Diagonal Recurrent Neural Networks for Speed Control of a DC motor

Alfonso Noriega[1], Carlos A. Silva[1] and Jesús Pichardo[2]

[1] Facultad de Ingeniería, Universidad Autónoma de Querétaro, Querétaro, Qro., México
[2] CICATA-IPN, Querétaro, Qro., México
{Alfonso Noriega, anoriega@uaq.mx}

Abstract. A new control algorithm based on diagonal recurrent neural network (DRNN) is presented. The architecture of DRNN is a modified model of the fully connected recurrent neural network with one hidden layer [1], and the hidden layer is comprised of self-recurrent neurons. Two DRNN's are utilized in a control system, one as an identifier called diagonal recurrent neuroidentifier (DRNI) and the other as a controller called diagonal recurrent neurocontroller (DRNC). A controlled plant is identified by the DRNI which then provides the sensitivity information of the plant to the DRNC. A generalized dynamic backpropagation algorithm (DBP) is developed and used to train both DRNC and DRNI. Due the recurrence, the DRNN can capture the dynamic behavior of the system. The proposed DRNN paradigm is applied to numerical problems and the simulation results for speed control of a DC motor are included.

Keywords: Recurrent Neural Networks, Dynamic Back-propagation, DC Motor, Diagonal Recurrent Neural Networks, Adaptive Control.

I Introduction

The development in the control area has been fueled by three major needs: the need to deal with increasingly complex systems, the need to accomplish increasingly demanding design requirements, and the need to attain these requirements with less precise advanced knowledge of the plant and its environment [2], [3]. Increasingly complex dynamical systems with significant uncertainty have forced system designers to turn away from conventional control methods. However, the fundamental shortcomings of current adaptive control techniques, such as nonlinear control laws which are difficult to derive, geometrically increasing complexity with the number of unknown parameters, and the general unsuitability for real time applications have compelled researchers to look for solutions elsewhere [4].

Several neural network models and neural learning schemes were applied to system controller design during the last three decades, and many promising result are reported [4]-[8]. Most people used the feedforward neural network and the backpropagation training algorithm [4], [9] to solve the dynamical problems; however, the feedforward network is a static mapping. On the other hand, recurrent neural networks [1]-[2] and [7]-[11] have important capabilities not found in feedforward network, such as attractor dynamics and the ability to store information

Industrial Informatics
Research in Computing Science 31, 2007, pp. 143-152

for later use. Thus the recurrent neural network is a dynamic mapping and is better suited for dynamic systems than the feedforward network.

In most control applications, the real-time implementation is very important, and thus the neurocontroller also needs to be designed such that it converge with a relative small number of training cycles. With the objective of a simple recurrent network and a shorter training time for a neural network model, a diagonal recurrent network (DRNN), as shown in figure 1, is developed. This model has considerably fewer weights and the network is simplified considerably.

This paper is organized as follows. In the section II, a DRNN model is developed and a dynamic backpropagation training algorithm is designed to train a DRNN based control system. Finally, in section III the practical relevance of the proposed control schemes is illustrated by simulation for speed control of a DC motor.

II Diagonal recurrent neural networks

Consider Fig. 1, where for each discrete time k, $I_i(k)$ is the ith input, $S_j(k)$ is the sum of inputs to the jth recurrent neuron, $X_j(k)$ is the output for the jth recurrent neuron, and $O(k)$ is the output of the network. Where W^I, W^O, W^D, represents input, output and diagonal weight vectors, respectively.

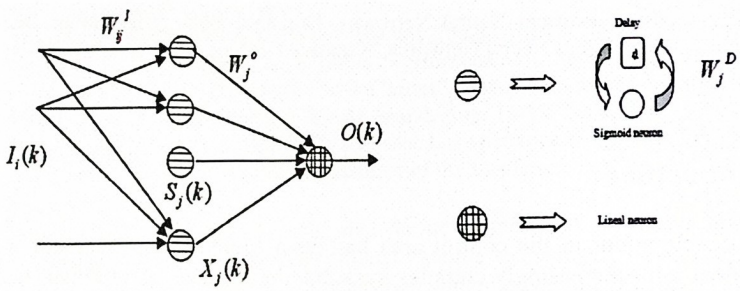

Fig1. Diagonal recurrent neural network

An approach for control and system identification using diagonal recurrent neural networks (DRNN) is presented in this section. An unknown plant is identified by a system identifier, called the diagonal recurrent neuroidentifier (DRNI), which provides information about the plant to a controller, called the diagonal recurrent neurocontroller (DRNC). The neurocontroller is used to drive the unknown dynamic system such that the between plant and desired output is minimized. A generalized algorithm, called the dynamic backpropagation (DBP), is developed to train both DRNC and DRNI. For simplicity, the plant is assumed to be single input/single output system.

