

Iterative Improvement of Trigonometric Networks

Iyab I. Sakhnini, Michael T. Manry, and Hema Chandrasekaran
Department of Electrical Engineering
The University of Texas at Arlington
Arlington, Texas 76019
e-mail: manry@uta.edu

Abstract

The trigonometric network, introduced in this paper, is a multilayer feedforward neural network with sinusoidal activation functions. Unlike the N -dimensional Fourier series, the basis functions of the proposed trigonometric network have no strict harmonic relationship. An effective training algorithm for the network is developed. It is shown that the trigonometric network performs better than the sigmoidal neural network for some data sets. A pruning method based on the modified Gram-Schmidt orthogonalization procedure is presented to detect and prune useless hidden units. Other network architectures related to the trigonometric network, such as the sine network, are shown to be inferior to the network proposed in this paper.

I. Introduction

The trigonometric feedforward neural network proposed in this paper has both sine and cosine activations for each hidden layer net function. Since the derivative of the activations for the net function are never simultaneously zero, the hope is that trigonometric networks can avoid the local minima seen in sigmoidal MLP training. In section II, we discuss the relationship between N -dimensional Fourier series and the trigonometric network. In section III, we describe the trigonometric network's architecture and training. Properties of trigonometric network basis functions are analyzed in section IV. In section V, we demonstrate the capabilities of the trigonometric neural network through simulations.

II. Trigonometric Neural Network and N -dimensional Fourier Series

Extending the concept of the multi-dimensional Fourier series to neural networks, consider a network having N input units and N_{out} output units. For the p th pattern, the N -dimensional

Fourier series $y_p(j)$ for the j th output unit can be expressed as [1-3]

$$y_p(j) = \sum_{i=1}^{N_F} [C_s(i, j) \sin(v_p(i)) + C_c(i, j) \cos(v_p(i))] \quad (1)$$
$$v_p(i) = \sum_{k=1}^N \frac{2\pi m(k, i)}{t(k)} x_p(k)$$

where N_F is the number of net functions. N_F depends on the maximum harmonic number, H equal to $(2H+1)^{N/2}$. $C_s(i, j)$ and $C_c(i, j)$ denote the coefficients of the sine and cosine terms of the Fourier series, respectively, for the i th harmonic and the j th output unit. $v_p(i)$ is an inner product of a scaled harmonic number and the inputs, $m(k, i)$ is an element of the coefficient matrix of harmonic number with value $|m(k, i)| \leq H$ and $m(k, 1) = 0$. The period $t(i)$ can be chosen such that $t(i) \geq \max(x(i)) - \min(x(i))$, where $\max(x(i))$ and $\min(x(i))$ denote the maximum and minimum values of the i th input feature, respectively.

One problem with multi-dimensional Fourier series is *combinatorial explosion*. This is where the number of harmonics needed to approximate the signal within a specified error increases drastically as the number of input variables increases. The problem is that the number of hidden nodes, which is equal to $2N_F$, can be very large and can drastically increase as a function of the number of inputs or the number of harmonics needed. Another problem with the Fourier series network is that the generalization property is limited because of the fixed values of input weights feeding into the hidden layer. This comes directly from equation (1). Also, the optimal period of each of the inputs is unknown.

Introducing the trigonometric network can solve the above problems. In this network the input weights in each hidden layer are allowed to adapt thus improving the generalization capability of the network and avoiding the need to know optimal periods for inputs. Another benefit is that the combinatorial explosion problem is brought under control. However, unlike the Fourier series network, the hidden unit basis functions are no longer orthogonal.

III. Architecture and Training of Trigonometric Networks

Trigonometric networks generally have three or more layers. Weights are connected in the forward direction. Input nodes are just connecting nodes with no transfer functions; the output nodes are generally linear nodes. The hidden nodes are nonlinear, with sines and cosines used for their activation functions, instead of the sigmoidal functions used in sigmoidal feedforward neural networks.

A. Trigonometric Networks with One Hidden Layer

Trigonometric networks with a single hidden layer can be implemented as shown in figure 1.

