_m}{\partial c_{nj}} \\ &= -\frac{\partial y_m}{\partial \theta_j(\boldsymbol{x}_m)} \frac{\partial \theta_j(\boldsymbol{x}_m)}{\partial c_{nj}} = -\frac{w_j \theta_j(\boldsymbol{x}_m)(x_{nm} - c_{nj})}{\sigma_j^2} \end{aligned} \tag{29}$$

$$\begin{aligned} \frac{\partial e_m}{\partial \sigma_j} &= -\frac{\partial y_m}{\partial \sigma_j} = -\frac{\partial y_m}{\partial \theta_j(\boldsymbol{x}_m)} \frac{\partial \theta_j(\boldsymbol{x}_m)}{\partial \sigma_j} \\ &= -\frac{w_j \theta_j(\boldsymbol{x}_m) \left\| \boldsymbol{x}_m - \boldsymbol{c}_j \right\|^2}{\sigma_j^3} \end{aligned} \tag{30}$$

$$\frac{\partial e_m}{\partial w_j} = -\frac{\partial y_m}{\partial w_j} = -\theta_j(\boldsymbol{x}_m) \tag{31}$$

The mean squared error (MSE) is utilized to evaluate the training process:

$$\mathrm{MSE}(t_2) = \frac{1}{M} \sum_{t_2=1}^{M} (y_{t_2} - y_{dt_2})^2 \tag{32}$$

It should be noted that a counter is needed to avoid dead loop. The MSE value decreases with the growing of RBF nodes and adjusting of parameters. Once the MSE value reaches the desired level, the construction and tuning process of the sub-network are completed.

After the task division and parallel design of sub-networks, the construction of NM-RBF neural network is finished, and the detailed process is explained in Table 1.

3.3 Outputs combining strategy

The output combining layer is not available during the construction process of NM-RBF neural network and only works in the network testing stage. In this study, the task is divided into different independent sub-tasks. Therefore, a competitive strategy is applied to the output layer; namely, only the selected sub-network contributes to the final output. Once a testing sample $(\boldsymbol{x}_v, y_{dv})$ is presented, activate all the hubs by this current sample, and calculate their activities by Eq. (6) to find the one with the highest activity:

$$h\mathrm{max} = \mathrm{argmax}[V(\boldsymbol{x}_v)] \tag{33}$$

In consequence, the present information is processed by the $h$max-th sub-network:

$$y_{h\mathrm{max}} = \sum_{j=1}^{J_{h\mathrm{max}}} w_j \mathrm{e}^{-\left\| \boldsymbol{x}_v - \boldsymbol{c}_j \right\|^2 / \sigma_j^2} \tag{34}$$

where $J_{h\mathrm{max}}$ is the hidden node number in the $h$max-th sub-network and $y_{h\mathrm{max}}$ is its output. Then, the final output of NM-RBF neural network is achieved by:

$$y_v = y_{h\mathrm{max}} \tag{35}$$

4 Experiments and analysis

In this section, four benchmark experiments those are different in problem type, input dimension and sample size are presented to assess and demonstrate the effectiveness and outperformance of the proposed NM-RBF neural network, including Mackey–Glass time series prediction, sin$E$ function approximation, nonlinear dynamic system identification and real-world problems. For simplicity and consistency, the initialization of the only parameter in all these experiments is the same: $\varepsilon = 0.3$. All the simulations were programmed with MATLAB R2012a and were run on a PC with a clock speed 3.4 GHz and 4 GB RAM, under a Microsoft Windows 8.0 environment.

