A NEW RECURRENT RADIAL BASIS FUNCTION NETWORK

Yiu-ming Cheung

Department of Computer Science
Hong Kong Baptist University
Hong Kong, China
ymc@comp.hkbu.edu.hk

ABSTRACT

Cheung and Xu 2001 has presented a dual structural recurrent radial basis function (RBF) network by considering the different scales in net's inputs and outputs. However, such a network implies that the underlying functional relationship between the net's inputs and outputs is linear separable, which may not be true from a practical viewpoint. In this paper, we therefore propose a new recurrent RBF network. It takes the net's input and the past outputs as an augmented input in analogy with the one in (Billings and Fung 1995), but introduces a scale tuner into the net's hidden layer to balance the different scales between inputs and outputs. This network adaptively learns the parameters in the hidden layer together with those in the output layer. We implement this network by using a variant of extended normalized RBF (Cheung and Xu 2001) with its hidden units learned by the rival penalization controlled competitive learning (RPCCL) algorithm (Cheung 2002). The experiments have shown the outstanding performance of the proposed network in recursive function estimation.

1. INTRODUCTION

In the past, radial basis function (RBF) network has been extensively studied due to its simple architecture and fast learning [2, 3, 6]. Typically, a RBF net describes its output to be a function of the inputs only. However, in some practical problems such as nonlinear adaptive noise cancellation problem [1] and the representation of finite state automata [7], the net's output depends on the past ones as well as the inputs. Under the circumstances, such a network cannot work well. In the literature, one improved RBF architecture is to take the net's input and the past outputs as an augmented input [1]. However, this method requests the scale of network's output to be the same as the inputs. Otherwise, it can lead to a poor clustering result in the hidden layer, whereby the net's performance considerably deteriorates.

The work described in this paper was supported by the Faculty Research Grant of Hong Kong Baptist University with Project Number: FRG/02-03/I-06.

In our recent paper [5], we have presented a dual structural radial basis function (DS-RBF) network, which is a hybrid system consisting of two sub-RBF networks. One subnetwork models the functional relationship between the current network's output and the past ones, and the other one describes the relationship between the current output value and the inputs. We have implemented each sub-RBF network by using a new variant of extended normalized RBF (ENRBF) net. The experiments in [5] has successfully shown the outstanding performance of DS-RBF in nonlinear recursive function approximation. However, we have also noticed that the DS-RBF supposes the underlying functions to be linear-separable, i.e., the function can be linearly decomposed into two ones that are the functions of the net's current inputs and past outputs respectively, which however may not be true from a practical viewpoint.

In this paper, we propose a new recurrent RBF network which takes the net's input and the past outputs as an augmented input in analogy with the one in [1], but introduces a scale tuner into the net's hidden layer to balance the scales between inputs and outputs. We learn the tuner's parameter together with those in the output layer, resulting in the parameters in the hidden layer and output layer are learned in an iterative way, rather than a two-separate steps as like in [6]. We have given out the learning algorithm of this network with its hidden units learned by the Rival Penalization Controlled Competitive Learning (RPCCL) algorithm [4] rather than k-means. The advantage is that the former can automatically drive the centers of extra hidden units far away from the input data set, whereby we can circumvent to pre-determine the number of hidden units. The experiments have shown the proposed net's outstanding performance.

2. PROBLEM

Given a set of N training data points $\{\mathbf{x}_t, \mathbf{z}_t\}_{t=1}^{N}$, where $\mathbf{x}_t = [x_{t,1}, x_{t,2}, \ldots, x_{t,d}]^T$ and $\mathbf{z}_t \in \Re^n$ are the input at time t and the corresponding desired output respectively, we describe the relations between $\mathbf{z}_t$'s and $\mathbf{x}_t$'s by the following

recursive function:

$$\mathbf{z}_t = F(\mathbf{Z}_{t-1}, \mathbf{x}_t) + \mathbf{e}_t \quad (1)$$

with $\mathbf{Z}_{t-1} = [\mathbf{z}_{t-1}^T, \mathbf{z}_{t-2}^T, \ldots, \mathbf{z}_{t-q}^T]^T$, where $F(.)$ is an unknown deterministic nonlinear function and $\mathbf{e}_t$ is white noise. The task of a recurrent RBF network is to approximate this function through the given training data set with the network's generalization ability as good as possible under a certain measurement.

