# Dynamic Self-organized Learning for Optimizing the Complexity Growth of Radial Basis Function Neural Networks

**Somwang Arisariyawong and Siam Charoenseang**[*]
Mechanical Engineering Department, Faculty of Engineering,
Srinakharinwirot University, Ongkharak, Nakornayok, 26120, THAILAND
E-mail: somwang@fibo.kmutt.ac.th
Center of Operation for FIeld roBOtics Development (FIBO)[*]
King Mongkut's University of Technology Thonburi
Suksawasd, 48 Bangmod, Bangkok, 10140, THAILAND
E-mail: siam@fibo.kmutt.ac.th

**Abstract**
This paper proposes a framework of automatically exploring the optimal size of a radial basis function (RBF) neural network. A dynamic self-organized learning algorithm is presented to adapt the structure of the network. The algorithm generates a new hidden unit based on the steady state error of network and the nearest distance from input data to the center of hidden unit. Furthermore, it also detects and removes any insignificant contributing hidden units. For optimizing the complexity growth of RBF neural network, the growing and pruning are combined during adaptation of RBF neural network structure. The examples of nonlinear dynamical system modeling are presented to illustrate the performance of the proposed algorithm.
**Keywords:** Self-organized Learning, Learning Algorithm, Radial Basis Function Networks, Function Estimation, Convergence.

## 1. Introduction

Neural network research has gained increasing attention in recent years. Researchers from diverse areas, such as neuroscience, engineering, and computer science, are interested in recreating the computational structures of the human brain. One of the most important models is the feedforward artificial neural network. This kind of network is used to model some unknown system or process having an unambiguous input/output mapping. The network size, which is often measured by the number of hidden units in a single hidden layer network, reflects the capacity of the network to approximate an arbitrary function. A continuing question in the research of neural networks is what size of a neural network is required to solve a specific problem. If the training starts with a small network, it is possible that learning cannot be achieved. On the other hand, if a large network is used, the learning process can be very slow and/or overfitting may occur. Hence, there is no standard on how one can implement a network which will solve a specific problem. Generally, a trial and error approach is adopted to find the suitable network size for a given problem. During a trial and error period, the searching will be terminated as soon as a satisfied performance is achieved.

Radial basis function (RBF) neural networks have been widely used for nonlinear function approximation. The original RBF method has been traditionally used for strict multivariable function interpolation [1] and it requires as many RBF units as data points. This is rarely practical because the large size of data points is usually required to fit the data relationship. D. Broomhead and D. Lowe [2] removed this strict interpolation restriction and provided a neural network architecture where the number of RBF neurons can be far less than the data points. Compared with other types of neural networks like backpropagation feedforward networks, the RBF neural network requires less computation time for learning [3] and also has a more compact topology [4]. J. Moody and C. Darken [3] proposed learning algorithm of RBF networks by employing unsupervised produces for selecting a fixed number of radial basis function centers. The algorithm offers good computational efficiency and convergence speed.

J. Platt [5] proposed a sequential or online learning algorithm for resource allocating network (RAN). In Platt's algorithm, hidden neurons are added based on the 'novelty' of the new data and the weights are also estimated using the well-known least mean square (LMS) algorithm. A new pattern is considered novel if the error between the network output and desired output is large. If no additional hidden neuron is added, the parameters of the existing hidden neurons, such as the centers, widths, and weights, are updated. V. Kadirkamanathan and M. Niranjan [6] interpreted Platt's RAN from a function approach and improved RAN by using an extended Kalman filter (EKF) instead of the LMS to estimate the network parameters. Their network, called RANEKF, is more compact and has faster convergence than RAN. S. Arisariyawong and S. Charoenseang [7] proposed the self-organized learning for growing of complexity of RBF neural networks. Their algorithm generates a new hidden unit based on the steady state error of network and the nearest distance from input data to the center of hidden unit. They showed that the self-organized learning algorithm for RBF neural networks yields good performance in terms of convergence and accuracy compared with conventional multilayer feedforward neural network.

However, one drawback of RAN [5], RANEKF [6] and self-organized learning [7] is that once a hidden unit is created, it will never be removed. This leads the network to produce some hidden units which are initially active but those end up to contribute a little to the network output. Furthermore, pruning becomes imperative for the identification of nonlinear systems with changing dynamics. The network is without pruning will result in numerous inactive hidden neurons being present. If inactive hidden units can be detected and removed during learning, a more parsimonious network topology can then be realized. Also, the problems of overparameterisation could be avoided when the neural networks are employed for control.

