ON-LINE STABLE NONLINEAR MODELLING BY STRUCTURALLY ADAPTIVE NEURAL NETS

Shaohua Tan, Yi Yu

Department of Electrical Engineering
National University of Singapore
10 Kent Ridge Crescent, Singapore 0511

Abstract

This paper proposes a neural net based *on-line* scheme for modelling discrete-time multivariable nonlinear dynamical systems. Taking the advantage of structural features of RBF (Radial-Basis-Function) neural nets, the method approaches the modelling problem by setting up a coarse RBF model structure in the light of the spatial Fourier transform and spatial sampling theory, then devising appropriate on-line algorithms to carry out refinements for both the RBF net structure and the associated weights. Main convergence results are established in the paper along with the analysis backing up the structure initialization and adaptation. The effectiveness of the scheme is illustrated with an simulation example.

1 Introduction

The problem to be addressed in this paper is the following: Let an input sequence $\{u_t\}$ and the corresponding output sequence $\{y_t\}$ $(t = 0, 1, \ldots)$ of an unknown nonlinear discrete-time multivariable dynamical system of the form

$$y_{t+1} = f(y_t, y_{t-1}, \ldots, y_{t-k+1}, u_t, u_{t-1}, \ldots, u_{t-l+1}) \tag{1}$$

be given. Where k and l denote the order, respectively, the time-delay of the system; $f(\cdot)$ is an unknown nonlinear vector function satisfying a few conditions to be detailed later; u_t is the q-dimensional input vector and y_t the m-dimensional output vector of (1) at discrete time instance t. Further assume that both u_t and y_t are confined within a bounded and closed set $\mathcal{C}$ for all t. We wish to construct a neural net model for (1) over $\mathcal{C}$ *on-line* so that when t progresses, the output sequence of the RBF model with u_t as its input will asymptotically converge to y_t.

This is the on-line version of the neural net modelling problem that has raised considerable interest recently [1] [2]. Along a conventional line of thought, the formulation naturally suggests a two-step approach to its solution: A suitable network structure needs to be postulated first. Certain parameters in the model structure should then be adapted on-line using given inputs and outputs to meet a pre-defined error criterion. This approach has been followed in many recent works including [2] [4] [5]. One of the obvious drawbacks of the approach is the difficulty in setting up an *appropriate* neural network structure. The structural determination often requires considerable insight into an unknown dynamical system to be modelled, which is simply unavailable in many situations.

This paper intends to propose a different modelling approach that incorporates the idea of on-line structural adaptation. Viewing a neural net as an assembly of standardized simple nonlinear components (neurons), the process for capturing the structure of an unknown system can be done by gradually adapting the number of neurons until the modelling accuracy can be attained by purely adjusting the weights.

The type of neural nets chosen for the approach is Radial-Basis-Function (RBF) nets for their universal approximation property as well as an important structural feature of the so-called linear-in-the-weights. This later feature allows the weights to be related directly to the output error, thus allowing relatively simple and stable weight updating schemes to be constructed. In addition, the structure initialization and adaptation can be guided by the spatial Fourier transform and spatial sampling theory.

We shall present the details of the on-line algorithms for adapting both the RBF net structure and the associated weights. Main convergence results will be proven along with considerations for the network structural initialization and adaptation. Simulation and comparison are carried out to evaluate the effectiveness of the scheme.

2 The formulation of the on-line modelling problem

This section will briefly discuss the structural feature of RBF neural nets, which will then be related to the formulation of the on-line modelling problem.

A RBF neural net is structured as a two-layer feedforward net with the first layer of RBF neurons and the second layer of linear neurons. The inputs to a RBF net are directly connected to the RBF neurons in the first layer. The outputs of these RBF neurons are then connected to the linear neurons in the second layer through weighted links.

