

# THE DESIGN OF MULTI-LAYER PERCEPTRONS USING BUILDING BLOCKS

Kamyar Rohani<sup>1</sup> and Michael T. Manry<sup>2</sup>

<sup>1</sup>Motorola Incorporation, 5555 North Beach Street,  
Fort Worth, Texas 76137

<sup>2</sup>Dept of Elec Eng, University of Texas at Arlington,  
Arlington, Texas 76019

## Abstract

In this paper, we describe a building block approach for constructing large back-propagation (BP) neural networks. It is shown that this approach results in considerably less training time than conventional BP, which starts from random initial weights. An example is given in which the randomized initial weight network fails to learn, but the assembled network succeeds.

## I. Introduction

Over the last few years, back-propagation training has become a popular method for training feed-forward neural networks. Several problems exist with this method:

- (1) Convergence of BP is highly dependent on the initial set of weights chosen. If the initial selection is close to a local minima, the algorithm converges to this point rather than the optimal solution.
- (2) Many iterations of this algorithm may be necessary for convergence. This again depends on the error surface and the initial set of weight values.
- (3) Training is done from a set of tuples selected from input/output spaces. The network may not train to an "optimal" solution if the training set is not properly chosen.

In trying to solve these problems, Waibel and his colleagues [1] designed separate sub-nets for the detection of different phonemes, in a TDNN network. In this paper we extend this work by designing modular feed-forward BP nets which perform various signal processing tasks. Unlike Waibel, our approach involves the mapping of conventional algorithms onto neural network structures. This approach has several benefits: First, it produces alternative parallel structures for implementation of conventional signal processing algorithms. Secondly, it produces a good set of initial weights for BP training. It is shown that improvements to conventional nonlinear signal processing algorithms may be obtained by further training the networks via BP method. Also, a systematic means for analysis of network structures is introduced.

## II. Design Strategy

In mapping conventional signal processing algorithms onto neural networks, first, the operations are broken down into small sub-tasks. Next, the required modules (blocks) are identified and designed. The design of each module can be carried out via conventional back-

---

<sup>1</sup>Proc. of IJCNN'91, Seattle WA., pp. II- 497 to II-502

propagation training or by polynomial mapping strategy, [2,3]. Once each module is designed and tested independently, it is connected to its adjacent modules via linear input/output layers.

At this point in the design phase the resulting network is complete. However, the linear connecting units, which are wasteful as far as training time and network size, are removed, producing a "compact" network. Figure 1 illustrates the process of linear layer removal. The objective is to determine the connection weight between unit  $N_1$  of the left network (in layer  $L_1$ ) and unit  $N_2$  of the right network (in layer  $L_2$ ). The linear layer to be removed is the layer  $L_r$ . All connection weights affected by this process are shown in thick lines. It should be clear that since there is no nonlinearity in the connecting layer, this layer can be removed by a simple linear operation. The connection weight can be computed from

$$W_{L_2 L_1}(N_2, N_1) = \sum_{n=1}^{M(L_r)} W_{L_r L_1}(n, N_1) W_{L_2 L_r}(N_2, n)$$

where  $M(L_r)$  denotes the number of units in the linear layer, and  $W_{mn}$  are the connection weights from unit  $n$  to unit  $m$ . For each nonlinear unit we modify the associated threshold weight as This

$$\theta_{L_2}(N_2) = \sum_{n=1}^{M(L_r)} \theta_{L_r}(n) W_{L_2 L_r}(N_2, n) + \theta_{L_2}(N_2)$$

process should be continued for all units of the right network having connections to the joining layer. Once this has been completed the linear layer can be removed without effecting the outcome of the original network.

### III. Building Blocks Examples

In this section, our system design strategy is demonstrated via two examples: matrix inversion and FM demodulation.