In this paper, we propose the learning algorithm to optimize the complexity growth of the RBF neural networks. It comprises growing and pruning algorithms for adapting the structure of the network.

## 2. Model description

In the RBF neural network model, the $j^{\text{th}}$ output, $y_j(i)$, is given by the following equation:

$$y_j(i) = \sum_{k=1}^{K} w_{jk} \Phi_k \left[ x(i), c_k, \sigma_k \right] \qquad j = 1, 2, \ldots, n \quad i = 1, 2, \ldots, N \tag{1}$$

where $K$ is the number of RBFs used, and $c_k \in R^m, \sigma_k \in R^m$, are the center value vector and the width value vector of RBF, respectively. These vectors are defined as:

$$\begin{aligned} c_k &= \left[ c_{k1} \;\; c_{k2} \;\; \ldots \;\; c_{km} \right]^T \in R^m, \quad k = 1, \ldots\ldots, K \\ \sigma_k &= \left[ \sigma_{k1} \;\; \sigma_{k2} \;\; \ldots \;\; \sigma_{\text{km}} \right]^T \in R^m, \quad k = 1, \ldots\ldots, K \end{aligned} \tag{2}$$

Also, $\{ w_{jk} | k = 1, 2, \ldots, K \}$ are the weights of RBFs connected with the $j^{\text{th}}$ output. Figure 1 shows the structure of a RBF neural network.

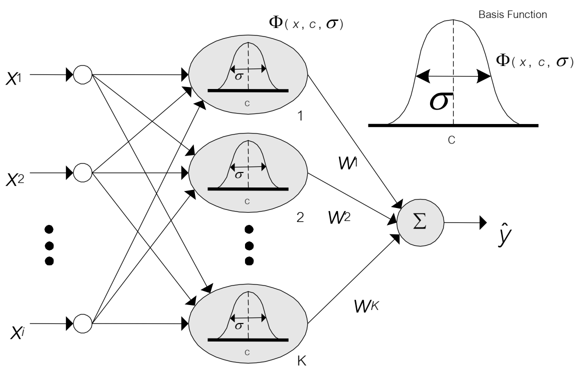


Fig. 1 Structure of RBF neural network

The RBF neural network representation can be implemented in the form of a two-layered network. For a given set of centers, the first layer performs a fixed nonlinear transformation which maps the input space onto a new space. Each term $\Phi_k(.)$ forms the activation function in a unit of the hidden layer. The output layer then implements a linear combination of this new space.

$$\Phi_k(x, c_k, \sigma_k) = \prod_{i=m}^{m} \phi_{ki}(x_i, c_{ki}, \sigma_{ki}) \tag{3}$$

Moreover, the most popular choice for $\phi(.)$ is the Gaussian form defined as

$$\phi_{ki}(x_i, c_{ki}, \sigma_{ki}) = \exp\left[ -\left( x_i - c_{ki} \right)^2 / \sigma_{ki} \right] \tag{4}$$

In this case, the $j^{\text{th}}$ output in equation (1) becomes

$$y_i(i) = \sum_{k=1}^{k} w_{jk} \exp\left\{ -\sum_{i=1}^{m} \left[ -(x_i - c_{ki})^2 / \sigma_{ki} \right] \right\} \tag{5}$$

Define the following sets:

$$\begin{aligned} C &= \left\{ c_{ki} \middle| k = 1, 2, \ldots, K, \quad i = 1, 2, \ldots, m \right\} \\ \Theta &= \left\{ \sigma_{ki} \middle| k = 1, 2, \ldots, K, \quad i = 1, 2, \ldots, m \right\} \\ W &= \left\{ w_{jk} \middle| k = 1, 2, \ldots, K, \; j = 1, 2, \ldots, n \right\} \end{aligned} \tag{6}$$

Given the $N$ input/output data and the specified root mean squared error $\varepsilon > 0$, the identification problem can be considered as obtaining the minimal number $K$ of RBFs and the optimal solution $\left( C^*, \Theta^*, W^* \right)$ which satisfies the inequality $E\left( C^*, \Theta^*, W^* \right) < \varepsilon$, where $E$ is the optimization criteria defined as:

$$E\left( C, \Theta, W \right) = 0.5 \sum_{k=1}^{K} \sum_{p=1}^{N} \left[ y_j^d(p) - y_j(p) \right]^2 \tag{7}$$

where $y_j(p)$ and $y_j^d(p)$ are the $j^{\text{th}}$ model output and the $j^{\text{th}}$ desired output, respectively, for the input of the training set.

