# Self-organized Learning in Complexity Growing of Radial Basis Function Networks

Somwang Arisariyawong[1] and Siam Charoenseang[2]
[1]Mechanical Engineering Department, Faculty of Engineering,
Srinakharinwirot University, Ongkharak, Nakornayok, 26120, Thailand
[2] Center of Operation for FIeld roBOtics Development (FIBO)
King Mongkut's University of Technology Thonburi
Suksawasd, 48 Bangmod, Bangkok, 10140, Thailand
e-mail : somwang@psm.swu.ac.th, siam@fibo.kmutt.ac.th

**Abstract:** To obtain good performance of radial basis function (RBF) neural networks, it needs very careful consideration in design. The selection of several parameters such as the number of centers and widths of the radial basis functions must be considered carefully since they critically affect the network's performance. We propose a learning algorithm for growing of complexity of RBF neural networks which is adapted automatically according to the complexity of tasks. The algorithm generates a new basis function based on the errors of network, the percentage of decreasing rate of errors and the nearest distance from input data to the center of hidden unit. The RBF's center is located at the point where the maximum of absolute interference error occurs in the input space. The width is calculated based on the standard deviation of distance between the center and inputs data. The steepest descent method is also applied for adjusting the weights, centers, and widths.

To demonstrate the performance of the proposed algorithm, general problem of function estimation is evaluated. The results obtained from the simulation show that the proposed algorithm for RBF neural networks yields good performance in terms of convergence and accuracy compared with those obtained by conventional multilayer feedforward networks.

## 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 networks. The feedforward artificial neural networks are 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, finding the suitable network size for a given problem, a trial and error approach is adopted. 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 neurons as data points. This is rarely practical because of the large size of data points. Broomhead and 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].

Although RBF neural networks are reported to be computationally efficient compared with feedforward neural networks but they have the important drawbacks. One drawback of RBF neural networks is that the number of radial basis functions are predetermined. This leads to similar problems as the determination of the number of hidden units for feedforward neural networks [5].

In this paper, we propose the learning algorithm that essentially allows for growing of complexity of RBF neural networks which is adapted automatically according to the complexity of tasks.

## 2. Model Description

In the RBF neural network model, the $j^{th}$ 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:

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

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

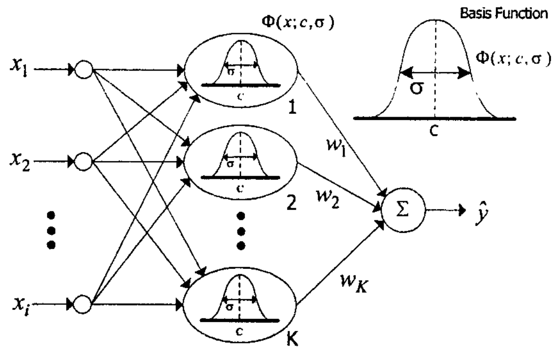


Figure 1. Structure of RBF neural network

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

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

In this case, the $j^{th}$ 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:

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

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 $(C^*, \Theta^*, W^*)$ which satisfies the inequality $E(C^*, \Theta^*, W^*) < \varepsilon$, where $E$ is the optimization criteria defined as:

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

where $y_j(p)$ and $y_j^d(p)$ are the $j^{th}$ model output and the $j^{th}$ 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 $C(h)$, $\sigma(h)$, $W(h)$ at iteration $h$ are computed from the sets $C(h-1)$, $\sigma(h-1)$, $W(h-1)$ by the following learning rule

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

$I = 1, \ldots, m$, $k=1, \ldots, K$, $j=1, \ldots, n$, $h$ is the iteration number.

### 3.1 The learning algorithm for growing of RBF neural networks

The learning algorithm for allowing RBF neural networks to grow during training is done by gradually increasing the number of hidden units. The network begins with only one hidden unit. The following three criteria decide whether an input $x(q)$ should be added to the hidden-layer of the network,

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

where $R(h)$ is the percentaged decreasing rate of errors 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 for a fixed number of RBFs, a new basis function is generated. The importance of this operation is that the model completeness depends not only on the complexity of the learning signal, but also on the structure of the model itself.

