ADAPTIVE COMPLEX MODIFIED PROBABILISTIC NEURAL NETWORK IN DIGITAL CHANNEL EQUALIZATION

James P. Young, Thomas Hanselmann, Anthony Zaknich and Yianni Attikiouzel
Center for Intelligent Information Processing Systems (CIIPS)
Department of Electrical and Electronic Engineering
The University of Western Australia, Australia
Young-j@ee.uwa.edu.au

A novel adaptive technique is proposed for the complex-valued Modified Probabilistic Neural Network (MPNN). The adaptive feature is desirable when using the MPNN in channel equalization to track time-varying channels. The MPNN is initially trained using the clustering technique. When training is completed, the network is switched to decision directed mode and the network parameters are adapted using stochastic gradient-based algorithms in an unsupervised manner. Simulations show that the equalizer was able to efficiently equalize 4-QAM symbol sequences transmitted through non-linear, slowly time-varying channels.

1 Introduction

The Modified Probabilistic Neural Network (MPNN), unlike the Multi-layer Perceptron (MLP), the Radial Basis Function Network (RBFN) or many other commonly used neural networks, is a memory-based network. Training does not involve adjusting its weights through a minimum error cost function at each iteration. Rather, the input-output relationship of the training data pairs is "remembered" and generalized through creating and clustering of centers and assigning appropriate weight for each center. When training is complete, given an evaluation input vector, the network structure generates an approximation to the maximal likely output in a Bayesian sense. The input-output mapping is non-linear. Therefore it can be used like many other neural networks in channel equalization problems for dealing with non-linear channels. For the application of digital channel equalization, the MPNN needs to be extended to operate with complex signals, as well as to be able to adapt to slowly time-varying channels.

The Multilayer Perceptron (MLP) and the Radial Basis Function Network (RBFN) have attracted attention as feasible solutions to equalization of non-linear channels. The MLP has been demonstrated to outperform most conventional receivers in terms of error probability and mean-square error [1], however it suffers from long training time and undesirable local minima. The RBFN on the other hand can be constructed to have an equivalent structure to an optimal Bayesian solution symbol-decision equalizer for binary signals [2]. It also has faster training and has better performance in terms of bit error rate (BER), because the cost function can be designed to minimize the total bit error rather than the mean square error [2]. The MPNN being structurally very similar to the RBFN, has most of the advantages that an RBFN equalizer has to offer. On top of that, for varying channels, the MPNN gives a more stable performance. This is demonstrated in the simulation.

2. Complex-valued MPNN

The MPNN was developed by Zaknich et al [3,4], inspired by Specht's work on Probabilistic Neural Network (PNN) [5]. Its basic structure (see Figure 2) directly simulates the Bayesian estimation theory. Network reduction is achieved through a k-means like clustering method, by grouping together centers that are close to each other. In channel equalization, clustering reduces the effect of noise. Modification to enable the network to process complex signals is simple. The only requirement is to replace the vector transpose operation with the Hermitian operation in the kernel function. This is detailed in the following section.

2.1. The Channel Equalization Problem

Consider an i.i.d. M-ary symbol sequence $\{\mathbf{S}_k\}$ is being transmitted through a dispersive channel $\mathbf{h}$ (see Figure 1). When the channel transfer function is linear, the output of the channel is the convolution of the input sequence $\mathbf{S}_k$ with $\mathbf{h}$ ($\mathbf{u} = \mathbf{S}_k * \mathbf{h}$). For non-linear channels, the output of the channel is $\mathbf{u} = \mathbf{h}(\mathbf{t}, \mathbf{S}_k)$. The output is further cor-

rupted by zero-mean Gaussian noise **v**. The output of the channel is **w** = **u** + **v**. The last m sequences from **w** are taken as the input to the MPNN equalizer to restore the original signal.

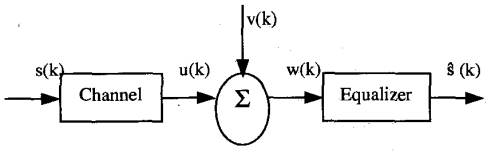

Figure 1: Discrete time model of data transmission system

2.2. The Complex-valued MPNN Equalizer

The training of the MPNN requires only a one-pass training. During training, the network stores the training input vectors as centers of the network. These centers are assigned the values of the desired output.

