A Comparison of Architectural Varieties in Radial Basis Function Neural Networks

Mehmet Önder Efe and Coşku Kasnakoğlu

Abstract— **Representation of knowledge within a neural model is an active field of research involved with the development of alternative structures, training algorithms, learning modes and applications. Radial Basis Function Neural Networks (RBFNNs) constitute an important part of the neural networks research as the operating principle is to discover and exploit similarities between an input vector and a feature vector. In this paper, we consider nine architectures comparatively in terms of learning performances. Levenberg-Marquardt (LM) technique is coded for every individual configuration and it is seen that the model with a linear part augmentation performs better in terms of the final least mean squared error level in almost all experiments. Furthermore, according to the results, this model hardly gets trapped to the local minima. Overall, this paper presents clear and concise figures of comparison among 9 architectures and this constitutes its major contribution.**

I. Introduction

DUE to their feature based operating principle, RBFNNs constitute a special class of neural network architectures. Many different forms of these types of networks have been studied in the past and many successful applications have been reported. However, works questioning the architectural varieties and associated performance measures were of special value for researchers and engineers practicing the approximation theory.

RBFNN have extensively been utilized for various purposes in the past. For instance, in [1], analysis of speech signals is studied with a recurrent RBFNN utilizing some delayed value of the hidden layer vector as external inputs. In [2], operational transconductance amplifier based electronic circuit implementation of a RBFNN is discussed and the standard architecture is considered with several function approximation examples. Horne et al [3] and Schilling et al [4] focus on issues of modeling nonlinear systems, and it is reported in [5] that RBFNN structures perform better than classical multilayer perceptron networks. A comparison of RBFNN structures is presented in [6], where Gaussian, raised cosine and multiquadric type basis functions are taken into consideration. It is concluded in [6] that raised cosine functions are more preferable than the others. Likewise, in [7], several alternatives of basis functions are considered, yet in [8], it is emphasized that the performance of the network is not strictly dependent upon the choice of neuronal nonlinearity. Several alternatives for basis functions are considered on a time series prediction problem. Cheung, in [9], introduces the concept of scale tuning in a recurrent RBFNN structure to balance the different scales in inputs and outputs. Cosine RBFNN structures are considered in [10] and the performance is assessed on some benchmarking data sets. Normalized Gaussian RBFNN are introduced by Bugmann, [11], where the output of the network is a weighted sum having the normalized values of hidden neuron outputs as weights. An identification problem is elaborated in [12], where the hidden layer evolves in time to improve performance. New neurons are added to the network based upon an algorithm of structural evolution. Lee et al [13] stress that if the map in hand displays constant values for some subsets of the input domain, conventional RBFNN approaches fail. As a remedy, the difference of two sigmoidal functions is proposed and the analysis of the improved performance is presented. Leonard et al [14], introduce a validity index concept to increase the extrapolation performance of RBFNNs, Ghosh and Nag present a comparison of RBFNNs with some immediate practical implications such as size selection, training method and so on, [15].

As seen from the above discussion, work towards improving the mapping performance of RBFNN is an active research involved generally with modifications in the nonlinearity that the hidden layer possesses. In this paper, our motivation is to figure out whether the structure changes play some considerable role in the final map to be realized. Since the RBFNN setting of knowledge representation is similar to support vector machines, [16], and fuzzy systems, [17], results applying this class of an approximator might have a potential applicability in the aforementioned tools originating from different viewpoints. To our best knowledge, no such study is reported for comparison of variations to this extent, and this is the originality and the contribution of the current paper.

II. Architectural Variants of RBFNNs

Although there are many alternatives, the RBFNN structures considered in this section utilize Gaussian basis functions due to their widespread use. The variable convention for the

Manuscript received November 5, 2007.

M.Ö. Efe is with TOBB Economics and Technology University, Turkey (phone: +90-312-292 4064; fax: +90-312-292 4180; e-mail: onderefe@etu.edu.tr).

C. Kasnakoğlu is with The Ohio State University, Electrical and Computer Engineering Department, OH43210, USA

978-1-4244-1821-3/08/$25.00©2008 IEEE

studied network architectures is described in Table 1, where a superscript k indicates the discrete time index.