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. Let $[x(q), y(q)]$ be an input/output vector pair such that the absolute inference error takes the maximum value at this point. That is to find $[x(q),y(q)]$ which satisfies

$$\left|y_j^d(q) - y_i(q)\right| = \underset{1\le p\le N}{MAX} \cdot \underset{1\le j\le n}{MAX} \left|y_j^d(p) - y_j(p)\right| \quad (11)$$

Then, the $(K+1)^{th}$ new radial basis function is generated according to

$$\begin{aligned} c_{K+1,i} &= x_i(q), \quad i = 1,\dots,m \\ \sigma_{K+1,i} &= \text{Standard deviation of distance between} \\ &\quad \text{the center and input data} \\ w_{K+1,i} &= y_j^d(q), \quad j = 1,\dots,n \end{aligned} \quad (12)$$

Let $K=K+1$ and $h=0$, the steepest descent method is used to determine the optimal solution according to the new number of RBFs. The initial values for the other model parameters are set identically to those obtained by the previous iteration. At any iteration $h$, if the $E(h) < \in$ is satisfied, the learning process is terminated.

## 4. Experimental Results

In this section, simulations with the proposed algorithm for a static problem, namely, the nonlinear function approximations are evaluated. The experiments consist of two parts. The first part considers the case of one-to-one mapping and the second part explores the case of one-to-many mapping. Results from the RBF neural network with our algorithm are compared with the results from multilayer feedforward neural network with backpropagation learning.

### 4.1 Part I. One-to-one mapping

Figure 1 shows the result of learning in one-to-one mapping. The error decreases as the number of hidden nodes increase. Figure 2 shows the result of sum squared errors obtained from the enhanced RBF neural network and the multilayer feedforward neural network with backpropagation learning. In Figure 2, the sum squared error of RBF neural network with our algorithm is reduced faster than the result obtained from multilayer feedforward neural network. Table 1 presents the parameters used in the enhanced RBF neural network and multilayer feedforward neural network.

Figure 1 and 2 showed that the RBF neural network with the proposed algorithm produced a good result with a small number of hidden nodes. To obtain the similar result, the multilayer feedforward neural network requires relatively larger number of hidden nodes as shown in Table 1.

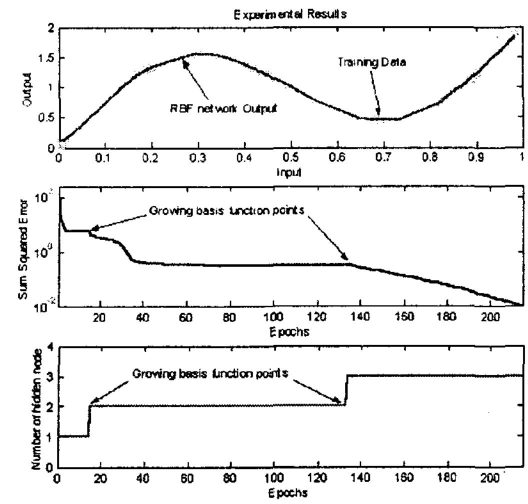


Figure 1. Function approximation by an enhanced self-organizing algorithm

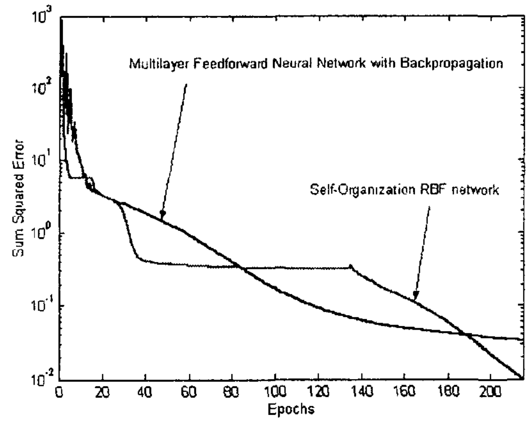


Figure 2. Sum squared errors obtained from two learning schemes

Table 1. The parameters of the neural network used in Part I with the sum squared error of 0.01.

