Approximating Transfer Functions using Neural Network Weights

Tarek A. Tutunji

Mechatronics Engineering Department
Philadelphia University
Amman, Jordan

Abstract— Artificial neural networks are widely used in the identification and control of complex systems. However, the network model which is based on neuron nodes, activation functions, and network weights is rarely related to the system transfer function. In this paper, a clear relationship between the network weights and the transfer function parameters is established. The developed mathematical equations are based on approximating the neuron activation function using Taylor expansion and relating the results to a linear transfer function based on Auto Regressive Moving Average model. Simulation results show that the approximated transfer function behavior resembles the original system function.

Keywords- Artificial Neural Networks, Transfer Functions, ARMA models.

I. INTRODUCTION

Artificial Neural Networks (ANN) are mathematical models that are used to mimic the biological neurons in the brain. They are used as black box models to map unknown functions. The concept of ANN was initiated in the 40's by McCulloch and Pitts where they described and modeled a single neuron called 'perceptron'. However, their model was capable of solving only linear separable problems. Research in the area was re-energized in the 70's when the backpropagation algorithm was developed and applied to multi-layer neural networks. Other researchers such as Hopfield and Hebbian also contributed to the recognition of neural networks [1]. Usually, ANN algorithms require intensive mathematical operations through many iterations updates and therefore, the fast development of computers in the 80's played a major role in the popularity of neural networks among researchers.

Artificial Neural Networks (ANN) are currently used in many intelligent control [2,3] and identification applications [4,5,6]. Many journals are dedicated to ANN research and many publications have been published in the area. However, the network structure and convergence is rarely related to the black box transfer function [7,8]. In this paper, a clear relationship between the network weights and the Auto-Regressive Moving-Average (ARMA) parameters will be developed for a single recurrent neuron. This in turn gives the system transfer function.

This paper is divided as follows: section two gives a review of ARMA models while section three gives a review of ANN architecture. In section four, the mathematical relationship

between the neuron weights and the ARMA parameters is derived. Simulation results are given in section five and conclusions are given in section six.

II. AUTO REGRESSIVE MOVING AVERAGE (ARMA) MODELS

ARMA models are linear regression models that can be used to identify black boxes by mapping input-output data. The general structure for an ARMA model [9] is given in Fig. 1.

The output is a linear difference equation of current and past inputs and past outputs. This ARMA equation is given next

$$y_{k,ARMA} = \sum_{j=1}^n a_j y_{k-j} + \sum_{i=0}^m b_i x_{k-i} \quad (1)$$

where x_k and y_k are the inputs and outputs at discrete-time k , and a_j 's and b_i 's are the ARMA parameters.

The Z-Transform of the ARMA(m,n) model is calculated to yield the transfer function of the model which is given by

$$Z \left\{ y_k - \sum_{j=1}^n a_j y_{k-j} \right\} = Z \left\{ \sum_{i=0}^m b_i x_{k-i} \right\}$$

$$\Rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_m z^{-m}}{1 - a_1 z^{-1} - \dots - a_n z^{-n}} \quad (2)$$

Therefore, the transfer function can be identified once the ARMA parameters are approximated. Researchers have used impulse response data within ARMA models to identify mechatronic systems [10].

Figure 1. ARMA Block Diagram

III. ARTIFICIAL NEURAL NETWORKS

A single recurrent neuron is considered where the input vector is composed of present input (x_k), delayed inputs (x_{k-i}), and delayed outputs (y_{k-j}). Those inputs are multiplied by weight vectors $W = [w_0 \ w_1 \ \dots \ w_m]$ and $V = [v_1 \ v_2 \ \dots \ v_n]$ respectively and taken to the activation function in the neuron to give the output $y_{k,net}$ as shown in Fig. 2.

