A New Approach Using Temporal Radial Basis Function in Chronological Series

Mustapha Guezouri and Larbi Mesbahi
Faculty of Electrical Engineering, University of Science and Technology, Algeria

Abstract: *In this paper, we present an extended form of the radial basis function network called temporal- radial basis function network. This extended network is used in decision rules and classification in spatio-temporal domain applications like speech recognition, economic fluctuations, seismic measurements and robotics applications. We found that such a network complies, with a relative ease, to constraints such as capacity of universal approximation, sensibility of node, local generalisation in receptive field, etc. For an optimal solution based on a probabilistic approach with a minimum of complexity, we developped two temporal radial basis function models. Application to the problem of Mackey-Glass time series, it has revealed that temporal radial basis function models are very promising, compared to traditional networks.*

Keywords: *Temporal RBF, classification, spatio-temporal, speech recognition, robotics applications.*

Received August 9, 2007; accepted Novembre 5, 2007

1. Introduction

One limitation of static network is their inability to respond to temporal pattern in the hidden node outputs. To respond to these patterns, the networks must have delay elements within each of its layers. The benefits of internal delays were demonstrated in [1] which is the dynamic approach, especially in signal prediction, wherein the input to the network is a time varying signal and the desired output is a prediction of the signal at a fixed lag. Other examples are speech recognition and signal production, when the output autonomously follows a desired trajectory [1].

First, day and davenport [1] have introduced Back-Propagation Through Time (BPTT) in chaotic signal prediction tasks based on the Mackey-Glass differential delay equation. Lin, ligomenides and dayhoff [2] have developped the Adaptive Temporal Delay Neural Network (ATDNN) [3], which consists of changing the delay time during learning and which gave good results in problems of eight and zero form. In problems which require information about the next event, it combines the Recurrent Neural Network (RNN) firstly and adds time delay connections progressively [4]. Another method named Long Short Term Memory (LSTM) applied to time series benchmark problems does not even require RNN at all, because all relevant information about the next event is in fact conveyed by a few recent events contained within a small time window [5].

Our method integrates the time aspect in the RBF neural network. The novelty is that, compared to other methods described above, this approach is not a black box, so we can alter the kernels in sub-neural networks. In addition, we can apply a Bayesian classifier which introduces the prior-probability, cost of punishment in the case of rejection of misclassification. It also incorporates basic aspects of static Radial Basis Function (RBF) in approximation, denseness, uniqueness of interpolation and convergence rate [6]. Other extensions such as moving centres, weighted norm and different types of basis function and multiple scales were also considered. These criteria provide a useful theoretical framework for investigating radial basis function networks and learning algorithms [6]. A variety of approaches for training radial basis function networks have been developed, most of which can be divided into two stages: 1 learning the centres and field receptor in the hidden layer, 2 learning the connection weights from the hidden layer to the output layer [7]. In this sense, we have proposed our model which integrates the time parameter in the network, in object to resolve some forgotten features in standard model, like memory state, dynamic measures, recalling phases...etc.

We therefore developped two models; first we introduce the delay time only to input neurones, second we insert delay time in both input and hidden neurones. In sections 2 and 3 we describe the TRBF neural network and the different models characterising this approach, mainly the occurrence of time in hidden and input layers disjointedly. In last section, we compare our approach with standard temporal neural networks especially in application to the famous Mackey-Glass chaotic time series.

2. T-RBF Approach

2.1. Definition

The standard RBF can be trained to accomplish pattern recognition tasks with complex non linear boundaries, but are limited to processing static patterns-patterns that are fixed rather than time-varying in nature [2]. The temporal RBF, like ATDNN, LSTM..., have been developped to overcome this limitation. Networks with this capability can play an important role in applications that are naturally time-varying and dynamic.

Also, like classical RBF, their goal is to approximate a desired approximation by a collection of functions, named kernels [8, 9]. A kernel is characterised by a centre *Ci* and receptive field r, and can be chosen by k-means clustering or vector quantification.