During classification, the "closeness" of a test input vector to each of the centers is determined via the kernel function. The output associated with the center closest to the input vector hence is the most likely output. It is said, therefore, that the network approximates an optimal Bayesian solution.

In equalization, the input, **x**, is taken as the last m sequence of the channel output.

$$\mathbf{x}_i = [w_i \quad w_{i-1} \quad \cdots \quad w_{i-m}]^T \qquad (1)$$

Suppose y is the output to be estimated. Using a statistical model, vector **x** can be treated as random variable with a conditional probability of **x** given y. During training the *a priori* distribution of y is specified based on what is known about y. It is then possible to obtain the conditional distribution of y given **x**. During classification phase, current data updates are used to yield posterior information about y.

Specht [6] has given the general regression equation of the scalar output y given an input vector random variable **x** as expressed by (2).

$$y(\mathbf{x}) = E[y \mid \mathbf{x}] = \frac{\int_{-\infty}^{\infty} yp(\mathbf{x}, y)dy}{\int_{-\infty}^{\infty} p(\mathbf{x}, y)dy} \qquad (2)$$

Where $p(\mathbf{x}, y)$ is the joint continuous probability density function. This is estimated *a posteriori* from the training set ($\mathbf{x}_n$, y_n).

The equation is derived via a semiparametric technique using the Parzen-Rosenblatt density estimator [7]. The key to this formulation is the kernel function, which has the underlying properties associated with a probability density function. The final estimator equation is given in (3), which includes a clustering variable Z_i. Without center clustering, the equation is also known as the Nadaraya-Watson Regression Estimator [7]. The kernel function shown in (4), has been formulated to accommodate for complex input vectors and complex outputs.

With the m-dimensional complex vector input **x** and complex output y implementing a mapping $\{\mathbf{f} : C^N \rightarrow C\}$, the transfer function is

$$\hat{y}(\mathbf{x}) = \frac{\sum_{i=1}^{M} Z_i y_i \Phi_i(\mathbf{x}, \mathbf{c}_i, \sigma)}{\sum_{j=1}^{M} Z_j \Phi_j(\mathbf{x}, \mathbf{c}_j, \sigma)} \qquad (3)$$

where the Kernel function is

$$\Phi_i(\mathbf{x}, \mathbf{c}_i, \sigma) = \exp\left(\frac{-[\mathbf{x}-\mathbf{c}_i]^H [\mathbf{x}-\mathbf{c}_i]}{2\sigma_i^2}\right) \qquad (4)$$

- **x** $\mathbf{x} \in C^N$ is the input vector to the network with complex members
- $\mathbf{c}_i$ $\mathbf{c}_i \in C^N$ is the center or mean vector for class i in the input space
- σ_i sigma is a real valued smoothing parameter
- y_i $y_i \in C$ is the weight / output relating to center c_i
- M is the total number of unique centers
- Z_i is the number of input training vector associated with center $\mathbf{c}_i$
- $[.]^H$ denotes Hermitian operator (or conjugate transpose)

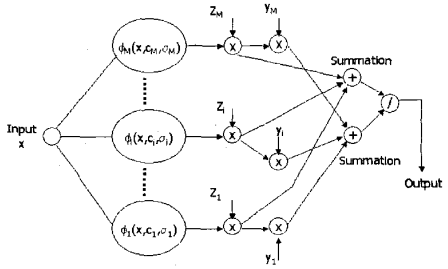

Figure 2: Basic MPNN Architecture

The output of each of the kernel functions is real. The complex MPNN equation is the same as the real MPNN equation in [3,4], except that the Hermitian operator is used inside the kernel function instead of a vector transpose operator. The network output is the sum of the product from

each center normalized by the sum of the output of the basis function scaled by Z_i.

2.3. Unsupervised Clustering of Centers

Learning involves the clustering of centers $\mathbf{c}_i$ and the assignment of an appropriate weight y_i and smoothing parameter σ_i to each center.