Both DRNI and DRNC use the same DRNN architecture shown in Fig 1, which has only one hidden layer with sigmoid type recurrent neurons. The block diagram of the DRNN based control system is shown in Fig. 2. The inputs to the DRNC are the

reference input, the previous plant output, and the previous control signal, and the output of the DRNC is the control signal to the plant. By using the dynamic backpropagation (DBP) algorithm developed in this paper, the weights of the DRNC are adjusted such that the error between the output of the plant and the desired output from a reference model approaches a small value after some training cycles. When the DRNC is in training, the information on the plant is needed. Since the plant is normally unknown, the DRNI is used to estimate the plant sensitivity y_u for the DRNC.

The current control signal generated from de DRNC and previous output of the plant are used as the inputs to the DRNI. The error between the output of the DRNI and plant is computed for each iteration, and is used to adjust the weights of the DRNI. By training the DRNI and DRNC alternately, the weights of the DRNC can be adjusted more effectively.

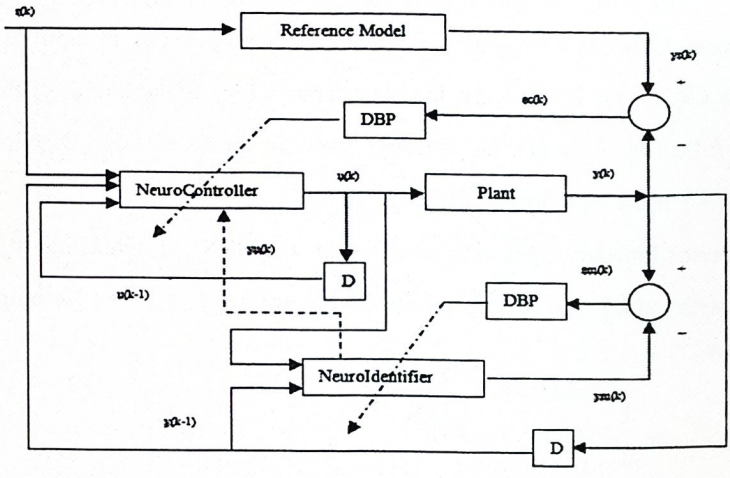

Fig 2. Block diagram of DRNN based control system

II.1 Dynamic backpropagation algorithm for diagonal recurrent neural networks

The mathematical model for the DRNN in Fig. 1 is shown below:

$$O(k) = \sum_j W_j^O X_j(k), \quad X_j(k) = f(S_j(k)) \tag{1}$$

$$S_j(k) = W_j^D X_j(k-1) + \sum_j W_{ij}^I I_i(k) \tag{2}$$

Where for each discrete time k, $I_i(k)$ is the ith input to the DRNN, $S_j(k)$ is the sum of inputs to the jth recurrent neuron, $X_j(k)$ is the output of the jth recurrent neuron, and $O(k)$ is the output of the DRNN. Here $f(\bullet)$ is the usual sigmoid function, and W^I, W^D, W^O are input, recurrent, and output weight vectors, respectively, in $\Re^{n_i}, \Re^{n_d}, \Re^{n_o}$.

Let $y_r(k)$ and $y(k)$ be the desired and actual responses of the plant, then an error function for a training cycle for DRNC can be defined as:

$$E_c = \frac{1}{2}(y_r(k) - y(k))^2 \tag{3}$$

In general, the plant response is a nonlinear mapping $G(\bullet)$ of input $u(k)$, i.e., $y(k) = G(i(), k)$. Here, the plant input $u(k)$ is the output of the DRNC, i.e., $u(k) = O(k)$ in (1). On the other hand, in the case of the DRNI, the plant input $u(k)$ is the input to the DRNI.

The error function (3) is also modified for the DRNI by replacing $y_r(k)$ and $y(k)$ with $y(k)$ and $y_m(k)$, respectively, where $y_m(k)$ is the output of the DRNI, i.e.,

$$E_m = \frac{1}{2}(y(k) - y_m(k))^2 \tag{4}$$

Where $y_m(k) = O(k)$ of (1).

