Adaptive Spline Neural Networks for Signal Processing Applications

Aurelio Uncini and Francesco Piazza

Dipartimento di Elettronica e Automatica - Università di Ancona Italy
via Brecce Bianche, I-60131 Ancona Italy - e-mail: aurel@eealab.unian.it

Abstract. In this paper, we study the properties of a new kind of real and complex domain artificial neural networks called adaptive spline neural networks (ASNN), which are able to adapt their activation functions by varying the control points of a Catmull-Rom cubic spline. Most of all, we are interested in generalization capability and we can show that this architecture can be seen as a sub-optimal realization of the additive spline based model obtained by the regularization theory. This new kind of neural network can be implemented as a very simple structure being able to improve the generalization capabilities using few training epochs. Due to its low architectural complexity this network can be used to cope with several nonlinear DSP problem at high throughput rate.

1. Introduction

RECENTLY in the neural network community, a new interest in adaptive activation functions has arisen. In fact, such a strategy seems to provide better fitting properties with respect to classical architectures with sigmoidal neurons.

The simplest solution we can imagine consists in involving gain $a$ and slope $b$ of the sigmoid $a(1-e^{-bx})/(1+e^{-bx})$ in the learning process [1]. A different approach is based on the use of polynomial functions [2], which allow to reduce the size of the network and, in particular, the connection complexity; in fact, the digital implementation of the activation function through a LUT (look-up-table) keeps the overall complexity under control. Drawbacks with this solution arise with the non-boundedness of the function (non-squashing) and with the adaptation of the coefficients in the learning phase. In [3] the direct adaptation of the LUT coefficients is proposed: this time the problems are a difficult learning process due to the large number of free parameters and the lack of smoothness of the neuron’s output. These are also the main reasons for the introduction of Hermite polynomials as substitutes for the so called supersmoother in the Projection Pursuit Learning approach of Hwang et al. [4].

The solution discussed in this paper makes use of spline based activation functions whose shape can be modified through some control points. In particular we’ll show how this architecture is related to cubic splines as obtained by the application of regularization theory [5]. In fact, our main goal is to demonstrate that an intelligent use of the activation function can reduce hardware complexity [6-7], while, at the same time, improving generalization ability.

Regularization theory is the basis of many techniques whose goal is a better generalization. Some of them are addressed to the training set; on the other hand, we are interested in deriving an architecture which embodies some regularity characteristics in its own activation function much better than the classical sigmoid can do.

The main advantages of this innovative structure, very useful for nonlinear adaptive signal processing, are: 1) the training sequence may be shorter than that required by the classical MLP; 2) the architecture is general and, unlike others approaches, it does not require a specific design; 3) the low hardware complexity (low overhead with respect to a simple adaptive linear combiner) makes it suitable for high speed data transmissions using a DSP device.

So, after seeing cubic splines in regularization theory and discussing some drawbacks in their use (Section 2), in Section 3 we present the backpropagation-like learning algorithm for complex domain ASNN. In Section 4, we’ll expose the results of some simulations on real and complex signal processing problems.

2. Cubic splines in regularization theory

Given a training set $T_{\mathcal{S}}=\{(x_1, t_1), ..., (x_n, t_n)\}$ (let’s consider for sake of simplicity $x_i \in \mathbb{R}^n$ and $t_i \in \mathbb{R}$), there are infinite possible hypersurfaces able to give a good approximation of its elements, but not all of them can be judged in the same way. Regularization theory offers a way to choose a compromise between data fitting and smoothness, through a regularizing term added to the classical squared error and weighted by a constant $\lambda$; see eqn. (1) where $H(f)$ represents the functional to be minimized. The stabilizer $P$ is a differential operator determining the kind of smoothness and the shape of the approximator, while $\| \|$ is a suitable norm. It is well

---

1 This work is supported in part by the Italian Ministry of University an Technological Research (MURTS).
known that the minimization of (1) leads to (2) where $G$ is the Green’s function corresponding to the operator $P$ and $c_i$ are coefficients determined by solving the $N\times N$ linear system $(G+\lambda I)c=t$, where $G$ is the Green’s matrix, $c$ is the column vector of the coefficients and $t$ is the column vector of the targets $t_i$.

