Neural Processing Letters (2005) 21:189–206
DOI 10.1007/s11063-004-7777-4

A Divide-and-Conquer Learning Approach to Radial Basis Function Networks

YIU-MING CHEUNG[1] and RONG-BO HUANG[2]
[1]*Department of Computer Science, Hong Kong Baptist University, Hong Kong, China. e-mail: ymc@comp.hkbu.edu.hk*
[2]*Department of Mathematics, Guangdong Pharmaceutical College, Guangzhou 510224, China. e-mail: hrongbo@tom.net*

Abstract. This paper presents a new divide-and-conquer based learning approach to radial basis function (RBF) networks, in which a conventional RBF network is divided into several RBF sub-networks. Each of them individually takes an input sub-space as its input. The original network's output then becomes a linear combination of the sub-networks' outputs with the coefficients adaptively learned together with the system parameters of each sub-network. Since this approach reduces the structural complexity of a RBF network by describing a high-dimensional modelling problem via several low-dimensional ones, the network's learning speed is considerably improved as a whole with the comparable generalization capability. The empirical studies have shown its outstanding performance on forecasting two real time series as well as synthetic data. Besides, we have found that the performance of this approach generally varies with the different decompositions of the network's input and the hidden layer. We therefore further explore the decomposition rule with the results verified by the experiments.

Key words. Divide and conquer learning, hidden-layer decomposition, input decomposition, radial basis function network, recurrent radial basis function network

1. Introduction

Due to simple architecture and learning, radial basis function (RBF) networks have become one of the most popular models in neural networks. In the literature, the RBF network has been intensively studied with a lot of applications in data mining [13], pattern recognition [18,9], time series forecasting [7,15], and so forth. In general, the structural complexity of a RBF network depends on the size of the hidden layer which is further somewhat proportional to the input dimension. Hence, effective dimension reduction of the network's input space can considerably decrease the network structural complexity, whereby the network's learning is faster. Traditionally, principal component analysis (PCA) is a popular statistical tool for input dimension reduction, through which first several principal components of the inputs (also called *observations* interchangeably hereinafter) are chosen as the RBF network new inputs. Since the PCA technique uses second-order statistics information only, it makes the principal components de-correlated but not really independent. Subsequently, some useful information in the nonprincipal components may be discarded as well during the dimension reduction process.

Consequently, the performance of the RBF network may become worse after PCA preprocessing [8].

In the past decade, independent component analysis (ICA) has been widely studied in the fields of neural networks and signal processing. It uses high-order statistics to map the multivariate observations into new representations with their component redundancy as reduced as possible. In the literature, it has been shown that ICA outperforms PCA in extracting the hidden feature information and structures from the observations [2,3,11,20]. However, ICA itself does not define the so-called "principal" order for the extracted components. Although some heuristic approaches [1,2,6,10] have explored the component ordering under different optimization criteria, the impact of different orderings on input dimension reduction still needs to be further investigated.

Recently, Kai Tokkola [17] applied a nonlinear dimension-reducing transformation to map high-dimensional space to low-dimensional one in which a nonparametric density estimator is coupled with a mutual information criterion to learn a discriminative dimension-reducing transform. However, the computation of such a transform is laborious because of the time-consuming density estimation. Further, this kind of transformation may lead to the RBF network performance degraded when the output dimension increases.

In this paper, we present a divide-and-conquer based RBF network learning approach (DCRBF), which divides a conventional RBF network into several RBF sub-networks. Each of them individually takes an input sub-space as its input. The original network's output is then a linear combination of the sub-networks' outputs with the coefficients learned together with the system parameters of each sub-network. For short, we hereinafter denote a RBF network learned by the DCRBF approach as *DCRBF* without further distinction. It can be seen that the DCRBF is actually a natural extension of our recently proposed recurrent RBF network named *Dual Structural RBF Network* [5] that models a recursive function by using two RBF sub-networks: one sub-network models the relationship between the current network's output and the past ones, and the other one describes the relationship between the current output and the inputs. Since the DCRBF describes a high-dimensional modelling problem via several low-dimensional ones, it essentially provides an implicit way to reduce the structural complexity of a RBF network such that its learning speed is considerably improved as a whole. We have used a new variant of Extended Normalized RBF (ENRBF) networks [16,18] to realize each sub-network in the DCRBF, whereby a detailed algorithm is presented to learn the parameters of DCRBF adaptively. The experimental results have shown its outstanding performance on forecasting two real time series as well as synthetic data in comparison with a conventional RBF network.

In general, the performance of the DCRBF varies with different decompositions of input space and the hidden layer. Suppose a d-dimensional input space is decomposed into q input sub-spaces, i.e., a d-dimensional input $\mathbf{x} = [x^{(1)}, x^{(2)}, \ldots, x^{(d)}]^T$ (T denotes the transpose operation of a matrix) is

decomposed into q parts. To optimize the DCRBF in a sense, we still have to answer at least two questions

Q1. Among $\left(P_d^d C_{q-1}^{d-1}\right) \Big/ P_q^q$ decomposition combinations of an $\mathbf{x}$, which one should be chosen?

Q2. Suppose the number of hidden units in a conventional RBF network is K. How to determine the unit number k_i of each RBF sub-network i with $\sum_{i=1}^{q} k_i = K$ such that the DCRBF performance is optimized in a certain sense?

In this paper, we have heuristically given out a decomposition rule in terms of optimizing the training speed with the experimental verification.

The paper is organized as follows: Section 2 describes the basic concept of DCRBF and its learning scheme. Particularly, a learning algorithm is given out when each RBF sub-network is implemented by the new ENRBF variant we propose. Also, the performance of DCRBF is experimentally demonstrated in comparison with the conventional RBF network. Section 3 explores the network decomposition rule with the experimental supports in Section 4. Finally, we draw a conclusion in Section 5.

2. DCRBF Learning Approach

2.1. DCRBF NETWORK AND ITS ALGORITHM

In the DCRBF, a conventional RBF network is decomposed into q RBF sub-networks, denoted as $RBF_r, r = 1, 2, \ldots, q$, respectively, as shown in Figure 1. The *Input Decomposer* of a DCRBF network decomposes the input space $\mathbf{V}$ into the direct sum of q input sub-spaces, denoted as $\mathbf{V}_r, r = 1, 2, \ldots, q$, respectively. That is, we have

$$\mathbf{V}_1 \cup \mathbf{V}_2 \cup \ldots \cup \mathbf{V}_q = \mathbf{V} \tag{1}$$

and

$$\mathbf{V}_i \cap \mathbf{V}_j = 0 \quad \text{for any} \quad i \neq j. \tag{2}$$

