Proceedings of the 29th Conference
on Decision and Control
Honolulu, Hawaii • December 1990

MODEL FOLLOWING USING MULTI-LAYER PERCEPTRONS

D. Hush, C. Abdallah, and B. Horne
Department of Electrical Engineering and Computer Engineering
University of New Mexico
Albuquerque, NM, 87131 USA.

Abstract

A model following controller is proposed for discrete nonlinear systems using a recursive multi-layer perceptron (MLP). The MLP network contains dynamics and is able to minimize the error between the plant and a desired model in many cases.

I. Introduction

In general, the control of nonlinear systems requires a compensator which performs some nonlinear mapping [1]. In practice, however, an analytical form of the compensator is difficult to find and implement. On the other hand, neural networks are being billed as massive computational networks which can implement nonlinear mappings from one linear vector space to another [3]. The proposed scheme attempts to minimize the error between the desired model and plant outputs by using an MLP neural network as a feedback compensator (see Fig. 1). If the plant is known, the weights of the MLP can be updated using a standard gradient search technique (back-propagation through the plant), an approach which has been recently suggested in many recent control applications [4-6]. By using the classic MLP algorithm [7] however, these controllers were limited to the case of static feedback laws. For output feedback, such a restriction will undeniably prevent us from dealing with many interesting problems. Recently, recurrent networks which contain dynamics have also been suggested [8] but have not caught on in the control community. In [5] and [6], a state was reconstructed by delaying the outputs and the inputs to the system so that the MLP could be used to implement a static mapping from a state space to an input space. The present work uses a similar idea but differs from [6] because it includes the controller directly in the feedback structure and has a different implementation of the dynamic backpropagation algorithm than the one in [5].

In section II, we describe the nonlinear control problem and present the neural network algorithm. An example is given in section III and our conclusions are given in section IV.

II. Recursive Multi-Layer Perceptrons

Our efforts in this research have concentrated on including the controller as an integral part of the update algorithm for a model-following control problem, rather than identifying the system or its inverse (compare to direct adaptive control). This was motivated by the fact that the estimated nonlinear system or its inverse, unlike their linear counterparts, are mathematical models only. They do not give useful information when the actual system is subjected to different inputs, except when the training input is white noise [5]. The model-following problem was selected for its generality. For example, by replacing the model with a unity gain one recovers the inverse identification problem [2,5]. Many control objectives may be achieved by selecting an appropriate model and trying to minimize the error between the plant/controller output and that of the model. The block diagram of Fig. 1 represents the general structure of the problem discussed and the controller is described next.

The controller in Fig. 1 consists of two subnetworks, a recursive net and a nonrecursive one. The outputs of these two networks are summed to produce the controller output,

$$u(k) = v_N(k) + v_R(k) \tag{1}$$

Each of these networks is a multi-layer perceptron. If we let $\mathbf{v}_i(k)$ represent the output vector of the *ith* layer then signals are propagated through the networks according to

$$\mathbf{v}_i(k) = \mathbf{f}(\,\mathbf{W}_i\mathbf{v}_{i-1}(k) + \mathbf{w}_{i0}) \tag{2}$$

$\mathbf{W}_i$ represents the weight matrix connecting the outputs of layer $i-1$ to the nodes in layer i, and $\mathbf{w}_{i0}$ are the bias weights for layer i. $\mathbf{f}(\cdot)$ is a vector function which applies the standard sigmoid function, $f(\gamma) = (1+e^{-\gamma})^{-1} - 0.5$, to each component.

The input vector to the nonrecursive net consists of current and delayed values of the input signal, that is for the nonrecursive net $\mathbf{v}_0(k) = \mathbf{d}(k)$, where

$$\mathbf{d}(k)^T = \left[d(k),\, d(k-1), \ldots,\, d(k-o_n)\right] \tag{3}$$

The input to the recursive net consists of previous net outputs, that is for the recursive net $\mathbf{v}_0(k) = \mathbf{u}(k-1)$ where

$$\mathbf{u}(k-1)^T = \left[u(k-1),\, u(k-2), \ldots,\, u(k-o_r)\right] \tag{4}$$

o_n and o_r are the orders of the nonrecursive and recursive net respectively. Note that both nets have only one output node so that $\mathbf{W}_L$ is a row vector and w_{L0} is a scalar. In addition the output node is linear for both nets, that is $f(\gamma) = \gamma$ for the output node.