3. NEW RECURRENT RBF NETWORK

3.1. General Structure

In analogy with the conventional RBF network [6], the proposed recurrent RBF network consists of a k-unit hidden layer and an n-unit output layer as shown in Figure 1, where the network takes the net's current input $\mathbf{x}_t$ and the past outputs $\mathbf{Z}_{t-1}$ as an augmented input. Although such a network is similar to that in [1], a new component named *Input Scale Tuner* (IST) is introduced into the hidden layer. Its main goal is to unify the different scales between inputs and outputs before performing clustering. In mathematics, the functionality of this component can be generally described as:

$$\begin{aligned} \tilde{\mathbf{x}}_t &= h_1(\mathbf{x}_t; \mathbf{\Theta}_1) \\ \tilde{\mathbf{Z}}_{t-1} &= h_2(\mathbf{Z}_{t-1}; \mathbf{\Theta}_2), \end{aligned} \quad (2)$$

where $\mathbf{\Theta}_i$, $i = 1, 2$, denotes the parameter set of function h_i. When an input $\mathbf{x}_t$ and the past outputs $\mathbf{Z}_{t-1}$ presented in the input layer, the IST transforms it to $\mathbf{U}_t = [\tilde{\mathbf{x}}_t^T, \tilde{\mathbf{Z}}_{t-1}^T]^T$. Then, the output of unit j in the hidden layer is:

$$O_j(\mathbf{U}_t) = \frac{\phi[(\mathbf{U}_t - \mathbf{m}_j)^T \mathbf{\Sigma}_j^{-1} (\mathbf{U}_t - \mathbf{m}_j)]}{\sum_{i=1}^k \phi[(\mathbf{U}_t - \mathbf{m}_i)^T \mathbf{\Sigma}_i^{-1} (\mathbf{U}_t - \mathbf{m}_i)]}, \quad (3)$$

where $\mathbf{m}_j$ is the center vector, and $\mathbf{\Sigma}_j$ is the receptive field of the basis function $\phi(.)$. In general, one common choice of function $\phi(.)$ is the Gaussian function $\phi(s^2) = \exp(-0.5s^2)$. That is,

$$O_j(\mathbf{U}_t) = \frac{\exp[-0.5(\mathbf{U}_t - \mathbf{m}_j)^T \Sigma_j^{-1} (\mathbf{U}_t - \mathbf{m}_j)]}{\sum_{i=1}^k \exp[-0.5(\mathbf{U}_t - \mathbf{m}_i)^T \Sigma_i^{-1} (\mathbf{U}_t - \mathbf{m}_i)]}. \quad (4)$$

Consequently, the net's actual output $\hat{\mathbf{z}}_t = [\hat{z}_{t,1}, \hat{z}_{t,2}, \ldots, \hat{z}_{t,n}]^T$ is

$$\hat{\mathbf{z}}_t = \sum_{j=1}^k \mathbf{g}_j(\mathbf{U}_t; \mathbf{\Theta}_g) O_j(\mathbf{U}_t), \quad (5)$$

where $\mathbf{g}_j(\mathbf{U}_t; \mathbf{\Theta}_g)$ with the parameter set $\mathbf{\Theta}_g$ is an $n \times 1$ vector function whose r^{th} component describes the relations between hidden unit j and output unit r.

With the desired output $\mathbf{z}_t$, the output error

$$\mathbf{e}_t = \mathbf{z}_t - \hat{\mathbf{z}}_t \quad (6)$$

is calculated out, and propagated to the output and hidden layers. Consequently, the two layers' parameters are modified. In the next sub-section, we will give out a general adaptive procedure to estimate these parameters.

