NEURAL EQUALIZER WITH ADAPTIVE MULTIDIMENSIONAL SPLINE ACTIVATION FUNCTIONS

Mirko Solazzi(*), Aurelio Uncini (**), Francesco Piazza (*)

(*) Dipartimento di Elettronica e Automatica - University of Ancona
Via Brecce Bianche, 60131 Ancona-Italy.
Fax +39 (071) 2204464 email: mrksol@eelab.unian.it - Internet: http://hmsp.eelab.unian.it/

(**) Dipartimento INFOCOM - University of Rome “La Sapienza”
Via Eudossiana 18, 00184 Rome - Italy.
Fax +39 (06) 4873300 email: aurel@infocom.ing.uniroma1.it

ABSTRACT
This paper presents a new neural architecture suitable for digital signal processing application. The architecture, based on adaptable multidimensional activation functions, allows to collect information from the previous network layer in aggregate form. In other words the number of network connections (structural complexity) can be very low respect to the problem complexity. This fact, as experimentally demonstrated in the paper, improve the network generalization capabilities and speed up the convergence of the learning process. A specific learning algorithm is derived and experimental results, on channel equalization, demonstrate the effectiveness of the proposed architecture.

1. INTRODUCTION
Reduction of effects of intersymbol interference (ISI), nonlinear channel distortions and interference in general are a recurrent issue in digital communications [1]. Different channel equalization techniques have been developed to deal with this problem. Adaptive channel equalization is practically required when channel perturbations are non-stationary. During the last decade, neural networks have been frequently applied for solving the equalization task [2][3], pursuing several addresses. Typically a neural equalizer attempts to invert the channel response to recover the original signal sequence before the final decision is made [4][6]. In these cases, neural networks act as non-linear maps between received samples and transmitted symbols. The overall network function is generally multivariate, in other words, the neural network receives as inputs more than one signal sample and returns an estimate of the transmitted symbol. The classical feedforward architectures with sigmoidal activation functions is demonstrated able to approximate any continuous function of several variables. Anyway, Vitushkin in [5] proved that not all the functions with a given degree of complexity can be represented in simple way by means of functions with a lower degree of complexity. Recently a new interest in adaptive activation functions has arisen. In [7] the authors have involved polynomial functions, which allow to increase the network complexity and reduce the size of the network. This solution implies some drawbacks, principally with the adaptation of the coefficients in the learning phase, due to spurious minima and maxima. In addition, polynomial functions are non-bounded and the resulting approximation is generally poorly smooth. Later adaptive spline activation function was introduced [8]. With this approach each neuron is characterized by a different activation function whose shape can be modified through some control points. Further works [9][10] demonstrated that such neuron architecture can improve approximation and generalization capabilities of the entire network. In this paper a more general activation function is introduced; in particular we first define the multidimensional neuron and the related network architecture. Next we implement its activation function by using cubic spline interpolation of some control points. Usually the introduction of a much number of adaptive coefficients leads to a poor generalization capability. In practical situation, when few control points for each neuron are locally updated to obtain a good approximation and, at the same time, the number of hidden units and connection weights are drastically reduced. Experimental trials on different non-linear channel models and symbol alphabets confirmed the validity of this architecture, from the points of view of both the average symbol error and the convergence speed.

2. M-DIM SPLINE ACTIVATION FUNCTIONS
2.1 Multidimensional Activation Functions
A feed-forward neural network consists of a given number of units called neurons, which are connected by weighted links. These units are arranged in multiple layers, namely an input layer, one or more hidden layers and an output layer; in total $S$ layers. The whole network function that maps the input vector onto the output vector is established by the network connection weights. Traditionally each neuron $j$ in the layer $l$ of the network is a simple processing unit that computes activation $y^{(l)}_j$ as a function of its incoming excitation, the so-called net input $net^{(l)}_j$. The output $y^{(l)}_j$ is computed by passing the net input through a non-linear activation function $\Phi^{(l)}(x)$ that is generally squashing and sigmoidal. We can think a generalization of this structure, introducing a more general activation function. In particular multi dimensional functions are now considered. Let $\Phi^{(l)}(x_1, x_2, ..., x_M)$ a generic $M$-variables function, following the notation of Figure 1, we define multiple net input as:

$$net_{k,l} = \sum_{i=0}^{N-1} w_{k,i}y_{i,l} = w_{k,l}^{(l)}$$

where $w_{k,i}$ is the connection weight from unit $i$ in the upper layer to unit $j$ for the $k$-th input $(k=1, ..., M)$ of function $\Phi^{(l)}(.)$ and $N^{(l)}$ represents the number of neurons in the layer $l$. Values $y_0^{(l)} = 1$, $\forall l:1 \leq l \leq S-1$ are the bias inputs. The activation $y_0^{(l)}$ of neuron $j$ in the layer $l$, is computed as:

$$y_0^{(l)} = \Phi^{(l)}(net_{k,1}^{(l)}, net_{k,2}^{(l)}, ..., net_{k,N^{(l)}})$$

are coefficients depending on control points of the function of a specific neuron in the same layer.

\[ \Phi^{(j)}(\mathbf{w}^{(j)}_1 y^{(1)}, \mathbf{w}^{(j)}_2 y^{(2)}, \ldots, \mathbf{w}^{(j)}_M y^{(M)}) \] 

(2)

in other words, the activation function’s inputs are collected as \( M \)-dimensional subset of the linear combiner outputs \( \mathbf{w}^{(j)}_i \). We can use \( M(l) \) in the place of \( M \) in the case of different layer dimension. Similarly, we introduce an additional index \( j \) to indicate the activation function of a specific neuron in the same layer.

\[ \Phi^{(j)}(\mathbf{w}^{(j)}_1 y^{(1)}, \mathbf{w}^{(j)}_2 y^{(2)}, \ldots, \mathbf{w}^{(j)}_M y^{(M)}) \]

Indices that are equally spaced along the 1-axis. Without loss of generality, \( \Phi^{(j)} \) are realized as the weighted average of the four control points \( 11 1 1() ( 1 ) ( 2 ) ( 3 ) \).

\[ \text{Fig. 1. Architecture of a neural network with multi-dimensional neurons.} \]

2.2 Adaptive Spline Activation Functions

Adaptive activation functions allow to increase the neuron complexity and reduce the size of the network. Several studies have been conducted in this direction [7]-[10]. In recent works, definition and implementation of adaptive spline activation functions is made [8]-[10]. With this approach each neuron is characterized by a different activation function whose shape can be modified through some control points.

Our idea consists in realizing multidimensional activation functions as hyper-surface interpolation of some control points using higher order derivatives. The \( M \)-dimensional formulation is developed by first considering 1D, 2D, ..., (M-1)D splines in the special case where the control points lie on a regular grid in the space of \( M \) dimension. The vector localization is performed by considering control points along each direction. These points are equally spaced and symmetrically disposed with respect to the axis origin. Let \( \Delta x \) and \( N_j \) the fixed step between points and the number of points along the \( k \)-th dimension respectively, we have:

\[ z_{k,j} = \frac{\text{net}_{k,j}}{\Delta x} + \frac{N_j - 1}{2} \]

(3)

\[ z_{k,j} = \left\{ \begin{array}{ll}
1 & \text{if } z_{k,j} < 1 \\
\frac{N_j - 3}{N_j} & \text{if } 1 \leq z_{k,j} \leq N_j - 3 \\
N_j - 3 & \text{if } z_{k,j} > N_j - 3
\end{array} \right. \] 

(4)

Constraints imposed by equation (10) are necessary to keep the net input within the active volume that encloses the control points. Next step is to separate indices \( z_{k,j} \) into integer and fractional part using the floor operator \( \lfloor \cdot \rfloor \)
The integer parts indicated as $n_{i,j}$ are used to address local control points of neuron $j$, while the fractional parts $u_{i,j}$ are passed as normalized inputs to multidimensional cubic spline function $h_{i,j}$.