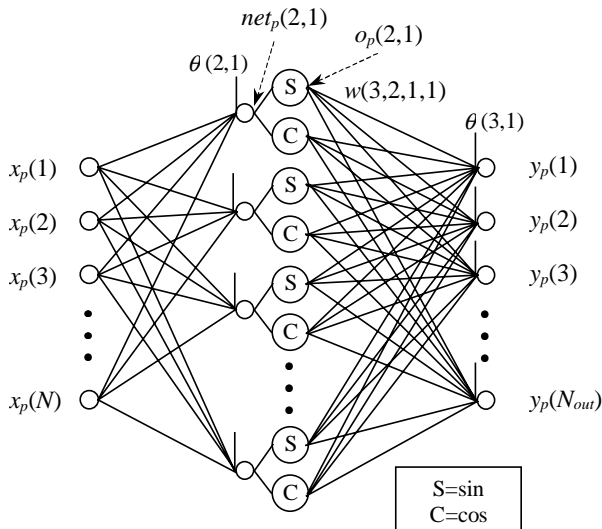


Figure 1

The net function for the m th-hidden node and the p th pattern can be given by:

$$net_p(2,m) = \theta(2,m) + \sum_{k=1}^N w(2,1,m,k)x_p(k) \quad (2)$$

where $1 \leq m \leq N_u(2)/2$, $N_u(2)$ being the number of sines and cosines in the hidden layer (second layer) and is equal to $2N_F$. $x_p(k)$ is the k th input for the p th pattern, $w(i,j,m,k)$ is the weight connecting the k th node of the j th source layer to the m th node of the i th destination layer. $\theta(2,m)$ is the threshold of the m th node of the hidden layer. In the above equation, the input weights are known in the case of a Fourier series network. In the case of the trigonometric network, the input weights are not predefined and are updated by the training algorithm.

The i th hidden node output is

$$o_p(2,i) = \sin(net_p(2,m)), \quad i = (2m-1) \quad (3a)$$

$$o_p(2,i) = \cos(net_p(2,m)), \quad i = 2m \quad (3b)$$

where $1 \leq m \leq N_u(2)/2$. It follows from equations (2) and (3) that the net function $net_p(2,m)$ in equation (2) is equivalent to the inner product $v_p(i)$ in equation (1). Finally, the j th network output for the p th pattern, where $1 \leq j \leq N_{out}$, can be given by

$$y_p(j) = \theta(3,j) + \sum_{i=1}^{N_u(2)} w(3,2,j,i)o_p(2,i) \quad (4)$$

B. Output Weight Optimization-Hidden Weight Optimization (OWO-HWO) Training

The backpropagation method traditionally used for training a feedforward neural network is not very efficient [8]. A more efficient updating algorithm is introduced in this section where the hidden weights are calculated by solving a set of linear equations [9]. This would require that the desired hidden net functions be given, which is not normally the case. However, approximations of these desired net functions can be made based on the current net function values plus a designed net change. Therefore, for the p th pattern, the j th unit of the hidden layer can have its desired net function approximated by [9,13].

$$net_{pd}(2,j) \cong net_p(2,j) + Z \cdot \delta_p(2,j) \quad (5)$$

where $1 \leq j \leq N_u(2)/2$, $net_{pd}(2,j)$ is the desired net function, $net_p(2,j)$ is the actual net function, Z is a learning factor, and $\delta_p(2,j)$, $\delta_p(3,j)$ are defined as follows:

$$\begin{aligned} \delta_p(3,k) &= -2(t_p(k) - y_p(k))f'(net_p(3,k)) \\ \delta_p(2,j) &= f'(net_p(2,j)) \sum_{l=1}^{N_{out}} \delta_p(3,l)w(3,2,l,j) \end{aligned} \quad (6)$$

where $1 \leq k \leq N_{out}$, $1 \leq j \leq N_u(2)$. In most cases, the output layer nodes are linear and therefore have $f'(net_p(3,j))=1$. In the case of the hidden nodes, the activation functions are sines and cosines and thus have $f'(net_p(2,j))$ equal to $\cos(net_p(2,j))$ and $-\sin(net_p(2,j))$, respectively. where $1 \leq j \leq N_u(2)/2$. The hidden weights can therefore be updated according to the following equation:

$$w(2,1,j,i)^{(l+1)} = w(2,1,j,i)^{(l)} + Z \cdot e(2,1,j,i) \quad (7)$$

for $1 \leq j \leq N_u(2)/2$, $1 \leq i \leq N$, and where $e(2,1,j,i)$ serves the same purpose as the negative gradient element,

$-\partial E/\partial w(2,1,j,i)$, in backpropagation. The hidden thresholds can be dealt with as weights having unit input.

