A Clustering Technique for Digital Communication Channel Equalization Using Wavelet Neural Networks

Hung-Ming Tsai[&] Cheng-Jian Lin[+]* Po-Yueh Chen[+]

[&]*Institute of Networking and Communication Engineering*
Chaoyang University of Technology
[+] *Department of Computer Science and Information Engineering*
Chaoyang University of Technology
168 Gifeng E. Rd., Wufeng, Taichung County, 413 Taiwan, R. O. C.

Abstract-*In this paper, we propose a wavelet neural network (WNN) for nonlinear time-invariant and time-varying channel equalizers. The WNN model is a four-layer structure which is comprised of an input layer, a wavelet layer, a product layer, and an output layer. A hybrid learning algorithm consists of structure and parameter learning algorithms. The structure learning is based on a self-clustering algorithm (SCA). It not only considers the original dilation and translation but also consider every translation and dilation's variation of dimension in the input data. The parameter learning is based on a simultaneous perturbation method for adjusting the parameters. Computer simulation results show that the bit error rate of the WNN equalizer is close to that of the optimal equalizer.*

Keywords: Cluster, wavelet neural network, digital communication, equalizer.

1. Introduction

During the past few years, applications of high-speed communication are required and fast increasing. Nonlinear distortion becomes a major factor which limits the performance of a communication system. High speed communications channels are often impaired by the channel inter-symbol interference (ISI), the additive white Gaussian noise (AWGN) [1]-[3] and the effects of time-varying channels [4]. All these effects are nonlinear and complex problems. Nevertheless, adaptive equalizers are used in digital communication system receivers to mitigate the effects of non-ideal channel characteristics and to obtain reliable data transmission.

Recently, some researches have been done on applications of wavelet neural networks (WNN), which combine the capability of artificial neural networks to learn from processes with the capability of wavelet decomposition, for the identification and control of dynamic systems. The main characteristics of WNN are the capability of incorporating the time-frequency localization properties of wavelets and the learning abilities of general neural networks. Therefore, the WNN can be applied to complex nonlinear system modeling [5]-[7].

In these networks, the learning scheme is of much interest, with the back-propagation method being widely used. A gradient type of learning rule is not easy to implement in a real system, since calculation of the gradients for all weights in the network is very difficult. The finite difference is a simple example. Jabri *et al.* pointed out the usefulness of such a learning rule. However, that learning rule has a fault in that it does not take good advantage of parallel processing of NN's. In the present technique, we add a perturbation to all weights one by one and obtain corresponding values of an error function, in order to use a difference approximation. Therefore, the learning rule using the finite difference approximation requires n-times forward operation of the network to obtain modifying quantities for all weights, where denotes the total number of weights of the network. Thus, if the NN is large, we cannot expect viability of the learning rule in the sense of the operating speed. More suitable is the simultaneous perturbation method [8]-[10], since the learning rule requires only forward operations of the network to modify parameters unlike the back-propagation method.

2. The Structure of WNN

The structure of the WNN model is shown in Fig.1. The input data in the input layer of the network

*Corresponding author
E-mail: cjlin@cyut.edu.tw

is $x = [x_1, x_2, \ldots, x_i, \ldots, x_n]$, where n is the number of dimensions. Then, the activation functions of the wavelet nodes in the wavelet layer are derived from the mother wavelet $\phi(x)$, with a dilation of d and a translation of t. The mother wavelet

$$\phi_{d,t} = 2^{d/2}\phi(2^d x - t) \quad (1)$$

The derivation of a differentiable Mexican-hat function is adopted as a mother wavelet herein,

$$\phi(x) = \left(1 - \|x\|^2\right)e^{-\|x\|^2/2}, \quad (2)$$

where $\|x\|^2 = x^T x$. Therefore, the activation function of the jth wavelet node connected with the ith input data is represented as:

$$\phi_{d_j,t_j}(x_i) = 2^{d_{ij}/2}\left(1 - \left\|2^{d_{ij}} x_i - t_{ij}\right\|^2\right)e^{-\left\|2^{d_{ij}} x_i - t_{ij}\right\|^2/2},$$

$$i = 1, \ldots, n \quad j = 1, \ldots, m \quad (3)$$

where n is the number of input-dimensions and m is the number of the wavelets. The wavelet functions of (3) with various dilations and translations are presented in Fig.2. Then, each wavelet in the product layer is labeled Π, i.e., the product of the jth multi-dimensional wavelet with n input dimensions $x = [x_1, x_2, .., x_i, .., x_n]$ can be defined as