The transfer function of the ith RBF neuron is of the form $z_i(x) = r(\sum_j (x^j - p_i^j)^2/(\sigma_i^j)^2)$, in which $z_i(x)$ is the ith output of the RBF neuron, and $r(\cdot)$ is a radial basis function which is chosen in the present paper to be the Gaussian function $z_i(x) = exp(-\sum_j (x^j - p_i^j)^2/(\sigma_i^j)^2)$. x^j is the jth element of the input vector x of dimension ℓ; p_i^j the jth element of the vector p_i which is also of dimension ℓ and is called the center for the ith RBF neuron. The positive scalar σ_i^j is the jth element of the width vector σ_i for the ith neuron. Both p_i and σ_i can be different for different neurons. Note that other choices of radial basis functions are equally valid for our modelling technique to be discussed in the ensuing sections.

The transfer function for the ith linear neuron is simply $y_i = \sum_{j=1}^{n} w_{ij} z_j(x)$, where y_i is the output of the ith neuron; w_{ij} is the weight from the jth RBF neuron to the ith linear neuron; and n is the number of RBF neurons in the first layer. With m such linear neurons in the second layer, we can rewrite this transfer function in the following matrix form

$$y = Wz(x), \tag{2}$$

where $y = [y_1\ y_2 \cdots y_m]^T$, $z(x) = [z_1(x)\ z_2(x) \cdots z_n(x)]^T$ and W is an $m \times n$ matrix of the form with w_{ij} being its ijth element. Equation (2) represents an explicit decomposition of the parameters involved in a RBF net into two parts: All the weights as in W, and all such structural parameters as the number of neurons n, the centers p_i's and the widths σ_i's in $z(x)$.

The usefulness of the RBF net for modelling a dynamical system lies in the so-called universal approximation property [3]. With the preceding notations, this property states that for any ϵ and an arbitrary nonlinear continuous function $f(\cdot)$ on a compact set $\mathcal{C}$, there always exists a weight matrix $\bar{W}$ and a vector $\bar{z}(x)$ such that $\max_{x \in \mathcal{C}} |f(x) - \bar{W}\bar{z}(x)| < \epsilon$. In the framework of system modelling, this property translates into the following: If we set $x = x_t \triangleq [y_t, y_{t-1}, \ldots, y_{t-k+1}, u_t, u_{t-1}, \ldots, u_{t-l+1}]^T$ and $y = y_{t+1}$ in (2), the existence of a weight matrix $\bar{W}$ and the structural vector $\bar{z}(x)$ are guaranteed so that the system (1) can be represented as

$$y_{t+1} = \bar{W}\bar{z}(x_t) \tag{3}$$

within a given error tolerance. The central task of the on-line nonlinear modelling problem is, therefore, to devise appropriate on-line algorithms that can find $\bar{W}$ and $\bar{z}(\cdot)$ so as to construct the model (3) for (1). This implies that both $\bar{W}$ and all the structural parameters n, p_i and σ_i will have to be established on-line.

For the modelling problem to be well-defined, the unknown nonlinear system given in (1) needs to meet a few conditions including the continuity and time-invariance of $f(\cdot)$ on a compact set $\mathcal{C}$, and the BIBO stability of (1). The former condition ensures the validity of the universal approximation property, the latter the existence of the compact set $\mathcal{C}$ which is often interpreted as the region of operation for the nonlinear system to be modelled.

Note that the nonlinear system (1) may not be fully representable by specific input and output sequences. Unlike linear systems where the concept of persistent excitation ensures all modes of a system to be excited, there is no corresponding theory for nonlinear systems. Therefore the best we can hope for is to model the part of the system that is being excited by particular input sequences during operation. This makes sense especially for on-line modelling where we normally have no control over the type of inputs. Thus, the constructed model is valid in the sense that it represents the part of the system that has been excited by past inputs. We shall see late how this idea is reflected in the simulation example.

3 On-line structural adaptation

We note that the spatial Fourier transform and spatial sampling theory constitutes a theoretical framework for the structural set-up of RBF nets in a compact set $\mathcal{C}$. By confining the approximation to $\mathcal{C}$ and properly truncating an unknown function $f(\cdot)$ on $\mathcal{C}$, [5] shows that the structure of a RBF net can be determined directly from a uniform grid drawn on C for a given error bound. [4] has developed a systematic structural determination method along this line. Thus, instead of trying to rely on pure heuristics or blind search, the problem of determining RBF neural net structure can be theoretically justified as setting up a sufficiently fine grid of RBF neurons with their centers located at the crossings of the grid, with the widths chosen properly. Such a choice then guarantees the existence of a set of weights to meet a pre-specified modelling precision.