After training with $N_e$ examples, activation function for $j$-th neuron can be expressed as:

$$
\Phi_j(\mathbf{w}, \mathbf{x}, \ldots, \mathbf{w}_{M_i}, \mathbf{x}) = \Phi_j(\text{net}_{i,1}, \ldots, \text{net}_{i,M_i}) = \\
\sum_{k=0}^{M_j} h_{i,j}(u_{i,j}, u_{i,j+1}, \ldots, u_{i,M_j})
$$

The resultant activation function for each neuron takes into account all modification locally performed during the learning phase. In other words, we can think it as the sum of locally adapted sub-surfaces.

$$
n_{i,j} = \lfloor z_{i,j} \rfloor \quad (11)$$

$$
u_{i,j} = z_{i,j} - n_{i,j} \quad (12)$$

The channel output is also corrupted by additive noise, usually modeled as a Gaussian white process $\{q[n]\}$ with zero mean.

The received sequence $\{s[n]\}$ must be processed to produce an estimate of the transmitted symbol $S[n]$. The equalizer structure is a decision-feedback architecture, represented in Figure 3. A feedforward delay chain of order $m$, and a feedback stage of order $l$ constitute the network input. The filtering process introduces a decision delay $d$. The non-linearity introduced by the neural network helps in contrasting non-linear distortions.

### 3. THE NEURAL EQUALIZER STRUCTURE

In a digital communication system the transmitter sends a sequence $\{S[n]\}$ of symbol waveforms, extracted from a given alphabet at discrete time intervals. Going through the transmission channel, symbols are affected by both linear and nonlinear distortions. The linear part of the channel is commonly modeled by a finite impulse response filter $[1]$:

$$
h[n] = \sum_{k=0}^{N} h_k S[n-k] \quad (14)
$$

where $N$ is the channel impulse response length.

$$
\hat{s}[n] = \sum_{k=0}^{N} h_k \hat{s}[n-k]
$$

A sequence of 2000 symbols was generated to train the equalizer for each SNR value. Once trained, the equalizer was tested on a sequence of 100000 symbols. All tests were repeated for 10 different realizations, where for each realization neural network parameters were reinitialized. Both experiments conducted with 2-PAM and 4-QAM modulations pointed out the better performances with respect to the others neural architectures (Fig.4-5).

### 4. EXPERIMENTAL RESULTS

In our simulations, a multilayer neural network with two layers of connections and 2D neurons was employed. Inputs, outputs and network connection weights were defined as complex values. Activation functions and learning algorithm were reconsidered for the complex domain in accordance with definitions in [11]. The performances of the equalizer (SP2D_2) were evaluated for several channel models and compared with equalizers based on fixed sigmoidal (NN_20) and adaptive mono-dimensional spline (SP1D_6) activation functions (the figures denote the number of hidden units). In every test the initial weight values were randomly fixed, while each adaptive activation function was implemented by mean of a grid of 30x30 control points, initially sampled from a signed two-dimensional sigmoid, with $\Delta x = \Delta y = 0.2$. The activation function of the output neuron was linear. We report here results for two typical channels with transfer functions given by:

$$
H_1(z) = 0.3482 + 0.8704 z^{-1} + 0.3482 z^{-2} \quad (15)
$$

$$
H_2(z) = -0.2052 - 0.2052 z^{-1} - 0.7183 z^{-2} - 0.3695 z^{-3} + 0.2052 z^{-4} \quad (16)
$$

A non-linearity was introduced at the channel output $\hat{r}[n]$ in (14) by the formula:

$$
x[n] = \hat{r}[n] + 0.2 (\hat{r}[n])^2 \quad (17)
$$

A sequence of 2000 symbols was generated to train the equalizer for each SNR value. Once trained, the equalizer was tested on a sequence of 100000 symbols. All tests were repeated for 10 different realizations, where for each realization neural network parameters were reinitialized. Both experiments conducted with 2-PAM and 4-QAM modulations pointed out the better performances with respect to the others neural architectures (Fig.4-5).