Recalling that

$$net_p(2, j) = \sum_{i=1}^{N+1} w(2,1,j,i) o_p(1,i) \quad (8)$$

where $o_p(1,N+1)=1$, $o_p(1,i) = x_p(i)$ for $1 \leq i \leq N$, and $w(2,1,j,N+1)=\theta(2,j)$, we can use the following equation to solve for the changes in the hidden weights,

$$net_p(2, j) + Z \cdot \delta_p(2, j) \equiv \sum_{i=1}^N [w(2,1, j, i) + Z \cdot e(2,1, j, i)] \cdot x_p(i) \quad (9)$$

By manipulating the above equation we can obtain a relation between $\delta(\cdot)$ and $e(\cdot)$:

$$\delta_p(2, j) \equiv \sum_{i=1}^N e(2,1, j, i) \cdot x_p(i). \quad (10)$$

We now define an objective function for the j th unit in the hidden layer as:

$$E_\delta(2, j) = \sum_{p=1}^{N_v} \left[\delta_p(2, j) - \sum_{i=1}^N e(2,1, j, i) \cdot x_p(i) \right]^2 \quad (11)$$

By taking the gradient of $E_\delta(2,j)$ with respect to desired weight changes $e(\cdot)$, we get:

$$g(1, l) = \frac{\partial E_\delta(2, j)}{\partial e(2,1, j, l)} \quad (12)$$

$$= -2 \left(s(1, l) - \sum_{i=1}^N e(2,1, j, i) \cdot r(i, l) \right)$$

where

$$r(i, l) = \sum_{p=1}^{N_v} x_p(i) \cdot x_p(l) \quad (13)$$

$$s(1, l) = \sum_{p=1}^{N_v} \delta_p(2, j) \cdot x_p(l)$$

$$= - \frac{\partial E}{\partial w(2,1, j, l)}$$

Here E is the mean square error (MSE) defined as

$$E = \sum_{k=1}^{N_{out}} E(k) = \frac{1}{N_v} \sum_{p=1}^{N_v} E_p \quad (14)$$

where $E(k)$ is the mapping error for the k th output defined as

$$E(k) = \frac{1}{N_v} \sum_{p=1}^{N_v} [t_p(k) - y_p(k)]^2 \quad (15)$$

where N_v is the total number of patterns in the training data. To minimize $E_\delta(2,j)$, set $g(1,l)$ to zero to get the set of linear equations:

$$\sum_{i=1}^N e(2,1, j, i) \cdot r(i, l) = - \frac{\partial E}{\partial w(2,1, j, l)} \quad (16)$$

By solving this set of linear equations for $e(2,1,j,i)$, we can update the hidden weights as in equation (7). The proper value of the learning factor Z is difficult to determine. However, an automated algorithm can be developed in order to calculate this learning factor [13].

IV. Analysis of Trigonometric Neural Networks

The nonlinearity within a trigonometric network is very distinctive when compared to activation functions of other neural networks. Being periodic in its nature, the sine and cosine activation functions can generate harmonics at the output layer. However, having sines and cosines as activation functions does not ensure any kind of orthonormality for the basis functions fed into the output layer. Because of this, Parseval's theorem may not apply to this kind of network.

A. Basis Functions in Trigonometric Networks

Throughout we refer to a fully connected feed-forward trigonometric network, where each layer is connected to all subsequent layers. This means that in a three-layer network we will have weights connecting the input to the output layer.

The basis functions consist of the set of inputs augmented with the outputs of the hidden layer and an element with the value 1, corresponding to the thresholds of the output layer. The basis functions can be represented by

$$\mathbf{b} = [x_{1p}, x_{2p}, \dots, x_{N_p}, 1, o_{1p}, o_{2p}, \dots, o_{N_h p}]^T \quad (17)$$

where $[\cdot]^T$ is the transpose operation, \mathbf{b} is of dimension $N_u = N + N_h + 1$, x_{kp} and o_{kp} are random variables, where N_h is the number of hidden units, and $1 \leq p \leq N_v$, N_v being the number of patterns.