A. Standard RBFNN Model

The structure of the model is depicted in Fig. 1(a). The output of the model is computed by (1), where the feature vector is comprised of the centers of the radial basis functions and every hidden neuron has its own spread (variance) parameter as seen in (2).

$$\tau^k = \sum_{i=1}^{R} y_i^k o_i^k \tag{1}$$

$$o_i^k = \exp\left(-\frac{\|\mathrm{u}-\mathrm{c}\|_2^2}{\left(\sigma_i^k\right)^2}\right) = \exp\left(-\frac{\sum_{j=1}^{m}\left(u_j^k - c_{i,j}^k\right)^2}{\left(\sigma_i^k\right)^2}\right),\ i=1,2,\cdots,R \tag{2}$$

TABLE I
NOMENCLATURE

Symbol	Meaning
τ^k	Output of the RBF network at time k
y_i^k	Output weight of the neuron i at time k
o_i^k $\left(\tilde{o}_i^k\right)$	Output of the neuron i in the hidden layer (context layer) at time k
$c_{i,j}^k$ $\left(\tilde{c}_{i,j}^k\right)$	Center of i-th basis function quantifying j-th input for the hidden layer (context layer)
$\sigma_{i,j}^k$ $\left(\tilde{\sigma}_{i,j}^k\right)$	Spread of i-th basis function qualifying j-th input for the hidden layer (context layer)
$\mu_{i,j}\left(u_j^k\right)$ $\left(\tilde{\mu}_{i,j}\left(u_j^k\right)\right)$	i-th radial basis function quantifying j-th input for the hidden layer (context layer)
R $\left(\tilde{R}\right)$	Number of RBF neurons in the hidden layer (context layer)
m	Number of external inputs (See Figure 1)

TABLE II
ADJUSTABLE PARAMETERS AND THE NUMBER OF ADJUSTABLE PARAMETERS FOR EACH MODEL

Section	Adjustable Parameters	#of Adjustables
II.A	$\left\{y_i^k, c_{i,j}^k, \sigma_i^k\right\}_{i,j=1}^{R,m}$	$R(m+2)$
II.B	$\left\{y_i^k, c_{i,j}^k, \sigma_{i,j}^k\right\}_{i,j=1}^{R,m}$	$R(2m+1)$
II.C	$\left\{y_i^k, c_{i,j}^k, \sigma_{i,j}^k\right\}_{i,j=1}^{R,m+1}$	$R(2m+3)$
II.D	$\left\{y_i^k, c_{i,j}^k, \sigma_{i,j}^k\right\}_{i,j=1}^{R,m+1}$	$R(2m+3)$
II.E	$\left\{y_i^k, c_{i,j}^k, \sigma_{i,j}^k\right\}_{i,j=1}^{R,m+C} \cup \left\{\tilde{c}_{i,j}^k, \tilde{\sigma}_{i,j}^k\right\}_{i,j=1}^{C,m}$	$R(2m+3)$
II.F	$\left\{y_i^k, c_{i,j}^k, \sigma_{i,j}^k\right\}_{i,j=1}^{R,m+R-1}$	$R(2R+2m-1)$
II.G	$\left\{y_i^k, c_{i,j}^k, \sigma_{i,j}^k\right\}_{i,j=1}^{R,m+R}$	$R(2R+2m+1)$
II.H	$\left\{c_{i,j}^k, \sigma_{i,j}^k\right\}_{i,j=1}^{R,m} \cup \left\{r_{i,j}^k\right\}_{i=1,j=0}^{R,m}$	$R(3m+1)$
II.I	$\left\{c_{i,j}^k, \sigma_{i,j}^k\right\}_{i,j=1}^{R,m} \cup \left\{r_{i,j}^k\right\}_{i=1,j=0}^{R,m}$	$R(3m+2)$

B. Extended RBFNN Model: Ellipsoid Basis Functions

The extended standard model assigns individual spread parameters to every radial basis function resulting in ellipsoid contours in the input space. The architecture of the network is as seen in Fig. 1(a) and the computation of the output value is the same as the standard model (See (1)). The hidden layer outputs are computed according to (3), where the chosen basis functions are Gaussians as described in (4).

$$o_i^k = \prod_{j=1}^{m} \mu_{i,j}\left(u_j^k\right),\ i=1,2,\cdots,R \tag{3}$$

$$\mu_{i,j}\left(u_j^k\right) = \exp\left(-\left(\frac{u_j^k - c_{i,j}^k}{\sigma_{i,j}^k}\right)^2\right) \tag{4}$$