4.1 Mackey–Glass time series prediction

The Mackey–Glass time series prediction has been frequently used to assess the performance of ANNs and other intelligent computing methods on dynamic nonlinear system modeling problems. In this study, the time series is generated from the following time-delay differential system:

Table 1 Construction of NM-RBF neural network

% Task division process	% Design of sub-networks
for i = 1: I % for all the training samples	**for** j = 1: $NMAX$ % for all the RBF units
while i = 1	Find the location of the largest residual error by (18)
Construct the first sub-network, and the initial parameters of this hub are set by (3), (4)	Add a new RBF unit and set the initial parameters by (19), (20) and (22)
end	**for** $iter$ =2: max_iter
Use the current input vector to activate the existing hubs by (5), (6)	Calculate Jacobian row using (28)–(31)
Find the hub with the highest activity (7)	Calculate quasi-Hessian matrix and gradient vector by (24)–(27)
if $V_{max}(t) < \varepsilon$	Update all the parameters by (23)
Add a new module and the corresponding parameters of the hub is set by (9), (10)	Calculate the current MSE value using (32)
else	**if** MSE($iter$) < desired training error
Compare the centrality between the new input vector and the hub, and determine the new hub by (11)–(15)	**break**
end	**end**
end	**end**

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1+x^n(t-\tau)} + bx(t) \tag{36}$$

where $n = 10$, $a = 0.2$, $b = -0.1$, $\tau = 17$ and $x(0) = 1.2$. The objective of this experiment is to predict the value $x(t + R)$ from $\{x(t), x(t - \Delta t) \ldots x(t - (n1 - 1)\Delta t)\}$, and $R = \Delta t = 6$, $n1 = 4$. Thus, the prediction model is given by

$$x(t+6) = g(x(t), x(t-6), x(t-12), x(t-18)) \tag{37}$$

During this simulation, 1000 data samples were extracted by Runge–Kutta method. The first 500 sets were used as the training data, while the other 500 were employed to evaluate the performance of the proposed NM-RBF neural network. When tackling this problem, only one sub-network was used for NM-RBF neural network. Figure 3 shows the testing results of the proposed algorithm, and the testing errors are exhibited in Fig. 4. Both two figures illustrate the good prediction capability of the proposed NM-RBF neural network on Mackey–Glass time series prediction problem.

To demonstrate the effectiveness and outperformance of the proposed algorithm, the results are compared with other well-known single networks and modular networks—GGAP [28], SVR [26], ErrCor [24], APSO-SORBF [33], CNNE [10] and OSAMNN [17]—in Table 2. On the one hand, when comparing with GGAP, SVR, ErrCor and APSO-SORBF, NM-RBF neural network with mere one sub-network yields a more compact structure and higher prediction accuracy. Although the training time of NM-RBF neural network is longer than SVR, the time of SVR needed to find the best parameters is not considered. Additionally, the NM-RBF has significant advantages on structure size and accuracy, which also can demonstrate the desirable performance of the sub-network. On the other hand, when comparing with other modular neural networks, both CNNE and OSAMNN need a more complex structure to realize this prediction. In conclusion, performance of NM-RBF and its sub-network has both been verified by this benchmark problem.

4.2 Sin*E* function approximation

After verifying the performance of the sub-network in the first experiment, the proposed NM-RBF neural network is utilized to approximate a rapidly changing function—sin$E$ function, which is given as follows:

$$y = 0.8\exp(-0.2x)\sin(10x) \tag{38}$$

In this problem, 3000 data points were chosen as the training sample, with the value of the input x randomly distributed in the interval [0, 10]. The size of the testing sample set was 1500, with the input distributed in the same range.