Parseval's theorem is a very useful tool in estimating the amplitudes of the Fourier coefficients of a given Fourier series. These coefficients correspond to the weights of the output layer in the case of a trigonometric network. This can be extremely useful since the weights of the output layer can be solved for if the basis function and the desired values of the output units are known. If the basis functions(inputs and hidden unit outputs)of the trigonometric net are orthonormal, Parseval's theorem can be applied. Let \mathbf{x} denote the input vector; In this case the theorem states that

$$\sum_{i=1}^{N_{out}} \int y_i^2(\mathbf{x}) f_x(\mathbf{x}) d\mathbf{x} = \sum_{i=1}^{N_{out}} \left[\frac{1}{4} \left\{ \theta_i^2 + \sum_{n=1}^{N_u(2)} (w^2(3,2,i,n)) \right\} \right] \quad (18)$$

where N_{out} is the number of outputs, $N_u(2)$ is the number of units in the hidden layer, θ_i is the thresholds of the i th output

unit and $w(3,2,i,n)$ is the weight connecting the n th unit in the hidden layer to the i th unit in the output layer and $f_x(\mathbf{x})$ is the probability density function of the input vector \mathbf{x} .

Parseval's equation for the trigonometric network, may not, however, be applied directly because it is only valid in the case of orthogonal basis functions. By orthonormalizing the basis functions we can derive a relationship between the MSE and the weights of the output layer and use this relation to determine the usefulness of a particular unit.

To test for the orthonormality of these basis vectors, a matrix \mathbf{P} is constructed having the dimensions $N_u \times N_u$. The \mathbf{P} matrix consists of the inner product of b_i and b_j , the i th and the j th row elements of \mathbf{b} . This inner product can be defined as [10]:

$$p_{ij} = \langle b_i, b_j \rangle \quad 1 \leq i, j \leq N_u. \quad (19)$$

$$\langle b_i, b_j \rangle \equiv E[b_i, b_j],$$

where $E[\cdot]$ is the expectation value operation. The above equation can be approximated as:

$$E[b_i, b_j] \approx \frac{1}{N_v} \sum_{p=1}^{N_v} b_i^p \cdot b_j^p. \quad (20)$$

where p_k^p is the k th element of \mathbf{b} for the p th pattern. From the above discussion, it can be seen that if the basis vectors are orthonormal, \mathbf{P} should be a diagonal matrix with $p_{ij} = 1$ for $i = j$ and $p_{ij} = 0$ for $i \neq j$. However, if this is applied directly to the trigonometric network, this is not be the case. These basis functions can be forced to be orthonormal after which Parseval's equation can be safely applied. We have employed modified Gram-Schmidt orthonormalization process to orthonormalize these basis functions [14].

B. Pruning Trigonometric Networks

Orthonormalizing the basis vectors simplifies the MSE expression which can be expressed as

$$E_k = E'_k - \sum_{i=1}^{N_u} w^2(3,2,k,i) \quad 1 \leq k \leq N_{out} \quad (21)$$

Thus MSE in an orthonormalized trigonometric network can be directly related to the square of the weights of the output layer. This equation can then indicate the most important as well as the least important units in the hidden layer. We then proceed to build an ordered orthonormal basis by iteratively applying Gram-Schmidt orthonormalization process. The ordering of the basis functions[14] will only involve ordering of the hidden layer units. This ordering process arranges the hidden units in the order of their importance. Pruning then amounts to disconnecting the units in the tail end of the order.

This would increase the MSE by an amount proportional to the square of its weights, as can be inferred from equation (21).

V. Numerical results

A fair comparison between the two types of networks restricts the choice of the number of hidden units in each network. For the comparison to be fair, the number of hidden units is chosen such that the two networks have the same pattern storage capability. Pattern storage can be defined as the number of patterns that a neural network can learn perfectly such that its outputs exactly match the desired output values.