$$\psi_j(x) = \prod_{i=1}^{n} \phi_{d_j,t_j}(x) \quad (4)$$

According to the theory of multi-resolution analysis (MRA), $f \in L^2(\Re)$ any can be regarded as a linear combination of wavelets at different resolution levels. For this reason, the function f is expressed as

$$Y(x) = f(x) \approx \sum_{j=1}^{m} w_j \psi_j(x) \quad (5)$$

if $\psi_j = [\psi_1, \psi_2 \ldots, \psi_m]$ is used as a nonlinear transformation function of hidden nodes and weight vectors and $w_j = [w_1, w_2 \ldots, w_m]$ defines the connection weights, then Eq. (5) can be considered the functional expression of the WNN modeling function Y.

3. A Hybrid Learning Algorithm for WNN

In this section, we propose a hybrid learning algorithm for the WNN model. The following two schemes are part of this learning algorithm [11]. First, a structure learning scheme is used to determine proper input space partitioning and to find the mean of each cluster. Second, a simple learning scheme is used to adjust the parameters for the desired outputs. The applied learning algorithm uses the self-clustering algorithm (SCA) to perform structure learning and the simultaneous perturbation algorithm to perform parameter learning.

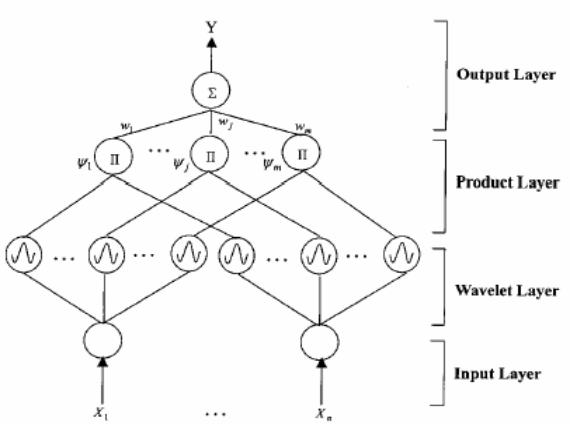

Figure 1.The architecture of the WNN model

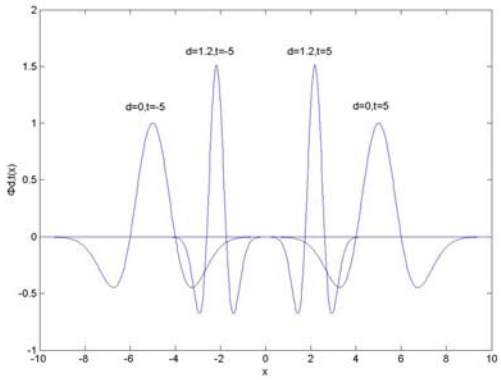

Figure 2.Wavelet bases with different translations and dilations

3.1. The Self-Clustering Algorithm

In the clustering process, the data examples come from a data stream, and the process starts with an empty set of clusters. When a new cluster is created, the cluster center, C, is defined, and its cluster distance and cluster width, Dc and Wd, is initially set to zero. When more examples are presented one after another, some created clusters will be updated by changing the positions of their centers and increasing the cluster

distances and cluster width. Which cluster will be updated and how much it will be changed depends on the position of the current example in the input space. A cluster will not be updated any more when its cluster distance, *Dc*, reaches the value that is equal to the threshold value D_{thr}.

Step 1: Create the first cluster by simply taking the position of the first example from the input stream as the first cluster center C_1, and setting its cluster distance Dc_1 and cluster width Wd_1_x and Wd_1_y to zero.

Step 2: If all examples of the data stream have been processed, the algorithm is finished. Otherwise, the current input example, P_i, is taken and the distances between this example and all *n* already created cluster centers C_j, $Dist_{ij} = \|P_i - C_j\|$, j=1,2,…,*n*, are calculated.

Step 3: If there is any distance value $Dist_{ij}$ equal to, or less than, at least one of the distance Dc_j, j=1,2,…,*n*, it means that the current example P_i belongs to a cluster C_m with the minimum distance

$$Dist_{im} = \|P_i - C_m\| = min(\|P_i - C_j\|), j = 1,2,...,n \quad (6)$$

In this case, neither a new cluster is created, nor any existing cluster is updated. The algorithm then returns to Step2. Otherwise, go to the next step.