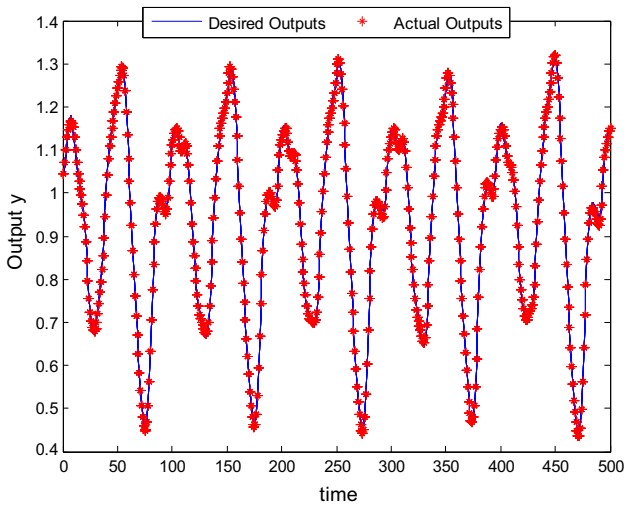

Fig. 3 Prediction results of NM-RBF neural network

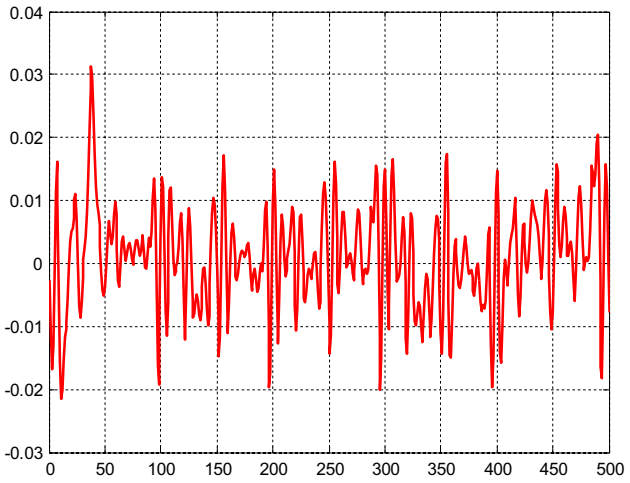

Fig. 4 Prediction errors of NM-RBF neural network

As shown in Fig. 5, through the partition method, the whole 3000 training samples are partitioned to 6 sets—798, 351, 463, 574, 605 and 209. Then, correspondingly, six sub-networks are constructed to handle the assigned data set, which are shown in Fig. 6. Obviously, Figs. 5 and 6 reveal the considerable partition performance of NM-RBF. In addition, the approximation results and the approximation errors of NM-RBF are presented in Figs. 7 and 8. As shown in Figs. 7 and 8, the proposed algorithm is able to approximate this function precisely.

Furthermore, in Table 3, the NM-RBF neural network is also compared with some other algorithms on training time, training RMSE, testing RMSE, number of sub-networks and hidden neurons. Although NM-RBF needs more sub-networks than other algorithms on this approximation problem, it is obvious that both NM-RBF and ErrCor have significant advantages over approximation accuracy and network size than GGAP, SVR and OSAMNN. Additionally, it should be noted that both GGAP and SVR need many trials to find the "best" initial parameters. On the other side, when comparing NM-RBF with ErrCor, ErrCor yields a more compact structure; nevertheless, the superiority of training time, training RMSE and testing RMSE make the NM-RBF more useful for those issues which require computational efficiency. Overall, this experiment exactly reveals the "divide-and-conquer" ability and associated performance improvements in the proposed NM-RBF neural network.

4.3 Nonlinear dynamic system identification

Nonlinear dynamic systems have always been used to evaluate the identification ability of different algorithms. In this study, the utilized equation is described as follows:

$$y(t+1) = \frac{y(t)y(t-1)[y(t)+2.5]}{1+y^2(t)+y^2(t-1)} + u(t), \tag{39}$$

Table 2 Performance comparison of different algorithms on Mackey–Glass prediction

Algorithms	Training time (s)	Training RMSE	Testing RMSE	Number of sub-networks	Number of hidden neurons
GAP-RBF	24.326[a]	0.0344[a]	0.0367[a]	1[a]	13[a]
SVR	0.169	0.0610	0.0638	1	17
ErrCor	6.034	0.00108	0.0098	1	6
APSO-SORBF	832.7[a]	–	0.0135[a]	1[a]	11[a]
CNNE	–	0.009[a]	0.0090[a]	8[a]	56[a]
OSAMNN	6.052	0.0091	0.0091	6	30
NM-RBF	6.416	0.0081	0.0081	1	6