## 3. The learning scheme

In order to solve the identification problem, the gradients of $E$ are derived with respect to the parameters $c_{ki}$, $\sigma_{ki}$, and $w_{jk}$. Assuming that the number of RBFs, K, is fixed,

$$\begin{aligned} \frac{\partial E}{\partial c_{ki}} &= \left( -\frac{2}{\sigma_{ki}^2} \right) \sum_{p=1}^{N} \sum_{j=1}^{n} \left[ w_{jk} \cdot \left[ y_j^d(p) - y_j(p) \right] \cdot \Phi_k \left[ x(p), c_k, \sigma_k \right] \cdot \left[ x_i(p) - c_{ki} \right] \right] \\ \frac{\partial E}{\partial \sigma_{ki}} &= \left( -\frac{2}{\sigma_{ki}^3} \right) \sum_{p=1}^{N} \sum_{j=1}^{n} \left[ w_{jk} \cdot \left[ y_j^d(p) - y_j(p) \right] \cdot \Phi_k \left[ x(p), c_k, \sigma_k \right] \cdot \left[ x_i(p) - c_{ki} \right]^2 \right] \\ \frac{\partial E}{\partial w_{ki}} &= -\sum^{N} \left[ y_j^d(p) - y_j(p) \right] \cdot \Phi_k \left[ x(p), c_k, \sigma_k \right] \\ j &= 1, \ldots, n \qquad k = 1, \ldots, K \end{aligned} \tag{8}$$

By using the gradients in Equation (8), the identification problem can be solved for a fixed number of neurons by using appropriate gradient methods such as the steepest descent method. For *N* input/output data, the sets of *C(h),* $\sigma(h)$, and *W(h)* at iteration *h* are computed from the sets of *C(h-1),* $\sigma(h-1)$, and *W(h-1)* by the following learning rule

$$\begin{aligned} c_{ki}(h) &= c_{ki}(h-1) - \alpha \cdot \frac{\partial E}{\partial c_{ki}} \\ \sigma_{ki}(h) &= \sigma_{ki}(h-1) - \alpha \cdot \frac{\partial E}{\partial \sigma_{ki}} \\ w_{jk}(h) &= w_{jk}(h-1) - \alpha \cdot \frac{\partial E}{\partial w_{jk}} \end{aligned} \tag{9}$$

*I = 1, ..., m, k=1, ..., K, j=1, ..., n* and *h* is the iteration number.

## 4. Dynamic Self-organized Learning Algorithm

Initially, the network begins with two hidden units. The following three criteria decide whether an input *x(q)* should be added to the hidden-layer of the network [7],

$$\begin{aligned} & e_h = y(h) - \hat{y}(h) > e_{\min} \\ & R(h) = 100 \cdot \left\| \left[ \frac{E(h) - E(h-1)}{E(h-1)} \right] \right\| < \varepsilon_R \\ & \| x(q) - c_{nr} \| > \varepsilon_n \end{aligned} \tag{10}$$

where *R(h)* is the percentaged decreasing rate of error at iteration *h*. $c_{nr}$ is center of a hidden unit whose distance from *x(q)* is the nearest among those of all of the other hidden unit centers. $e_{min}$, $\varepsilon_n$, and $\varepsilon_R$ are thresholds which are selected appropriately. If these criteria are found during the training process, a new basis function is generated. A new basis function is generated in such a way that its center is located at the point where the maximum of absolute inference error occurs in the input space.

When a new basis function is added to the network the parameters associated with the unit are assigned as follows:

$$\begin{aligned} & c_{K+1,i} = x_i(q), \quad i = 1, \ldots, m \\ & \sigma_{K+1,i} = \text{Standard deviation of distance between the center and input data} \\ & w_{K+1,i} = y_j^d(q), \quad j = 1, \ldots, n \end{aligned} \tag{11}$$

The steepest descent method is also applied for adjusting the weights, centers, and widths of the network.

In addition, to keep the RBF neural network in an optimal size and make sure that there are no superfluous basis function units, the pruning algorithm is combined during adaptation of RBF neural network structure. For this purpose, the measurement criteria for RBF neural network pruning are defined as follows;

(a) The normalized output value $r_k^n$ which can be expressed by the following equation:

$$r_k^n = \left\| \frac{o_k^n}{o_{\max}^n} \right\|, (k = 1, \ldots, K) \tag{12}$$

where $o_k^n$ is the output for the $k^{th}$ basis function unit, $o_{\max}^n$ is the largest output of basis function unit for the $n^{th}$ input.