A. Pattern Storage Bounds

The upper bound on pattern storage can be derived from that of the functional link network (FLN). The FLN network is capable of storing patterns with 100% efficiency. The pattern storage of a FLN is the number of weights and thresholds per output [4,6,7,11] which is written as

$$M_{FLN} = \frac{N'_w}{N_{out}} \quad (22)$$

where N'_w is the total number of weights and thresholds in the network and N_{out} is the number of outputs. This storage capability of the FLN can be regarded as the upper bound on the pattern storage for the trigonometric network as well as the MLP. For the other network can be found by equating the upper bounds on pattern storage for the two networks. For the three-layer network, the number of MLP hidden units can be given by

$$N_{u,MLP}(2) = \frac{N_{u,Trig}(2) \cdot (2N_{out} + N + 1)}{2 \cdot (N_{out} + N + 1)} \quad (23)$$

where $N_{u,MLP}(2)$ and $N_{u,Trig}(2)$ are the number of hidden units in the MLP and the trigonometric networks, respectively.

B. Simulation results

1. Training

We trained sigmoidal, trigonometric and sine activation neural networks using the OWO-HWO algorithm and Figures 2,3 and 4 depict the results.

Figure2: The training data file Twod.tra has 8 inputs, 7 outputs and contains 1768 training patterns. The testing data file Twod.tst has 1000 patterns. The inputs consist of eight theoretical values of backscattering coefficient parameters at V and H polarizations and four incident angles. The outputs are the corresponding values of permittivity, upper surface height, lower surface height, normalized upper surface

correlation length, normalized lower surface correlation length, optical depth and single scattering albedo which has a joint uniform pdf. [12].

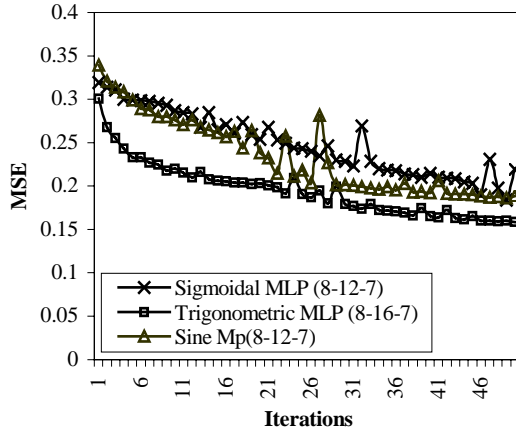


Figure 2

Figure 3: The data set *Single2* has 16 inputs and 3 outputs, and represents the training set for inversion of surface permittivity ϵ , the normalized surface rms roughness $k\sigma$, and the surface correlation length kL found in backscattering models from randomly rough dielectric surfaces [12]. The training data file *Single2.tra* has 5992 patterns and the testing data file *Single2.tst* has 4008 patterns.

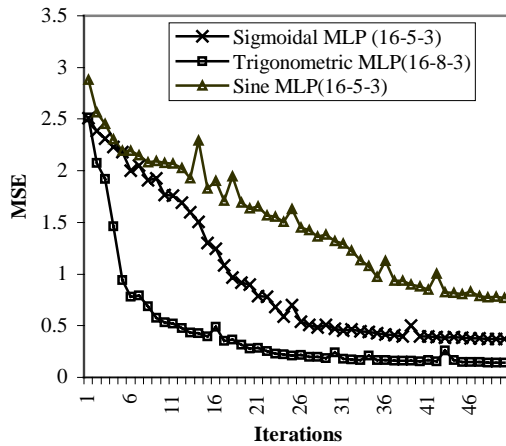


Figure 3

Figure 4: The third example is the data set *F17*, which contains 2823 training patterns and 1922 testing patterns for onboard *Flight Load Synthesis* (FLS) in helicopters. There are 17 inputs and 9 outputs for each pattern. This data was obtained from the M430 flight load level survey conducted in Mirabel Canada in early 1995 by Bell Helicopter of Fort Worth.