C. Recurrent RBFNN: Feedback Connections Only in the Hidden Layer

The first type of recurrent RBF neural network model provides the past output of the neurons in the hidden layer back to itself. This is shown in Fig. 1(b). The output of every neuron and the network output are computed as given in (5) and (1) respectively. Clearly, this model exploits the strength of memory use in the feature space characterized by the neurons in the hidden layer.

$$o_i^k = \left(\prod_{j=1}^{m} \mu_{i,j}\left(u_j^k\right)\right) \times \mu_{i,m+1}\left(o_i^{k-1}\right),\ i=1,2,\cdots,R \tag{5}$$

D. Recurrent RBFNN: Feedback Connection from Network Output to Network Input

This type of recurrent model is depicted in Fig. 1(c), where the past value of the network response is fed back to the network as an input signal so that the RBFNN gains the ability of responding according to the past output. The output of a neuron in the hidden layer is computed by (6) and the network output is computed through (1).

$$o_i^k = \left(\prod_{j=1}^{m} \mu_{i,j}\left(u_j^k\right)\right) \times \mu_{i,m+1}\left(\tau^{k-1}\right),\ i=1,2,\cdots,R \tag{6}$$

E. Context Layered RBFNN Structure

In this structure, $\tilde{R}$ neurons form a context layer, which provides extra information to the hidden layer of the network. The output of a neuron in the context layer is computed by (7), where a basis function in the context layer is described in (8) and the expression for computing the output of a neuron in the hidden layer is given by (9).

$$\tilde{o}_i^k = \prod_{j=1}^{m} \tilde{\mu}_{i,j}\left(u_j^k\right),\ i=1,2,\cdots,\tilde{R} \tag{7}$$

$$\tilde{\mu}_{i,j}\left(u_j^k\right)=\exp\left(-\left(\frac{u_j^k-\tilde{c}_{i,j}^k}{\tilde{\sigma}_{i,j}^k}\right)^2\right) \tag{8}$$

$$o_i^k=\left(\prod_{j=1}^{m}\mu_{i,j}\left(u_j^k\right)\right)\times\left(\prod_{p=1}^{\tilde{R}}\mu_{i,m+p}\left(\tilde{\sigma}_p^k\right)\right),\ i=1,2,\cdots,R \tag{9}$$

F. Recurrent RBFNN: Feedback Connection from One Neuron in the Hidden Layer to All Others

A slightly extended form of the recurrent architecture shown in Fig. 1(b) is illustrated in Fig. 1(e), where the hidden neurons communicate with each other. As seen from the figure, a neuron in the hidden layer provides information to all other hidden neurons except itself. The output of a hidden neuron is computed according to (10) and the network output is computed by (1).

$$o_i^k=\left(\prod_{j=1}^{m}\mu_{i,j}\left(u_j^k\right)\right)\times\left(\prod_{j=1,\,j\neq i}^{R}\mu_{i,j+m}\left(o_j^{k-1}\right)\right),\ i=1,2,\cdots,R \tag{10}$$

G. Recurrent RBFNN: Hidden Layer Output is Fully Fed Back

The structure of the network is depicted in Fig. 1(f), where the hidden neurons communicate with each other. As seen from the figure, hidden layer output is fully fed back. The output of a hidden neuron is computed according to (11) and the network output is computed by (1).

$$o_i^k=\left(\prod_{j=1}^{m}\mu_{i,j}\left(u_j^k\right)\right)\times\left(\prod_{j=1,}^{R}\mu_{i,j+m}\left(o_j^{k-1}\right)\right),\ i=1,2,\cdots,R \tag{11}$$

H. RBF Network with Linear Part Augmentation

This type of network architecture separates the map being approximated into linear and nonlinear components to be realized individually by a linear part and a classical RBF network, respectively. The structure of the network realizing the nonlinear part is depicted in Fig. 1(a), where the output of the hidden neurons is computed by (3) and that of the entire network is computed by (12). Clearly, such a setting considers the linear terms in the Taylor expansion of a map described indirectly by the data separately and is therefore capable of realizing the deviations from a linear behavior.

$$\tau^k=\sum_{j=1}^{m}r_j^k u_j^k+\sum_{i=1}^{R}y_i^k o_i^k,\qquad u_0^k=1 \tag{12}$$

I. RBF Network with Functional Weights

As in the case with Takagi-Sugeno type fuzzy systems, we adopt linear functions of the input variables as the weighting entities, and end up with the representation in (13)

$$\tau^k=\sum_{i=1}^{R}y_i^k o_i^k,\qquad y_i^k=r_0^k+\sum_{j=1}^{m}r_j^k u_j^k \tag{13}$$

where r_{ij}s are the adjustable parameters of the weighting polynomials.