Each RBF_r models the functional relationship between the current desired output $\mathbf{y}_t$ and the sub-input

$$\mathbf{x}_{r,t} = [x_t^{(i_1)}, x_t^{(i_2)}, \ldots, x_t^{(i_{dr})}] \in \mathbf{V}_r, \tag{3}$$

where $\{i_1, i_2, \ldots, i_{d_r}\} \subseteq \{1, 2, \ldots, d\}$, d_r and d are the dimensions of $\mathbf{V}_r$ and $\mathbf{V}$, respectively, with $\sum_{r=1}^{q} d_r = d$. Further, $\hat{\mathbf{y}}_t$ is the actual output of the DCRBF network with

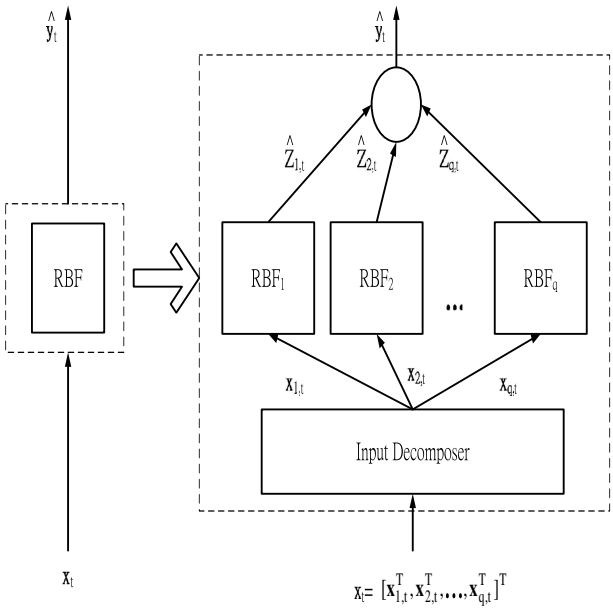

Figure 1. Decomposition of an RBF network into q RBF sub-networks by the DCRBF learning approach.

$$\hat{\mathbf{y}}_t = \sum_{r=1}^{q} c_r \hat{\mathbf{z}}_{r,t}, \tag{4}$$

where $\hat{\mathbf{z}}_{r,t}$ denotes the actual output of RBF_r, and c_r is a coefficient of linear combination. At each time step t, given the desired output $\mathbf{y}_t$, we can calculate the output residual

$$\mathbf{e}_t = \mathbf{y}_t - \hat{\mathbf{y}}_t. \tag{5}$$

Consequently, we can learn the combination coefficients $c_1, c_2, \ldots, c_q$ in Equation (4) and the parameters of each RBF_r's by minimizing the cost function

$$J(\Theta) = \frac{1}{N} \sum_{t=1}^{N} (\mathbf{y}_t - \hat{\mathbf{y}}_t)^T (\mathbf{y}_t - \hat{\mathbf{y}}_t), \tag{6}$$

where N is the number of input data points, $\mathbf{C} = \{c_1, c_2, \ldots, c_q\}$, and $\Theta = \mathbf{C} \cup \Theta_1 \cup \Theta_2 \cup \ldots \cup \Theta_q$ with Θ_r being the parameters of the RBF_r. In implementation, at each time step t, we adaptively tune Θ with a small step along the gradient descent direction of minimizing $(\mathbf{y}_t - \hat{\mathbf{y}}_t)^T (\mathbf{y}_t - \hat{\mathbf{y}}_t)$. That is, we adjust Θ by

$$c_r^{\text{new}} = c_r^{\text{old}} + \eta \mathbf{e}_t^T \hat{\mathbf{z}}_{r,t}, \quad r = 1, 2, \ldots, q, \tag{7}$$

$$\Theta_r^{\text{new}} = \Theta_r^{\text{old}} - \eta \left. \frac{\partial (\mathbf{y}_t - \hat{\mathbf{y}}_t)^T (\mathbf{y}_t - \hat{\mathbf{y}}_t)}{\partial \Theta_r} \right|_{\Theta_r^{\text{old}}}, \tag{8}$$

where η is a small positive learning rate.

The detailed steps of Equation (8) depend on the explicit implementation of each RBF_r, which can be realized by a variety of RBF network models. In this paper, we adopt a new variant of existing ENRBFs [16, 19], whose architecture is shown in Figure 2. There are three layers: d_r-unit input layer, k_r-unit hidden layer, and n-unit output layer. Given the input $\mathbf{x}_{r,t} = [x_{r,t}^{(1)}, x_{r,t}^{(2)}, \ldots, x_{r,t}^{(d_r)}]^T$ at time step t, the output of RBF_r network is

$$\hat{\mathbf{z}}_{r,t} = \sum_{j=1}^{k_r} O_{r,j}(\mathbf{x}_{r,t}) \mathbf{g}_{r,j}(\mathbf{x}_{r,t}), \tag{9}$$

where $\hat{\mathbf{z}}_{r,t} = [\hat{\mathbf{z}}_{r,t}^{(1)}, \hat{\mathbf{z}}_{r,t}^{(2)}, \ldots, \hat{\mathbf{z}}_{r,t}^{(n)},]^T$, $\mathbf{g}_{r,j}(\mathbf{x}_{r,t})$ is an $n \times 1$ vector function whose τth component describes the relationship between hidden unit j and output unit τ in the RBF_r. $O_{r,j}(\mathbf{x}_{r,t})$ is the output of unit j in the hidden layer with

$$O_{r,j}(\mathbf{x}_{r,t}) = \frac{\phi[(\mathbf{x}_{r,t} - \mathbf{m}_{r,j})^T \sum_{r,j}^{-1} (\mathbf{x}_{r,t} - \mathbf{m}_{r,j})]}{\sum_{i=1}^{k_r} \phi[(\mathbf{x}_{r,t} - \mathbf{m}_{r,i})^T \sum_{r,i}^{-1} (\mathbf{x}_{r,t} - \mathbf{m}_{r,i})]}, \tag{10}$$

where $\mathbf{m}_{r,j}$ and $\sum_{r,j}$ are the center vector and receptive field of the basis function $\phi(.)$, respectively, in hidden unit j. In common, the Gaussian function $\phi(s) = \exp(-0.5s)$ is chosen. Consequently, Equation (9) becomes

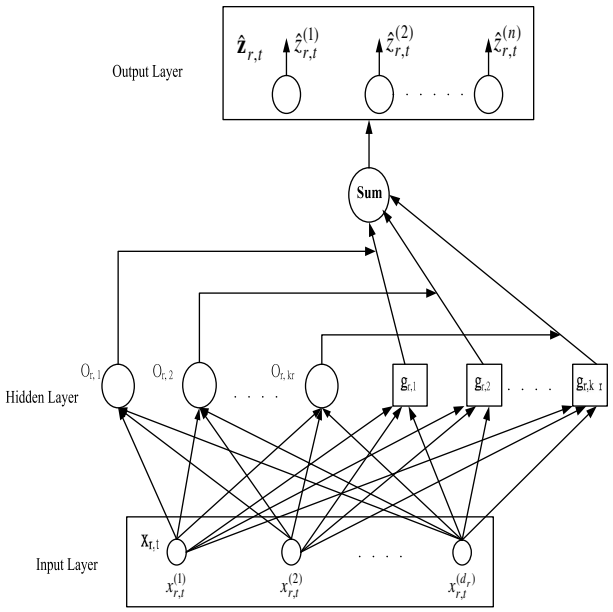

Figure 2. An Extended Normalized RBF Network model used to implement the RBF sub-network RBF_r.