Step 4: Find a cluster with center C_m and cluster distance Dc_m from all *n* existing cluster centers by calculating the values $S_{ij} = Wd_{ij} + Dc_j$, j=1,2,…,*n*, and then choosing the cluster center C_m with the minimum value S_{im}:

$$S_{im} = Wd_{im} + Dc_m = min(S_{ij}), j = 1,2,...,n \quad (7)$$

Step 5: If S_{im} is greater than D_{thr}, the example P_i does not belong to any existing clusters. A new cluster is created in the same way as described in Step 1, and the algorithm returns to Step2.

Step 6: If S_{im} is not greater than D_{thr}, the cluster C_m is updated by moving its center, C_m, and increasing the value of its cluster distance, Dc_m, and cluster width Wd_m_x, Wd_m_y. The parameters are updated by the following equation:

$$Wd_m\_x^{new} = (\|C_m\_x - P_i\_x\| + Wd_m\_x)/2 \quad (8)$$

$$Wd_m\_y^{new} = (\|C_m\_y - P_i\_y\| + Wd_m\_y)/2 \quad (9)$$

$$C_m\_x^{new} = P_i\_x - D_m\_x^{new} \quad (10)$$

$$C_m\_y^{new} = P_i\_y - D_m\_y^{new} \quad (11)$$

$$Dc_m^{new} = S_{im}/2 \quad (12)$$

where C_m_x is a value of the *x* dimension for C_m, C_m_y is a value of the *y* dimension for C_m, P_i_x is a value of the *x* dimension for P_i, and P_i_y is a value of the *y* dimension for P_i. The algorithm returns to Step 2.

3.2. Simultaneous Perturbation Algorithm

Details of the simultaneous perturbation method as a learning rule of NNs have been described previously [8]-[10] and are reiterated in this section.

First of all, we define the following perturbation vector c_l that gives small disturbances to all weights

$$c_l = (c_l^1, \ldots, c_l^n)^T \quad (13)$$

where the subscript *l* denotes an iteration.

The perturbation vector c_l has the following properties.

1. c_l^i is a uniformly random number in an interval $[-c_{max}, c_{max}]$ except an interval $[-c_{min}, c_{min}]$ and is independent with respect to time *l* for $i = 1, \ldots, n$.
2. $E(c_l) = 0$.
3. $E(c_l^i c_l^j) = \begin{cases} 0 & if\ i = j \\ \sigma^2 & if\ i \neq j \end{cases}$

   $E(\cdot)$ denotes expectation. σ^2 is a variance of the perturbation c_l^i.

Learning Rule 1: First, we estimate $\partial J / \partial w$ overall, then the *i*th component of the modifying vector of the weights Δw_l, translations Δt_l and dilations Δd_l are defined as follows:

$$\Delta w_l^i = \frac{J(u(w_l + c\_w_l)) - J(u(w_l))}{c\_w_l^i} \quad (14)$$

$$\Delta t_l^i = \frac{J(u(t_l + c\_t_l)) - J(u(t_l))}{c\_t_l^i} \quad (15)$$

$$\Delta d_l^i = \frac{J(u(d_l + c\_d_l)) - J(u(d_l))}{c\_d_l^i} \quad (16)$$

The parameters are updated in the follows manner:

$$w_{l+1} = w_l - \alpha \Delta w_l \quad (17)$$

$$t_{l+1} = t_l - \alpha \Delta t_l \quad (18)$$

$$d_{l+1} = d_l - \alpha \Delta d_l \quad (19)$$

where α is a positive learning coefficient.

Learning Rule 2: Next, we estimate the first differential coefficient of the unknown plant with

respect to the parameters of the NN, then we can obtain the following modifying quantity:

$$\Delta w_l^i = \varepsilon_l \frac{f(u(w_l + c\_w_l)) - f(u(w_l))}{c\_w_l^i} \tag{20}$$

$$\Delta t_l^i = \varepsilon_l \frac{f(u(t_l + c\_t_l)) - f(u(t_l))}{c\_t_l^i} \tag{21}$$

$$\Delta d_l^i = \varepsilon_l \frac{f(u(d_l + c\_d_l)) - f(u(d_l))}{c\_d_l^i} \tag{22}$$

where $\varepsilon_l = (y_l - y_{dl})$. Since the error ε_l can be easily measured, in this learning rule, only $\partial f / \partial w$ is estimated by means of the simultaneous perturbation.