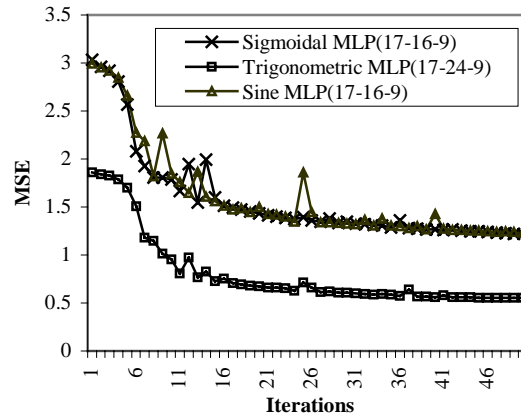


Figure 4

2. Generalization Performance

The trained networks were then tested on the respective testing data files to evaluate their generalization capability. The results are tabulated in Table 1. We have demonstrated that the trigonometric neural networks train more easily and have good generalization performance.

Table 1. Training and Testing results

	Sigmoidal MLP	Trigono-metric MLP	Sine MLP
Twod.tra Training	0.184123	0.158191	0.186347
Twod.tst Testing	0.20198	0.224112	0.207087
Single2.tra Training	0.373223	0.142646	0.76717
Single2.tst Testing	0.398947	0.168315	0.76038
F17.tra Training	1.22043	0.5501	1.23591
F17.tst Testing	1.28761	0.64062	1.31366

V. Conclusions

In this paper we have developed an effective training algorithm for trigonometric networks. We have given the theoretical framework for understanding the trigonometric neural networks in terms of conventional basis function analyses. Using data sets from remote sensing applications and flight load synthesis, we have established that trigonometric networks train well and have good generalization capabilities. Trigonometric networks almost always perform better than sine networks and often perform better than sigmoidal MLPs.

Acknowledgement

This work was funded by the Advanced Technology Program of the state of Texas under grant 003656-063.

References

- [1] W. Rudin, *Principles of Mathematical Analysis*. McGraw-Hill, second edition, 1964.
- [2] H.S. Carslaw, *Theory of Fourier's Series and Integral*. Dover Publications, third edition, 1930.
- [3] Geogri P. Tolstov, *Fourier Series*, Prentice-Hall, 1962.
- [4] K.Rohani, M.S. Chen and M.T.Manry, "Neural Subnet Design by Direct Polynomial Mapping," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 1024-1026 (Nov. 1992).
- [5] P.E. Gill, W. Murray, and M.H. Wright, *Practical Optimization*. New York: Academic Press, 1981.
- [6] M. S. Chen and M. T. Manry, "Nonlinear Modeling of Back-Propagation Neural Networks," *Proceedings of International Joint Conference on Neural Networks*, Seattle, pp. 899 (1991).
- [7] M. S. Chen and M.T.Manry, "Conventional Modeling of the Multi-Layer Perceptron Using Polynomial Basis Function" *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 164-166 (Jan. 1993).
- [8] P.Werbos, "Backpropagation: Past and Future," *The Proceedings of the IEEE International Conference on Neural Networks*, pp. 343-353 (1988).
- [9] R.S.Scalero and N.Tepedelenlioglu, "A Fast New Algorithm for Training Feedforward Neural Networks," *IEEE Transactions on Signal Processing*, vol. 40, no.1, pp. 202-210 (Jan. 1992).
- [10] Papoulis, *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 1965.
- [11] A.Gopalakirshnan, X.Jiang, M.S.Chen, and M.T. Manry, "Constructive Proof of Efficient Pattern Storage in the Multi-layer Perceptron," *Twenty Seventh Asilomar Conference on Signals, Systems and Computers*, pp. 386-390 (Nov. 1993).
- [12] A.K.Fung, Z.Li, and K.S.Chen, "Backscattering from a randomly rough dielectric surface", *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 30, No. 2, pp. 356-369, 1992.
- [13] H.H.Chen, M.T. Manry and Hema Chandrasekaran , "A Neural Network Training Algorithm Utilizing Multiple sets of Linear Equations", *Neurocomputing* Vol. 25 No.1-3, pp 55-72, April 1999.
- [14] Z.J.Yang, "Hidden-layer Size Reducing for Multilayer Neural Networks Using the Orthogonal Least-Squares Method", *Proceedings of the 36th SICE Annual Conference*, vol. 37, pp. 1089-1092 (July 1997).