$$\hat{\mathbf{z}}_{r,t} = \sum_{j=1}^{k_r} \frac{\exp[-0.5(\mathbf{x}_{r,t} - \mathbf{m}_{r,j})^T \sum_{r,j}^{-1} (\mathbf{x}_{r,t} - \mathbf{m}_{r,j})]}{\sum_{i=1}^{k_r} \exp[-0.5(\mathbf{x}_{r,t} - \mathbf{m}_{r,i})^T \sum_{r,i}^{-1} (\mathbf{x}_{r,t} - \mathbf{m}_{r,i})]} \mathbf{g}_{r,j}(\mathbf{x}_{r,t}). \tag{11}$$

In the existing ENRBF networks as shown in [16, 19], $\mathbf{g}_{r,j}(\mathbf{x}_{r,t})$ is a linear function

$$\mathbf{g}_{r,j}(\mathbf{x}_{r,t}) = \mathbf{W}_{r,j}\mathbf{x}_{r,t} + \beta_{r,j}, \quad j = 1, 2, \ldots, k_r, \tag{12}$$

where $\mathbf{W}_{r,j}$ is an $n \times d_r$ parameter matrix, and $\beta_{r,j}$ is an $n \times 1$ bias vector. Here, we extend Equation (12) to

$$\mathbf{g}_{r,j}(\mathbf{x}_{r,t}) = \mathbf{W}_{r,j}\mathbf{diag}[sign(\mathbf{x}_{r,t})]|\mathbf{x}_{r,t}|^{p_{r,j}} + \beta_{r,j} \tag{13}$$

with

$$\begin{aligned} sign(\mathbf{x}_{r,t}) &= [sign(x_{r,t}^{(1)}), sign(x_{r,t}^{(2)}), \ldots, sign(x_{r,t}^{(d_r)})]^T, \\ |\mathbf{x}_{r,t}|^{p_{r,j}} &= [|x_{r,t}^{(1)}|^{p_{r,j}}, |x_{r,t}^{(2)}|^{p_{r,j}}, \ldots, |x_{r,t}^{(d)}|^{p_{r,j}}]^T, \end{aligned} \tag{14}$$

where sign(.) is the sign function, $\mathbf{diag}(\mathbf{x}_{r,t})$ is a diagonal matrix whose (i,i)th element is $x_{r,t}^{(i)}$, and $p_{r,j}$ is the exponent of the polynomia $|x_{r,t}^{(j)}|$ that denotes the absolute value of $x_{r,t}^{(j)}$. That is, we use a single $p_{r,j}$-order polynomial term, rather than a conventional linear one, to model the relations between a hidden unit and an output unit. Since such a flexible polynomial regression extends the fitting ability of the linear regression used in most existing ENRBF networks, it is therefore expected that this new variant generally has better performance in function approximation under the moderate number of hidden units. By putting Equation (13) into Equation (11), we then have

$$\begin{aligned} \hat{\mathbf{z}}_{r,t} = \sum_{j=1}^{k_r} & \frac{\exp[-0.5(\mathbf{x}_{r,t} - \mathbf{m}_{r,j})^T \sum_{r,j}^{-1} (\mathbf{x}_{r,t} - \mathbf{m}_{r,j})]}{\sum_{i=1}^{k_r} \exp[-0.5(\mathbf{x}_{r,t} - \mathbf{m}_{r,i})^T \sum_{r,i}^{-1} (\mathbf{x}_{r,t} - \mathbf{m}_{r,i})]}, \\ & [\mathbf{W}_{r,j}\mathbf{diag}[sign(\mathbf{x}_{r,t})]|\mathbf{x}_{r,t}|^{p_{r,j}} + \beta_{r,j}]. \end{aligned} \tag{15}$$

In Equation (15), two parameter sets should be learned. One is $\{\mathbf{m}_{r,j}, \sum_{r,j}\}_{j=1}^{k_r}$ in the hidden layer, and the other is $\{\mathbf{W}_{r,j}, p_{r,j}, \beta_{r,j}\}_{j=1}^{k_r}$ in the output layer. In the paper [19], the learning of parameters in the hidden layer and output layer have been connected with the mixture-of-experts model, whereby an expectation-maximization (EM) based single-step learning algorithm is proposed. Here, for simplicity, we prefer to learn the two parameter sets in the same way as the traditional approaches [14] with the two separate steps: Learn parameters in hidden layer via a clustering algorithm such as k-means [12], followed by learning parameters in the output layer via minimizing $J(\Theta)$ in Equation (6). Eventually, the detailed learning algorithm of DCRBF is given out as follows:

Step 1: Given an input space $\mathbf{V}$, we decompose it into q input sub-space: $\mathbf{V}_1, \mathbf{V}_2, ..., \mathbf{V}_q$. Subsequently, given an input $\mathbf{x}_{r,t} \in \mathbf{V}_r$ at time step t, we learn the parameters of RBF_r by **Step 2** and **Step 3**.

Step 2: We learn $\{\mathbf{m}_{r,j}, \mathbf{\Sigma}_{r,j} | j = 1, 2, \ldots, k_r\}$ in the hidden layer of RBF_r via the RPCCL [4] rather than k-means because of its robust performance without knowing exact cluster number. Interested readers can refer to the paper [4] for more details. In the following, we show its main steps only.

Step 2.1: Randomly take a sample $\mathbf{x}_{r,t}$ from the data set $D = \{\mathbf{x}_{r,t}\}_{t=1}^{N}$, and for $j = 1, 2, \ldots, k_r$, let

$$I(j|\mathbf{X}_{r,t}) = \begin{cases} 1 & \text{if } j = c \\ -1 & \text{if } j = \tau, \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

with

$$\begin{aligned} c &= \arg\min_j \gamma_j \|\mathbf{x}_{r,t} - \mathbf{m}_{r,j}\|^2, \\ \tau &= \arg\min_{j \neq c} \gamma_j \|\mathbf{x}_{r,t} - \mathbf{m}_{r,j}\|^2, \end{aligned} \tag{17}$$

where $\gamma_j = (n_j) \Big/ \left(\sum_{\rho=1}^{k_r} n_\rho\right)$ is the relative winning frequency of the seed point $\mathbf{m}_{r,j}$ in the past, and n_j is the cumulative number of the occurrences of $I(j|\mathbf{x}_{r,t}) = 1$ in the past.