The hyper tangent function can be used as the activation function and therefore

$$y_{k,net} = \frac{1 - e^{-net_k}}{1 + e^{-net_k}} \quad (3)$$

$$net_k = \sum_{i=0}^m w_i x_{k-i} + \sum_{j=1}^n v_j y_{k-j} \quad (4)$$

The goal is to minimize the error between the desired output d_k and the neuron output $y_{k,net}$. The least square error is defined to be

$$E = \frac{1}{2} \sum_{k=1}^K (y_{k,net} - d_k)^2 \quad (5)$$

The weights updates are constructed using the steepest descent method and are given next

$$w_i = w_i - \alpha (y_{k,net} - d_k) \left(\frac{2e^{-net_k}}{(1 + e^{-net_k})^2} \right) x_{k-i} \quad (6)$$

$$v_j = v_j - \alpha (y_{k,net} - d_k) \left(\frac{2e^{-net_k}}{(1 + e^{-net_k})^2} \right) y_{k-j} \quad (7)$$

Similar results can be derived for multi-layer networks. However, in the multi-layer case the backpropagation [1] algorithm is usually used to derive an error equation such as (5) for the hidden nodes. This research is concerned in developing an approximation formula and algorithm for transfer functions and therefore will consider single output neurons (regardless of the number of hidden neurons). The architecture of multi-layer with three hidden nodes and single output is given in Fig. 3. The transfer function approximation for multi-layer is much more involved as will be shown in the next section.

Figure 2. Single Recurrent Neural Network

IV. NEURON WEIGHTS TRANSFORMATION TO ARMA PARAMETERS

In order to find the transfer function for a single recurrent neuron, (1) and (3) should be equaled. However, the solution is not apparent because of the exponential term in (3). In order to simplify this problem, the 2nd order Taylor expansion was used at the approximation point '0'. The result is

$$y_{k,net} = \frac{1 - e^{-net_k(0)}}{1 + e^{-net_k(0)}} + \frac{2e^{-net_k(0)}}{(1 + e^{-net_k(0)})^2} (net_k - 0) \quad (8)$$

$$\Rightarrow y_{k,net} = 0.5 \sum_{i=0}^m w_i x_{k-i} + 0.5 \sum_{j=1}^n v_j y_{k-j}$$

Now, setting the ARMA output in (1) equal to (8) results in

$$a_j = 0.5v_j, b_i = 0.5w_i \quad (9)$$

Equation (9) can be used to approximate the ARMA parameters (and therefore the transfer function as shown in (2)) from the neuron weights.

Figure 3. Multi-Layer Network

Figure 4. Algorithm flowchart

For a more general network, such as the one given in Fig. 3, more work needs to be done. Consider the 2-layer network with 3 hidden nodes. Each neuron will generate its own transfer function and therefore the overall transfer function would be

$$\hat{H}(z) = (\hat{H}_1(z) + \hat{H}_2(z) + \hat{H}_3(z)) \times \hat{H}_{out}(z) \quad (10)$$

Here, each transfer function can be derived independently using (9). Then the overall transfer function can be calculated using (10). Fig. 4 describes the algorithm used to approximate the system transfer function using neural networks

V. SIMULATION RESULTS

This section gives simulation results that show the behavior of the described algorithm in Fig.4. In the first example used, a 3rd order transfer function was generated in MATLAB which is shown next

$$H(z) = \frac{1.3573 + 1.0266z^{-1} - 1.0378z^{-2}}{1 - 0.0927z^{-1} - 0.1821z^{-2} - 0.5280z^{-3}} \quad (11)$$

TABLE I. POLE ZERO LOCATIONS

Systems	Poles and Zeros	
	Poles	Zeros
Original	0.9178	-1.3309
	-0.4126 + 0.6364i	0.5745
	-0.4126 - 0.6364i	
Approximated	0.9400	-1.3768
	-0.4243 + 0.6688i	0.5613
	-0.4243 - 0.6688i

Figure 5. Output comparisons (for 3rd order example) among original (solid), neural network (o), and approximated transfer function (x)

The neural network weights converged and were transformed to give the following transfer function:

$$\hat{H}(z) = K \frac{0.4175 + 0.3405z^{-1} - 0.3227z^{-2}}{1 - 0.09314z^{-1} - 0.1704z^{-2} - 0.5897z^{-3}} \quad (12)$$

Here K is the adjusted gain (1.3573/0.4175). The approximated transfer function does not equal the original transfer function exactly. This is due to the fact that a local optimal point (i.e. not the global optimal) is usually reached. The pole/zero locations of both systems are given in Table I.