All these parameters can be taken in account to introduce the Bayesian probabilistic classifier and prior knowledge about the problem. Moreover, we can combine this approach with other techniques like Hidden Markov Models (HMM) by using the generating probabilities. In general the temporal discrimination function of class k is written in the following form [2]:

$$y_k(tn) = \sum_{j=1}^{ml} w_{jk}\,\varphi\Big(\sum_{l=0}^{pl} w_j(l)\,x(tn-l) + b_j\Big) + b_0 \tag{1}$$

Next, we describe the network architecture and learning algorithm.

2.2. Network Architecture

First, we have the following architectures; see Figures 1 and 2:

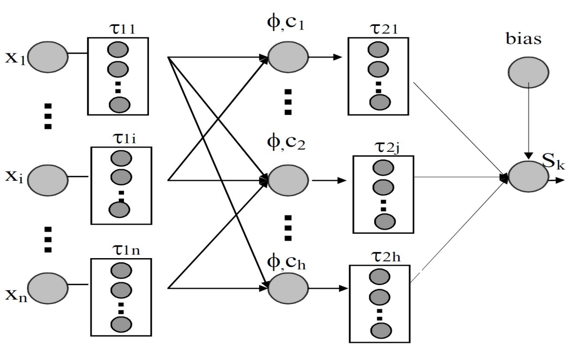


Figure 1. General model for T-RBF.

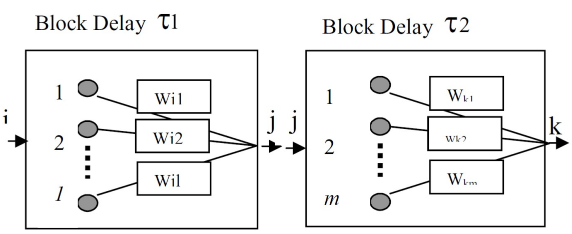


Figure 2. Block delay (t_1, t_2) representation.

Now, we define the k^{th} output activation function as follow:

$$S[k] = \varphi\Big[W_{ok} + \sum_{j=1}^{m}\sum_{i=1}^{h} Yh[i][j]\,W_{kij}\Big] \tag{2}$$

where φ is an activation function (linear or sigmoid...*etc*.) and W_{ok} is the bias. The kernel function is:

$$Yh[i][j] = \varphi_j \,.\, \tau(\|Ci - X\|) \tag{3}$$

where *i*=1..*h, h* is number of centres, *j*=1..*m, m* is the size of hidden time delay (τ_2). Dimension of C_i = dimension of *X*= *n* x l, *n*= number of features of input vector, l is time delay size of input vector (τ_l), θ_j is the kernel function characterised by time delay *j*. Second, we must unfold the architecture of the Figure 1, by using the algorithm described in next section.

2.3. Unfolding Network

The main problem is how estimating the activation functions for each block delay elements in hidden layer. One solution, is reducing this block to only one delay. For this object we apply the Network Unfolding Algorithm (NUA) developped by Lin [10]. This algorithm allows us to simplify the architecture of the Figure 1 and reduces the complexity of computation. For this purpose, we must apply two phases:

Phase 1: in this phase we must respect the following steps:

Step 1: Unfold inputs
For each hidden node j do in parallel:
For each input node i do in parallel:
Duplicate the new input nodes.
Spread these nodes horizontally next to each original input node.
Move the original connections to next input nodes correspondingly.
Retain the weight and time-delay on each connection.

Step 2: Re-adjust input time lag
For each hidden node j do in parallel:
Remove the time-delays between input and hidden units.
Set the input values as the signal value vector transmitted from i^{th} input node to j.

Step 3: Unfold hidden nodes
For each output node k do in parallel:
For each hidden node j do in parallel:
Duplicate the new input nodes.
Spread these nodes horizontally next to each original hidden node.
Move the original connections to new hidden nodes correspondingly.