Step 2.2: Update the winner $\mathbf{m}_{r,c}$ (i.e., $I(c|\mathbf{x}_{r,t}) = 1$) and its rival $\mathbf{m}_{r,\tau}$ only by

$$\mathbf{m}_{r,k}^{\text{new}} = \mathbf{m}_{r,k}^{\text{old}} + \triangle \mathbf{m}_{r,k}, \quad \kappa = c, \tau \tag{18}$$

with

$$\begin{aligned} \triangle \mathbf{m}_{r,c} &= \eta_c (\mathbf{x}_{r,t} - \mathbf{m}_{r,c}), \\ \triangle \mathbf{m}_{r,\tau} &= -\eta_c p_\tau(\mathbf{x}_{r,t})(\mathbf{x}_{r,t} - \mathbf{m}_{r,\tau}), \\ p_\tau(\mathbf{x}_{r,t}) &= \frac{\min(\|\mathbf{m}_{r,c} - \mathbf{m}_{r,\tau}\|, \|\mathbf{m}_{r,c} - \mathbf{x}_{r,t}\|)}{\|\mathbf{m}_{r,c} - \mathbf{m}_{r,\tau}\|}, \end{aligned} \tag{19}$$

where η_c is the small positive learning rate.

Steps 2.1 and **2.2** are iterated until those seed points $\mathbf{m}_{r,1}, \mathbf{m}_{r,2}, \ldots, \mathbf{m}_{r,k_r}$ converge. We then directly calculate each $\mathbf{\Sigma}_{r,j}$ by

$$\mathbf{\Sigma}_{r,j} = \frac{\sum_{t=1}^{N} I(j|\mathbf{x}_{r,t})[(\mathbf{x}_{r,t} - \mathbf{m}_{r,j})(\mathbf{x}_{r,t} - \mathbf{m}_{r,j})^T]}{\sum_{t=1}^{N} I(j|\mathbf{x}_{r,t}) - 1}. \tag{20}$$

Step 3: Learn $\{\mathbf{W}_{r,j}, \beta_{r,j} | j = 1, 2, \ldots, k_r\}$ in the output layer under the least mean-square-error (MSE) criterion. That is, we learn them as well as $\mathbf{C}$ by minimizing Equation (6). Consequently, we have

Step 3.1: Given $\mathbf{x}_t$ and $\mathbf{y}_t$, we calculate $\hat{\mathbf{y}}_t$ via Equation (4) by fixing $\boldsymbol{\Theta}$.
Step 3.2: Update $\mathbf{C}$ by Equation (7). Also, we update $\boldsymbol{\Theta}_r$ by

$$\begin{aligned}
\mathbf{W}_{r,j}^{\text{new}} &= \mathbf{W}_{r,j}^{\text{old}} + \eta \triangle \mathbf{W}_{r,j}, \\
p_{r,j}^{\text{new}} &= p_{r,j}^{\text{old}} + \eta \triangle p_{r,j}, \\
\beta_{r,j}^{\text{new}} &= \beta_{r,j}^{\text{old}} + \eta \triangle \beta_{r,j}
\end{aligned}$$

with

$$\begin{aligned}
\triangle \mathbf{W}_{r,j} &= c_r^{\text{old}} O_{r,j}(\mathbf{x}_{r,t}) \mathbf{e}_t |\mathbf{x}_{r,t}|^{p_{r,j}{}^T} \mathbf{diag}[sign(\mathbf{x}_{r,t})], \\
\triangle p_{r,j} &= c_r^{\text{old}} O_{r,j}(\mathbf{x}_{r,t}) \boldsymbol{\mu}_{r,t,j}^T \mathbf{diag}[sign(\mathbf{x}_{r,t})] \mathbf{W}_{r,j}^T \mathbf{e}_t, \\
\triangle \beta_{r,j} &= c_r^{\text{old}} O_{r,j}(\mathbf{x}_{r,t}) \mathbf{e}_t,
\end{aligned} \tag{21}$$

where $\boldsymbol{\mu}_{r,t,j} = [|x_{r,t}^{(1)}|^{p_{r,j}} \ln |x_{r,t}^{(1)}|, |x_{r,t}^{(2)}|^{p_{r,j}} \ln |x_{r,t}^{(2)}|, \ldots, |x_{r,t}^{(d_r)}|^{p_{r,j}} \ln |x_{r,t}^{(d_r)}|^{p_{r,j}}|]^T$.

The iterations of **Steps 3.1** and **3.2** do not stop until the parameters converge.

2.2. EXPERIMENTAL RESULTS

On the time series forecasting, we performed three experiments to compare the performance of the DCRBF with the conventional RBF network learning under the MSE criterion.

Experiment 1. We generated 5100 data points, denoted as $\{(\mathbf{x}_t, y_t)\}_{t=1}^{5100}$ from the following time series:

$$\begin{aligned}
h_t = {} & 0.08h_{t-1}^2 - 0.33h_{t-2} + \sin(h_{t-3}) + 0.08h_{t-4} \\
& + 0.2h_{t-5} + 0.064h_{t-6}^2 h_{t-7} - 0.6h_{t-8}h_{t-9},
\end{aligned} \tag{22}$$

where $\mathbf{x}_t = [x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(9)}]^T = [h_{t-1}, h_{t-2}, \ldots, h_{t-9}]^T$ and $y_t = h_t$ are the input and the desired output at time step t respectively. We let the first 5000 data points be the training set, and the remaining 100 data points be the testing set.

In the DCRBF, we decomposed the input space into three sub-spaces with the input dimension $d_1 = 2, d_2 = 3, d_3 = 4$, respectively. We further let the number of hidden units in each sub-network be $k_1 = 2, k_2 = 2, k_3 = 2$, respectively. For comparison, we also implemented a conventional RBF network hereinafter by the ENRBF network described in Section 2, whose input is $\mathbf{x}_t$ and the hidden-layer size is $k = k_1 + k_2 + k_3 = 6$. In other words, the DCRBF has decomposed the conventional

RBF into three sub-networks in this regard. During the experiment, we simply fixed the learning rate $\eta = 0.0001$, and repeatedly scan the training data set for 100 epochs. We plotted the MSE values of the DCRBF and the conventional RBF network on testing set in Figure 3. It can be seen that the converged performance of DCRBF on the testing set was the same as the conventional RBF network, but the former converged much faster. Actually, the DCRBF has reached the MSE value much lower than the conventional RBF network after the first epoch, although their parameters were initialized in the same way. This scenario implies that the underly mechanism of DCRBF can indeed reduce the structural complexity of a RBF network in effect. Subsequently, the learning speed is considerably improved. In this experiment, the convergence of DCRBF parameters needs only 6 epochs in contrast to 60 epochs of the conventional RBF network.

