Comparative Analysis of Neural Network Filters and Adaptive Volterra Filters

D. J. Krusienski and W. K. Jenkins
Department of Electrical Engineering
The Pennsylvania State University
University Park, PA 16802

Abstract

This paper analyzes the relationship between the traditional adaptive Volterra filter (AVF) and the multilayer perceptron (MLP) neural network filter. These two filter structures are inherently similar in form and function. This relationship is clarified by analyzing the details of both methods within the context of adaptive nonlinear filtering.

1. Introduction

The Volterra series expansion is an extension of the Taylor series expansion used to represent nonlinear systems with memory. Volterra functions take the form

$$\begin{aligned} y[n] &= h_0 + \sum_{m_1=0}^{\infty} h_1[m_1]x[n-m_1] \\ &+ \sum_{m_1=0}^{\infty}\sum_{m_1=0}^{\infty} h_2[m_1,m_2]x[n-m_1]x[n-m_2]+... \\ &+ \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty}...\sum_{m_p=0}^{\infty} h_p[m_1,m_2,...,m_p]x[n-m_1]x[n-m_2]..x[n-m_p] \\ &+... \end{aligned} \tag{1}$$

where x[n] and y[n] represent the input and output of the discrete-time causal nonlinear system, respectively. The h_p terms represent Volterra kernels, which act as the weighting coefficients of the ordered terms. These kernels are analogous to the tap weights of a linear digital filter, although in the nonlinear filter many of these kernels serve as tap weights for products and cross products of delayed input terms.

Adaptive Volterra filtering is special type of polynomial filtering based on the Volterra series expansion. The Volterra model is advantageous because it can accurately represent nonlinearities based on causal systems with memory, which applies to many real systems. Some common applications suitable for Volterra modeling are high power amplifiers used in wireless communications, harmonic distortion in loudspeakers, distortions in digital magnetic recording systems, and enhancement of noisy images.

Because an infinite Volterra series is not possible to implement in a practical filter, a truncated Volterra series is used. Figure 1 shows the standard block diagram of an adaptive Volterra filter (AVF). The adaptive filter is updated by the error e[n] that represents the difference between the desired signal d[n] and the filter output y(n).

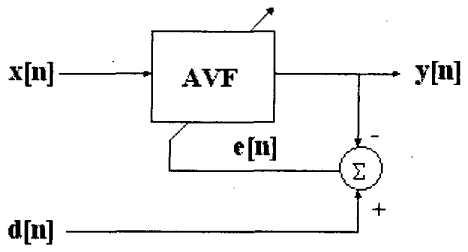


Figure 1. The Adaptive Volterra Filter

In order to use these well known linear techniques for a nonlinear system, the incoming data must be manipulated in such a manner that the resulting output can be related linearly. This is accomplished by performing rank order operations (ROO) on the incoming data [3]. The block diagram for this system is shown in Figure 2.

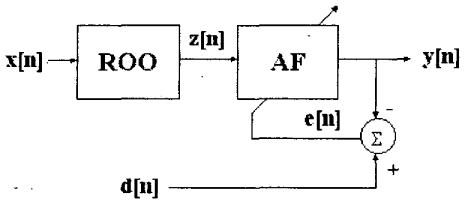


Figure 2. The traditional adaptive Volterra filter

The input vector $X[n]=[x[n],x[n-1],...,x[n-M]]^T$ contains the delayed input samples. The rank order operator combines these delayed inputs in such a manner as to produce all of the polynomial combinations of the inputs. The resulting vector is

$$Z[n]=[z_1[n],z_2[n],...,z_L[n]]^T,$$

where $z_i[n]=x_{i1}[n]x_{i2}[n]...x_{ik}[n]$, $i=1,2,...,L$ and each $i_k \leq M$ contains a subset of the elements in x[n] arranged in some orderly manner [3]. The incoming data are now arranged such that they can be related linearly in terms of its Volterra kernels from equation (1). As a result, any of the standard adaptive linear filter techniques can be used to achieve the optimal Volterra kernels. The number of arithmetic operations and the number of outputs of the ROO increase exponentially with the number of inputs. Thus a very large filter may be required to achieve accurate results.

0-7803-7150-X/01/$10.00@2001 IEEE

2. Multi-layer Perceptron Neural Network Filters

Another adaptive Volterra filtering structure is the multi-layer perceptron neural network filter. This type of network is a feedforward neural network, consisting of multiple layers of neurons in parallel. These architectures are motivated by the structure of biological neurons, and are best known for their application to prediction and pattern recognition. A neuron consists of a weighted input and a summing node whose output is passed through a nonlinear transfer characteristic. The representation of a neuron is shown in Figure 3.