–: the results are not listed in the original literature

[a]The results are the same as the original literature

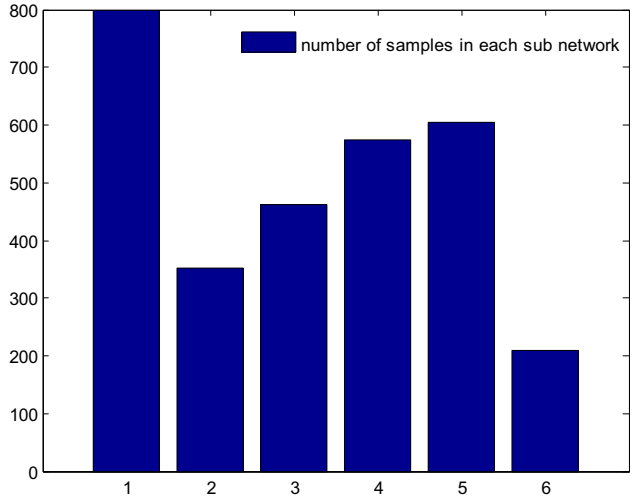

Fig. 5 Distributions of training samples and hidden neurons

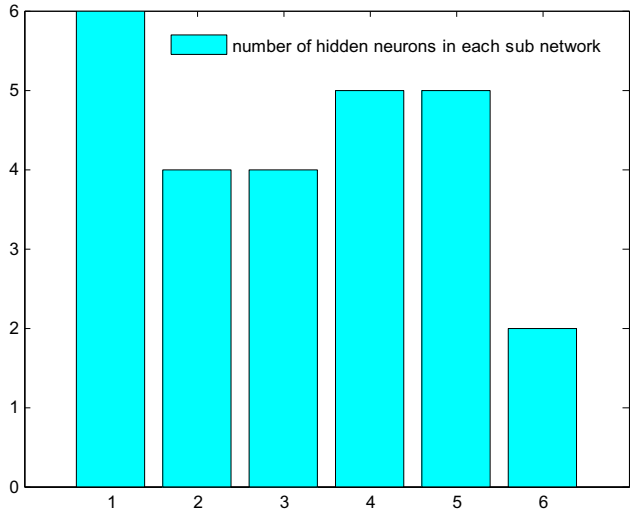

Fig. 6 Distributions of hidden neurons

where $u(t) = \sin(2\pi t/25), y(0) = 0, y(1) = 0$.

The model is described by the following equation:

$$\hat{y}(t+1) = f(y(t), y(t-1), u(t)). \tag{40}$$

There are three inputs ($y(t)$, $y(t-1)$, $u(t)$) and one output $y(t+1)$ in this model. In this experiment, the observations from epoch $t = 1$ to $t = 5000$ were selected as training samples, while the observations from epoch $t = 5001$ to $t = 5200$ were chosen to test the proposed algorithm.

Comparisons between the desired outputs and outputs of the NM-RBF are exhibited in Figs. 9 and 10, from which we can see the considerable identification performance of the proposed NM-RBF. Besides, some indices including training time, training RMSE, testing RMSE, number of