The clustering method is similar to the k-means clustering. Close centers that are mapped to the same output are clustered together to form one new center. The location of the new center is the mean of all the centers being clustered. The total number of centers belonging to this new center is Z_i. As a result, a smaller network size can be realized

Define a parameter, Rx, called the radius of influence. Training starts with the first training pair $\{\mathbf{x}_i, y_i\}$ and establishing a center $\mathbf{c}_i$ at $\mathbf{x}_i$ with a corresponding weight of y_i. Given n training data pairs, the pseudo-code for the clustering method is as follows:

for i = 1: n,
  if
    $([\mathbf{x}_i - \mathbf{c}_i]^H[\mathbf{x}_i - \mathbf{c}_i] \le R_x)$ and $(y_i(k) = y(k))$ (5)
  then
    $Z_i^{new} = Z_i^{old} + 1$
    $\mathbf{c}_i^{new} = \frac{(Z_i - 1)\mathbf{c}_i^{old} + \mathbf{x}_i}{Z_i}$ (6)
  else
    $Z_i^{new} = 1$
    $c_i = x_i$
  end
end

When all training data have been passed, training is completed.

3. Complex Stochastic Gradient Adaptation

This section of the paper deals with adapting the MPNN equalizer to slowly time varying channels.

The equalizer is in decision directed (DD) mode. The output of the equalizer is compared to its closest signal constellation. The difference is the error. The parameters of the equalizer are adjusted to this error term.

For convenience let us rewrite the estimation equation and denote the numerator term as 'b' and the denominator term as 'a'.

$$\hat{y}(\mathbf{x}) = \frac{\sum_{i=1}^{M} Z_i y_i \Phi_i(\mathbf{x}, \mathbf{c}_i, \sigma)}{\sum_{j=1}^{M} Z_j \Phi_j(\mathbf{x}, \mathbf{c}_j, \sigma)} = \frac{b}{a} \quad (7)$$

Real and imaginary term of a variable are denoted by the subscript R and I respectively.

The error term E(k) is complex.

$$E(k) = dec(k) - y(k) = e_R(k) + je_I(k) \quad (8)$$

The objective J is to minimize the square of both the real and the imaginary error components E(k).

$$J = 1/2\left[e_R^2(k) + e_I^2(k)\right] \quad (9)$$

J is a non-constant real-valued function. Its complex derivative is not defined. We can however calculate its gradient with respect to the network's free parameters (y_i, $\mathbf{c}_i$, σ_i^2) by using the following equations.

$$\nabla_{y_i} J(k) = \frac{\partial J[k]}{\partial y_{R,i}} + j\frac{\partial J[k]}{\partial y_{I,i}} \quad (10)$$

$$\nabla_{c_i} J(k) = \frac{\partial J[k]}{\partial c_{R,i}} + j\frac{\partial J[k]}{\partial c_{I,i}} \quad (11)$$

$$\nabla_{\sigma^2_i} J(k) = \frac{\partial J[k]}{\partial \sigma^2_i} \quad (12)$$

Evaluation gives the update equations (13)– (21) as follows:

$$y_{i,k+1} = y_{i,k} + \mu_y e_k \Phi_i(\mathbf{x}, \mathbf{c}_i, \sigma_i^2) \quad (13)$$

$$\sigma^2_{i,k+1} = \sigma^2_{i,k} + \mu_{\sigma^2}\left(re(e_k)\varsigma_R + im(e_k)\varsigma_I\right) \quad (14)$$

$$c_{i,k+1} = c_{i,k} + \mu_c \frac{Z_i \Phi_i(\mathbf{x}, \mathbf{c}_i, \sigma_i^2)}{a^2\sigma_i^2} \bullet \left[re(e_k)\lambda_{i,RR} + im(e_k)\lambda_{i,IR} + j*(re(e_k)\lambda_{i,RI} + im(e_k)\lambda_{i,II})\right] \quad (15)$$