4. Illustrative Examples

A discrete time model of a digital communication system is depicted in Fig.3. A random sequence x_i is passed through a dispersive channel of finite impulse response (FIR), to produce a sequence of outputs $\hat{y}_i$ .A term, e_i, which represents additive noise in the system, is then added to each $\hat{y}_i$ to produce an observation sequence y_i. The problem to be considered is that of utilizing the information represented by the observed channel outputs $y_i, y_{i-1}, \ldots, y_{i-m+1}$ to produce an estimate of the input symbol x_{i-d} . A device that performs this function is known as an equalizer. The integers *m* and *d* are known as the order and the delay of the equalizer, respectively. Throughout, the input samples are chosen from {-1 1} with equal probability and assumed to be independent of one another.

The equalizer performance is described by the probability of misclassification with respect to the signal-to-noise ratio (SNR). With the assumption of independent identically distributed (i.i.d.) sequence the SNR can be defined as

$$SNR = 10\log_{10}\frac{\sigma_s^2}{\sigma_e^2} \tag{23}$$

where σ_s^2 represents the signal power and σ_e^2 is the variance of the Gaussian noise.

4.1. Application of Time-Invariant Channel

The equalizer order is chosen as m=2. Let the channel transfer function be

$$\hat{y}(n) = l(n) + 0.1l(n)^3 \tag{24}$$
$$l(n) = 0.5x(n) + x(n-1)$$

then the output channel signal is

$$\hat{y}(n) = (0.5x(n) + x(n-1)) + 0.1 \times (0.5x(n) + x(n-1))^3 \tag{25}$$

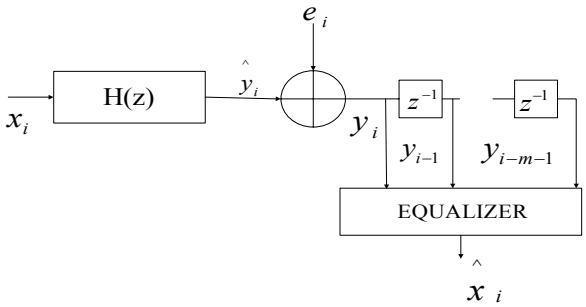

Figure 3.Schematic of the data transmission system

All the combinations of *x(n)* and the desired channel states are listed in Table 1. To see the actual bit error rate (BER), a realization of 10^6 points of sequence *x(n)* and *e(k)* are used to test the BER of trained WNN equalizer. The resulting BER curve of the WNN equalizer under the different SNR is show in Fig. 4.

We now compare the performance of our model with Bayesian equalizer. The Bayesian equalizer is near optimal method for communication channel equalizer. Computer simulation results show that the bit error rate of the WNN is close to the optimal equalizer.

4.2. Bayesian equalizer

The Bayesian decision theory provides the optimal solution to the general decision problem. Therefore, the optimal symbol-by-symbol equalizer can be formed from the Bayesian probability theory and is termed a Bayesian or maximum a posteriori probability (MAP) equalizer.

This minimum error probability decision can be rewritten as

$$f_B(y(n)) = \sum_{i=1}^{n_s^+} \exp\left(-\left\|y(n) - y_i^+\right\|^2 / 2\sigma^2\right) - \sum_{i=1}^{n_s^-} \exp\left(-\left\|y(n) - y_i^-\right\|^2 / 2\sigma^2\right) \tag{26}$$

$$\hat{x}(n) = \mathrm{sgn}(f_B(y(n))) = \begin{cases} 1, & f_B(y(n)) \geq 0 \\ -1, & f_B(y(n)) < 0 \end{cases} \quad (27)$$

where y_i^+ and y_i^- refer to the channel states which are +1 and –1 signal states, which have estimates of the noise-free received signal vector. Therefore, we can base on this function to make the optimal decision boundary. Form this point of view, the equalizer can be viewed as a classifier, and the problem can be considered as a classification problem.

For noise is 10db, 1000samples of *y(n)* are plot using dots in fig. 5. The shaded region is the region where the transmitted signal is classified as -1, otherwise it is classified as 1. This way of making decisions is optimal because it produces the minimum average error probability or bit error rate.