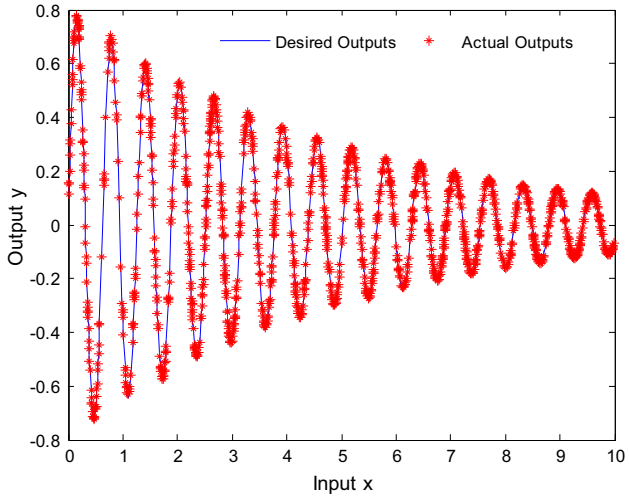

Fig. 7 Approximation results of NM-RBF neural network

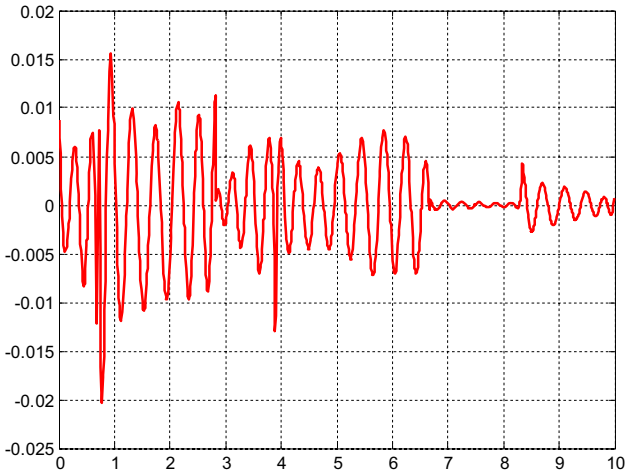

Fig. 8 Approximation errors of NM-RBF neural network

sub-networks and network size are listed in Table 4 to evaluate different methods: GAP, SVR, ErrCor, OSAMNN and NM-RBF. Regardless of the time for seeking appropriate initial parameters, SVR has the shortest training time, but its structure size is overlarge. GAP and OSAMNN are able to obtain nearly the same identification performance on this issue. Furthermore, ErrCor and NM-RBF can achieve higher identifying accuracy with more compact structures than other techniques. Nevertheless, the "divide-and-conquer" characteristic of NM-RBF neural network makes it yield a higher accuracy during a shorter time than ErrCor.

4.4 Real-world problems

After the above three simulations, in this section, the NM-RBF neural network is tested on a series of benchmark

Table 3 Performance comparison of different algorithms on sin$E$ function

Algorithms	Training time (s)	Training RMSE	Testing RMSE	Number of sub-networks	Number of hidden neurons
GAP	24.808	0.0261	0.0269	1	45
SVR	0.759	0.0346	0.0361	1	500
ErrCor	68.64	0.0057	0.0056	1	20
OSAMNN	15.689	0.0217	0.0240	4	28
NM-RBF	22.93	0.0041	0.0045	6	26

regression problems from UCI database, such as Abalone, Auto-MPG, Boston Housing and Delta-Elevators. The specifications of these benchmark data sets are shown in Table 5, with multiple input dimensions and hundreds or thousands of samples.

In each problem, the sample data are randomly divided into two sets: 70% of the data are as the training samples, and the remaining are used as testing data. Besides, all the inputs are normalized into the range $[-1, 1]$ while the normalized outputs are in the range [0, 1]. Detailed comparison results between NM-RBF and other algorithms—GAP [27], SVR [26], ErrCor [24] and OSAMNN [17]—on each problem are presented in Tables 6, 7, 8 and 9.

The first problem is to address the prediction of Abalone age based on eight features. As shown in Table 6, although SVR is able to complete the training process fast, its network structure is too large. NM-RBF and ErrCor can obtain much more compact structures in comparison with other algorithms, but NM-RBF has an advantage on the prediction accuracy.

When predicting the fuel consumption of different models of cars in the Auto-MPG problem, there is only one sub-network constructed in the NM-RBF. Results in Table 7 show that OSAMNN achieves the highest accuracy, but it should be noted that its structure is too large, which may not be useful for practical applications.

As for Boston Housing and Delta-Elevators, OSAMNN needs too many sub-networks to solve these two problems due to its unsuitable partition method. On the contrary, all the results reveal the desirable partition ability of the proposed NM-RBF. Furthermore, in comparison with other single RBF networks, the "divide-and-conquer" strategy makes the NM-RBF obtain high accuracy with fast learning speed.