| Radial Basis Function Network with a proposed self-organizing algorithm | | Multilayer Feedforward Network with Backpropagation Learning (Learning rate = 0.01, Momentum = 0.9) | |
|---|---|---|---|
| Number of Basic Function | Epochs | Number of Hidden Nodes | Epochs |
| 3 | 215 | 3 | >10,000 |
| | | 12 | 2746 |
| | | 48 | 1343 |
| | | 128 | 338 |

### 4.2 Part II. One-to-many mapping

Figure 3 shows the capability of RBF neural network with our proposed algorithm when it is used in one-to-many mapping. The RBF neural network can reduce the sum squared error when the number of hidden nodes increase. Figure 4 shows the sum squared error obtained from a multilayer feedforward neural network with backpropagation learning. Although the number of hidden nodes increase, the multilayer feedforward neural network still could not converge.

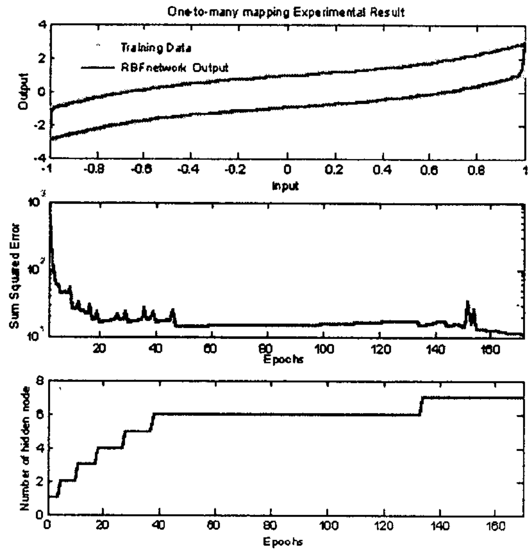


Figure 3. Function approximation by an enhanced self-organizing algorithm

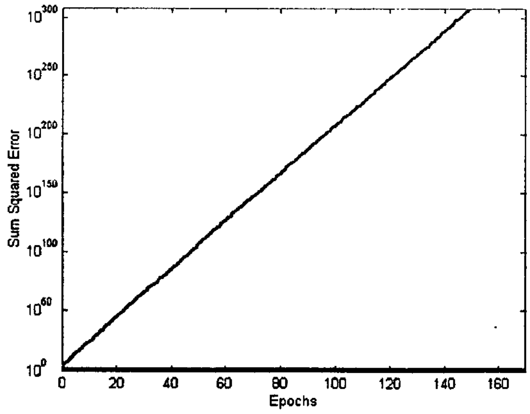


Figure 4. Sum squared error obtained from a multilayer feedforward network with backpropagation learning

## 5. Conclusions

In this paper we presented the learning algorithm that allows for growing of complexity of RBF neural networks. It is also adapted automatically according to the complexity of tasks. Experimental results obtained from both one-to-one mapping and one-to-many mapping were presented. The analysis and simulation showed that our learning algorithm provides the necessary and sufficient enhancement for growing of complexity of RBF neural network. Specifically, the enhanced RBF neural network showed excellent results compared with ones obtained from a multilayer feedforward neural network with backpropagation learning. Finally, this proposed algorithm could be implemented for solving several practical problems, e.g., human-to-robot skill transfers as well as rigorous mathematical analysis.

## 6. References


[1] POWELL, M.J.D., "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] BROOMHEAD, D.S., and LOWE, D., "Multivariable functional interpolation and adaptive networks", *Complx Syst.,* 1988, 2, pp.321-355.

[3] MOODY, J., and DARKEN, C.J., "Fast learning in network of locally-tuned processing units", *Neural Comput.,* 1989, 1, pp. 281-294.

[4] LEE, S., and KIL, R.M., "A Gaussian potential function network with hierarchically self-organizing learning", *Neural Netw.,* 1991, 4, pp.207-224.

[5] N. B. KARAYIANNIS and A. N. VENETSANOPOULOS., "Artificial Neural Networks: Learning Algorithms, Performance Evaluation, and Applications", Kluwer Academic, Boston, MA, 1993.