NO	x(n)	x(n-1)	x(n-2)	y(n)	y(n-1)
1	1	1	1	1.8375	1.8375
2	1	1	-1	1.8375	-0.5125
3	1	-1	1	-0.5125	0.5125
4	1	-1	-1	-0.5125	-1.8375
5	-1	1	1	0.5125	1.8375
6	-1	1	-1	0.5125	-0.5125
7	-1	-1	1	-1.8375	0.5125
8	-1	-1	-1	-1.8375	-1.8375

Table 1.Input and desired channel states. m=2 and d=0

4.3. Application of Time-Varying Channel

Let the nonlinear time-invariant channel transfer function be Eq. 25 where a_1=0.5 and a_2=1. Since we assume the channel is time-varying, a_1 , a_2 are two time-varying coefficients. Those time-varying coefficients are generated by passing the white Gaussian noise through a Butterworth lowpass filter (LPF). The example is centered at a_1=0.5 and a_2=1 and the input for the Butterworth filter is a white Gaussian sequence with standard deviation (std) β. Applying the function provided by the Matlab Signal Processing Toolbox, we can generate a second-order lowpass digital Butterworth filter with cutoff frequency β=0.1.

The main portion of the program for Matlab is shown as follows:

```
[B, A]=butter (2, 0.1);
a1= 0.5+ filter (B, A, beta * randn (1, 1000));
a2= 1+ filter (B, A, beta * randn (1, 1000));
```

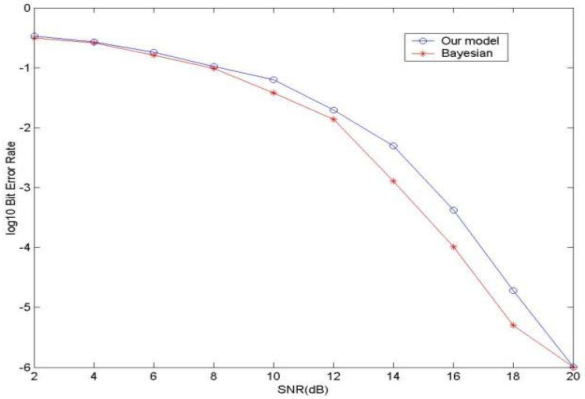

Figure 4.Comparison of bit-error-rate curves for the Bayesian, WNN equalizer.

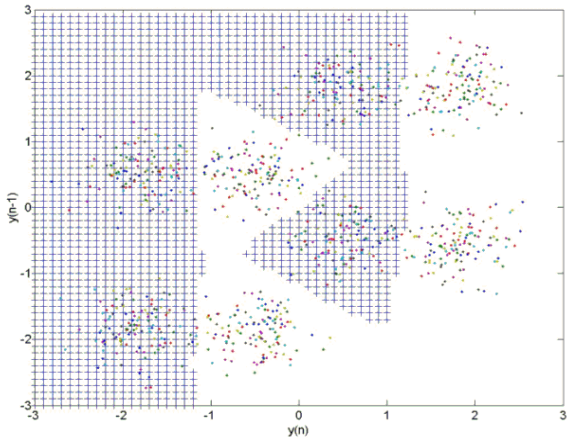

Figure 5.Desired channel, data clusters, SNR=10db, 1000 samples of *y(n)*, and decision boundary.

Adjusting the time-varying coefficients, the coefficients and the corresponding channel states are plotted in Figure 6(a) and (b) respectively.

To see the actual bit error rate (BER), a realization of 10^6 points of sequence *x(n)* and *e(k)* are used to test the BER of trained WNN equalizer. The resulting BER curve of the WNN equalizer under the different SNR is show in Fig. 7. We compare the performance of our model with Bayesian equalizer. Computer simulation results show that the bit error rate of the WNN is close to the optimal equalizer.

In the next experiment, we fixed SNR at 20 dB and ran simulations for eight different β values ranging from β = 0.04 to β=0.32, with step size 0.04 (set d = 0). The result is shown in Figure 8. We compare the performance of our model with Bayesian equalizer. Computer simulation results show that the bit error rate of the WNN is close to the optimal equalizer.

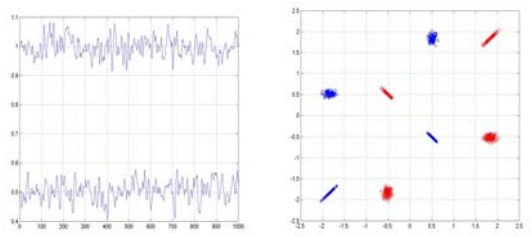

Figure 6.For the channel (a) an example of a time-varying channel with β=0.1. (b) Channel states (noise free) of the time-varying channel.