The preceding way of structural determination is, however, inherently off-line. In what follows, we shall propose an on-line structural determination scheme within the same theoretical framework. Our scheme is built on the following considerations.

To start with, we need to assume the existence of a compact set $\mathcal{C}$ in the form of hypercube for (1) to be modelled. We also need to initialize the RBF net structure by choosing a grid with the size Δ_j and, consequently, σ_i^j using the rules given in [4]. In accordance with the theory, Δ_j needs to be sufficiently small to ensure the desired accuracy. From the on-line standing-point, however, the exact nature of $\mathcal{C}$ is not needed as we are confined within $\mathcal{C}$ by the BIBO stability of the system. Note that the initial grid only has its nominal value and will be modified in the on-line process. After the initialization, the subsequent structure formation is divided into two phases: Generation and deletion.

During the generation phase, only neurons with the centres close to the on-line data will have their weights adapted. This is substantiated by the fact that the Gaussian function decreases dramatically outside certain effective area. Thus, those nodes that are centered sufficiently far (relative to their radii) from the current input vector will contribute almost nothing to the formation of the unknown function. To be precise, we can completely neglect the contribution of the nodes that are centered further than $(\kappa+1)\Delta_j$ from the input vector while still maintaining a given error bound. The effect of this generation rule is that for each updating cycle, only the weights for a few neurons (depending on the maximum value of σ_i, this number is around 9) are updated. Note that this local updating rule considerably reduces the computational demand to almost a minimal, and makes the computation load independent from the size of the RBF net. Yet the convergence analyses of the weight update is not affected, as we shall see later.

During the deletion phase, a neuron at a grid point will be deleted on the basis of the following considerations. The on-line nature of the problem dictates that the operation of the system may not be able to fill up the entire $\mathcal{C}$ that has be covered with a grid and designated as the region of operation. This implies the existence of such grid points that have never been closer to input data. Consequently, if there are neurons placed at these grid points, their corresponding weights will never get updated, and thus do not contribute to the model. These neurons will be deleted.

Introducing $\rho_j = (\kappa+1)\Delta_j$, the modeling error ϵ_f, the fixed time period T, we can summarize the preceding discussions into the following algorithmic steps for the determination of all the structural parameters.

Step 1 Choose a set of appropriate Δ_j to define a rectangular grid on $\mathcal{C}$. Define the set of all structural neurons be $\mathcal{S}_d$, which has no element initially.

Step 2 For each training datum at time t, we define a current neuron set $\mathcal{S}_t$ that contains all the neurons with their centers located less than ρ_j from the training datum. Carry out the weight update on the neurons in $\mathcal{S}_t$.

Step 3 Update $\mathcal{S}_d$ as follows: Remove all the neurons in $\mathcal{S}_d$ that have not got their weights updated for a prescribed peorid T, and $\mathcal{S}_d = \mathcal{S}_d \cup \mathcal{S}_t$.

Step 4 Empty $\mathcal{S}_t$. Repeat *Step 2* till the modelling error drops below a prescribed limit.

An obvious advantage of the algorithm is that it appears to be able to bring down considerably the number of neurons in the RBF model. This is especially important for modelling MIMO systems. Another advantage is that there is no need to have the explicit knowledge about the operation region $\mathcal{C}$. The knowledge of $\mathcal{C}$ is obviously crucial in the off-line situation. For our on-line algorithm, however, the knowledge of $\mathcal{C}$ does not affect the structural adaptation which is done locally, and is under the assumption that the grid only exists nominally.

4 On-line weight determination

We now consider the problem of determining the weight matrix $\bar{W}$ on-line. Note that although only part of the weight matrix is updated at any time instance due to the local structural generation nature, we can treat the weight updating as if it is done directly on the whole matrix W. This will be justified as we proceed to prove the stability.