The time domain results are shown in Fig. 5. A total of 50 randomly generated inputs were applied. The random generator had a Gaussian distribution with a mean of zero and variance of one. It can be seen that the outputs of the desired (original transfer function), the neural network, and the re-constructed (approximated) transfer function are similar. The statistical results are displayed in Table 2.

The impulse response between the original and approximated transfer functions are shown in Fig. 6 while the bode plot responses of the systems are shown in Fig. 7. Note that the approximate transfer function output was able to follow the behavior of the original system with minimal error.

TABLE II. STATISTICAL ERROR RESULTS

Systems	Sum Square Error (SSE) Calculations		
	Average	Variance	Maximum
Neural Network	0.0052	0.0045	0.1943
Approximated ARMA	0.0114	0.0063	0.2168

Figure 6. Impulse response for 3rd order system (a) original (b) approximated

In the second example, the original transfer function was given by

$$H(z) = \frac{z^2 - r \cos \theta z}{z^2 - 2r \cos \theta z + r^2} \quad (13)$$

Values of $r=0.5$ and $\theta = 0.785$ (i.e. 45 degrees) were used. In this case, a two layer network was used with 3 hidden nodes. A total of 50 inputs were generated randomly and applied to the systems. The results are shown in Fig. 8. Here, the neural network was able to follow the original system behavior (average and maximum errors were 0.0070 and 0.2812) while the approximated transfer function did not do as good (average and maximum errors were 0.2578 and 1.4002). This might be due to the fact that (10) had compounded errors from four transfer functions. Research is still at work to analyze the multi-layer case.

Figure 7. Bode plots for 3rd order system (a) original (b) approximated

Figure 8. Output comparisons (for 2-layer example) among original (solid), neural network (o), and approximated transfer function (x)

It is worth noting here that the sensitivity of the original transfer function has a high effect on the convergence of the approximated transfer function. Simulations of such results are not included here for lack of space.

VI. CONCLUSIONS

Neural networks have been widely applied to identification and control applications. However, the network structure and convergence is rarely related to the system transfer function. Transfer functions can provide valuable information about the system such as the pole/zero locations and bode plots. In this paper, a mathematical relationship between the neuron weights and the ARMA parameters were derived. Simulation results show that the developed algorithm was able to approximate the system transfer function successfully for a single recurrent neuron. Current work is already in progress to further explore the proposed algorithm, specifically for multi-layer networks.

REFERENCES

- [1] Haykin, S., Neural Networks: a comprehensive foundation. Macmillan College Publishing Company 1994
- [2] Ryckaert, Salaets, White, and Stoker, "A Comparison of Classical, State-Space and Neural Network Control to Avoid Slip" *Mechatronics Journal* March 2005.
- [3] Burton, Kamran, Harley, Habetler, Brooke, and Podder, "Identification and Control of Induction Motor Stator Currents Using Fast On-Line Random Training of a Neural Network" *IEEE Transactions on Industry Applications* Vol 33, No. 3, 1997
- [4] I. Gabrijel and A. Dobnikar, "On-line identification and reconstruction of finite automata with generalized recurrent neural networks" *Neural Networks* v. 16 pp. 101-120, 2003.
- [5] Liu, Kadirkamanathan, and Billings. "On-line Identification of nonlinear systems using Volterra polynomials basis function neural networks". *Neural Networks* 11. pp. 1645-1657, 1998.
- [6] Efe and Kaynak, "A Comparative Study of Neural Network Structures in Identification of Nonlinear Systems". *Mechatronics* 9 pp. 287-300, 1999.
- [7] Fung, Billings, and Zhang, "Generalised Transfer Functions of Neural Networks", *Mechanical Systems and Signal Processing* 11 (6), pp. 843-863, 1997.
- [8] Y. Chen and Z. Chen, "A neural-network-based experimental technique for determining z-transfer function coefficients of a building environment" *Building and Environment* 35 pp. 181-189, 2000.
- [9] Soderstrom and Stoica, *System Identification*, Prentice Hall, 1989.
- [10] Tutunji, Molhem, and Turki. *Mechatronic, "Systems Identification using an Impulse Response Recursive Algorithm". Simulation Modeling and Practice Theory*, 2007.