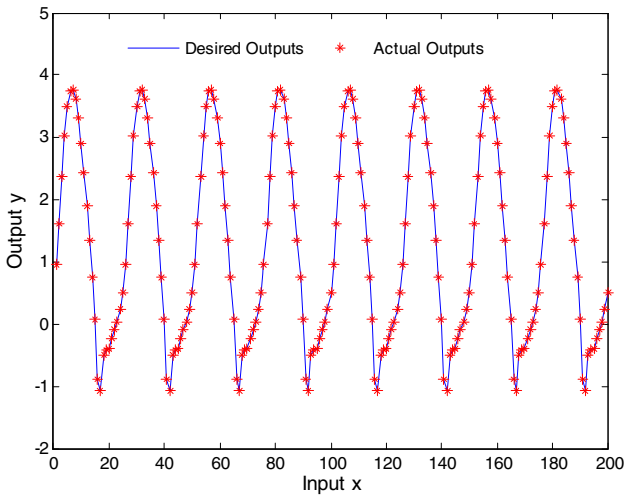

Fig. 9 Testing results of NM-RBF neural network

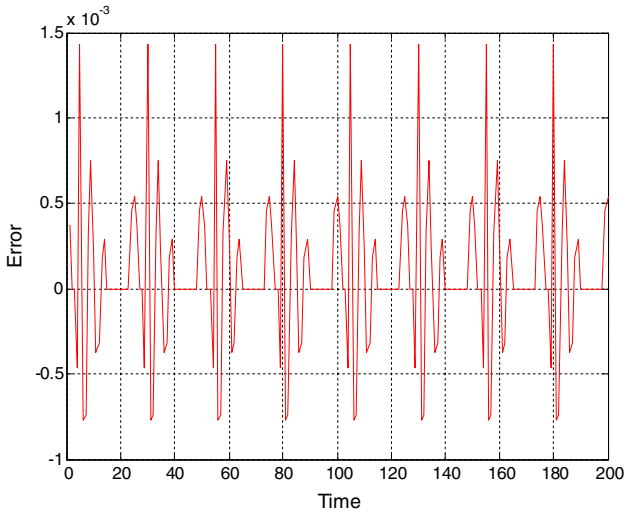

Fig. 10 Testing errors of NM-RBF neural network

5 Conclusion

This paper develops a novel algorithm for constructing modular neural networks. This algorithm works by dividing the training samples to heterogeneous subsets through a density-based brain-like partition method. Then, each subset is processed by an RBF network, which is designed based on an incremental mechanism and an improved second-order algorithm. This algorithm takes advantages of both the "divide-and-conquer" characteristic and the effectiveness of RBF networks. There are three reasons

Table 4 Performance comparison of different algorithms on system identification

Algorithms	Training time (s)	Training RMSE	Testing RMSE	Number of sub-networks	Number of hidden neurons
GAP	27.203	0.0285	0.0293	1	42
SVR	9.837	0.1350	0.1700	1	416
ErrCor	71.921	0.0082	0.0088	1	20
OSAMNN	17.410	0.0237	0.0306	10	40
NM-RBF	16.109	0.0011	0.0015	9	23

Table 5 Specifications of data sets

Real-world problem	Training patterns	Testing patterns	Input dimensions
Abalone	3000	1177	8
Auto-MPG	320	78	7
Boston Housing	481	25	13
Delta-Elevator	4000	5517	6

why the NM-RBF neural network could outperform some other popular algorithms:

1. The NM-RBF neural network takes a great advantage of the human brain when tackling complex problems—divide and conquer. Through dividing a large complex issue to different individual simple problems, the network can handle the problem better and faster.
2. Unlike some other modular neural networks, this study does not only apply a conventional clustering approach to achieve the task decomposition. In contrary, the density-based partition method for NM-RBF neural network can illustrate the feature of human brain.
3. Last but also important, performance of sub-networks is an essential guarantee of the entire modular neural network. In this study, the compact sub-networks with fast learning speed and considerable generalization performance ensure the parsimonious and generalization of the whole NM-RBF neural network.