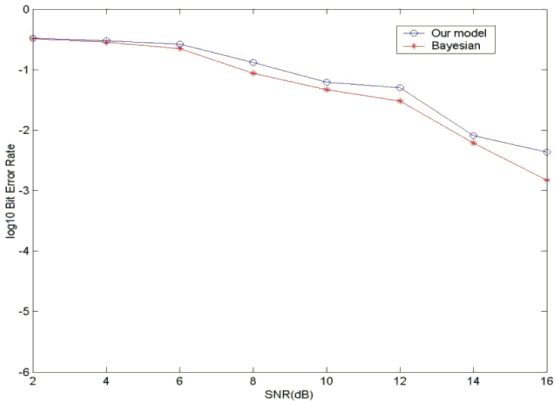

Figure 7.Comparison of bit-error-rate curves for the Bayesian and WNN equalizer, in time-varying channel with $\beta = 0.1$.

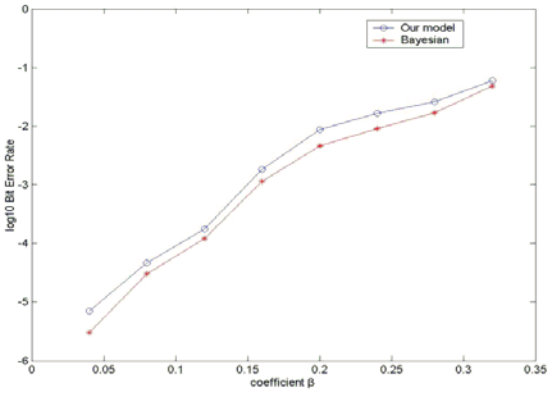

Figure 8.Comparison of bit-error-rate curves for the Bayesian and WNN equalizer, in time-varying channel with SNR=20dB, β = 0.04 to 0.32.

5. Conclusion

This paper describes the wavelet neural network with simultaneous perturbation, which is a stochastic gradient-like learning rule. The rules need only twice operations to obtain the modifying quantities of all parameters. We applied a WNN model to the time-invariant and time-varying channel problems. Simulation results show that the bit error rate of the proposed WNN model is close to the Bayesian equalizer.

Acknowledgment

This research is supported by the National Science Council of R.O.C. under grant NSC 92-2213-E-324-002.

Reference

[1] Gavin J. Gibson, Sammy Siu, and Colin F. N. Cowan, "The Application of Nonlinear Structure to the Reconstruction of Binary Signal," IEEE Transaction on Signal Processing, Vol. 39, No. 8, August 1991.

[2] Khalid A. Al-Mashouq, "The Use of Neural Nets to Combine Equalization with Decoding for Severe Intersymbol Interference Channels," IEEE Transaction on Neural Networks, Vol. 5, No. 6, November 1994.

[3] Sheng Chen, Bernard Mulgrew, and Peter M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," IEEE Transaction on Neural Networks, Vol. 4, No. 4, July 1993.

[4] Qilian Liang and Jerry M. Mendel, "Equalization of nonlinear time-varying channels using type-2 fuzzy adaptive filters," IEEE Transactions on Fuzzy Systems, Vol. 8, No. 5, October 2000.

[5] P. Cristea, R. Tuduce, and A. Cristea "Time Series Prediction with Wavelet Neural Networks," *Proceedings* of IEEE Neural Network Applications in Electrical Engineering, 2000.

[6] J. Zhang, G. G. Walter, Y. Miao, and W. N. W. Lee, "Wavelet Neural Networks for Function Learning," IEEE Trans. Signal Processing, Vol. 43, p1485-1497, June 1995.

[7] Daubechies, I. "The wavelet transform, time frequency localization and signal analysis," *IEEE Transforms on Information Theory,* Vol. 36, No. 5, pp. 961-1005, 1990.

[8] Yutaka Maeda and Toshiki Tada, "FPGA Implementation of a Pulse Density Neural Network With Learning Ability Using Simultaneous Perturbation," IEEE Transaction on Neural Networks, Vol. 14, No. 3, May 2003.

[9] Yutaka Maeda and Rui J. P. De Figueiredo, "Learning Rules for Neuro-Controller via Simultaneous Perturbation," IEEE Transaction on Neural Networks, Vol. 8, No. 5, September 1997.

[10] Y. Maeda, H. Hirano, and Y. Kanata, "A learning rule of neural networks via simultaneous perturbation and its hardware implementation," *Neural Networks*, vol. 8, 1995

[11] Cheng-Jian Lin, Chi-Yung Lee and Cheng-Hung Chen, "A Self-Adaptive Quantum Radial Basis Function Network for Classification Applications," International Joint Conference on Neural Networks, Budapest, July 200