The gradient of error in (3) with respect to an arbitrary weight vector $W \in \Re^n$ is represented by

$$\frac{\partial E_c}{\partial W} = -e_c(k)\frac{\partial y(k)}{\partial W} = -e_c(k)y_u(k)\frac{\partial u(k)}{\partial W} = -e_c(k)y_u(k)\frac{\partial O(k)}{\partial W} \tag{5}$$

Where $e_c(k) = y_r(k) - y(k)$ is the error between the desired and output responses of the plant, and the factor $y_u(k) \equiv \frac{\partial y(k)}{\partial u(k)}$ represents the sensitivity of the plant with respect to its input.

Since the plant is normally unknown, the sensitivity needs to be estimated for the DRNC. However, in the case of the DRNI, the gradient of error in (4) simply becomes

$$\frac{\partial E_m}{\partial W} = -e_m(k)\frac{\partial y_m(k)}{\partial W} = -e_m(k)\frac{\partial O(k)}{\partial W} \tag{6}$$

Where $e_m(k) = y(k) - y_m(k)$ is the error between the plant and the DRNI responses.

The output gradient $\frac{\partial O(k)}{\partial W}$ is common (5) and (6) and needs to be computed for both DRNC and DRNI. Its computation is summarized in the following paragraph: Given the DRNN shown in Fig 1 and described by (1) y (2), the output gradients with respect to output, recurrent, and input weights, respectively, are given by:

$$\frac{\partial O(k)}{\partial W_j^O} = X_j(k) \tag{7a}$$

$$\frac{\partial O(k)}{\partial W_j^D} = W_j^O P_j(k) \tag{7b}$$

$$\frac{\partial O(k)}{\partial W_{ij}^I} = W_j^O Q_{ij}(k) \tag{7c}$$

Where $P_j(k) \equiv \frac{\partial X_j(k)}{\partial W_j^D}$ and $Q_{ij}(k) \equiv \frac{\partial X_j(k)}{\partial W_{ij}^I}$ and satisfy:

$$P_j(k) = f'(S_j)(X_j(k-1) + W_j^D P_j(k-1)), \qquad P_j(0) = 0 \tag{8a}$$

$$Q_{ij}(k) = f'(S_j)(I_i(k) + W_j^D Q_{ij}(k-1)), \qquad Q_{ij}(0) = 0 \tag{8b}$$

From (1), the gradient with respect to the output weight is found as $\frac{\partial O(k)}{\partial W_j^O} = X_j$. Again, from (1), the gradient with respect to the recurrent weight is

$$\frac{\partial O(k)}{\partial W_j^D} = W_j^O \frac{\partial X_j(k)}{\partial W_j^D} = W_j^O P_j(k)$$

From (1) and (2),

$$\frac{\partial X_j(k)}{\partial W_j^D} = \frac{\partial X_j(k)}{\partial S_j(k)} \frac{\partial S_j(k)}{\partial W_j^D} = f'(S_j(k)) \frac{\partial S_j(k)}{\partial W_j^D}$$

and

$$\frac{\partial S_j(k)}{\partial W_j^D} = X_j(k-1) + W_j^D \frac{\partial X_j(k-1)}{\partial W_j^D}$$ Which lead to (8a).

The procedure of deriving the gradient with respect to input weight is similar to the above derivation, and the corresponding equations, (7c) and (8b) follow.

Equations (8a) and (8b) are nonlinear dynamic recursive equations for the state gradients $\frac{\partial X_j(k)}{\partial W}$, and can be solve recursively with given initial conditions. For the usual FNN, the current weight W_j^D is zero and the equations become algebraic.

II.2 Dynamic backpropagation for DRNI

From (6), the negative gradient of the error with respect to a weight vector in $\Re^n$ is

$$-\frac{\partial E_m}{\partial W} = e_m(k) \frac{\partial O(k)}{\partial W} \tag{9}$$

Where the output gradient is given by (7) and (8), and W represents W^D, W^O, or W^I in $\Re^{n_d}$, $\Re^{n_o}$, or $\Re^{n_i}$, respectively.

The weights can now be adjusted following a gradient method, i.e., the update rule of the weights become:

$$W(n+1) = W(n) + \eta\left(-\frac{\partial E_m}{\partial W}\right) \tag{10}$$

Where η is a learning rate. The equations (7)-(10) define the dynamic backpropagation algorithm (DBP) for DRNI.

II.3 Dynamic backpropagation for DRNC

In the case of DRNC, from (5), the negative gradient of the error with respect to a weight vector in $\Re^n$ is

$$-\frac{\partial E_c}{\partial W} = e_c(k) y_u(k) \frac{\partial O(k)}{\partial W} \tag{11}$$

Since the plant is normally unknown, the sensitivity term $y_u(k)$ is unknown. This unknown value can be estimated by using the DRNI. When the DRNI is trained, the dynamic behavior of the DRNI is close to the unknown plant, i.e., $y(k) \approx y_m(k)$ where $y_m(k)$ is the output of the DRNI.