Table 6 Performance comparison of different algorithms on Abalone age prediction

Algorithms	Training time	Training RMSE	Testing RMSE	Number of sub-networks	Number of hidden neurons
GAP	83.784[a]	0.0963[a]	0.0966[a]	1	23.62[a]
SVR	0.445[a]	0.0759[a]	0.0846[a]	1	310[a]
ErrCor	5.967	0.0758	0.0792	1	4
OSAMNN	10.025	0.0847	0.0847	10	50
NM-RBF	8.625	0.0733	0.0747	2	4

[a]The results are the same as the original literature

Table 7 Performance comparison of different algorithms on Auto-MPG

Algorithms	Training time	Training RMSE	Testing RMSE	Number of sub-networks	Number of hidden neurons
GAP-RBF	0.4520[a]	0.1144[a]	0.1404[a]	1	3.12[a]
SVR	0.0210[a]	0.0465[a]	0.0785[a]	1	96[a]
ErrCor	0.5030	0.0671	0.0792	1	3
OSAMNN	1.1637	0.0614	0.0614	18	90
NM-RBF	0.889	0.0634	0.0739	1	2

[a]The results are the same as the original literature

Table 8 Performance comparison of different algorithms on Boston Housing

Algorithms	Training time	Training RMSE	Testing RMSE	Number of sub-networks	Number of hidden neurons
GAP-RBF	1.2399[a]	0.1507[a]	0.1418[a]	1	3.5[a]
SVR	–	–	0.1155[a]	1[a]	47[a]
ErrCor	5.9672	0.0989	0.0989	1	4
OSAMNN	3.4273	0.0950	0.0950	26	130
NM-RBF	1.442	0.0826	0.0826	2	4

–: the results are not listed in the original literature

[a]The results are the same as the original literature

Table 9 Performance comparison of different algorithms on Delta-Elevator

Algorithms	Training Time	Training RMSE	Testing RMSE	Number of sub-networks	Number of hidden neurons
GAP-RBF	50.424	0.0944	0.0930	1	6
SVR	–	–	0.0603[a]	1	261[a]
ErrCor	31.3576	0.0534	0.0538	1	3
OSAMNN	383.219	0.0528	0.0528	51	255
NM-RBF	26.262	0.0530	0.0542	2	4

–: the results are not listed in the original literature

[a]The results are the same as the original literature

Comparison results on a series of benchmark problems have demonstrated the aforementioned outperformance of the NM-RBF neural network. Consequently, this study can be considered as a previous step for practical applications. In our future work, the NM-RBF neural network will be applied to some industrial applications.