III. LEVENBERG-MARQUARDT TRANINING ALGORITHM

Since the algorithm is fast and applicable to all structures discussed in Section II, we choose LM algorithm as the tuning scheme for parameter optimization. The LM algorithm is an approximation to the Newton's method, and both of them have been designed to solve the nonlinear least squares problem [18]. Consider a neural network having K outputs, and N adjustable parameters denoted by the vector $\underline{\omega}$. If there are P data points (or patterns) over which the interpolation is to be performed, a cost function qualifying the performance of the interpolation can be given as $E\left(\underline{\omega}^k\right)=\sum_{i=1}^{P}\sum_{j=1}^{K}\left(d_{ji}^k-\tau_{ji}^k\left(\underline{\omega}^k\right)\right)^2$, where τ_{ji}^k is the observation at the j^{th} output of the structure in response to the i^{th} pattern, and d_{ji}^k is the corresponding target entry. The parameter update prescribed by Newton's algorithm is given as $\underline{\omega}^{k+1}=\underline{\omega}^k-\left(\nabla_{\omega}^2E\left(\underline{\omega}^k\right)\right)^{-1}\nabla_{\omega}E\left(\underline{\omega}^k\right)$ where the second derivative is computed as $\nabla_{\omega}^2E\left(\underline{\omega}^k\right):=2J\left(\underline{\omega}^k\right)^{\mathrm{T}}J\left(\underline{\omega}^k\right)$ and the first derivative is $\nabla_{\omega}E\left(\underline{\omega}^k\right):=2J\left(\underline{\omega}^k\right)^{\mathrm{T}}\underline{e}\left(\underline{\omega}^k\right)$ with $\underline{e}$ and J being the error vector and the Jacobian as given in (14) and (15) respectively, the Gauss-Newton algorithm can be formulated as $\underline{\omega}^{k+1}=\underline{\omega}^k-\left(J\left(\underline{\omega}^k\right)^{\mathrm{T}}J\left(\underline{\omega}^k\right)\right)^{-1}J\left(\underline{\omega}^k\right)^{\mathrm{T}}\underline{e}\left(\underline{\omega}^k\right)$, and the Levenberg-Marquardt update can be constructed as $\underline{\omega}^{k+1}=\underline{\omega}^k-\left(\mu I_{N\times N}+J\left(\underline{\omega}^k\right)^{\mathrm{T}}J\left(\underline{\omega}^k\right)\right)^{-1}J\left(\underline{\omega}^k\right)^{\mathrm{T}}\underline{e}\left(\underline{\omega}^k\right)$ where $\mu>0$ is a scalar design parameter. Clearly, the LM technique improves the rank deficiency problem of the matrix $J\left(\underline{\omega}^k\right)^{\mathrm{T}}J\left(\underline{\omega}^k\right)$.

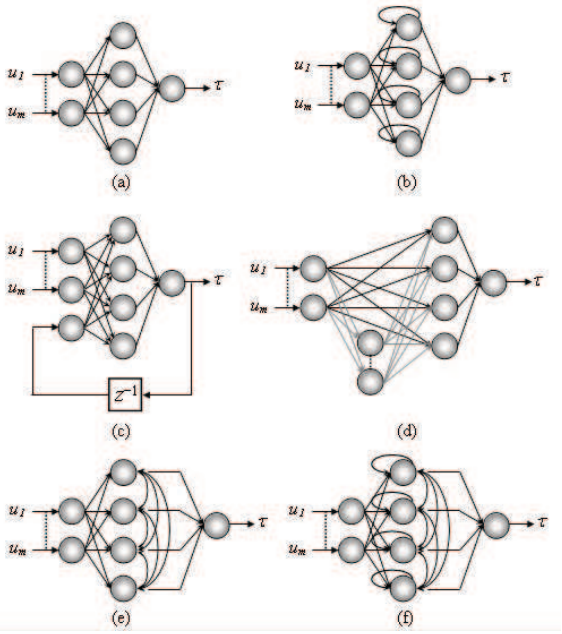


Fig. 1: RBF Network Architectures

Contrary to what is postulated in the case of error backpropagation (gradient descent), the framework of LM optimization technique utilizes the second order partial derivatives of the cost measure, and therefore extracts better path towards the goal in the adjustable parameter space. The cost of this is the computational burden primarily due to the matrix inversion taking place at each epoch.