Let $\hat{y}_t$ be the neural net estimate of y_t, and

$$\hat{y}_t = W_{t-1} z^T(x_{t-1}), \tag{4}$$

where W_{t-1} is the weight matrix at $t-1$. Then $e_t = \hat{y}_t - y_t$ is naturally defined to be the modelling error at time t. Let the weight updating formula be given as

$$W_t = W_{t-2} - \frac{\alpha e_{t-1} z(x_{t-2})}{\beta + z^T(x_{t-2}) z(x_{t-2})}. \tag{5}$$

The subsequent lemma summarizes a few important features of this updating formula.

Lemma 1 *The on-line formula (5) has the following properties*

(i). $||W_t - \bar{W}|| \le ||W_{t-1} - \bar{W}||$;

(ii). $\lim_{t\to\infty} \frac{|e_t|}{\sqrt{\beta + z^T(x_{t-1}) z(x_{t-1})}} = 0$;

(iii). $\lim_{t\to\infty} ||W_t - W_{t-k}|| = 0$ *for any finite* k.

The proof for the lemma will be omittd for lack of space. This lemma helps to establish the following convergence theorem.

Theorem 1 *The on-line weight updating formula (5) guarantees that* $\lim_{t\to\infty} |e_t| = 0$.

Although the formula (5) is designed for the whole matrix W, it actually includes our local weight updating scheme as a special case. For the local updating scheme corresponds to setting to zero those entries in $z(x)$ that have negligible values. Thus, the convergence analysis is perfectly valid for our structural varying situation.

5 An Illustrative example

We use a simple example to demonstrate the use of the on-line modelling scheme discussed in the preceding sections for modelling a MIMO system.

In this example, the following 2-input and 2-output system is used to generate the input and output sequences for the modelling scheme.

$$\begin{bmatrix} y_1(t+1) \\ y_2(t+1) \end{bmatrix} = \begin{bmatrix} \frac{y_1(t)}{1+y_2^2(t)} \\ \frac{y_1(t)y_2(t)}{1+y_2^2(t)} \end{bmatrix} + \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \tag{6}$$

We choose the following periodic excitation as the system input

$$u(t) = \begin{bmatrix} 0.5\sin(2\pi t/25) \\ 0.5\cos(2\pi t/25) \end{bmatrix} \tag{7}$$

Δ_i^j is chosen to be 0.31,0.31,1.20,1.08. Using the rules given in [4], σ_i^j can be set to be 0.21,0.21,0.80,0.72.

After about 500 structural and weight updates, the modelling error converges to within the mean-square error of 0.01 as shown in Figure 1. The computational efficiency is among the important advantages of the scheme.

6 Conclusions

We have presented a RBF neural net based on-line technique for modelling nonlinear discrete-time multivariable dynamical systems. The salient features of the technique are the use of structural adaptation and the localization of the weight adaptation that guarantees the stability yet reduces the computation to a near minimal. The structural adaptation feature also contributes to reduction in the RBF neural net size.

References

[1] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Net.*, Vol. NN-1, pp. 4-27, 1990.

[2] S. Chen, S. A. Billings and P. M. Grant, "Nonlinear system identification using neural networks," *Int. J. Control*, Vol. 51, pp. 1191-1213, 1990.

[3] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, Vol.3, pp.246-257, 1990.

[4] S. Tan, J. Hao and J. Vandewalle, "Stable and efficient neural network modeling of discrete multi-channel signals," *ESAT Internal Report*, Dept. Electrical Eng., KUL, Belgium, 1993.

[5] R.M. Sanner and J. E. Slotine (1992), "Gaussian networks for direct adaptive control," *IEEE Trans. Neural Net.*, Vol. 3, pp. 837-863.