Where

$$\varsigma_R = \frac{Z_i \Phi_i(\mathbf{x}, \mathbf{c}_i, \sigma_i^2)[\mathbf{x} - \mathbf{c}_i]^H[\mathbf{x} - \mathbf{c}_i]}{2a^2(\sigma^2_{i,k})^2}[a*re(y_i) - re(b)] \quad (16)$$

$$\varsigma_I = \frac{Z_i \Phi_i(\mathbf{x}, \mathbf{c}_i, \sigma_i^2)[\mathbf{x} - \mathbf{c}_i]^H[\mathbf{x} - \mathbf{c}_i]}{2a^2(\sigma^2_{i,k})^2}[a*im(y_i) - im(b)] \quad (17)$$

$$\lambda_{i,RR} = [\mathbf{x}_R - \mathbf{c}_{i,R}][ay_{i,R} - b_R] \quad (18)$$

$$\lambda_{i,RI} = [\mathbf{x}_I - \mathbf{c}_{i,I}][ay_{i,R} - b_R] \quad (19)$$

$$\lambda_{i,IR} = [\mathbf{x}_R - \mathbf{c}_{i,R}][ay_{i,I} - b_I] \quad (20)$$

$$\lambda_{i,II} = [\mathbf{x}_I - \mathbf{c}_{i,I}][ay_{i,I} - b_I] \quad (21)$$

$\mu_y, \mu_{\sigma^2}, \mu_c$ are learning rates for weights, smoothing parameters and centers respectively

re(.) denotes real operator

im(.) denotes imaginary operator

4. Simulation Results

Without loss of generality, the source signal, s(k), were 4-QAM symbols. The constellation of s(k) were { s_1=1+j ; s_2=1-j ; s_3=-1+j ; s_4=-1-j }. The slowly time varying channel had a transfer function:

$$H(z) = (1 - j0.03434) + (0.5 + 0.015t + j0.2912 + j0.007t)z^{-1} \quad (22)$$

where t = k / baud rate

The non-linearity was modeled by a third order Volterra non-linearity and the noise was additive Gaussian white noise

$$w(k) = u(k) + 0.2u^2(k) + 0.1u^3(k) + v(k) \quad (23)$$

A plot of the received symbols is shown in Figure 3a, taking into account the non-linearity, the noise and the time varying component. Due to the time varying component in the transfer function, the location of the received signals moved from starting positions indicated by 'o' to positions indicated by 'x' (Figure 3b).

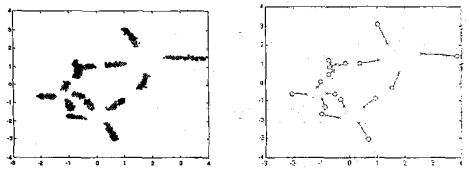

Figure 3a, 3b: Non-linear channel with time-varying transfer functions (a) actual received signals with noise variance σ^2 = 0.03 (b) locations of received signals drifted from starting position marked by 'o' and end position marked by 'x'

The dimension of the input of the equalizer, m, is chosen to be 2 and the noise variance to be 0.03. 300 samples of random training data were found to be adequate to train the network.

The channel characteristic was continuously changing. New errors were introduced at any time induced by the slowly drifting channel states. Therefore, the instantaneous error can best represent the effectiveness of the equalizer. The error was measured as the sum of squared distance of both real and imaginary parts from the original symbol.

The performances of a complex RBFN and a complex MPNN were compared without adaptation (Figure 4). The methods of a complex RBFN are detailed in [8].

The symbols in Figure 4 and Figure 5 were given sequentially with time. The initial symbols were the results of received signals at the starting positions. Consequent symbols were the result of received signals having drifted.

Initially, the parameters of the RBFN had been tuned to give an almost perfect equalization (without any miss-classification). This can be observed in Figure 4a. For the first 100 symbols or so the error rate was low. The MPNN trained only with the clustering technique also gave an almost perfect equalization.

For the RBFN, as the location of the received signal drifted from its original position, the input of the equalizer was no longer close to any centers. The overall output of the radial basis functions quickly converged to zero and produced errors of 2 (Figure 4a). The MPNN on the other hand, measures the relative distance of a drifted input to all centers, therefore it remains a good Bayesian estimator despite the actual drift of channel states from the centers. It is shown in Figure 4b that MPNN retains its ability to equalizer the channel. Please note that the vertical scales in Figure 4a and 4b are different.