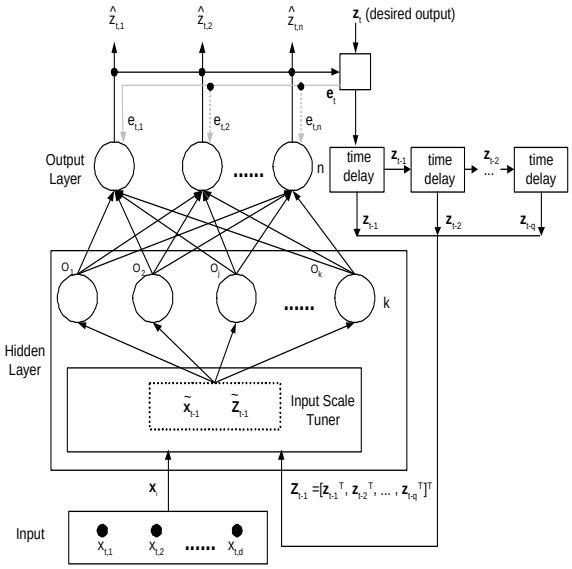

Figure 1: General structure of the proposed recurrent RBF network.

3.2. A General Procedure for Parameter Estimations

We learn the parameters by minimizing the mean square error (MSE) between network's actual outputs $\hat{\mathbf{z}}_t$s and the desired outputs $\mathbf{z}_t$s with the cost function:

$$\begin{aligned} Q(\mathbf{\Theta}) &= \frac{1}{N} \sum_{t=1}^N (\mathbf{z}_t - \hat{\mathbf{z}}_t)^T (\mathbf{z}_t - \hat{\mathbf{z}}_t), \\ &= \frac{1}{N} \sum_{t=1}^N J_t(\mathbf{\Theta}) \end{aligned} \quad (7)$$

with $J_t(\mathbf{\Theta}) = (\mathbf{z}_t - \hat{\mathbf{z}}_t)^T (\mathbf{z}_t - \hat{\mathbf{z}}_t)$, and $\mathbf{\Theta} = \{\mathbf{\Theta}_1, \mathbf{\Theta}_2, \mathbf{\Theta}_g\}$. In implementation, at each time step t, we adaptively tune $\mathbf{\Theta}$ with a small step size along the descent direction of minimizing $J_t(\mathbf{\Theta})$. That is, we adjust $\mathbf{\Theta}$ by

$$\mathbf{\Theta}^{\text{new}} = \mathbf{\Theta}^{\text{old}} - \eta \frac{\partial J_t(\mathbf{\Theta})}{\partial \mathbf{\Theta}}|_{\mathbf{\Theta}^{\text{old}}}, \quad (8)$$

where η is a small positive learning rate.

In Sub-section 3.1, we have two parameter sets: $\mathbf{\Theta}_h = \{\mathbf{m}_j\text{s}, \mathbf{\Sigma}_j\text{s}, \mathbf{\Theta}_1, \mathbf{\Theta}_2\}$ in the hidden layer, and $\mathbf{\Theta}_g$ in the

output layer. Since $\mathbf{m}_j$s and $\mathbf{\Sigma}_j$s are both learned with the inputs $\mathbf{U}_t$s, whose values however depend on the parameter $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$. Consequently, we cannot learn $\mathbf{\Theta}_h$ and $\mathbf{\Theta}_g$ in a two separate steps, i.e., learn $\mathbf{\Theta}_h$ followed by learning $\mathbf{\Theta}_g$. Hence, we alternatively give out an iterative learning procedure for them as follows:

Step 1 Initialize $\mathbf{\Theta}_1$, $\mathbf{\Theta}_2$, and $\mathbf{\Theta}_g$.

Step 2 At current time step t with $1 \leq t \leq N$, by fixing $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$, we transform the augmented input $[\mathbf{x}_t, \mathbf{Z}_{t-1}]^T$ into $\mathbf{U}_t$. we then adjust $\{\mathbf{m}_j, \mathbf{\Sigma}_j\}$s with a small step size via an adaptive clustering algorithm. Here, we choose RPCCL [4] rather than the k-means or RPCL [8] upon the fact that not only it can automatically deactivate the extra hidden units without pre-determining the size of the hidden layer, but also it circumvents the selecting problem of the de-learning rate in the RPCL.

Step 3 Fixing $\{\mathbf{m}_j, \mathbf{\Sigma}_j\}$s and calculate out the output error $\mathbf{e}_t$, we adjust $\mathbf{\Theta}_1$, $\mathbf{\Theta}_2$, and $\mathbf{\Theta}_g$ with a small step size along the direction of minimizing the mean square error between the network's output and its desired value.