$$\underline{e} = \begin{bmatrix} e_{11} & \cdots & e_{K1} & e_{12} & \cdots & e_{K2} & \cdots & e_{1P} & \cdots & e_{KP} \end{bmatrix}^{\mathrm{T}} \tag{14}$$

$$J(\underline{\omega}^k) = \begin{bmatrix} \frac{\partial e_{11}(\underline{\omega}^k)}{\partial \omega_1^k} & \frac{\partial e_{11}(\underline{\omega}^k)}{\partial \omega_2^k} & \cdots & \frac{\partial e_{11}(\underline{\omega}^k)}{\partial \omega_N^k} \\ \frac{\partial e_{21}(\underline{\omega}^k)}{\partial \omega_1^k} & \frac{\partial e_{21}(\underline{\omega}^k)}{\partial \omega_2^k} & \cdots & \frac{\partial e_{21}(\underline{\omega}^k)}{\partial \omega_N^k} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_{K1}(\underline{\omega}^k)}{\partial \omega_1^k} & \frac{\partial e_{K1}(\underline{\omega}^k)}{\partial \omega_2^k} & \cdots & \frac{\partial e_{K1}(\underline{\omega}^k)}{\partial \omega_N^k} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_{1P}(\underline{\omega}^k)}{\partial \omega_1^k} & \frac{\partial e_{1P}(\underline{\omega}^k)}{\partial \omega_2^k} & \cdots & \frac{\partial e_{1P}(\underline{\omega}^k)}{\partial \omega_N^k} \\ \frac{\partial e_{2P}(\underline{\omega}^k)}{\partial \omega_1^k} & \frac{\partial e_{2P}(\underline{\omega}^k)}{\partial \omega_2^k} & \cdots & \frac{\partial e_{2P}(\underline{\omega}^k)}{\partial \omega_N^k} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_{KP}(\underline{\omega}^k)}{\partial \omega_1^k} & \frac{\partial e_{KP}(\underline{\omega}^k)}{\partial \omega_2^k} & \cdots & \frac{\partial e_{KP}(\underline{\omega}^k)}{\partial \omega_N^k} \end{bmatrix} \tag{15}$$

IV. Process Models

A. Bioreactor Benchmark Problem

Bioreactor benchmark problem is considered for testing the performance of RBFNN models studied in this paper. The process is valuable as its dynamics consists of coupled and nonlinear differential equations, which display a rich set of behaviors containing limit cycles, attractors and repellers. The bioreactor is a tank containing a mixture of water, biological cells and nutrients. The tank is fed by an inflow rate ($w^k \in [0,2]$) and the same amount of mixture is removed from the tank, so that the reaction volume is kept constant. Though characterized by few state variables, the process is a good test bed for benchmarking the performance of numerical data based models. The state of the process is the amount of cells denoted by $c_1{}^k \in [0,1]$ and the amount of nutrients denoted by $c_2{}^k \in [0,1]$. The two difference equations obtained after the discretization of the continuous time process are given in (16)-(17), where γ=0.48and β = 0.02 are the nutrient inhibition parameter the growth rate parameter, respectively, [19-20].

$$c_1^{k+1} = c_1^k + \Delta\left(-c_1^k w^k + c_1^k \left(1 - c_2^k\right) e^{c_2^k/\gamma} \right) \tag{16}$$

$$c_2^{k+1} = c_2^k + \Delta\left(-c_2^k w^k + c_1^k \left(1 - c_2^k\right) e^{c_2^k/\gamma} \frac{1+\beta}{1+\beta-c_2^k} \right) \tag{17}$$

In above, Δ=0.01sec. is the discretization period. The performance of a RBFNN structures is evaluated in predicting the cell mass (c_1^k) under various operating conditions. An initial condition is chosen randomly and (16)-(17) are executed for 50 discrete time instants. This operation is repeated for 50 different initial conditions yielding a total of 2500 training samples. A similar strategy is followed for generating the validation data set containing 750 samples. The input vector of the network contains c_1^k, c_2^k, w^k and the auxiliary variables if any (See Fig. 1).