### 5. CONCLUSIONS

Neural network architecture, based on adaptive multidimensional activation function has been proposed. Such activation functions were implemented by means of cubic spline interpolation of some control points. Notwithstanding the large number of adaptable parameters the number of adapted parameters for each iteration is fixed (16 points). After the learning phase, as in the mono-dimensional case [9], we can observe that only a little subset of the total adaptable parameters is changed. This fact, in addition to small computational load per iteration, can be interpreted as an intrinsically generalization capabilities of the proposed architecture (see [9] for more details). Experimental trials in equalization performed on different channel models and symbol alphabets confirm the validity of proposed neural architecture, from the points of view of both the average symbol error and the convergence speed.
\[ \delta - \partial \Delta = - \partial \Delta \]

\[ \partial \Delta = \partial \Delta \]

**Figure 4.** Average symbol error rate versus the SNR for the channel \( H_z(z) \): a) 2-PAM; b) 4-QAM signaling.

**Figure 5.** Average symbol error rate versus the SNR for the channel \( H_z(z) \): a) 2-PAM; b) 4-QAM signaling.

## 6. REFERENCES


## 7. APPENDIX

The backpropagation (BP) algorithm is a well-known method to tune the weights such that the network performs a specific mapping of input onto a desired output vector. An extension of the classical BP algorithm for multidimensional spline networks is now derived. Being \( p \) the learning step, the weights in the network are updated along a search direction opposite of the cost function gradient:

\[ w_j^{(p+1)}[p] = w_j^{(p)}[p] + \Delta w_j^{(p)}[p] \]  

(18)

Omitting the time \( p \) for the aim of readability, we have:

\[ \Delta w_j^{(p)} = -\mu \frac{\partial E}{\partial w_j^{(p)}} = \mu \delta_j^{(p)} x_j^{(p-1)} \]  

(19)

\[ \delta_j^{(p)} = e_j^{(p)} \sum_i \delta_i^{(p+1)} \frac{\partial f_i}{\partial x_i} \]  

(20)

\[ e_j^{(p)} = \begin{cases} f_j - y_j^{(p)} & l = S \sum_i \delta_i^{(p+1)} w_j^{(p+1)}(i) l = S - 1, \ldots, 1 \end{cases} \]  

(21)

with \( 0 \leq i \leq N^{(p-1)} \), \( 1 \leq j \leq N^{(p)} \) and \( 1 \leq k \leq M(l) \).

Let’s now apply the multidimensional neuron backpropagation to derive the learning algorithm for spline activation function networks

\[ \delta_j^{(p)} = e_j^{(p)} \sum_{i=1}^M \frac{\partial h_i^{(p)}(u_{i-1},u_M)}{\partial u_i} \Big|_{(i-1),...,u_M=0} \left( \frac{1}{\Delta h_i^{(p)}} \right) \]  

(22)

where \( e_j^{(p)} \) is defined in expression (21).

For example, in the case of two-dimensional spline activation functions we have:

\[ \frac{\partial h_l^{(p)}(u_{i-1},u_M)}{\partial u_i} = \begin{cases} 1, & i = l \sum_{l=1}^M \left( T_i^2 u_i + 1 \right) \end{cases} \]  

where \( T_l = [3 u_l^2, 2 u_l, 1, 0] \).

Control points of spline activation functions \( h_l^{(p)} \) are also updated after each step using a the adaptation rule:

\[ \Delta Q_{l-1}^{(p)}(u_{l-1},u_M) = \mu_l e_j^{(p)} \sum_{i=1}^M \frac{\partial h_i^{(p)}(u_{i-1},u_M)}{\partial u_i} \Big|_{(i-1),...,u_M=0} \left( \frac{1}{\Delta h_i^{(p)}} \right) \]  

(23)

with \( l_s = (0,1,2,3) \). The number of points that we have to adjust is \( 4^{N^{(p)}} \).