Step 2 and **Step 3** are implemented for each time step until all the parameters converge.

In the following section, we will give out the detailed learning algorithm by implementing the proposed recurrent RBF with using the ENRBF net in [5].

4. LEARNING ALGORITHM

We suppose that h_1 and h_2 in Eq.(2) are linear functions:

$$\begin{aligned} h_1(\mathbf{x}_t, \mathbf{\Theta}_1) &= \mathbf{A}_1\mathbf{x}_t + \mathbf{B}_1 \\ h_2(\mathbf{Z}_{t-1}, \mathbf{\Theta}_2) &= \mathbf{A}_2\mathbf{Z}_{t-1} + \mathbf{B}_2 \end{aligned} \tag{9}$$

with $\mathbf{\Theta}_1 = \{\mathbf{A}_1, \mathbf{B}_1\}$ and $\mathbf{\Theta}_2 = \{\mathbf{A}_2, \mathbf{B}_2\}$. That is,

$$\begin{aligned} \mathbf{U}_t &= \begin{pmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 \end{pmatrix} \begin{pmatrix} \mathbf{x}_t \\ \mathbf{Z}_{t-1} \end{pmatrix} + \begin{pmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{pmatrix} \\ &= \tilde{\mathbf{A}}\mathbf{u}_t + \tilde{\mathbf{B}}, \end{aligned}$$

where $\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 \end{pmatrix}$, $\mathbf{u}_t = \begin{pmatrix} \mathbf{x}_t \\ \mathbf{Z}_{t-1} \end{pmatrix}$, and $\tilde{\mathbf{B}} = \begin{pmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{pmatrix}$. Under the circumstances, the proposed RBF network will degenerate to the one in [1] when $\mathbf{A}_1 = \mathbf{A}_2$ and $\mathbf{B}_1 = \mathbf{B}_2 = \mathbf{0}$. Furthermore, we let $\mathbf{g}_j(\mathbf{U}_t)$ be a single-term polynomial term to fit the relations between each hidden units and its corresponding output unit as given in the ENRBF variant [5]. That is,

$$g_j(\mathbf{U}_t; \mathbf{\Theta}_g) = \mathbf{W}_j dg[sign(\mathbf{U}_t)]|\mathbf{U}_t|^{p_j} + \boldsymbol{\beta}_j \tag{10}$$

with

$$\begin{aligned} sign(\mathbf{U}_t) &= [sign(U_{t,1}), \ldots, sign(U_{t,d+qn})]^T \\ |\mathbf{U}_t|^{p_j} &= [|U_{t,1}|^{p_j}, |U_{t,2}|^{p_j}, \ldots, |U_{t,d+qn}|^{p_j}]^T \end{aligned} \tag{11}$$

where $\mathbf{\Theta}_g$ consists of $\mathbf{W}_j$s, p_js and $\boldsymbol{\beta}_j$s, $dg(\mathbf{U}_t)$ denotes the diagonal matrix whose $(i,i)^{\text{th}}$ element is $U_{t,i}$, $\mathbf{W}_j$ is an $n \times (d+qn)$ parameter matrix, and $\boldsymbol{\beta}_j$ is an $n \times 1$ constant vector. By putting Eq.(10) into Eq.(5), we then obtain

$$\hat{\mathbf{z}}_t = \sum_{j=1}^{k} [\mathbf{W}_j dg[sign(\mathbf{U}_t)]|\mathbf{U}_t|^{p_j} + \boldsymbol{\beta}_j] O_j(\mathbf{U}_t) \tag{12}$$

with $O_j(\mathbf{U}_t)$ given by Eq.(4). In this specific implementation, the parameters in this net are: $\{\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \mathbf{m}_j\text{s}, \mathbf{\Sigma}_j\text{s}\}$ in the hidden layer and $\{\mathbf{W}_j, p_j, \boldsymbol{\beta}_j\}$'s in the output layer. Consequently, the previous **Step 2** and **Step 3** can be explicitly given as follows:

Step 2 At current time step t with $1 \leq t \leq N$, given $\tilde{\mathbf{A}}$s and $\tilde{\mathbf{B}}$s, we do the three sub-steps:

Step 2.1 Calculate $\mathbf{U}_t$ by Eq.(2) and Eq.(9).