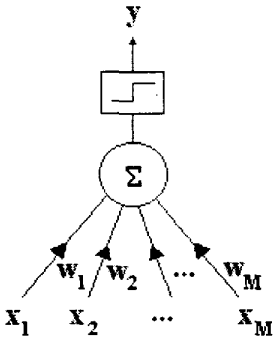


Figure 3. A single neuron model

In a feed forward neural network, the neurons of each layer are connected to all neurons in adjacent layers through unidirectional weighted links. There are no interconnections of neurons within the same layer or to nonadjacent layers. Because weighted links are unidirectional, the incoming data must flow from the first "input" layer to the next consecutive layers, i.e. "feedforward". Any additional layers between the input layer and the output layer are referred to as hidden layers. The diagram of a feed-forward MLP network is shown below in Figure 4.

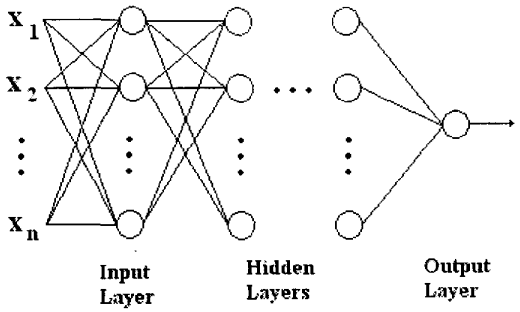


Figure 4. Feed-forward MLP

Typical nonlinearities used for neural networks are soft limiters because they are bounded, monotonically increasing, and differentiable (which is a necessary condition for the backpropogation training technique described later). The logsigmoid and tansigmoid (see Figure 5) functions are common examples of soft limiters. The Taylor series expansion for the tansigmoid is given below.

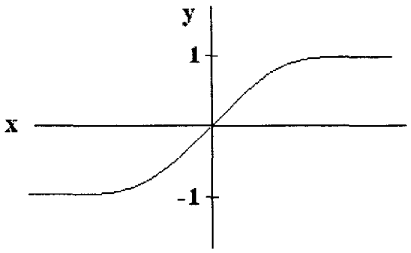


Figure 5. Tansigmoid Function

$$y = \tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$

$$= \frac{2x + \frac{(2x)^2}{2!} + \frac{(2x)^3}{3!} + \ldots}{2 + 2x + \frac{(2x)^2}{2!} + \frac{(2x)^3}{3!} + \ldots} \qquad (2)$$

It is evident from this expansion that the output of a neural network is composed of an infinite order combination of the summed/weighted inputs. This allows the network to accurately approximate nonlinearities of a very high order depending on the structure. As the number of inputs, neurons in parallel, and layers increase, more complex functions can be realized. It should also be pointed out that if the transfer characteristic of the neuron is linear, it will behave as a linear FIR filter. This is the basis that indicates a relationship between the two types of filters.

The neural network is trained adaptively using the back propagation algorithm, on which the LMS algorithm is based on. For backpropogation, the transfer characteristics are fixed and the weights are initialized randomly. The error is determined by propagating the errors from the output layer to the input layer in terms of derivatives. This is why the transfer characteristic must be differentiable, as is the case with sigmoidal functions. The corresponding weights are updated according to the error, similar to the LMS algorithm.

In filtering applications the neural network will have tapped delay inputs and a single output neuron, whose output corresponds to the output of a typical filter. It has been shown that any function can be realized with only three layers with unlimited neurons [12], but in most cases the total number of neurons can be decreased by adding additional layers. It is currently not possible to analytically determine the optimum number of neurons and layers needed for a given system. The number of degrees of freedom achievable by the input weights is directly related to the number of inputs,

neurons, and layers. This is more obvious when the output of a single neuron with a tansigmoid transfer characteristic is considered. Referencing the neuron in Figure 3, the input to the transfer characteristic given in equation (2) is characterized by:

$$x = \sum_{i=1}^{M} w_i x_i \tag{3}$$

The corresponding output is given by:

$$y = \frac{2\left(\sum_{i=1}^{M} w_i x_i\right) + \frac{\left(2\left(\sum_{i=1}^{M} w_i x_i\right)\right)^2}{2!} + \frac{\left(2\left(\sum_{i=1}^{M} w_i x_i\right)\right)^3}{3!} + \ldots}{2 + 2\left(\sum_{i=1}^{M} w_i x_i\right) + \frac{\left(2\left(\sum_{i=1}^{M} w_i x_i\right)\right)^2}{2!} + \frac{\left(2\left(\sum_{i=1}^{M} w_i x_i\right)\right)^3}{3!} + \ldots} \tag{4}$$

It is evident from equation (4) that increasing the number of weighted inputs will more accurately approximate the Volterra kernels.