Retain the weight and time-delay on each connection.
For each newly created node do in parallel:
Copy the whole branch which associates with the original hidden node in step 2
Connect to that new node as its branch and retain the weights.
Step 4: Re-adjust input time lag by
For each hidden node j and its newly cretaed node from step 3 do in parallel:
Remove the associated time delays between hidden and outputs units.
Re-adjust the input node time lag such that each the input node of each branch takes the signal value vector delayed by τ_2 *transmitted from node i to j.*

To simplify well understanding the algorithm, we apply the NUA algorithm on the scheme represented in Figure 3 and in result we obtain a new unfolded scheme showed on Figure 4.

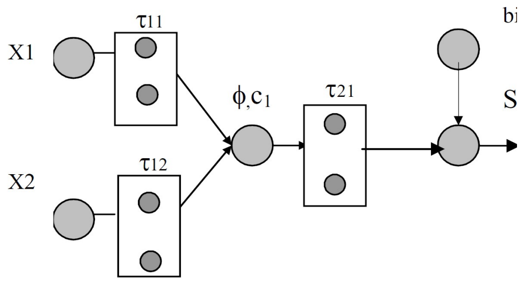


Figure 3. Initial network with two blocks (t_1,t_2).

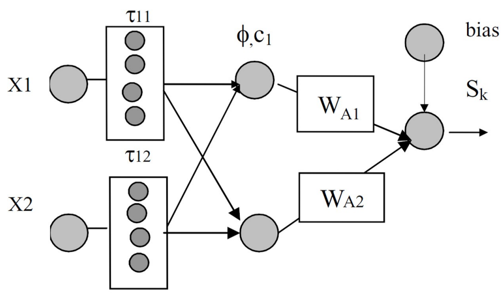


Figure 4. Deletion of block delay t_2 by unfolding.

Phase 2: this phase consists to fuse two hidden neurones in one hidden neurone, in order to represent only one centre at each iteration (see the transition between Figures 4 and 5). This phase allows us to apply the Orthogonal Least Square (OLS) learning algorithm [6] described in next section. The OLS method is an incremental learning technique such: at each iteration, it creates a new node representing a new cluster until obtaining a sufficient number of kernels.

In summary, phases 1 and 2 yield the following tasks:

- The connection between the hidden layer and the output layer is based only on the weights, without time delay.
- The connection between the input layer and the hidden layer is based only on time delay, without weights.

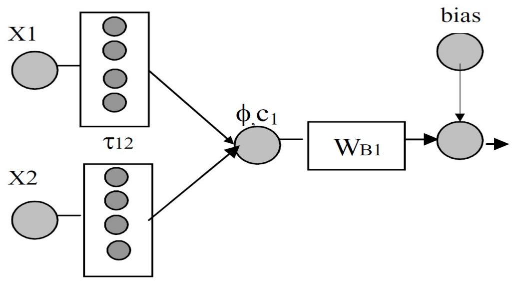


Figure 5. Resulting network after fusion.

2.4. Learning Algorithm

The OLS algorithm has been applied for learning step. We suppose that the kernel function ϕ is fixed and have the same form in all hidden neurones, the initial set must be fixed. Therefore this algorithm allows us to do an incremental learning [8].

- First, it makes a linear separation between the input layer and the hidden layer and creates the hidden neurones automatically [11], with application of Gram-Schmidt orthogonalisation.
- Second, the synaptic weights between hidden and output layers are calculated by the least square method.

We consider the RBF network as a particular case of a linear regression model [6], defined as follow:

$$d(t)=\sum_{i=1} P_i\phi_i+\varepsilon(t) \tag{4}$$

where *d(t)* is the desired output at time t; ϕ_I are the search parameters. $\varepsilon(t)$ is the approximation error of *d(t);*
$P_i(t)=P_i(x(t))$ the fixed functions of $x\ (t)$. The calculated function by the RBF network is the same described in equation 4, the analogy is:

$$d=P.\phi+E \tag{5}$$

where d is the desired output vector, $d=[d(1)\ ...\ d(N)]^T$, N is then number of examples of learning basis; M correspond to the initial number of centers. P is the matrix of the hidden layer outputs: $P=[P1..\ PM]$. P_i is the vector of i^{th} hidden cell $P_i=[P_i(1)...\ P_i(N)]^T$. θ is the weights of output layer: $\theta=[\theta_1....\ \theta_M]$. *E correspond* to error between the estimated and desired outputs: $E=[\varepsilon(1)..\ \varepsilon(N)]^T$.