Funding This study was funded by the National Natural Science Foundation of China (Grants 61533002 and 61603009), Beijing Science and Technology Project (Grant Z1511000001315010) and "Rixin Scientist" Foundation of Beijing University of Technology (Grant 2017-RX (1)-04).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Zhang Y, Chai TY, Li Z, Yang C (2012) Modeling and monitoring of dynamic processes. IEEE Trans Neural Netw Learn Syst 23(2):277–284
2. Pukish MS, Rozycki P, Wilamowski BM (2015) Polynet: a polynomial-based learning machine for universal approximation. IEEE Trans Ind Inf 11(3):708–716
3. Han HG, Zhou WD, Qiao JF, Feng G (2015) A direct self-constructing neural controller design for a class of nonlinear systems. IEEE Trans Neural Netw Learn Syst 26(6):1312–1322
4. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
5. Wilamowski BM (2009) Neural network architectures and learning algorithms. IEEE Ind Electron Mag 3(4):56–63
6. Hoori AO, Motai Y (2017) Multicolumn RBF network. IEEE Trans Neural Netw Learn Syst 29(4):766–778
7. Bullmore E, Sporns O (2009) Complex brain networks: graph theoretical analysis of structural and functional systems. Nat Rev Neurosci 10(3):186–198
8. Park HJ, Friston K (2013) Structural and functional brain networks: from connections to cognition. Science 342(6158):1238411
9. Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE (1991) Adaptive mixtures of local experts. Neural Comput 3(1):79–87
10. Islam MM, Yao X, Murase K (2003) A constructive algorithm for training cooperative neural network ensembles. IEEE Trans Neural Netw 14(4):820–834
11. Tseng HC, Almogahed B (2009) Modular neural networks with applications to pattern profiling problems. Neurocomputing 72(10–12):2093–2100
12. Tokunaga K, Furukawa T (2009) Modular network SOM. Neural Netw 22(1):82–90
13. Goltsev A, Gritsenko V (2009) Modular neural networks with Hebbian learning rule. Neurocomputing 72(10):2477–2482
14. Cimino MG, Pedrycz W, Lazzerini B, Marcelloni F (2009) Using multilayer perceptrons as receptive fields in the design of neural networks. Neurocomputing 72(10):2536–2548
15. Ding Y, Feng Q, Wang T, Fu X (2014) A modular neural network architecture with concept. Neurocomputing 125:3–6
16. Yang J, Zeng X, Zhong S, Wu S (2013) Effective neural network ensemble approach for improving generalization performance. IEEE Trans Neural Netw Learn Syst 24(6):878–887
17. Qiao JF, Zhang ZZ, Bo YC (2014) An online self-adaptive modular neural network for time-varying systems. Neurocomputing 125:7–16
18. Wang XL, Chen YY, Zhao H, Lu BL (2014) Parallelized extreme learning machine ensemble based on min–max modular network. Neurocomputing 128:31–41
19. Valdez F, Melin P, Castillo O (2014) Modular neural networks architecture optimization with a new nature inspired method using a fuzzy combination of particle swarm optimization and genetic algorithms. Inf Sci 270:143–153
20. Yoo SH, Oh SK, Pedrycz W (2015) Optimized face recognition algorithm using radial basis function neural networks and its practical applications. Neural Netw 69:111–125
21. Wang H, Zhang N, Creput JC (2017) A massively parallel neural network approach to large-scale Euclidean traveling salesman problems. Neurocomputing 240:137–151
22. Sanchez D, Melin P (2014) Optimization of modular granular neural networks using hierarchical genetic algorithms for human recognition using the ear biometric measure. Eng Appl Artif Intell 27:41–56
23. Mozaffari A, Scott KA, Chenouri S et al (2017) A modular ridge randomized neural network with differential evolutionary distributor applied to the estimation of sea ice thickness. Soft Comput 21(16):4635–4659
24. Yu H, Reiner PD, Xie TT, Bartczak T, Wilamowski BM (2014) An incremental design of radial basis function networks. IEEE Trans Neural Netw Learn Syst 25(10):1793–1803
25. Xie TT, Yu H, Hewlett J, Rozycki P, Wilamowski BM (2012) Fast and efficient second-order method for training radial basis function networks. IEEE Trans Neural Netw Learn Syst 23(4):609–619
26. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
27. Huang GB, Saratchandran P, Sundararajan N (2004) An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. IEEE Trans Syst Man Cybern B Cybern 34(6):2284–2292

28. Huang GB, Saratchandran P, Sundararajan N (2005) A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Trans Neural Netw 16(1):57–67
29. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501
30. Hunter D, Yu H, Pukish MS III, Kolbusz J, Wilamowski BM (2012) Selection of proper neural network sizes and architectures—a comparative study. IEEE Trans Ind Inf 8(2):228–240
31. Hilgetag CC, Hutt MT (2014) Hierarchical modular brain connectivity is a stretch for criticality. Trends Cogn Sci 18(3):114–115
32. Rodriguez A, Laio A (2014) Clustering by fast search and find of density peaks. Science 344(6191):1492–1496
33. Han HG, Lu W, Hou Y, Qiao JF (2018) An adaptive-PSO-based self-organizing RBF neural network. IEEE Trans Neural Netw Learn Syst 29(1):104–117