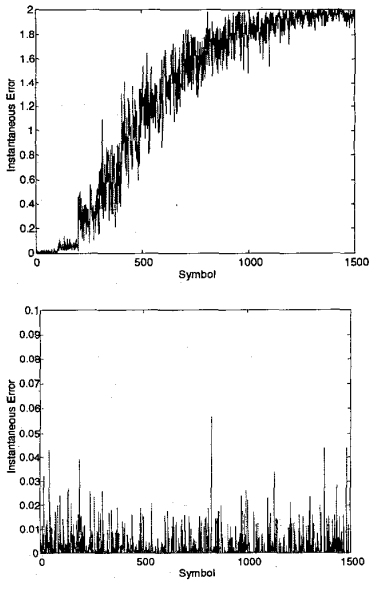

Figure 4a, 4b: Comparison between RBFN and MPNN equalizing drifting channel states (a) RBFN (b) MPNN

For a more severe case where the noise variance σ^2 was 0.05 the MPNN could no longer equalize the channel without adjusting for the changing channel. The equalizer was switched to DD mode for adaptation. The results with and without adaptation were compared and are shown in Figure 5. All three parameters were updated with learning rate $\{\mu_{yi} = 0.05; \mu_{\sigma i}2 = 0.001; \mu_{ci} = 0.05\}$. There was a clear converging behavior to perfect equalization even though new errors were supposedly being introduced continuously.

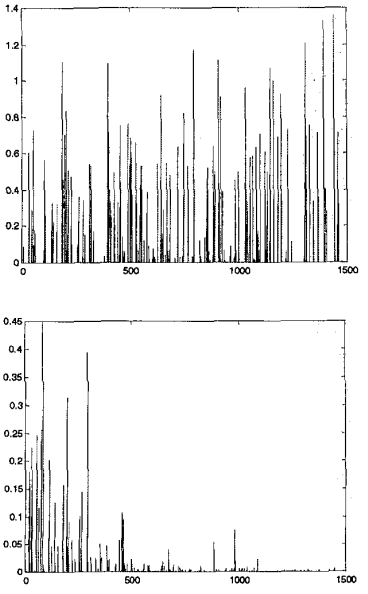

Figure 5a, 5b: Application of complex MPNN adaptation algorithms on non-linear drifting channel states (a) without adaptation (b) adapting network

5. Conclusion

In the simulation, MPNN was tested on a low-order, non-linear and slowly time-varying channel and has exhibited good performance. For higher order channels with greater signal alphabet size, the system can suffer from the curse of dimensionality requiring a great number of centers. A center reduction technique for the MPNN is proposed in [9] to tackle this problem.

Good tracking behavior of the MPNN equalizer was demonstrated for slowly drifting channel states. However, as with all adaptive systems, the parameter learning rates must be chosen so that they are small enough for the system to converge and are large enough so that the rate of adaptation is greater than the rate change.

6. References

[1] G.Gibson, S.Siu and C.Cowan, "Multilayer Perceptron Structures Applied to Adaptive Equalisers for Data Communications", in *Proc. of ICASSP'89*, pp. 1183-1186, Glasgow, Scotland, May 1989.

[2] S.Chen, B.Mulgrew and P.M.Grant, "A Clustering Technique for Digital Communications Channel Equalization Using Radial Basis Function Networks", *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 570-579, 1993.

[3] A.Zaknich, "Introduction to the Modified Probabilistic Neural Network for General Signal Processing Applications", *IEEE Trans. Signal Processing*, vol. 46., no. 7, pp. 1980-1990, July 1998.

[4] A.Zaknich, C.deSilva and Y.Attikiouzel, "The Probabilistic Neural Network for Non-linear Times Series Analysis", in Proc. *IEEE Int. Joint Conf. Nerual Networks* (IJCNN), Singapore, Nov. 1991.

[5] D.F.Specht, "Probabilistic Neural Networks", Int. Neural Network Soc., *Neural Networks*, Vol.3, pp.109-118, 1990.

[6] D.F.Specht, "A General Regression Neural Network", *IEEE Trans. Neural Networks*, Vol.2, Nov. 1991.

[7] S.Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd Ed, Prentice-Hall, 1999.

[8] S.Chen, P.M.Grant,S.McLaughlin and B.Mulgrew, "Complex-valued Radial Basis Function Networks", *IEEE Third International Conference on Artificial Neural Networks*, pp. 148-152, 1993.

[9] J.Young, A.Zaknich and Y.Attikiouzel, "Center Reduction Algorithm for the Modified Probabilistic Neural Network Equalizer", to appear in *IJCNN 2001*.