Step 2.2 Adjust $\mathbf{m}_j$s by the RPCCL [4]. That is, given the input $\mathbf{x}_t$, and for $j = 1, 2, \ldots, k$, let

$$I(j|\mathbf{x}_t) = \begin{cases} 1, & \text{if } j = c, \\ -1, & \text{if } y = r, \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$

with

$$\begin{aligned} c &= \arg\min_j \gamma_j \|\mathbf{x}_t - \mathbf{m}_j\|^2, \\ r &= \arg\min_{j \neq c} \gamma_j \|\mathbf{x}_t - \mathbf{m}_j\|^2, \end{aligned}$$

where $\gamma_j = \frac{n_j}{\sum_{r=1}^{k} n_r}$ is the relative winning frequency of $\mathbf{m}_j$ in the past, and $\mathbf{n}_j$ is the cumulative number of the occurrences of $I(j|\mathbf{x}_t) = 1$ in the past. Then, update the winner $\mathbf{m}_c$ (i.e., $I(c|\mathbf{x}_t) = 1$) and its rival only by

$$\mathbf{m}_\tau^{\text{new}} = \mathbf{m}_\tau^{\text{old}} + \Delta\mathbf{m}_\tau, \quad \tau = c, r$$

with

$$\begin{aligned} \Delta\mathbf{m}_c &= \alpha_c(\mathbf{x}_t - \mathbf{m}_c) \\ \Delta\mathbf{m}_r &= -\alpha_c p_r(\mathbf{x}_t)(\mathbf{x}_t - \mathbf{m}_r) \end{aligned}$$

where $p_r(\mathbf{x}_t) = \frac{\min(\|\mathbf{m}_c - \mathbf{m}_r\|, \|\mathbf{m}_c - \mathbf{x}_t\|)}{\|\mathbf{m}_c - \mathbf{m}_r\|}$, and α_c is a small positive learning rate.

Step 2.3 Update $\mathbf{\Sigma}_c$ only. Since the algorithm just involves its inverse, to save computing

costs and calculation stability, we here prefer to update $\mathbf{\Sigma}_c^{-1}$ directly by

$$\mathbf{\Sigma}_c^{-1^{\text{new}}} = \frac{\mathbf{\Sigma}_c^{-1^{\text{old}}}}{1-\eta_s}[\mathbf{I} - \frac{\eta_s \boldsymbol{\zeta}_t \boldsymbol{\zeta}_t^T \mathbf{\Sigma}_c^{-1^{\text{old}}}}{1-\eta_s+\eta_s \boldsymbol{\zeta}_t^T \mathbf{\Sigma}_c^{-1^{\text{old}}} \boldsymbol{\zeta}_t}],$$

where $\mathbf{I}$ is an identity matrix, $\boldsymbol{\zeta}_t = \mathbf{U}_t - \mathbf{m}_c$, and η_s is a small positive learning rate. To make the covariance learned smoothly, by rule of thumb, η_s should be chosen much smaller than η, e.g., $\eta_s = 0.1\eta$.

Step 3 After computing $\mathbf{e}_t$ by Eq.(6), we update $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ by

$$\begin{aligned} \tilde{\mathbf{A}}^{\text{new}} &= \tilde{\mathbf{A}}^{\text{old}} + \eta \Delta \tilde{\mathbf{A}} \\ \tilde{\mathbf{B}}^{\text{new}} &= \tilde{\mathbf{B}}^{\text{old}} + \eta \Delta \tilde{\mathbf{B}} \end{aligned}$$

with

$$\begin{aligned} \Delta \tilde{\mathbf{A}} &= dg(\Delta \tilde{\mathbf{B}} \mathbf{u}_t^T) \\ \Delta \tilde{\mathbf{B}} &= \sum_{j=1}^{k} [O_j(\mathbf{U}_t) p_j dg(|\mathbf{U}_t|^{p_j - 1}) \mathbf{W}_j^T \mathbf{e}_t] \\ &\quad + \sum_{j=1}^{k} c_{t,j} [\sum_{i=1}^{k} O_i(\mathbf{U}_t) \mathbf{\Sigma}_i^{-1} (\mathbf{U}_t - \mathbf{m}_i) \\ &\quad - \mathbf{\Sigma}_j^{-1} (\mathbf{U}_t - \mathbf{m}_j)], \end{aligned}$$