The resolution of equation 5 is a trivial problem. The solution vector θ can be defined by the mean square method. The original idea of OLS method resides in transformation of P matrix to matrix with orthogonal columns one by one. The orthogonalisation of columns *Pi* can be obtained by the decomposition of P matrix in two matrices W and A :

$$P=W.A \tag{6}$$

where W of size $N \times M$ is the orthogonal image of P matrix., A of size $M \times M$ is superior triangular matrix, it contains the orthogonalisation coefficients.
The A matrix is defined as follow:

	1	$\alpha_{1,2}$				$\alpha_{1,M}$
	0	1				$\alpha_{2,M}$
	…	…	…	…	…	…
A=	…	…	…	1	$\alpha_{M-2,M-1}$	…
	0	0		0	1	$\alpha_{M-1,M}$
	0	0		0	0	1

The breeding space from the P_i vectors is the same space breeding by the Wi vectors, and the equation 5 can be rewritten in new form

$$d = W * G + E \tag{7}$$

where $G=A.\ \theta$ is the researching solution. Noting that

$$H = W^T.WE \tag{8}$$

Since the columns of W matrix are orthogonal one by one, H is diagonal matrix with h_i elements we have

$$h_i = w_i^t w_i = \sum_{j=l} w_{ij} w_{ji}, \quad l \le i \le M \tag{9}$$

This propriety makes useful the OLS method, for the following reason: the orthogonal solution G is calculated by:

$$G = H^{-1}.W^t d \tag{10}$$

We can calculate the G_i by

$$g_i = \frac{w_i^t d}{w_i^t w_i}, \quad l \le i \le M \tag{11}$$

This signifies that g_i elements of orthogonal solution G are dependant only on wi column, in other words they depend on orthogonal image of calculated output for each centre. This part defines the quotient of approximation error reduction, introduced by each wi vector and can be expressed by

$$[err]_i = \frac{G_i^2 w_i^t w_i}{d^t d}, \quad l \le i \le M \tag{12}$$

This equation is used to construct the RBF network in iterative manner. Beginning with initial set of M centres, the network is constructed to each iteration, by adding a centre which have the $[err]_i$ maximal, and we take the correspondent G_i. For each iteration, we calculate the A and W elements by

$$\alpha_{j,k}^i = \frac{w_j^t * pi}{w_j^t * wj} \tag{13}$$

$$w_k^i = Pt - \sum_{j=l}^{j=k-l} \alpha_{j,k}^i * wj \tag{14}$$

we use the criterion Akaike to stop the iterations:

$$l - \sum_{i=l}^{M} err_i > \varepsilon \tag{15}$$

In the end of iterations, we calculate θ_l (the synaptic weights), by the following system:

$$G = A.\theta \tag{16}$$

3. T-RBF Models

3.1. Model 1

In this model, we consider the network with bloc delay $\tau 1$, without bloc delay $\tau 2$ as shown in Figures 1 and 2, it means that we take into account the 1 delays of measures of input layer with simply one connection between hidden and output layers, for each hidden neurone.

3.2. Model 2

In this case, we consider the network with bloc delay $\tau 1$ and bloc delay $\tau 2$, it means that we take in consideration the "l" delays of measures of input layer, with m delays of hidden layer, representing the m history states for each hidden neurones.

4. Experimentation

4.1. Application of Mackey-Glass Series

The Mackey-Glass series is a case of typical dynamic system [4], which describes the production of white blood cells, and can be generated from the delay differential equation:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{l + x^c(t-\tau)} - bx(t) \tag{17}$$

with a=0.2, b=0.1, c=10, τ=17 and $x(t)$=0.8 for t<=17 in our application, for τ>17 the series become chaotic and for τ=17 is quasi periodic series [4, 5, 2].