Experiment 2. In this experiment, we used 4774 FOREX daily foreign exchange rates of 9 countries, which are from the famous Rob Hyndman's Time Series Data Library, during the period between December 31, 1979 and December 31, 1998. We let the first 4674 data be the training set, and the remaining 100 data be the testing set. Also, we set the dimension of input space to $d = 9$, and decomposed it into three subspaces with $d_1 = 2, d_2 = 3, d_3 = 4$. Subsequently, the DCRBF consisted of three RBF sub-networks whose input dimensions are d_1, d_2 and d_3, respectively. Further, we let their hidden-layer size be $k_1 = 2, k_2 = 3$, and $k_3 = 3$. Similar to Experiment 1, we implemented the conventional RBF network by setting $d = d_1 + d_2 + d_4 = 9$ and $k = k_1 + k_2 + k_3 = 8$ for comparison. We plotted the

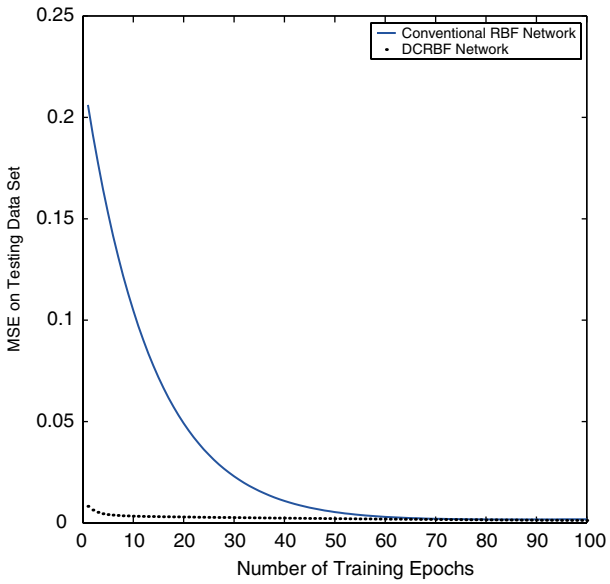

Figure 3. The performance comparison between the DCRBF network and the conventional RBF one on the synthetic time-series data in Experiment 1.

MSE values of the DCRBF and the conventional RBF network on testing set, as shown in Figure 4(a). It can be seen again that the DCRBF network converges faster than the conventional one. Actually, the performance of DCRBF tended to converge within the first training epoch. Furthermore, we found that the DCRBF gives a slight improvement on the generalization error as shown in Figure 4(b). This scenario implies that, through reducing the network's structural complexity, the DCRBF learning is capable of circumventing the over-fitting problem, and reducing the possibility of the system parameters falling into some sub-optimal solutions.

Experiment 3. We applied the DCRBF to forecast the time series of sunspot from year 1700 to 1979, observed by Rudolph Wolf. We used the first 250 data to be the training set, and the remaining 30 to be the testing set. The number of hidden units of the conventional RBF network was $k = 8$, while the hidden units of the three sub-networks in DCRBF were $k_1 = 2, k_2 = 3, k_3 = 3$. We let the input dimension of the conventional RBF network be $d = 9$, and the input dimension of three decomposed sub-network in DCRBF be $d_1 = 3, d_2 = 3, d_3 = 3$. The experimental results are shown in Figure 5. Once again, we found that the DCRBF converges much faster than the conventional one with slightly better generalization ability.

3. The Decomposition Rule for DCRBF

In general, as shown in Section 2.2, the performance of DCRBF varies with the different input and hidden-layer decompositions. Supposing a DCRBF consists of

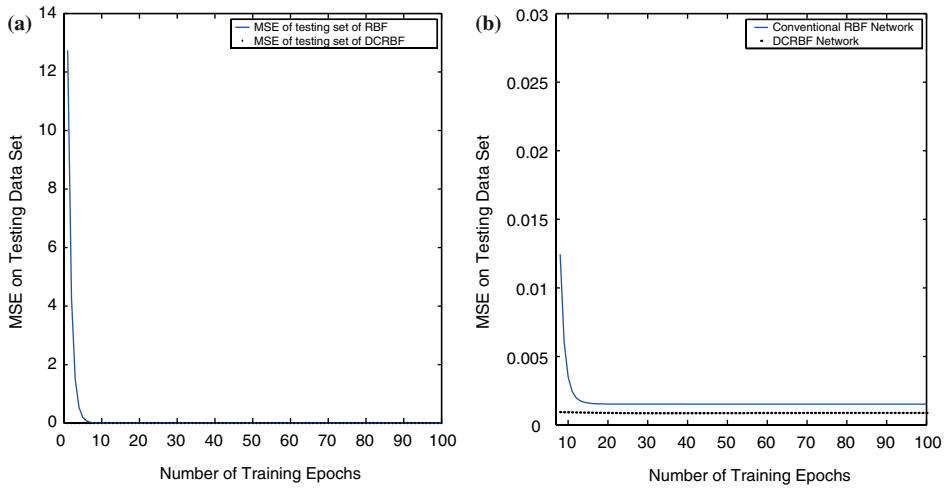

Figure 4. (a) The performance comparison between the DCRBF network and the conventional RBF network on FOREX daily foreign exchange data in Experiment 2; (b) The slight better convergent performance of the DCRBF in contrast to the conventional one.

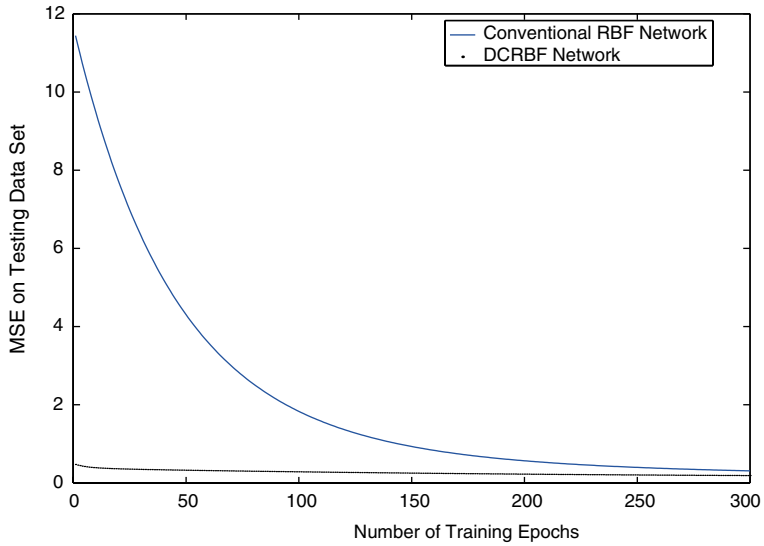

Figure 5. The performance comparison between the DCRBF and the conventional RBF networks on sunspot data in Experiment 3.

q RBF sub-networks, we can heuristically determine an appropriate input decomposition in the previous **Step 1** by the following two sub-steps:

Step 1.1 Give an appropriate order of the d input variables in $\mathbf{V}$;
Step 1.2 Separate these ordered d input variables into q groups, each of which spans an input sub-space accordingly.