where $c_{t,j} = O_j(\mathbf{U}_t) tr[\mathbf{e}_t^T g_j(\mathbf{U}_t)]$. Also, we update $\mathbf{W}_j$s, p_js, and β_js by

$$\begin{aligned} \mathbf{W}_j^{\text{new}} &= \mathbf{W}_j^{\text{old}} + \eta \Delta \mathbf{W}_j \\ p_j^{\text{new}} &= p_j^{\text{old}} + \eta \Delta p_j \\ \beta_j^{\text{new}} &= \beta_j^{\text{old}} + \eta \Delta \beta_j \end{aligned}$$

with

$$\begin{aligned} \Delta \mathbf{W}_j &= O_j(\mathbf{U}_t) \mathbf{e}_t (|\mathbf{U}_t|^{p_j})^T dg[sign(\mathbf{U_t})] \\ \Delta p_j &= O_j(\mathbf{U}_t) \mathbf{v}_{t,j}^T dg[sign(\mathbf{U_t})] \mathbf{W}_j^T \mathbf{e}_t \\ \Delta \beta_j &= O_j(\mathbf{U}_t) \mathbf{e}_t, \end{aligned}$$

where $\mathbf{v}_{t,j} = [|U_{t,1}|^{p_j} \ln |U_{t,1}|, |U_{t,2}|^{p_j} \ln |U_{t,2}|, \ldots, |U_{t,d+qn}|^{p_j} \ln |U_{t,d+qn}|]^T$.

5. EXPERIMENTAL RESULTS

We conducted two experiments to demonstrate the performance of the proposed network in the recursive function estimation. In Experiment 1, we let the data be from a linear-separable function, while the data in Experiment 2 are from a linear non-separable one.

5.1. Experiment 1

We generated $1,100$ data points $\{\mathbf{x}_t, \mathbf{z}_t\}$s from the following equation:

$$y_t = 0.7(\sin x_t)^3 + 0.3 y_{t-1}^2 + \varepsilon_t, \quad t \geq 1 \qquad (14)$$

with $y_0 = 0$, where $x_t \in [1, 11]$, $\varepsilon_t \in [-0.1, 0.1]$ is white noise with uniformly distributed. We let the first $1,000$ data points be training set, and the remaining 100 points be testing set. In the experiment, we fixed $\eta = 0.001$ and set the size of hidden layer be $k = 5$. We measured the net's performance under the MSE criterion.

Figure 2 shows the net's performance curve on the training set, where it can be seen that the network tends to converge with $MSE = 0.0070$ after 15 epoches, i.e., repeatedly scan the training data set 15 times. After net's performance convergence, We then tested the network on the testing set with obtaining $MSE = 0.0147$. This result is comparable with that from the dual structural RBF in [5]. Furthermore, for comparison, we also implemented the RBF (denoted as RBF-R hereafter) with the current inputs and past outputs as an augmented inputs, but without considering their scales. We found that the network's performance deteriorate significantly with the MSE value on the testing set to be 0.0379.

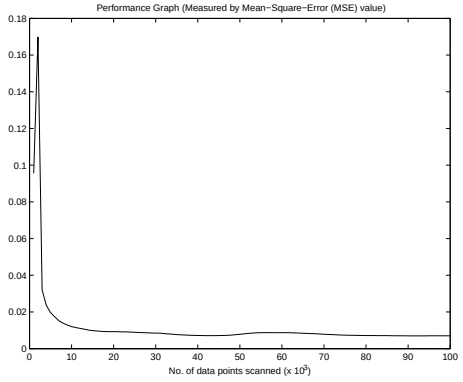

Figure 2: The performance curve of the proposed recurrent RBF network on the training data set in Experiment 1.