B. A Model Studied by Narendra and Parthasarathy, [21]

Although the bioreactor benchmark problem displays a quite rich set of dynamical properties, due to the discretization in (16)-(17), the network is guided dominantly by the previous response of the variable being predicted. For very small Δ, this becomes more apparent in (16)-(17). We therefore perform the tests on two different models reported by many research studies. The model considered in [21] is given in (18). The training data is generated exactly as explained above. The input vector of the network contains c_1^k, w^k and the auxiliary variables if any (See Fig. 1). This model is a good candidate to investigate learning of nonlinearities influencing the behavior dominantly in a discrete map.

$$c_2^{k+1} = \frac{c_1^k}{1 + \left(c_1^k\right)^2} + \left(w^k\right)^3 \tag{18}$$

For this model, the operating range is chosen as $c_1{}^k \in [-1,1]$ and $w^k \in [-1,1]$.

V. Modeling Results

During the performance comparison phase, we have chosen R = 5, i.e. 5 hidden neurons for each RBFNN structure and C = 2 for the model assuming the context layer. This is deliberate as we wish to compare the performance of small sized networks, which are generally more preferable than those introducing computational intensity for better performance. Although one may follow a structural optimization procedure to find out the best network configuration, the goal here is to present a comparison between networks that are fairly compact for real time applications. Initially, the parameters of all RBFNN structures are assigned to small random numbers.

The models introduced in Section II are executed 100 times and the final Mean Squared Error (MSE) is recorded for each trial and this array is sorted so that a conclusion can be drawn from the distribution seen by the facet with Exp. Number as the horizontal axis (See Fig. 2). The type of the RBFNN structure is coded by the subsection number in Section II and the logarithm of the final MSE levels for all 100 experiments are plotted for each RBFNN structure. The data used during the training contain noisy samples where the noise sequences are uniformly distributed signals from

the interval ±0.01. The results for both processes are depicted in Fig. 2.

According to the results seen in the top subplot of Fig. 2, we infer that all models find the achievable minimum of the cost function but the model in Subsection II-H is the most successful one whereas the model in Subsection II-C is the least successful in terms of the trends. Looking at Fig. 2(a), we interpret the curves as follows: RBFNN model in Subsection II-H finds the global minimum in all 100 experiments while the model in Subsection II-C generally fails but finds a good value for few of the 100 experiments. The distribution for the other methods is interpreted in the same manner. According to the results obtained with bioreactor benchmark problem, the experiments with best performance values indicate almost indistinguishable final MSE value seen on the front facet. On the other hand, the results obtained with the model in (18) let us have the same best and worst RBFNN structures with an emphasis that the RBFNN in Subsection II-B is in the top three and the model in Subsection II-I is in the worst three in both cases.

In Figs. 3 and 4, we illustrate the generalization performance observed with the use of model introduces in Subsection II-H. In the left column of the figures, the results in the time domain are presented. The top left subplots illustrate the desired values together with the response of the neural network. Since the two trends are close to each other, the discrepancy between them is illustrated in the bottom left subplots. Similarly, the evolution of the MSE level is depicted in the top right subplot, which indicates the trend during training. The bottom right subplot gives the trend of the checking MSE level which is used for stopping the training. According to the presented figures, clearly the predictions are very accurate for the unseen data. This particularly demonstrates the usefulness of the mentioned approach and together with the statistical facts presented in Fig. 2(a) and (b), one infers that a model with these settings will converge quickly and it will provide good generalization of the presented training data.

VI. Conclusions

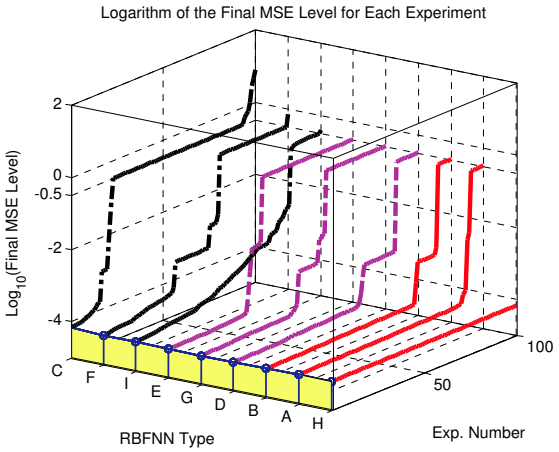
This paper presents nine structural variants of RBFNNs in predicting the behavior of nonlinear processes. Many experiments have been carried out to unfold the convergence properties with each approach statistically and these are presented graphically. It is seen that the model separating the linear part of the process is the best performing approach (Subsection II-I). The model considered in Subsection II-B, which is the extended standard model, performs satisfactory in both cases. When the poor ones are taken into consideration, the model providing the past value of the hidden layer outputs to the hidden layer itself (Fig. 1(b), Subsection II-C) yields the worst results in both cases and the model introduced in Subsection II-I is within the worst three of the considered models for both processes. Although we make no claim that the deduced results will necessarily extend to other problems, it is seen that the studied problems are good test beds for distinguishing the learning and generalization performances of considered RBFNN structures which may assume many different structural configurations influencing the performance substantially.