In **Step 1.1**, we use PCA ordering, i.e., the inputs $x^{(1)}, x^{(2)}, \ldots, x^{(d)}$ in $\mathbf{V}$ are transformed into $x^{(i_1)}, x^{(i_2)}, \ldots, x^{(i_d)}$, where $i_1, i_2, \ldots, i_d$ is the decreasing order of the eigen values of the covariance matrix of $\mathbf{x}$. In **Step 1.2**, we uniformly decompose the ordered input variables into q groups. That is, the dimension of each input sub-space should be as equal as possible. In the following, we will present a theorem to show that such a uniform input separation leads to an optimal decomposition in a sense of training time cost. Moreover, when the total number of hidden units is fixed, the theorem tells us that the number of hidden units in each RBF sub-network should be uniformly distributed as well. Before presenting the theorem, we first give out some definitions.

DEFINITION 1. *Let* $d_i, i = 1, 2, \ldots, q$, *be positive integers. A* q*-tuple* $(d_1, d_2, \ldots, d_q)$ *is called* $q_{\text{decomposition}}$ *of* d *if* $d_1 + d_2 + \cdots + d_q = d$.

DEFINITION 2. *If a* $q_{\text{decomposition}}$ *makes the training time cost of a DCRBF minimized, it is called* **t***ime* **o***ptimal* **d***ecomposition (TOD).*

DEFINITION 3. *The total number K of hidden units in DCRBF is defined as the module of DCRBF, written as $\|DCRBF\| = K$, where K is the sum of those hidden radial function units in each RBF sub-network.*

DEFINITION 4. *A product among the parameters and constants in the DCRBF is called a basic computation term (BCT).*

The theorem is then presented as follows.

THEOREM 1.

1. *Let the dimension of input space* $\mathbf{V}$ *be* d *and* $(d_1, d_2, \ldots, d_q)$ *be a* $q_{\text{decomposition}}$ *of* d. *A conventional RBF network is therefore decomposed into the DCRBF consisting of* q *sub-networks with input dimension* $d_1, d_2, \ldots, d_q$, *respectively.*

2. *Let* $\|DCRBF\| = K$, *and* $(k_1, k_2, \ldots, k_q)$ *be a* $q_{\text{decomposition}}$ *of* K;

3. *The time cost of each BCT in the DCRBF is equal.*

The $q_{\text{decomposition}}$ *of* $(d_1, d_2, \ldots, d_q)$ *and* $(k_1, k_2, \ldots, k_q)$ *is TOD if and only if*

$$d_1 = d_2 = \cdots = d_q = \frac{d}{q}, \tag{23}$$

$$k_1 = k_2 = \cdots = k_q = \frac{K}{q}. \tag{24}$$

To prove Theorem 1, we first give out the lemma as follows.

LEMMA 1. *Given the objective function*

$$f(\xi_1, \xi_2, \cdots, \xi_q, \varsigma_1, \varsigma_2, \cdots, \varsigma_q) = \sum_{i=1}^{q} \xi i \varsigma_i^2 \tag{25}$$

with the constraints

$$\sum_{i=1}^{q} \xi_i = Q, \quad \sum_{i=1}^{q} \varsigma_i = K, \quad \xi_i > 0, \quad \varsigma_i > 0, i = 1, 2, \ldots, q \tag{26}$$

the necessary and sufficient condition for a minimum point of $f(\xi_1, \xi_2, \ldots, \xi_q, \varsigma_1, \varsigma_2, \ldots, \varsigma_q$ *is*

$$\xi_1 = \xi_2 = \cdots = \xi_q = \frac{Q}{q}, \tag{27}$$

$$\varsigma_1 = \varsigma_2 = \cdots = \varsigma_q = \frac{K}{q}. \tag{28}$$

Proof. We construct the Lagrange function as follows:

$$L(\xi_1, \xi_2, \ldots, \xi_q, \varsigma_1, \varsigma_2, \ldots, \varsigma_q, \lambda_1, \lambda_2) = \sum_{i=1}^{q} \xi_i^2 \varsigma_i^2 + \lambda_1 \left(Q - \sum_{i=1}^{q} \xi_i \right) + \lambda_2 \left(K - \sum_{i=1}^{q} \varsigma_i \right). \tag{29}$$

We let the partial derivatives of L with respect to $\xi_i, \varsigma_i, \lambda_1, \lambda_2, i = 1, 2, \ldots, q$ be zero, i.e.,

$$\frac{\partial L}{\partial \xi_i} = \varsigma_i^2 - \lambda_1 = 0, \tag{30}$$

$$\frac{\partial L}{\partial \varsigma_i} = 2\xi_i \varsigma_i - \lambda_2 = 0, \tag{31}$$

$$\frac{\partial L}{\partial \lambda_1} = \left(Q - \sum_{i=1}^{q} \xi_i \right) = 0, \tag{32}$$

$$\frac{\partial L}{\partial \lambda_2} = \left(K - \sum_{i=1}^{q} \varsigma_i \right) = 0. \tag{33}$$

From Equations (30) and (31), we have

$$\xi_1 = \xi_2 = \cdots = \xi_q, \tag{34}$$

$$\varsigma_1 = \varsigma_2 = \cdots = \varsigma_q. \tag{35}$$

Substitute Equations (34) and (35) into Equations (32) and (33), Lemma 1 is therefore proved. □

Now we present the proof of Theorem 1 as follows.

Proof. Let $(d_1, d_2, \ldots, d_q)$ be a $q_{\text{decomposition}}$ of d, ζ be the training time of DCRBF, and ζ_r be the training time of RBF_r with $r = 1, 2, \ldots, q$, respectively. We have

$$\zeta = \sum_{r=1}^{q} \zeta_r. \tag{36}$$

Since ζ_r depends on the number of BCT only in the learning process, we therefore just need to consider the BCT in the following term:

$$[\mathbf{x}_r - \mathbf{m}_{r,j}]^T \mathbf{\Sigma}_{r,j}^{-1} [\mathbf{x}_r - \mathbf{m}_{r,j}], \tag{37}$$

where $j = 1, 2, \ldots, d_r, r = 1, 2, \ldots, q$, and $\mathbf{x}_r \in \mathbf{V}_r$ is a vector. It can be seen that the number of BCT in (37) is d_r^2 as given $\mathbf{x}_r$. Without loss of generality, we suppose the computing time cost of each BCT is one time unit. Hence, ζ_r is a function of d_r and k_r, which can be further expressed as

$$\zeta_r(d_r, k_r) = k_r d_r^2. \tag{38}$$

Putting Equation (38) into Equation (36), we then have

$$\zeta(d_1, d_2, \ldots, d_q, k_1, k_2, \ldots, k_q) = \sum_{r=1}^{q} k_r d_r^2. \tag{39}$$

That is, we try to optimize the following constraint problem:

$$\text{minimize } \zeta(d_1, d_2, \ldots, d_q, k_1, k_2, \ldots, k_q) = \sum\nolimits_{r=1}^{q} k_r d_r^2, \tag{40}$$

$$\text{subject to } \sum\nolimits_{r=1}^{q} d_r = d, \quad \sum\nolimits_{r=1}^{q} k_r = K, \quad d_r > 0, \quad k_r > 0. \tag{41}$$