The sigmoid functions are traditionally used in neural networks applied in pattern recognition because they act as limiters, making a decision or classifying the incoming data. Since this is not necessary for filtering applications any differentiable transfer characteristic can be implemented. A more general case of a transfer characteristic represented by its power series expansion is given as:

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \ldots a_n x^n$$

The a_i terms represent the constant coefficients of the expansion.

In order for the number of neurons to be adequate for representing a given system the number of unknown weights must be at least equal to the number of desired kernels. This condition does not guarantee an optimal solution because the kernels are represented by a nonlinear system of equations, which is not easily solvable. Adding additional layers will further increase the number of higher order terms in the expressions.

3. Computer Simulations

To compare the performance of the traditional and the neural network adaptive Volterra filters a system identification configuration was used. In this system identification, the adaptive filter attempts to accurately model the plant, which is a fixed order Volterra system. The block diagram of this is shown in figure 6.

For the simulations, three classes of adaptive Volterra filters were examined: the traditional AVF, the MLP filter, and the modified MLP filter (MMLP). The first two filters are the filters described previously. The MMLP filter is simply the MLP with a ROO performed on the input samples and a linear transfer characteristic for the neurons, as in Figure 2, with the adaptive filter

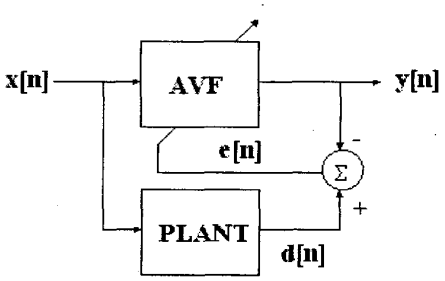


Figure 6. The Simulation model

block acting as a neural network. This enables the MMLP filter to mimic the AVF in terms of function. A block diagram of the MMLP is given in Figure 7. It is used to establish a common ground in the performance comparisons of the adaptive neural network and the adaptive linear filter.

The traditional AVF was realized using the standard LMS gradient descent method with variable order and tapped delay inputs. Both of the neural network filters were realized using the backpropagation method of adaptation along with variable transfer characteristics, tapped delay inputs, order#(MMLP only), and number of neurons/layers.

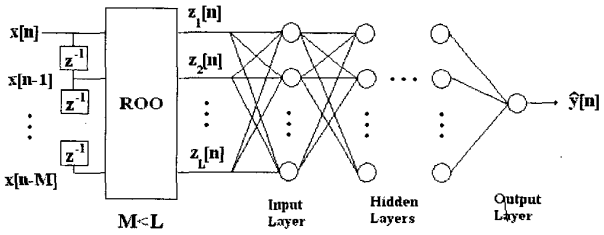


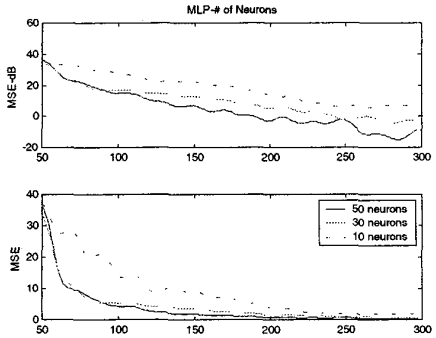
Figure 7. The MMLP

The experimental truncated Volterra plant equation is given in equation (5). The input is a sum of sinusoids in uniformly distributed random noise v[n] of unit variance, as described by equation (6). For the arbitrary second order plant the vector a=[10, -2, 3, -4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], for the third order plant the vector a=[10, -2, 3, -4, 1, 1, -2, 1, 4, -5, 2, 4, 3, -3, 4, 1, -5, 1, 1].

$$\begin{aligned} y[n] &= a_1 x[n] + a_2 x[n-1] + a_3 x[n-2] + a_4 x^2[n] \\ &+ a_5 x^2[n-1] + a_6 x^2[n-2] + a_7 x[n]x[n-1] \\ &+ a_8 x[n]x[n-2] + a_9 x[n-1]x[n-2] + a_{10} x^3[n] \\ &+ a_{11} x^3[n-1] + a_{12} x^3[n-2] \\ &+ a_{13} x[n]x[n-1]x[n-2] + a_{14} x^2[n]x[n-1] \\ &+ a_{15} x^2[n]x[n-2] + a_{16} x^2[n-1]x[n] \\ &+ a_{17} x^2[n-1]x[n-2] + a_{18} x^2[n-2]x[n] \\ &+ a_{19} x^2[n-2]x[n-1] \end{aligned} \tag{5}$$