#### Matrix Inversion

It has been shown that it is possible to compute inverse of a matrix via Hopfield nets, [4]. In this section, it is shown that this problem can also be solved using feed-forward nets. The inverse of a 2 by 2 non-singular matrix and its inverse are, respectively,

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}; \quad A^{-1} = B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \frac{1}{|A|} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

where the determinant of  $\mathbf{A}$  is  $|\mathbf{A}| = a_{11}a_{22} - a_{12}a_{21}$ . To invert  $\mathbf{A}$ , its elements must be rearranged and normalized by its determinant. The matrix inversion neural network is shown in Figure 2. This network has four basic modules. Module (a) uses two product sub-nets to compute the determinant. Module (b), which is a 1-4-1 scalar inversion net, avoids the singular point at  $x=0$  by limiting the input's dynamic range to the interval  $[0.5, 1]$ . Module (c) multiplies each input by the inverse of the determinant. Module (d) swaps and scales the normalized inputs to obtain the final outputs. The network of Figure 2 contains linear layers, which are removed to yield the compact 4-4-4-8-4 network of Figure 3.

To check the validity of the model several experiments were conducted. First, effect of further training was studied by initialization of back-propagation training with weights obtained from building blocks. Figure 4 (bottom plot) shows that further training reduces the error. Overlaid on the same plot are also two different cases initialized with random weights. Clearly, the case initialized from building blocks converges much faster and produces a lower error than the other two cases.

### FM Demodulator

Parallel implementation of communication systems is of considerable interest, [5]. As a first step toward this direction using neural network filters, we describe a parallel implementation of a frequency discriminator-type FM demodulator. A FM modulated carrier can be written as

$$C_{FM}(t) = A \cdot \cos\left[\omega_c t + k_f \int_0^t f(\tau) d\tau\right]$$

where  $A$ ,  $\omega_c$ ,  $k_f$ ,  $f(t)$  are constant carrier amplitude, carrier frequency, modulation index, and message signal, respectively. An FM discriminator is built by differentiating the FM modulated carrier. The output of an ideal differentiator is

$$\frac{dc}{dt} = A[\omega_c + k_f f(t)] \sin\left[\omega_c t + k_f \int_0^t f(\tau) d\tau\right]$$

where  $k_f f(t) \ll \omega_c$ . This equation corresponds to an AM modulated signal with a variable carrier frequency. The AM signal envelope is

$$A \cdot \omega_c \left[1 + \frac{k_f}{\omega_c} f(t)\right]$$

with carrier frequency of  $\omega_c + k_f f(t)$ . Thus, the FM demodulator consists of a differentiator and an envelope detector. An FM demodulator network was designed using building block units. This network is composed of the following modules:

1. A 6-5 linear layer for differentiation was formed. This layer is composed of sub-units which compute the difference between two adjacent input values.
2. Squaring was done via a 5-5-5 parallel network. This layer is made up of 1-1-1 sub-units which compute the square of their input values.
3. A 5-1 linear layer was designed for low pass filtering by computing the average of input values.
4. Square-root operation was approximated by a 1-8-1 network in the range of [0,1].

The resulting network after removal of linear layers is a 6-5-8-1 network (Figure 5). This network is functionally equivalent to the FM demodulator.

As an experiment, we initialized a 6-5-8-1 network with random weights and trained it using back-propagation. The test/training waveform was generated from Where  $A$ ,  $f_c$ ,  $A_m$ ,  $f_m$  were set to 0.5, 0.1, 5.0, and 0.01, respectively. The resulting output after

$$C_{FM}(t) = A \cdot \cos[\omega_c t + A_m \sin(\omega_m t)]$$

long training is illustrated in Figure 6. Clearly, the network does not converge to the correct signal. This is partially due to the high correlation between cycles of the sinewave. This indicates the importance of the training sequence in designing feed-forward networks.

For the original network, which was initialized through the building blocks approach, the desired and actual outputs are shown in Figure 7. Note the presence of high frequency terms due to the short length of the averaging filter. After training via back-propagation, we obtained the output waveform of Figure 8. This plot indicates that further training improves the network output by eliminating most of the high frequency signals from the output waveform.

#### IV. Conclusions

In this paper, we have described and demonstrated a modular approach for constructing large back-propagation (BP) neural networks. Unlike previous approaches, ours involves the mapping of conventional algorithms onto neural network structures. This approach has several benefits. First, it produces alternative parallel structures for implementation of conventional signal processing algorithms. Secondly, it produces a good set of initial weights for BP training. It is shown that improvements to conventional nonlinear signal processing algorithms may be obtained by further training the networks via BP method. Also, a systematic means for analysis of network structures is introduced.