Our goal here is to find the set of network weights that minimizes the total error,

$$E = \frac{1}{2}\sum_{k=0}^{N} e^2(k) = \sum_{k=0}^{N}(r(k)-y(k))^2 \tag{5}$$

where $r(k)$ is the output of the model and $y(k)$ is the output of the plant. The plant is in general nonlinear and its output is a function of previous outputs as well as the current and previous inputs,

$$y(k) = g(\,u(k), \ldots,\, u(k-p_n),\, y(k-1), \ldots,\, y(k-p_r)\,) \tag{6}$$

where p_n and p_r are the nonrecursive and recursive orders of the plant respectively. The weights are found using a gradient search. As such the update equation for weight j is

$$\begin{aligned} w_j(m+1) &= w_j(m) - \mu\frac{\partial E}{\partial w_j} \\ &= w_j(m) + \mu\sum_{k=0}^{N} e(k)\beta_j(k) \end{aligned} \tag{7}$$

where

$$\beta_j(k) = \frac{\partial y(k)}{\partial w_j} = \mathbf{q}_u^T(k)\mathbf{a}_j^{p_n}(k) + \mathbf{q}_y^T(k-1)\mathbf{b}_j(k-1) \tag{8}$$

and

$$\mathbf{q}_u^T(k) = \left[\frac{\partial g(\cdot)}{\partial u(k)}, \frac{\partial g(\cdot)}{\partial u(k-1)}, \cdots, \frac{\partial g(\cdot)}{\partial u(k-p_n)}\right] \tag{9}$$

$$\mathbf{q}_y^T(k-1) = \left[\frac{\partial g(\cdot)}{\partial y(k-1)}, \frac{\partial g(\cdot)}{\partial y(k-2)}, \cdots, \frac{\partial g(\cdot)}{\partial y(k-p_r)}\right] \tag{10}$$

$$\mathbf{a}_j^n(k)^T = \left[\alpha_j(k), \alpha_j(k-1), \cdots, \alpha_j(k-n)\right] \tag{11}$$

$$\mathbf{b}_j(k-1)^T = \left[\beta_j(k-1), \beta_j(k-2), \cdots, \beta_j(k-p_r)\right] \tag{12}$$

The $\mathbf{q}$ vectors above represent partials of the plant with respect to its inputs and outputs. When the plant is unknown $\mathbf{q}$ must be estimated. The $\alpha_j(k)$ term in Eq. 11 is given by

$$\alpha_j(k) = \frac{\partial u(k)}{\partial w_j} = \frac{\partial v_{NR}(k)}{\partial w_j} + \frac{\partial v_R(k)}{\partial w_j} \tag{13}$$

It can be shown that [9]

$$\alpha_j(k) = BP_j(k) + FF(\mathbf{a}_j^{o,-1}(k-1)) \tag{14}$$

where $BP_j(k)$ are the standard backpropagation equations and $FF(\mathbf{a}_j^{o,-1}(k-1))$ represents a recursive term due to the feedback. The standard backpropagation term for the *nth* weight in the *mth* node of layer l can be expressed

$$BP(k) = v_{l-1,n}(k) h_{lm}(k) \mathbf{w}_{l+1,m}^T \mathbf{H}_{l+1}(k) \cdots \mathbf{W}_{L-1}^T \mathbf{H}_{L-1}(k)\, \mathbf{W}_L^T \mathbf{1} \tag{15}$$

where 1 is a vector of 1s, and

$$\mathbf{H}_i(k) = diag\left[h_{i1}(k),\, h_{i2}(k),\ \cdots,\, h_{in}(k)\right] \tag{16}$$

where

$$h_{im}(k) = v_{im}(k)[1 - v_{im}(k)] \tag{17}$$

The recursive term in Eq. (14) can be shown to take on the form [9]

$$FF(\mathbf{a}_j^{o,-1}(k-1)) = \mathbf{W}_L^r \mathbf{H}_{L-1}(k)\, \mathbf{W}_{L-1}^r \cdots \mathbf{H}_1(k)\, \mathbf{W}_1^r \mathbf{a}_j^{o,-1}(k-1) \tag{18}$$

In this operation the $\mathbf{a}_j^{o,-1}(k-1)$ vector is fed *forward* through the recursive net in a fashion which is almost the reverse of the $BP(k)$ operation.