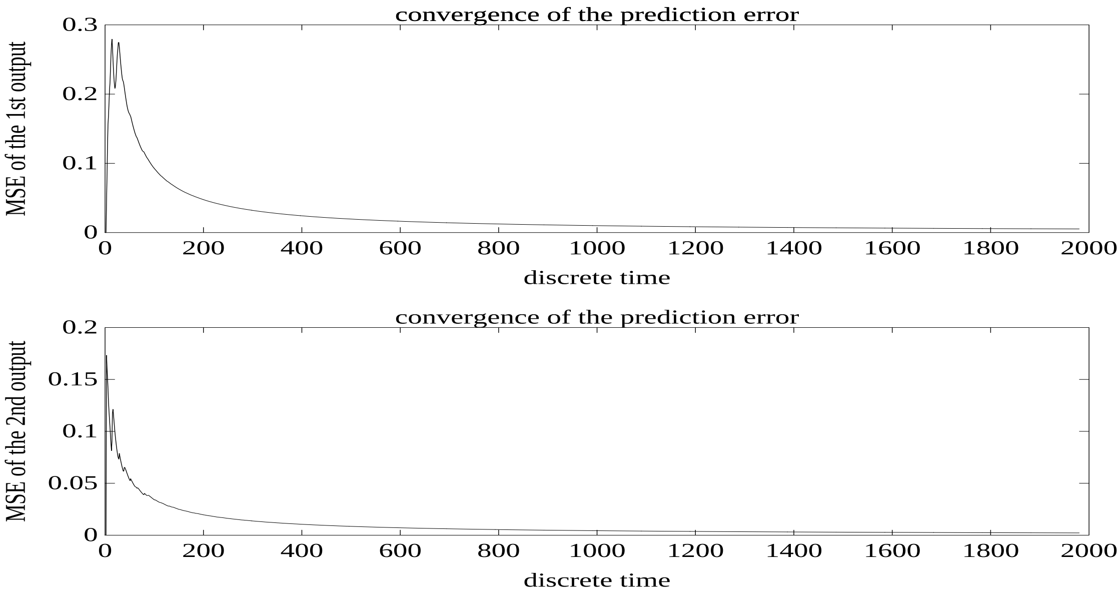

Figure 1: the convergence of the modelling error for the Example.

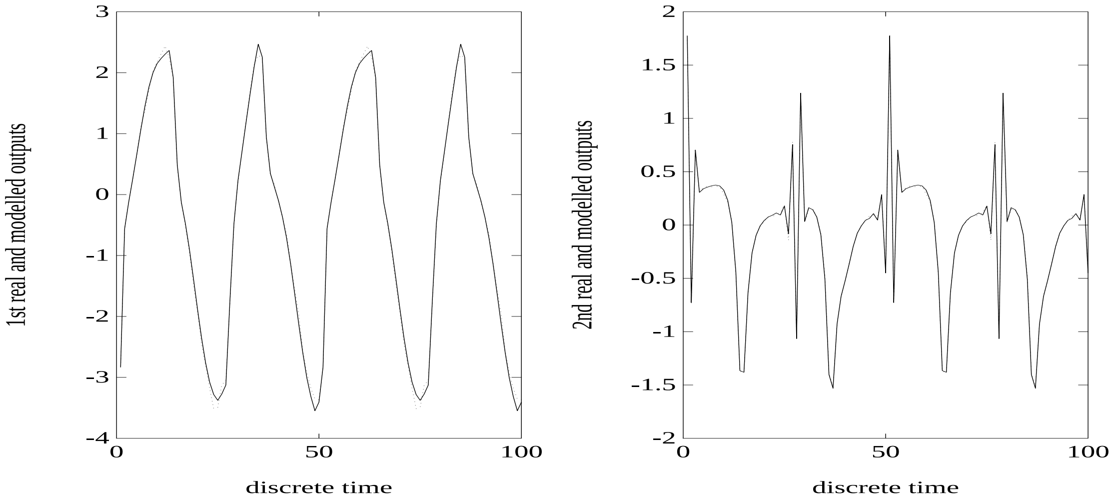

Figure 2: The modelling accuracy the Example after 500 structural and wights updates. (a) y_1 and (b) y_2. Solid line: the exat output; dotted line: the RBF model output.