(b) The variance of the weight among hidden-output connections $HOWV_i$, which can be expressed by the following equation:

$$HOWV_i = \frac{1}{N-1} \sum_{j=1}^{N} \left( W_{ij} - \overline{W_i} \right)^2 \tag{13}$$

where *N* is the number of hidden units, $W_{ij}$ is the weight between hidden node *i* and output node *j*, and $\overline{W_i}$ is defined as:

$$\overline{W_i} = \frac{1}{N} \sum_{j=1}^{N} W_{ij} \tag{14}$$

This measurement is the calculation of the variance of the weight among hidden-output connections. From our observation in the case of RBF neural networks, the hidden units play an important role tending to have the small variance, compared with the other less significant hidden units. This measure correlates well with the *HOV* criteria, which was proposed by S. Erdogan, Ng. Geok, and P. Chan [8]. However, *HOWV* requires at least two hidden units. Importantly, if $r_k^n$ is less than a threshold $\delta_r$ or *HOWV* is larger than a threshold $\delta_{HOWV}$ during training process, the $k^{th}$ hidden unit should be removed.

## 5. Experimental Results

In this section, results obtained from the simulation, which implements the proposed algorithm for the nonlinear system identification, are presented. They consist of two parts. First, the case of a static nonlinear system similarly to one in [9] [10] is considered. The second part studies the case of dynamic nonlinear system whose dynamical properties are changed online by adding and removing some functions to/from the system model.

Furthermore, obtained results are compared with the ones obtained from a self-organized learning algorithm, which was proposed by S. Arisariyawong and S. Charoenseang [7]. The advantages of a dynamic self-organized learning algorithm are also highlighted. Experiments and their results are described as follows:

## 5.1 Static nonlinear system identification

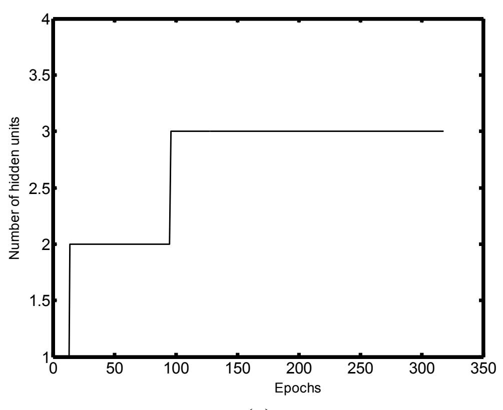
For this problem, a nonlinear function, which consists of three exponential functions, is shown in the following way:

$$y(x) = \exp\left[-\frac{(x-0.3)^2}{0.01}\right] + \exp\left[-\frac{(x-0.7)^2}{0.01}\right] + \exp\left[-\frac{(x-0.5)^2}{0.02}\right] \quad (15)$$

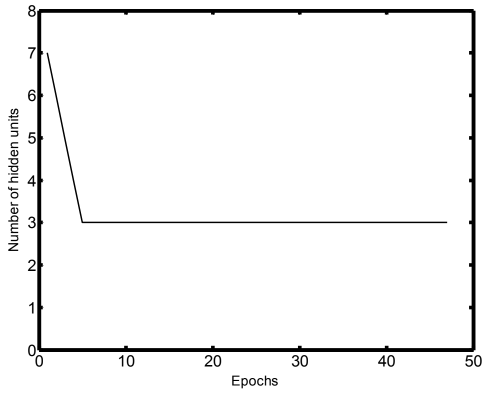
The aim is to evolve a minimal basis function using our pruning algorithm and a self-organized learning algorithm [7] to approximate the function with small error. For this approximation, 121 training patterns *(x,y)* are provided, where $x \in [0,...,1]$. These algorithms will stop learning when the sum squared error is less than 0.005.

Table 1: Actual and estimated centers and width Values

| The proposed pruning algorithm | | | |
|---|---|---|---|
| Actual center | 0.3 | 0.7 | 0.5 |
| Estimated center | 0.2944 | 0.7066 | 0.5030 |
| Actual width $\sigma^2$ | 0.01 | 0.01 | 0.02 |
| Estimated width $\sigma^2$ | 0.0089 | 0.0086 | 0.0346 |
| Self-organized learning algorithm [7] | | | |
| Actual center | 0.3 | 0.7 | 0.5 |
| Estimated center | 0.3098 | 0.6919 | 0.5020 |
| Actual width $\sigma^2$ | 0.01 | 0.01 | 0.02 |
| Estimated width $\sigma^2$ | 0.0115 | 0.0110 | 0.0128 |



(a)



(b)

Fig. 2 Number of hidden units during network adaptation
(a) Self-organized learning algorithm [7]
(b) The proposed pruning algorithm

Fig. 2 shows the number of hidden units during training process with a self-organized learning algorithm [7] and our proposed pruning algorithm. These algorithms require careful selection of threshold parameters as defined in the above section. Based on parametric studies, the parameters required for the optimal basis function unit are chosen as follows: $e_{\min} = 0.02$, $\varepsilon_R = 0.2$, $\varepsilon_n = 0.3$, $\delta_{HOWV} = 0.02$.