In general, the stability of the gradient approach in a feedback loop is not guaranteed even in the linear case. As might be suspected, certain initial conditions and design parameters will lead to unstable behavior of the closed-loop system. This will be an area of further research. On the other hand, we were able to solve many model-following problems, with different external inputs and many different models. A particular model-following example is given next.

III. Example

Let the plant be given by [5]

$$y(k) = \frac{y(k-1)}{1+y^2(k-1)} + u^3(k).$$

and the reference input be

$$d(k) = \sin(0.05\pi k)$$

Let the reference model be the unity gain so that an inverse model is sought. The algorithm was run for 50,000 iterations with white noise training input, after which, the weights were fixed and the results of Figure 2 were obtained. As can be seen, the recursive neural network has learned the inverse of the plant for the particular input d(k).

IV. Conclusions

A recursive neural network was derived and used for the model-following control of a discrete-time nonlinear systems. The network is updated in the feedback loop directly and was able to minimize the model-following error for a number of plants. More recently, we have included another MLP in the feedback path of the plant's output, and generalized the algorithm to higher order MLP's. The results will be reporteu elsewhere. Many issues however, remain to be resolved. Such issues include the stability of the closed-loop system, the choice of the number of delays and other design parameters. It is felt that the recursive MLP approach presents an alternative to the hard problem of controlling a nonlinear system.

REFERENCES

[1] A. Isidori, *Nonlinear Control Systems: An Introduction*, Lecture Notes in Control and Information Sciences, M. Thoma, Ed., Springer Verlag, Berlin, 1985.

[2] R.H. Hirschorn,"Invertibility of Multivariable Nonlinear Control Systems," *IEEE trans. Automat. Contr.*, Vol. AC-24, No.6, pp.855-865, Dec. 1979.

[3] Funahashi, K., "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, Vol. 2, pp. 183-192, 1989.

[4] Special Section on Neural Networks For Control Systems, *IEEE Control Systems Magazine*, Vol.9, No.3, pp.25-59, April 1989.

[5] K.S. Narendra, and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. on Neural Networks*, Vol.1, No.1, pp.4-27, March 1990.

[6] W. Li, and J.J.E. Slotine, "Neural Network Control of Unknown Nonlinear Systems," *Proc. IEEE Auto. Contr. Conf.*, pp. 1136-1141, Pittsburgh, PA 1989.

[7] Rumelhart, D.E., McClelland, J.L., & Williams, R.J., "Learning internal representations by error propagation," In D.E. Rumelhart & J.L. McClelland (Eds.), *Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1986.

[8] Pineda, F.J., "Generalization of backpropagation to recurrent neural networks," *Physical Review Letters*, Vol. 18, pp. 2229-2232, 1987.

[9] D. Hush, C. Abdallah, and B. Horne, " A Learning Algorithm for Recursive Neural Networks," *UNM Technical Report, 1990.*

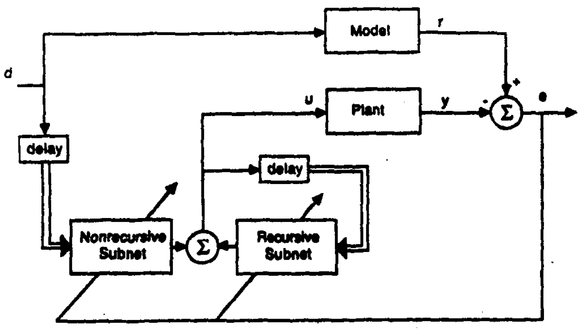

Figure 1

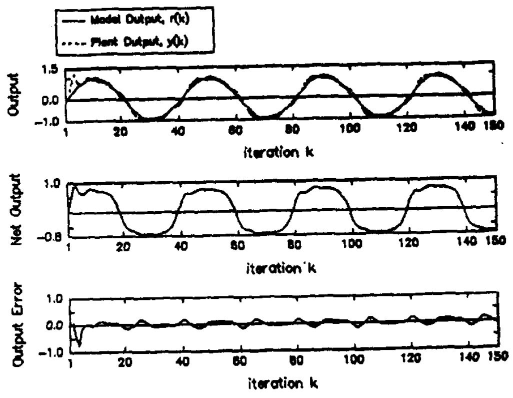

Figure 2