Once the training process is done, we assume the sensitivity can be approximated as

$$y_u(k) \equiv \frac{\partial y(k)}{\partial u(k)} \approx \frac{\partial y_m(k)}{\partial u(k)} \tag{12}$$

Where $u(k)$ is an input to the DRNI.

Applying the chain rule to (12), and noting that $y_m(k) = O(k)$ of (1),

$$\frac{\partial y_m(k)}{\partial u(k)} = \frac{\partial O(k)}{\partial u(k)} = \sum_j \frac{\partial O(k)}{\partial X_j(k)} \frac{\partial X_j(k)}{\partial u(k)} = \sum_j W_j^O \frac{\partial X_j(k)}{\partial u(k)} \tag{13}$$

Also from (1),

$$\frac{\partial X_j(k)}{\partial u(k)} = f'(S_j(k)) \frac{\partial S_j(k)}{\partial u(k)} \tag{14}$$

Since inputs to the DRNI are $u(k)$ and $y(k-1)$ from Fig 2, (2) becomes

$$S_j(k) = W_j^D X_j(k-1) + W_{1j}^I u(k) + W_{2j}^I y(k-1) + W_{3j}^I b_I \tag{15}$$

Where b_I is the bias for DRNI. Thus

$$\frac{\partial S_j(k)}{\partial u(k)} = W_{1j}^I \tag{16}$$

From (13), (14) and (16),

$$y_u(k) = \frac{\partial y_m(k)}{\partial u(k)} = \sum_j W_j^O f'(S_j(k)) W_{1j}^I \tag{17}$$

Where the variables and weights are those found in DRNI.

Using the negative gradients in (11), the weights for DRNC can now be adjusted using the update rule similar to (10). The equations (7), (8), (19), (11), and (17) define the dynamic backpropagation algorithm for DRNC.

III SIMULATION RESULTS

Although the above described algorithm has been tested in many examples, here we will only show the results obtained in one simulated case corresponding to the following model [12] of a motor CD. The transfer function from the armature voltage to the angular velocity is:

$$\frac{W(s)}{V_a(s)} = G(s) = \frac{20.16}{s^2 + 4.244s + 14.34}$$

Discrete the plant G (s) the following is obtained:

Y=(-0.6542*Y_2)+(1.539*Y_1)+(0.0754*U_2)+(0.0869*U_1);

Period of sampling T = 0.1

After the reference is reached a change it is made in the constant of time of the motor obtaining the following model:

$$\frac{W(s)}{V_a(s)} = G(s) = \frac{20.16}{s^2 + 4s + 20}$$

Discrete the plant G (s) with T=0.1 the following is obtained:

Y=(-0.6703*Y_2)+(1.508*Y_1)+(0.07623*U_2)+(0.08716*U_1);

As conditions initials we have:

Ym = 0, Yu = 1, η_I = 0.2, η_C = 0.2

Where η_I, η_C are the constants of learning for neuroidentifier and neurocontroller respectively.

In the figure 3, it is shown the proposed schemes for DRNI and DRNC. The figure 4, illustrates the tracking-reference sequence $y_m(k)$, the plant output and the corresponding adaptive neural control input when the parameter's change occurs at

t_1. It can be observed that the tracking error converges to zero in few sampling periods. The transient period could be reduced by changing the learning rate η at the expense of more input energy.

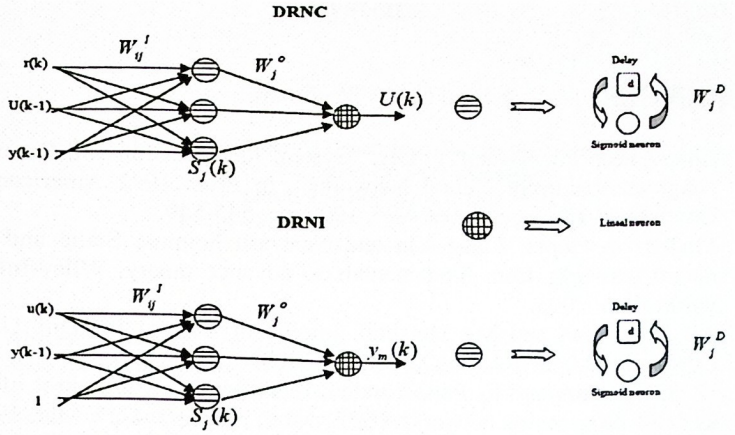

Fig. 3 Schemes DRNN for neuroidentifier and neurocontroller respectively.