Furthermore, the final values of the MSE levels are now different as seen from the front facet of the box on the right. For this case the model in Subsection II-E is not counted as a good model as it is more likely to get trapped to the local minima than those yielding higher MSE levels. This is clear in Fig. 2(b).

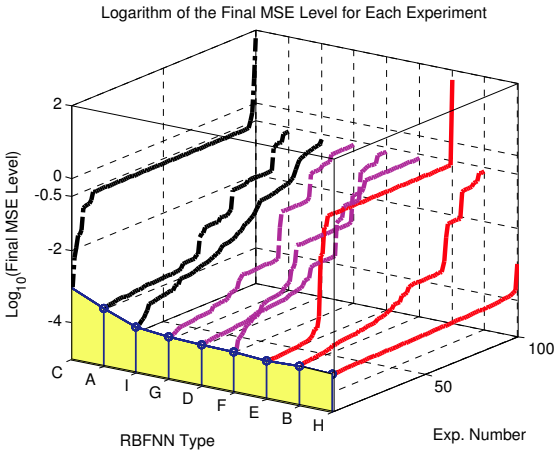
VII. Acknowledgements

This work is supported in part by TÜBİTAK Project No 107E137 and in part by TOBB ETU BAP Program 2006/04.

The first author gratefully acknowledges the facilities of TOBB ETU Library.



(a)



(b)

Fig. 2:(a) Results for the bioreactor benchmark problem (b) Result for the nonlinear process in (18)

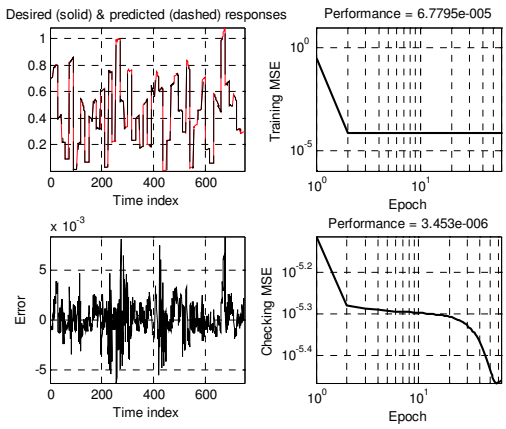


Fig. 3 Results for the bioreactor benchmark problem

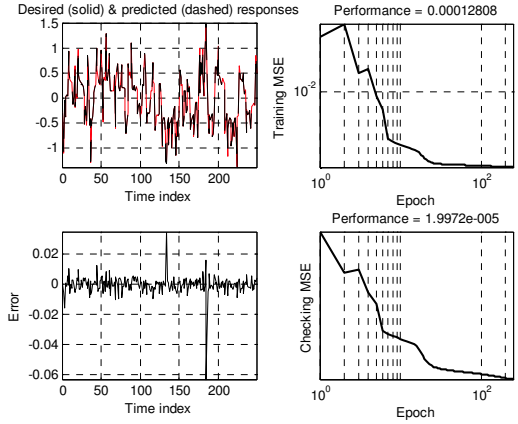


Fig. 4 Results for the nonlinear process in (18)

REFERENCES

[1] Varoglu, E. and Hacioglu, K.: A Recurrent Neural Network Speech Predictor Based On Dynamical Systems Approach, IEEE Eurasip Worskshop on Nonlinear Signal and Image Processing (NSIP'99), June 20-23, Antalya, Turkey, 1999.

[2] Lucks, M.B. and Oki, N.: A Radial Basis Function Network (RBFN) for Function Approximation, 42nd Midwest Symposium on Circuits and Systems, August 8-11, Las Cruces, New Mexico, pp.1099-1101, 1999.