Since the function for being approximated is in the form of summation of three Gaussian functions, one can say theoretically that the RBF neural network should have three hidden units with Gaussian functions in its hidden layer. As shown in Fig.2 (a) and (b), both algorithms can learn to get the optimal number of hidden units but our proposed pruning algorithm can learn more quickly than the self-organized learning algorithm [2].

Table 1 presents the comparison of the center and width estimation from both algorithms. From the Table 1, it is obvious that both of the center and width values for the Gaussian functions in the hidden units are close to the actual values. In addition, both learning algorithms can accurately approximate a nonlinear function with an optimal network's size.

## 5.2 Dynamic nonlinear system identification

To further study the capability of adaptation of the dynamic self-organized learning algorithm, the system equation is changed during simulation as follows:

$$y(x) = \exp\left[-\frac{(x-0.3)^2}{0.01}\right] + \exp\left[-\frac{(x-0.7)^2}{0.01}\right] + \exp\left[-\frac{(x-0.5)^2}{0.02}\right] + f(x) \quad (16)$$

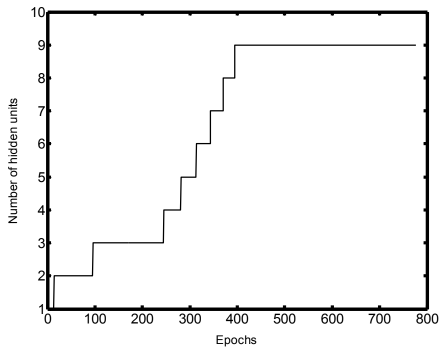
where *f(x)* is defined as follows:

$$f(x) = 0 \text{ ; if epochs } \le 190 \text{ or epochs} > 400 \qquad (17)$$

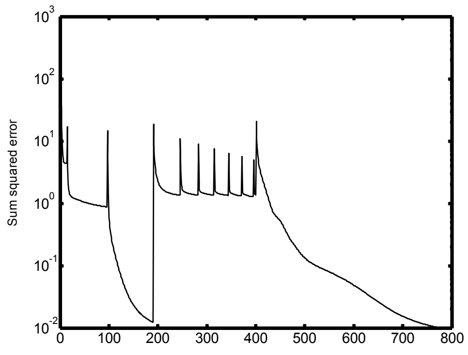
$$f(x) = \exp\left[-\frac{(x-0.9)^2}{0.02}\right] + \exp\left[-\frac{(x-0.1)^2}{0.01}\right] \text{ ; if epochs} > 190 \qquad (18)$$

The aim is to further study the adaptive capability in the dynamic self-organized learning algorithm compared with one in the self-organized learning algorithm proposed by [7]. For this approximation, 121 training patterns *(x,y)* are provided, where $x \in [0,...,1]$. These algorithms will stop learning when the sum squared error is less than 0.005.

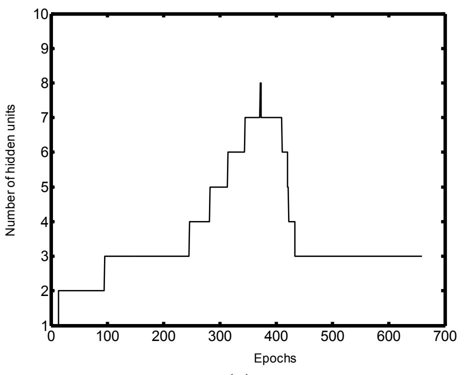
It can be seen that all parameters in equation (16) are fixed during the epochs are equal or less than 190. After epochs are greater than 190, the function *f(x)* was added in the way expressed by the equations (17)-(18). The dynamic self-organized learning algorithm and a self-organized learning algorithm [7] are used for online identification of this time-varying nonlinear system.