**Acknowledgement;** The contributions to this work by Michael T. Manry were funded by the State of Texas through the Advanced Technology Program.

#### References

- [1] A.Waibel, H.Sawai, K.Shikano, "Consonant Recognition by Modular Construction of Large Phonemic Time-Delay Neural Networks", International Conf. on Acoust, Speech, and Signal Processing, ICASSP-89, Vol. S3.9, pp.112-115, May 1990.
- [2] K.Rohani, M.Chen, M.Manry, "A Mapping Approach for Designing Neural Sub-Nets", Submitted to IJCNN'91.
- [3] M. Chen, M. Manry, "Back-Propagation Representation Theorem using Power Series", International Joint Conf. on Neural Net, IJCNN, Vol.I, pp.643-648, San Diego, California, June 17-21, 1990.
- [4] R.Steriti, J.Colman, M.Fiddy, "A Neural Based Matrix Inversion Algorithm", International Joint Conf. on Neural Net, IJCNN, Vol.I, pp.467-470, San Diego, California, June 17-21, 1990.
- [5] J.Bingham, "Multicarrier Modulation for Data Transmission: An Idea Whose Time Has Come", IEEE Comm Magazine, pp.5-14, May 1990.

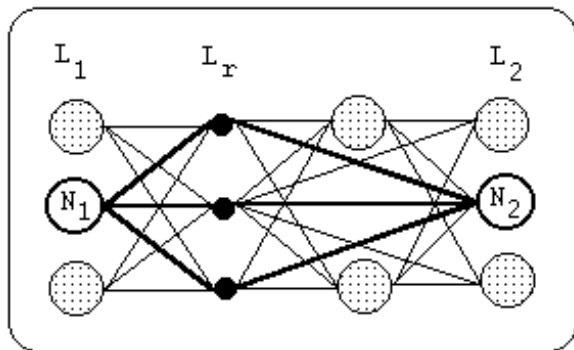


Figure 1

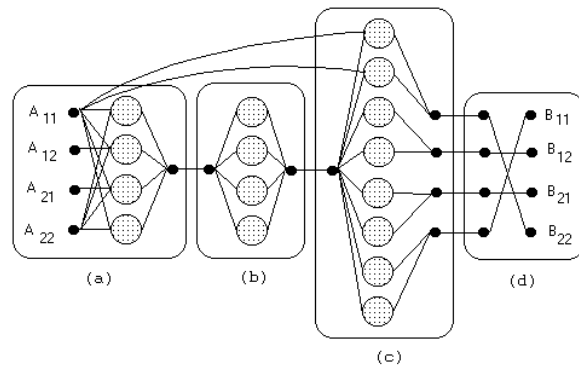


Figure 2

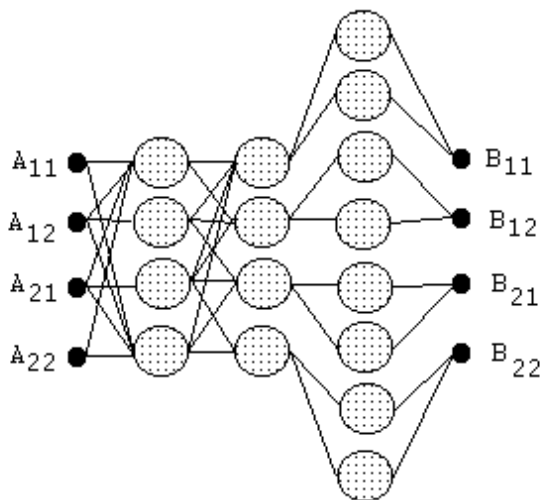


Figure 3

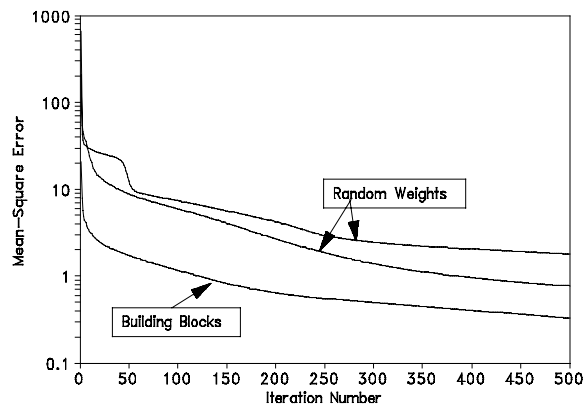
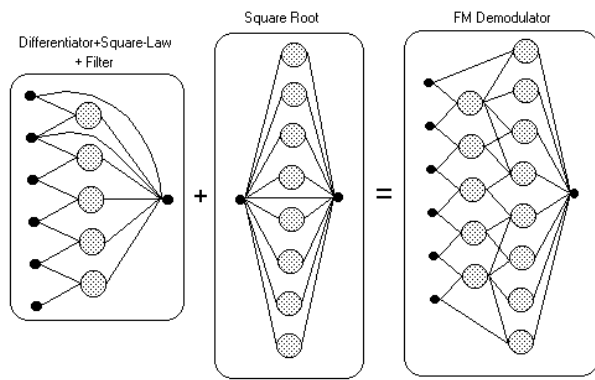
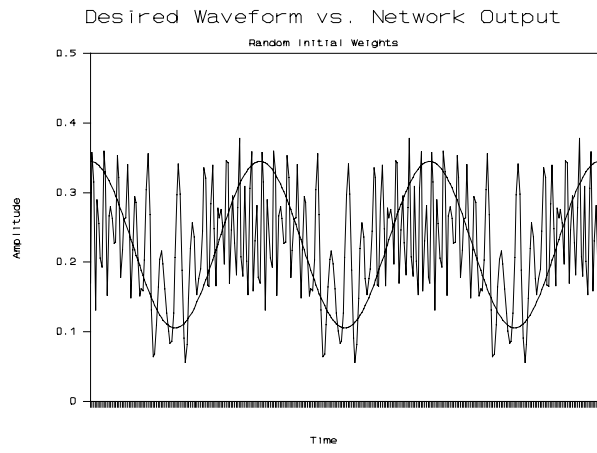


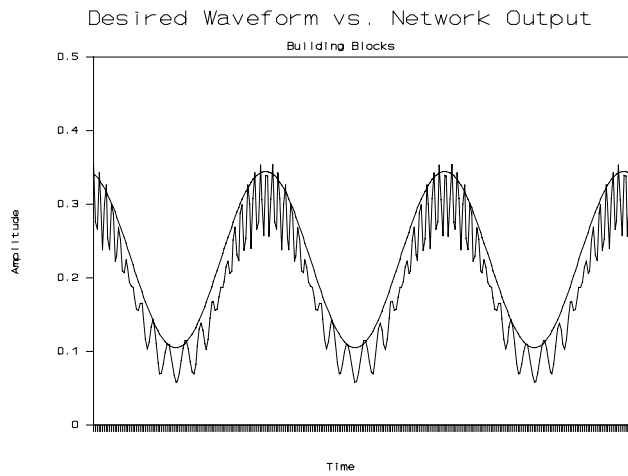
Figure 4



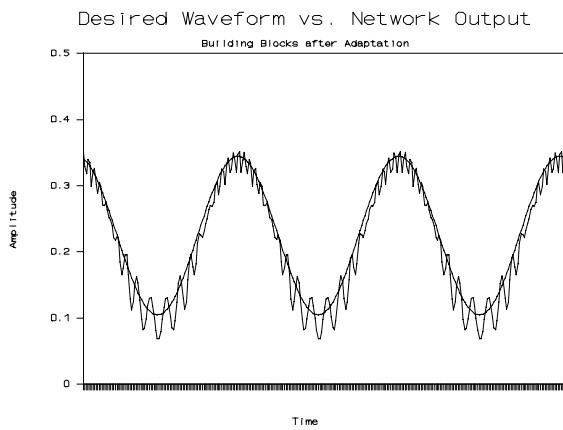
**Figure 5**



**Figure 6**



**Figure 7**



**Figure 8**