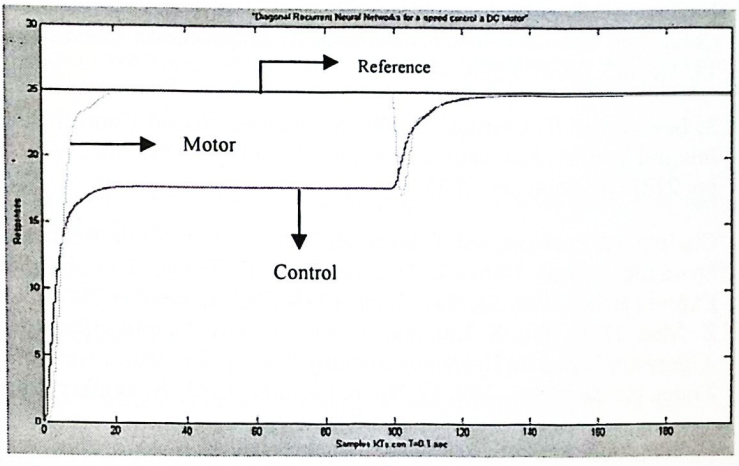

Fig. 4 Responses of the motor, signal of control and the reference

IV CONCLUSION

The control algorithm based on a diagonal recurrent neural network presented in this paper, promise to be a very interesting option for the control of processes with a difficult dynamics that could not be adequately controlled with PID regulators, even

in their self-tuning versions, as it was shown in the realized simulation cases. The class of processes in which the algorithm could be applied is very wide and it includes most of the cases that can appear in practice. In the near future, we plan to apply the algorithm to some real laboratory processes and to extend the obtained results to the case of multivariable and multiconnected systems.

REFERENCES

1. Chao-Cheen ku, Kwang Y. Lee, "System Identification and Control Using Diagonal Recurrent Neural Networks", in Proc. 1992 American Control Conference, Chicago, June 24-26, 1992, pp. 545-549.
2. Madan M. Gupta, Liang Yin and Noriyasu homma, Static and Dinamic neural networks from fundamentals to advance theory, Wiley-Interscience publication, 2003.
3. Ali Zilouchian and Mo Jamshidi, Intelligent Control Systems Using Soft Computing Methodologies, CRC press, 2001.
4. K. S. Narendra and K. Parthasarathy, Identification and control of dynamic systems using neural networks, IEEE, vol. I, No 1, pp. 4-27, Mar. 1990.
5. Richard P. Lippmann. "An Introduction to Computing Whit Neural Nets". IEEE ASSP Magazine, pages. 4-22, April 1987.
6. N. Bhat and T. J. McAvoy, "Use of Neural Nets for Dynamic Modelling and Control of Chemical Process Systems", Computers on Chem. Engng., Vol 14, No. 4/5, pp. 573-583, 1990.
7. S. Baruch and R. Garrido, "A Direct Adaptive Neural Control Scheme with Integral Terms", International Journal of Intelligent Systems, Vol. 20, No. 2, pp. 213-224, February, 2005.
8. Chairez, A. Poznyak and T. Poznyak, New Sliding-Mode Learning Law for Dynamic neural Network Observer, IEEE Trans. Circuits Systems-II: Express Briefs, Vol. 53, No. 12, pp. 1338-1342, December 2006.
9. Z. Man, H. R. Wu, S. Liu, and X. Yu, "A new Adaptive Backpropagation Algorithm Based on Lyapunov Stability Theory for Neural Networks", IEEE Trans. Neural Netw., Vol. 17, No. 6, pp. 1580-1591, November 2006.
10. O. De Jesús and M. T. Hagan, "Backpropagation Algorithm for a Broad Class of Dynamic Networks", IEEE Trans. Neural Netw., Vol. 18, No. 1, pp. 14-27, 2007.
11. Y. D. Jou and F. K. Chen, "Least-Squares Design of FIR Filters Based on a Compacted Feedback Neural Network", IEEE Trans. Circuits Syst. II, Exp. Briefs, Vol. 54, No. 5, pp. 427-431, May., 2007.
12. Khaled Nouri, Rached Dhaouadi, Naceur Benhadj Braiek, Nonlinear speed control of a dc motor drive system with online trained recurrent neural network, IEEE, AMC'06-Istanbul, Turkey, 2006.