4.2. Data base

We have constructed a base of 300 (examples and targets), each example being represented by $<x(t_i), x(t_i-\tau)>$ and his corresponding target is $<x(t_i+1)>$, where

$$x(t_{i+1}) = x(t_i) + \frac{dx(t)}{dt}\Big|_{t=t_i} \tag{18}$$

4.3. Parameters

- Complexity: we have used the second model of TRBF, for the length of bloc delay τ_1 is 1 and the length of bloc delay τ_2 is 2. We need to one input layer node, one output layer node and sufficiently some hidden layer nodes.
- Akaike criterion: this criterion fixed at 0.001 is used in OLS Algorithm. First part it stops the iterations

based on quadratic error, second part it determine the complexity of hidden layer (number of kernel functions characterised by theirs centres and receptor field). In our case, we have obtained 3 hidden nodes.

- Gaussian kernel: this kernel function is characterised by their asymptotic properties and getting accuracy in learning and generalisation phases. It is presented by the cluster (meaning centre) and spreading deviation.

4.4. Evaluation

- Mean Square Error (MSE): to take part of veracity and the goodness results we have computed the MSE between calculated outputs from TRBF and targets from the learning base. We have obtained a score of MSE equal to 0.02.
- Comparison: we show the results obtained by the application of analytic differential equation towards the results obtained by TRBF approach, we see the different cases in the following Figures 6 and 7. For example in Figure 6 the curve of $x(t)$ towards $x(t-1)$ in our approach falls approximately on the same position of curve obtained by analytical differential equations.

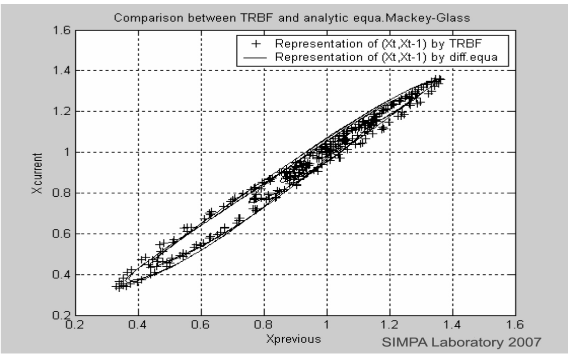


Figure 6. State space (current and previous states).

We conclude this work with a comparative table which summarise the results obtained by different temporal neural network approaches [5] against our model which have given a good results with minimum of parameters.

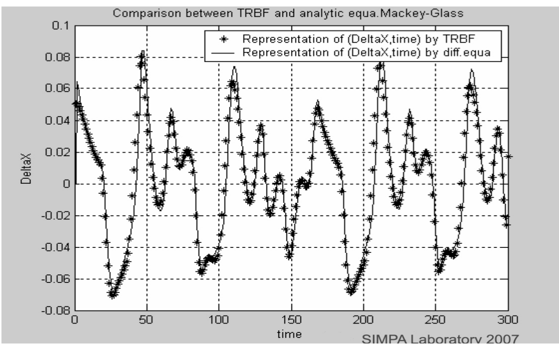


Figure 7. Strange attractors on state space.

We conclude this work with a comparative table which summarise the results obtained by different temporal neural network approaches [5] against our model which have given a good results with minimum of parameters. We define the abstract terms given in Table 1:

- ATNN: Adaptive temporal neural network.
- SOM: Self Organising Map.
- MLP: Multi Layers Perceptron.
- AMB: An improved Memory Based regression.
- LSTM: Local Short Term Memory.
- TRBF: Temporal Radial Basis Function.

Temporal Methods	Units	Para-meters	NMSE		
			T=2	T=6	T=84
ATNN	20	120	-	0.005	-
SOM	-	10x10	-	0.013	0.06
MLP	4	25	-	0.0511	0.46
AMB	-	-	-	-	0.054
LSTM	4	113	0.0214	0.1184	0.47
TRBF	5	9	0.0198	0.005	-

Table 1. Position of TRBF among different temporal methods.