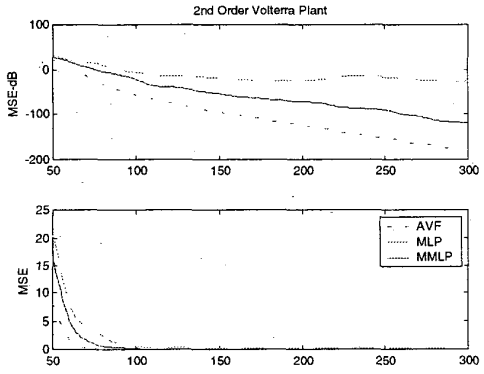
$$x[n] = 0.5\sin(4\pi t) + 0.5\sin(7\pi t) + v(t) \qquad (6)$$

The initial simulations are general experiments used to examine the effects of different MLP structures and to determine an adequate working model for the MLP filter. They were executed using the third order Volterra plant described previously with variable transfer characteristics, tapped delay inputs, layers, and neurons, respectively.

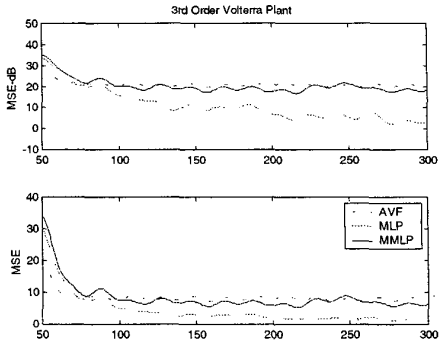


The first condition tested the effect of increasing the number of neurons in a single layer. The input delays were again held constant at three, and the transfer characteristics were again tansigmoid. As expected, more neurons exhibits better convergence properties. The degrees of freedom for approximating the Volterra kernels increases for each neuron added.

With a reasonable MLP model established: tansigmoid transfer characteristics, 50 neurons, two layers, and one delay; the three types of filters were used to identify the second and third order Volterra plants. One delay was chosen in order to match the single delay of the AVF used, so as to achieve a common standpoint.



The above learning curve shows that all three models perform fairly well in identifying the second order plant, the AVF clearly demonstrating the best performance. This is due in part to the fact that the order and delays of the ROO match the plant. The neural networks require more weight updates, so naturally they display inferior convergence properties in this case. The MMLP indicates the second best performance, partially due to the fact that the ROO is carried out and the network is left with a simplified linear optimization task. From the inception, the MLP must perform a complete nonlinear optimization, which makes the optimal weights more difficult to achieve using the backpropogation algorithm.



For the third order plant, the MLP gives the best performance. This is partly due to the fact that the ROO of the AVF and MLP are only computing the second order combinations of the inputs, therefore they cannot exactly approximate a higher order plant. All three models are constrained by the fact that they are equipped with one delay to model the two delays of the plant.

4. Conclusion

Two methods of nonlinear adaptive filtering were compared in this paper: i) performing rank order operations to obtain the nonlinear terms and then linearly filtering those terms, and ii) using a neural network to produce the nonlinear terms via nonlinear transfer. By including a sufficient number of neurons from the onset, the network can accurately approximate a greater number of higher order terms in the nonlinearity being modeled.

References

[1] Bose N., Liang P., *Neural Networks: Graphs and Algorithms*, McGraw-Hill Book Company, N.Y., 1996.

[2] Mathews J.V., Sicuranza G.L., *Polynomial Signal Processing*, John Wiley & Sons, Inc., N. Y., 2000.

[3] Luo F., Unbehauen R., *Applied Neural Networks for Signal Processing*, Cambridge University Press, N.Y., 1997.

[4] Mathews J.V, "Adaptive Polynomial Filters", *IEEE SP Magazine*, July 1991.

[5] Cao J., Yahagi T., "Nonlinear Adaptive Digital Filters Using Parallel Neural Networks", *Proceedings of the IEEE International Conference on Neural Networks*, Volume 2, 1995.

[6] Govind G., Ramamoorthy P.A., "Muli-layered Neural Networks and Volterra Series: The Missing Link", *IEEE International Conference on Systems Engineering*, 1990.