It can be seen that, as $\xi_j = d_j$ and $\varsigma_j = k_j$ with $j = 1, 2, \ldots, q$, the solution of this problem actually becomes the special case of Lemma 1. Hence, Theorem 1 is held. □

4. Experimental Simulations

4.1. EXPERIMENT 1 AND 2

To justify the above theorem, we showed two experiments to compare the train time cost of DCRBF with the different decompositions. The experimental environment is given in Table I. In Experiment 1, we used 8100 data points generated from the nonlinear function

$$y_t = x_t^{(1)} \cos(x_t^{(2)}) + x_t^{(3)} \sin(x_t^{(4)}) - 0.4(x^{(5)}t)^2 + 0.5x_t^{(6)} x_t^{(7)} + 0.2(x_t^{(8)})^2 x^{(9)} + \varepsilon_t,$$

where $\mathbf{x}_t = [x_t^{(1)}, x_t^{(2)}, x_t^{(3)}, x_t^{(4)}, x_t^{(5)}, x_t^{(6)}, x_t^{(7)}, x_t^{(8)}, x_t^{(9)}]^{\mathrm{T}}$ is the input of RBF network, y_t is the desired output of RBF network, and ε_t is zero-mean Gaussian white noise with the variance being 0.001. We let $\|DCRBF\| = 9$ and decomposed the input space of the conventional RBF network into the direct sum of three input sub-spaces. The decomposition of input space and experimental results are shown in Table II. It can be seen that the learning speed of uniform decomposition is the fastest in all cases we have tried so far.

In Experiment 2, we performed an experiment on the benchmark data acquired from the famous Rob Hyndman's Time Series Data Library. We used the FOREX daily foreign exchange rates of Australia to USA from December 31, 1979

Table I. The experimental environment.

CPU	Pentium III 650 MHZ
Memory	256M
Operating System	Windows 2000
Running software	Matlab 5.3

Table II. The training time cost of DCRBF under different input decompositions.

3-Decomposition of d	3-Decomposition of k	$\sum_{r=1}^{q} k_r d_r^2$	Mean training time
(3,3,3)	**(3,3,3)**	**81**	**11.1723**
(3,3,3)	(2,2,5)	81	11.2679
(2,2,5)	(3,3,3)	99	11.6658
(2,3,4)	(2,2,5)	106	11.7269
(2,2,5)	(2,2,5)	141	12.0171

to December 31, 1998 with the 4774 data points, written as $\{h_t\}_{t=1}^{4774}$. In the experiment, we let the input of RBF network be $\mathbf{x}_t = [h_{t-1}, h_{t-2}, h_{t-3}, h_{t-4}, h_{t-5}, h_{t-6}, h_{t-7}, h_{t-8}, h_{t-9}]^T$, and $y_t = h_t$ be the desired output. Similar to Experiment 1, we also let $\|DCRBF\| = 9$, and decomposed the input space of RBF network into three input sub-spaces. Table III shows the results under different decompositions. Again, the DCRBF with the uniform decomposition needs the least training time cost.

4.2. EXPERIMENT 3

In the above experiments, we have investigated the uniform decomposition on the learning speed of DCRBF without considering the net's generalization capability. In the following, we will further demonstrate the generalization capability of DCRBF when uniform decomposition and PCA input-variable ordering are used. We set $\mathbf{x}_t = [x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(11)}]^{\mathrm{T}}$ to be the input of DCRBF, where $x_t^{(2)}, x_t^{(4)}, x_t^{(6)}, x_t^{(7)}, x_t^{(8)}, x_t^{(9)}$ were uniformly distributed, $x_t^{(1)}, x_t^{(3)}, x_t^{(11)}$ were Gaussians and $x_t^{(5)}, x_t^{(10)}$ were from a mixture of two Gaussians. The desired outputs of the network were given by

$$y_t = x_t^{(1)} \cos x_t^{(2)} + x_t^{(3)} \sin x_t^{(4)} \cos x_t^{(10)} - (x_t^{(5)})^2 \sin x_t^{(12)} + 0.5 x_t^{(6)} x_t^{(7)} + 0.2 (x_t^{(8)})^2 x_t^{(9)} \sin x_t^{(11)} + e_t,$$

where e_t is white noise.

We generated 1100 data points. The first 1000 were the training data, and the remaining 100 data were the testing data. In the experiment, the PCA

Table III. The training time cost of DCRBF under different input decompositions.

3-Decomposition of d	3-Decomposition of k	$\sum_{r=1}^{q} k_r d_r^2$	Mean training time
(3,3,3)	**(3,3,3)**	**81**	**15.4161**
(3,3,3)	(2,2,5)	81	15.4914
(2,2,5)	(3,3,3)	99	15.7441
(2,3,4)	(2,2,5)	106	15.9068
(2,2,5)	(2,2,5)	141	16.2129

Table IV. The MSE of DCRBF on testing set under different decompositions, where $\bigoplus$ denotes the direct sum of input sub-spaces.

Trials	Input space	k	MSE
1	$(x_2, x_1, x_2, x_4, x_5) \bigoplus (x_{11}, x_{10}, x_9, x_8, x_7, x_6)$	**(5,6)**	**0.3200**
2	$(x_3, x_4, x_5, x_{10}, x_{11}) \bigoplus (x_1, x_2, x_6, x_7, x_8, x_9)$	(5,6)	0.3235
3	$(x_1, x_2, x_{11}) \bigoplus (x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10})$	(3,8)	0.3451
4	$(x_4, x_5) \bigoplus (x_1, x_2, x_3, x_6, x_7, x_8, x_9, x_{10}, x_{11})$	(2,9)	0.3534
5	$(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11})$	11	0.4322

order of input is $x^{(11)}, x^{(10)}, x^{(9)}, x^{(8)}, x^{(7)}, x^{(6)}, x^{(3)}, x^{(1)}, x^{(2)}, x^{(4)}, x^{(5)}$. We decomposed the inputs into two parts: $\mathbf{x}(1) = \{x^{(11)}, x^{(10)}, x^{(9)}, x^{(8)}, x^{(7)}, x^{(6)}\}$ and $\mathbf{x}(2) = \{x^{(3)}, x^{(1)}, x^{(2)}, x^{(4)}, x^{(5)}\}$. Further, we fixed the learning rate to 0.001. During the network learning process, the MSE curve on the testing set was shown in Figure 6. After scanning training set data 200 times, a snapshot of the MSE value on the testing set was 0.32 as shown in the Trial 1 of Table IV. In contrast, we also investigated another input uniform decomposition in Trial 2 without considering PCA ordering. Table IV shows that the performance of DCRBF deteriorates a little bit in comparison with Trial 1. Further, in Trial 3 and Trial 4, we tested two other different input decompositions without uniform decomposition and PCA ordering. It can be seen that the performance of DCRBF has been further moderately deteriorated, but they were better than the DCRBF without input decomposition (i.e., a DCRBF has degenerated to a conventional RBF network) as shown in Trial 5 of Table IV. Actually, Figure 6 has shown that the DCRBF with PCA ordering and uniform decomposition learns much faster than the conventional RBF network, and has a moderately improved generalization capability.