5. Conclusion

Inspect to the comparative table and all figures we can say that the spatio-temporal estimator based on temporal radial basis function is effective and accurate in time modelisation, with a minimum of complexity. The advantage of this work is to show that a complicated problem based on the Mackey-Glass time series prediction can be recognised by such a simple TRBF architecture. We thus conclude from this study that the TRBF is a good tool to tackle temporal signal processing and estimation problems.

The network accomplishes the identification of background model of Mackey-Glass delay differential equation again with high accuracy (NMSE =0.0198). This proposed neural architecture can be useful and promising in biomedical prediction tasks, recognition of trajectories from moving targets, motion of visual images, robotic application and speech recognition.

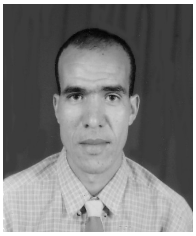
References

[1] Bonne R. and Cruciano M., "An Algorithm for the Addition of Time-Delayed Connections to Recurrent Neural Network," *in Proceedings of the European Symposium on Artificial Neural Networks (ESANN'2000)*, pp. 293-298, Bruge, 2000.

[2] Day S. and Davenport M., "Continuous-Time Temporal Back-Propagation with Adaptable Time Delays," *IEEE Transactions on Neural*

Networks, vol. 4, no. 2, pp. 348-354, 1993.
[3] Gers F., Eck D., and Schmidhuber J., "Applying LSTM to Time Series Predictible Through Time Window Approaches," *Technical Report*, *IDSIA-IDSIA-22-00, Institute Dalle Molle di studi Sull Intelligenza Artificiale*, Switzerland, 1999.
[4] Haykin S., *Neural Networks: A Comprehensive Foundation*, Prentice Hall, USA, 1999.
[5] Karyianis N., "Reformulated Radial Basis Function Neural Network Trained by Gradient Descent," *IEEE Transactions on Neurals Networks*, vol. 10, no. 3, pp. 657-669, 1999.
[6] Lin D. and Dayhoff J., "Network Unfolding Algorithm and Universal Spatio-Temporal Function Approximation," *Technical Research Report TR95-6*, *Institute for System Research ISR, University of Maryland*, College park MD, 1994.
[7] Lin DT., "The Adaptive Time Delay Neural Network Characterization and Application to Pattern Recognition," *Prediction and Signal Processing Thesis Report PHD*, University of Maryland, 1994.
[8] Mekkakia Z., Mesbahi L., and Lakehal M., "Initialising of RBF Centers by SOM Algorithm," *International Conference on Modeling and Simulation in Technical and Social Sciences*, pp. 717-723, Spain, 2002
[9] Titsias M. and Likas A., "Shared Kernel Model for Class Conditional Density Estimation," *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 987-996, 2001
[10] Wohler C. and Anlauf J., "Real Time Object Recognition on Image Sequences with Adaptable Time Delay Neural Network Algorithm: Application to Autonomous Vehicles," *Image and Vision Computing Journal*, vol. 19, no. 9, pp. 593-618, 2001.
[11] Zheng G. and Billings S., "Radial Basis Function Network Configuration Using Mutual Information and the Orthogonal Least Squares Algorithms," *IEEE Transactions on Neurals Networks*, vol. 9, no. 9, pp. 1619-1637, 1996.

Mustapha Guezouri received the engineer degree from the national Algerian Institute of Telecommunications in 1988. He received the Master and PhD degrees in electrical engineering in 1990 and 2007, respectively from the Science and Technology University in Oran.



Larbi Mesbahi obtained his engineering diploma in computer science in 1994 from Oran University, and MSc in electronic option vision and pattern recognition in 2001. Currently, in final phase of preparing the PhD in computer science. His research interests include pattern recognition application, neural network, mobile robot applications, speech synthesis and speech recognition.