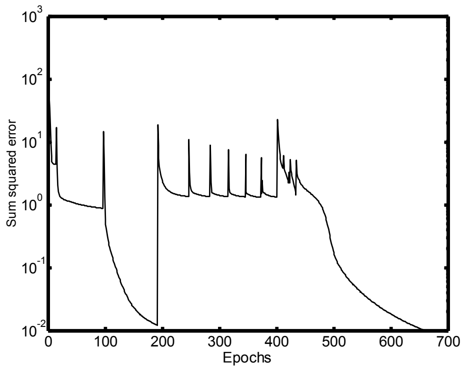
(a)



(b)



(c)



(d)

Fig. 3 Number of hidden units and sum squared error during network adaptation
(a) Number of hidden units in the self-organized learning algorithm proposed in [7]
(b) Sum squared error in the self-organized learning algorithm proposed in [7]
(c) Number of hidden units in the dynamic self-organized learning algorithm
(d) Sum squared error in the dynamic self-organized learning algorithm

Fig. 3(a)-3(d) show the simulation results. In Fig. 3(c), the dynamic self-organized learning algorithm can learn to obtain the optimal number of hidden units during the epochs are equal or less than 190. From the 191th epoch to the 400th epoch, some hidden units are generated to reduce the sum squared error. Subsequently, the superfluous hidden units are detected and removed from the hidden layer to keep the network's size optimal when the system equation is set back to the original one again.

The self-organized learning algorithm [7], is also applied to this time-varying nonlinear system. Its results are also shown in Fig. 3(a) and 3(b) for comparison purpose. Since there is no method of reducing the number of hidden units in the self-organized learning algorithm [7], the size of the network is not reduced during the process of identification for this time-varying system. At the end of process, the number of hidden units is highly excessive compared with the optimal one obtained from the dynamic self-organized learning algorithm. Furthermore, time required to reduce the sum squared error of a self-organized learning algorithm [7] is larger. Finally, this clearly indicates that the excessive number of hidden units generated by the self-organized learning algorithm [7] makes the network to be a highly overparameterised model with all attendant problems.

## 6. Conclusions

In this paper, an algorithm for automatically exploring the optimal size of a radial basis function (RBF) neural network for identifying the nonlinear systems was proposed. This algorithm provides an effective way to increase and decrease the number of hidden units depending on the normalized output value of hidden units and the variance of the weight among hidden-output connections. Experiments of this algorithm for both static and dynamic nonlinear system identification were presented. Specially, the adaptive capability of the algorithm for tracking the time-varying dynamics of a nonlinear system was also demonstrated.

## 7. References

[1] M. Powell, “Radial basis function for multivariate interpolation: A review”, in Mason, J.C., and Cox, M.G. (Eds.), “Algorithm for approximation” (Clarendon Press, Oxford, 1987), pp. 143-168.

[2] D. Broomhead and D. Lowe, “Multivariable functional interpolation and adaptive networks”, *Complx Syst.,* 1988, Vol. 2, pp. 321-355.

[3] J. Moody and C. Darken, “Fast learning in network of locally-tuned processing units”, *Neural Comput.,* 1989, Vol. 1, pp. 281-294.

[4] S. Lee and R. Kil , “A Gaussian potential function network with hierarchically self-organizing learning”, Neural Networks, 1991,Vol. 4, pp. 207-224.

[5] J. Patte, “A resource allocating network for function interpolation”, Neural Computing, 1991, Vol. 3, pp. 213-225.

[6] V. Kadirkamanathan and M. Niranjan, “A function estimation approach to sequential learning with neural network”, Neural Computing, 1993, Vol. 5, pp. 954-975.

[7] S. Arisariyawong and S. Charoenseang, “Self-organized Learning in Complexity Growing of Radial Basis Function Networks”, In Proc. 2002 Int. Tech. Conf. on Circuits/Systems, Computers and Communications, Phuket, Thailand, July 2002, Vol.1, pp. 30-33.

[8] S. Erdogan, Ng. Geok, and P. Chan, “Measurement Criteria for Neural Network Pruning”, In Proc. IEEE TENCON – Digital Signal Processing Applications, Perth, WA, Australia, Nov. 1996, Vol. 1, pp. 83-89.

[9] S. Chen, A. Billings, and M. Grant, “Recursive hybrid algorithm for non-linear system identification using radial basis function networks”, Int. J. Control, 1992, Vol. 55, pp. 1051-1070.

[10] L. Chen, C. Chen, and Y. Cheng, “Hybrid learning algorithm for Gaussian potential function network”, In Proc. IEE D, 1993, Vol. 140, pp. 442-448.