5. Conclusion

We have presented a divide-and-conquer learning approach to RBF network that is a hybrid system consisting of several RBF sub-networks. Each RBF sub-network takes an input sub-space as its own input. The whole DCRBF network output is a combination of RBF sub-networks' outputs. Since this system divides a high-dimensional modelling problem into several low-dimensional ones, its structural complexity is generally lower than a conventional RBF network. The experiments have shown that the learning of the proposed approach is much faster than the conventional RBF network learning. Furthermore, we have studied the decompositions in the DCRBF network. Not only is the PCA input-variable ordering suggested, but also a uniform decomposition is presented on both the number of inputs and the number of hidden units, which is optimal in the sense of learning speed. The numerical results have justified such a decomposition.

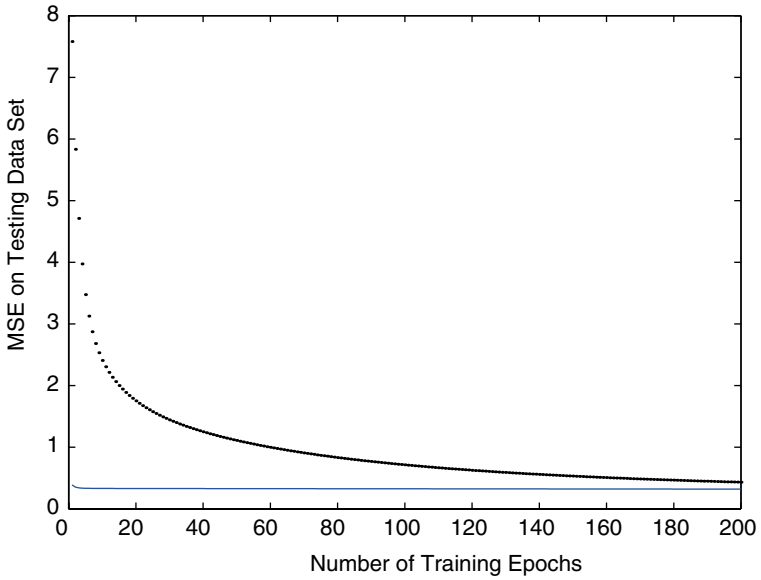

Figure 6. The MSE curve of DCRBF and a conventional RBF network on the testing set in Experiment 3 of Section 4.2, where the solid line is from the DCRBF, and the dashed line is from the conventional one.

Acknowledgements

This work was supported by a Faculty Research Grant of Hong Kong Baptist University with the project code: FRG/02-03/II-40.

References

1. Back, A. D. and Trappenberg, T. P.: Input variable selection using independent component analysis, In: *Proceedings of International Joint Conference on Neural Networks* **2**, pp. 989–992, 1999.
2. Back, A. D. and Weigend, A. S.: A first application of independent component analysis to extracting structure from stock returns, In: *International Journal of Neural System*, **8**, (4), pp. 473–484, 1997.
3. Bartlett, M. S., Lades, H. M. and Sejnowski, T. J.: Independent component representations for face recognition, In: *Proceedings of the SPIE Symposium on Electronic Imaging: Science and Technology; Conference on Human Vision and Electronic Imaging* **III** , pp. 528–539, 1998.
4. Cheung, Y. M.: Rival penalization controlled competitive learning for data clustering with unknown clustering number, In: *Proceedings of 9th International Conference on Neural Information Processing* (ICONIP'2002) **2**, pp. 467–471, 2002.
5. Cheung, Y. M. and Xu, L.: A dual structure radial basis function network for recursive function estimation, In: *Proceedings of International Conference on Neural Information Processing* (ICONIP'2001), **2**, pp. 1903–1097, 2001.
6. Cheung, Y. M. and Xu, L.: Independent component ordering in ICA time series analysis, *Neurocomputing* **41** (2001), 145–152.

7. Davey, N., Hunt, S. P. and Frank, R. J.: Time series prediction and neural networks, *Journal of Intelligent and Robotic Systems* **31** (2001), 91–103.
8. Huang, R. B., Law, L. T. and Cheung, Y. M.: An experimental study: On reducing RBF input dimension by ICA and PCA, In: *Proceedings of 1st International Conference on Machine Learning and Cybernetics 2002* (ICMLC'02), **4**, pp. 1941–1946, 2002.
9. Huang, D. S.: Application of generalized radial basis function networks to recognition of radar targets, *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 13, No. 6, pp. 945–962, 1999.
10. Hyvärien, A.: survey on independent component analysis, *Neural Computing* (1999), 94–128.
11. Jang, G. J., Lee, T. W. and Oh, Y. H.: Learning statistically efficient features for speaker recognition, In: *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Salt Lake City, Utah, May, 2001.
12. MacQueen, J.: Some methods for classification and analysis of multivariate observations, In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, **1**, pp. 281–297, Berkeley: University of California Press, 1967.
13. McGarry, K. J., Wermter, S. and MacIntyre, J.: Knowledge extraction from radial basis function networks and multilayer perceptrons, In: *Proceeding of International Joint Conference on Neural Networks*, **4**, pp. 2494–2497, 1999.
14. Moody, J. and Darken, J.: Fast learning in networks of locally-tunned processing units, *Neural Computation*, **1** (1989), 281–294.
15. Saranli, A. and Baykal, B.: Chaotic time-series prediction and the relocating LMS (RLMS) algorithm for radial basis function networks, *European Signal Processing Conference* (EUSIPCO) **2** (1996), 1247–1250.
16. Stokbro, K., Umberger, D. K. and Hertz, J. A.: Exploiting neurons with localized receptive fields to learn chaos, *Complex Systems* **4** (1990), 603–622.
17. Torkkola, K. and Campbell, W. M.: Mutual information in learning feature transformations, In: *Proceedings of International Conference on Machine Learning* (ICML), Stanford: CA, 2000.
18. Verma, B.: Handwritten hindi character recognition using RBF and MLP neural networks, *IEEE International Conference on Neural Networks* (ICNN) pp. 86–92, Perth, 1995.
19. Xu, L.: RBF nets, mixture experts, and bayesian Ying-Yang learning, *Neurocomputing* **19** (1–3), (1998), 223–257.
20. Ziehe, A., Nolte, G., Sander, T., Muller, K. R. and Curio, G.: A comparison of ICA-based artifact reduction methods for MEG, *12th International Conference on Biomagnetism*, Finland, Helsinki University of Technology, 2000.