5.2. Experiment 2

We also generated $1,100$ data points $\{\mathbf{x}_t, \mathbf{z}_t\}$s that were from the following equation:

$$y_t = 0.4(\sin x_t)^3 + 0.3 y_{t-1}^2 + 0.3 y_{t-1} \cos(x_t) + \varepsilon_t, \quad t \geq 1,$$

where y_0, x_t, and ε_t were set in the same way as Eq.(14). The data distribution graph is shown in Figure 3. Under the same experimental environment as Experiment 1, we used

the first $1,000$ data points to train the network, whose performance learning curve is shown in Figure 4. It can be seen that its performance has converged after 30 epoches with $MSE = 0.0071$. We then used the remaining 100 data points to test the network's performance. The MSE value was 0.0057. Also, we tested the RBF-R on the same data set, and obtained $MSE = 0.0415$. It shows again that the proposed network outperforms the RBF-R. The former works well in the case of linear non-separable function estimation as well as linear-separable one, but the latter cannot.

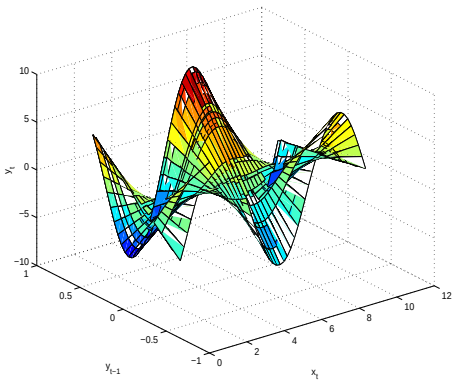

Figure 3: The graph of function value y_t versus x_t and y_{t-1}.

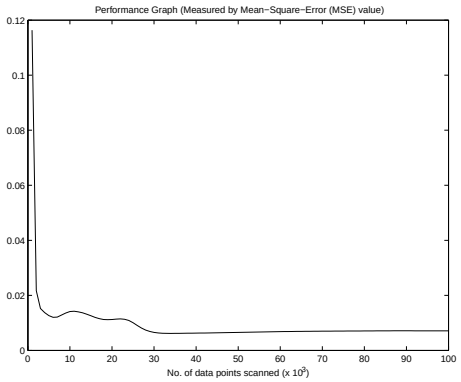

Figure 4: The performance curve of the proposed recurrent RBF network on the training data set in Experiment 2.

6. CONCLUSIONS

We have proposed a new recurrent RBF network, which takes the net's input and the past outputs as an augmented input, but introduces a scale tuner into the net's hidden layer to balance the different scales between inputs and outputs. This network adaptively learns the parameters in the hidden layer together with those in the output layer. We have implemented this network by using a variant of ENRBF [5] with its hidden units learned by the RPCCL algorithm [4]. The experiments have shown the outstanding performance of the proposed network in recursive function estimation.

7. REFERENCES

[1] S.A. Billings and C.F. Fung, "Recurrent Radial Basis Function Networks for Adaptive Noise Cancellation", *Neural Networks*, Vol. 8, No. 2, pp. 273-290, 1995.

[2] D.S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks", *Complex System*, Vol. 2, pp. 321-323, 1988.

[3] S. Chen, C.F.N. Cowan and P.M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", *IEEE Transactions on Neural Networks*, Vol. 2, pp. 302-309, 1991.

[4] Y.M. Cheung, "Rival Penalization Controlled Competitive Learning for Data Clustering with Unknown Cluster Number", to appear in *9th International Conference on Neural Information Processing* (ICONIP'02), Singapore, November 18-22, 2002.

[5] Y.M. Cheung and L. Xu, "A Dual Structural Radial Basis Function Network for Recursive Function Estimation", *Proceedings of* 8^{th} *International Conference on Neural Information Processing*, Vol. 2, pp. 1093-1097, 2001.

[6] J. Moody and J. Darken, "Fast Learning in Networks of Locally-tunned Processing Units", *Neural computation*, 1, pp. 281-294, 1989.

[7] P. Frascont, M. Cori, M. Maggini and G. Soda, "Representation of Finite State Automata in Recurrent Radial Basis Function Networks", *Machine Learning*, Vol. 23, pp. 5-32, 1996.

[8] L. Xu, A. Krzyzak and E. Oja, "Rival Penalized Competitive Learning for Clustering Analysis, RBF Net and Curve Detection", *IEEE Transactions on Neural Networks*, Vol. 4, No. 4, pp. 636–649, 1993.