[3] Horne, B.G. and Giles, C.L.: An Experimental Comparison of Recurrent Neural Networks, Advances in Neural Inf. Processing Systems, 7, pp.697-704, The MIT Press, 1995.

[4] Schilling, R.J., Carroll J.J. and Al-Ajlouni, A.F.: Approximation of Nonlinear Systems with Radial Basis Function Neural Networks, *IEEE Trans. on Neural Networks*, v.12, no.1, pp.1-15, 2001.

[5] Park, J.-W., Harley, R.G. and Venayagamoorthy, G.K.: Comparison of MLP and RBF Neural Networks Using Deviation Signals for On-Line Identification of a Synchronous Generator, IEEE Power Eng. Society Winter Meeting, Jan. 27-31, pp.274-279, 2002.

[6] Kobetski, A., Coulomb, J.-L., Costa, M.C., Marechal, Y. and Jönsson, U.: Comparison of Radial Basis Function Approximation Techniques, COMPEL: The Int. J. for Computation and Mathematics in Electrical and Electronic Engineering, v.22, no.3, pp.616-629, 2003.

[7] Webb, A.R. and Shannon, S.: Shape-Adaptive Radial Basis Functions, *IEEE Trans. on Neural Networks*, v.9, no.6, pp.1155-1166, 1998.

[8] Chen, S., Cowan, C.F.N. and Grant, P.M.: Orthogonal Least Squares Algorithm for Radial Basis Function Networks, *IEEE Trans. on Neural Networks*, v.2, no.2, pp.302-309, 1991.

[9] Cheung, H.-M., A New Recurrent Radial Basis Function Network, Proc. Of the 9th Int. Conf. on Neural Information Processing, Nov. 18-22, pp.1032-1036, 2002.

[10] Karayiannis, N.B. and Randolph-Gips, M.M.: On the Construction and Training of Reformulated Radial Basis Function Neural Networks, *IEEE Trans. on Neural Networks*, v.14, n.4, pp.835-846, 2003.

[11] Bugmann, G.: Normalized Gaussian Radial Basis Function Networks, *Neurocomputing*, v.20, pp.97-110, 1998.

[12] Obradovic, D.: On-Line Training of Recurrent Neural networks with Continuous Topology Adaptation, *IEEE Trans. on Neural Networks*, v.7, no.1, pp. 222-228, 1996.

[13] Lee, C.-C., Chung, P.-C., Tsai, J.-R. and Chang, C.-I.: Robust Radial Basis Function Neural Networks, *IEEE Trans. on Systems, Man and Cybernetics-Part B*, v.29, no.6, pp.674-685, 1999.

[14] Leonard, J.A., Kramer, M.A. and Ungar, L.H.: Using Radial Basis Functions to Approximate a Function and Its Error Bounds, *IEEE Trans. on Neural Networks*, v.3, n.4, pp. 624-627, 1992.

[15] Ghosh, J. and Nag, A.: An Overview of Radial Basis Function Networks, in Radial Basis Function Neural Network Theory and Applications (Eds.) R.J. Howlerr and L.C. Jain, Physica-Verlag, 2000.

[16] Gunn, S.R.: Support Vector Machines for Classification and Regression, ISIS Technical Report, Univ. of Southampton, United Kingdom, 1998.

[17] Jang, J.-S.R. Sun, C.-T. and Mizutani, E.: *Neuro-Fuzzy and Soft Computing*, PTR Prentice-Hall, 1997.

[18] Hagan MT and Menhaj MB. Training feedforward networks with the Marquardt algorithm. *IEEE Trans. on Neural Networks*, 1994; NN-5:989-993.

[19] Ungar, LH.. A Bioreactor Benchmark for Adaptive-Network Based Process Control. *Neural Networks for Control*, W. T. Miller III, R. S. Sutton, P. J. Werbos, Eds, MIT Press, pp.387-402, 1997.

[20] Efe MÖ., Abadoğlu E and Kaynak O. A Novel Analysis and Design of a Neural Network Assisted Nonlinear Controller for a Bioreactor. *Int. Journal of Robust and Nonlinear Control*, v.9, no:11, pp.799-815, 1999.

[21] Narendra KS and Parthasarathy K. Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Transactions on Neural Networks*, 1, 4-27, 1990.