\[ H(f) = \sum_{i=1}^{N} [t_i - f(x_i)]^2 + \lambda \| P f \|_2^2 ; \quad (1) \]

\[ f(x) = \sum_{i=1}^{N} c_i G(x-x_i) ; \quad (2) \]

\[ \| P f \|_2^2 = \int \left[ \frac{d^2 f(x)}{dx^2} \right]^2 dx ; \quad (3) \]

\[ f(x) = \sum_{i=1}^{N} f_i(x) \quad (4) \]

\[ G(x) = \sum_{i=1}^{N} \mu_i G(x_i) = \sum_{i=1}^{N} \mu_i |x_i|^j ; \quad (5) \]

\[ f(x) = \sum_{i=1}^{N} \sum_{j=1}^{N} \mu_i G(x_i - x_j) ; \quad (6) \]

\[ f(x) = \sum_{i=1}^{N} \mu_i G(w_i x - \alpha_i) ; \quad (7) \]

\[ \varphi_j(w,x) = \sum_{i=1}^{N} c_i |w_i x - \alpha_i|^j \quad j=1,...,n \quad (8) \]

\[ f(x) = \sum_{j=1}^{N} \mu_j \varphi_j(w,x) \quad (9) \]

In particular, we are interested in the one-dimensional stabilizer (eqn. (3)) which corresponds to the kernel $G(x)=|x|^3$. For the multidimensional case, in [5] it is shown that we can use the same stabilizer, just decomposing the function $f$ in the sum of $n$ functions, each in charge of one component of the vector $x$ (eqn. (4)).

Then, the overall kernel is shown in (5) where $\mu_i$ and $n$, are constants. The final aspect of the approximating function is (6) where the symbol $x_{ij}$ indicates the $j$-th component of the $i$-th input $x_i$ in $T_N$.

An important extension of the previous function involves a change in the coordinates system for the space $X$, as reported in [5], the choice of a proper “point of view” can be important when representing a multivariate function as the sum of a number of functions equal to the dimension of the input space. Calling $w_{j}, j=1,...,n$, the vectors which determine the axis of the new system and $\alpha_j$ the new centers in such a system, we can write eqn. (7), inverting the order of summation of equation (6). Eqn. (7) is the starting point of our considerations.

Our idea consists in realizing a neuron with a more complex activation function than the sigmoid, able to reproduce the shape of a whole cubic spline along the directions specified by $w_{j}, j=1,...,n$; see eqn. (8). Then, $f(x)$ can be written as in (9). Now $\mu_{ij}$, and the components of $w_j$, for all the indexes $j$, can be found by backpropagation, thus solving the problem of the optimal set of the parameters $\mu_i$ and of the ideal system of coordinates (although we can get trapped in local minima). The open question is about $\varphi_j$. Once again, the exact implementation of equation (8) would require the knowledge of all the coefficients $c_i$, so we choose a different solution, that is using a cubic spline of simpler structure. Its main characteristics are the adaptation of its shape through some control points and a suitable degree of smoothness. Notice that in our implementation a bias parameter $w_0$ similar to the one used in sigmoidal neurons has been introduced and so we’ll deal with a function $\varphi_j(w,x+w_0)$. The last point to discuss is the approximation capability of the function in equation (8): of course, it will behave well on targets which are likely to have an additive structure, but, in general, a number of hidden units equal to the dimension of the input space is not enough to obtain universal approximation of continuous functions on a compact set. However, we can extend the idea of a neuron with a cubic and smooth activation function to architectures involving a larger number of hidden units (or even more than one hidden layer), though they cannot be directly derived from regularization theory.

3. The Adaptive Spline Neural Networks

3.1 Cubic spline based adaptive activation function

In this section we give some notes on such realization of an adaptive activation function. As we have anticipated in section 2.1, the goal is to give a global approximation of the curve drawn by the functions $\varphi_j, j=1,...,n$, using a structure as tractable as possible. In equation (8) we find a spline with $N+1$ tracts: each of them is described by a different combination of the coefficients $c_i$, because of the change in the sign of the kernels at $\alpha_j$. We have chosen to represent the activation functions through the concatenation of even more local spline basis functions, controlled by only four coefficients. As we want to keep the cubic characteristic, we have used a Catmull-Rom cubic spline [8]. Using this type of spline we could exactly reproduce function (8), but, of course, this is not the cheapest solution so we’ll take a different approach. Referring to Figure 1, the $i$-th tract is expressed as

\[
F_{i}(u) = \begin{bmatrix}
F_{i}(0) \\
F'_{i}(0)
\end{bmatrix} = \frac{1}{2} \begin{bmatrix}
u & u & u^2
\end{bmatrix} \begin{bmatrix}
-1 & 3 & -3 & 1 \\
2 & -5 & 4 & -1 \\
-1 & 0 & 1 & 0 \\
0 & 2 & 0 & 0
\end{bmatrix} \begin{bmatrix}
u
\end{bmatrix},
\]

where $u \in [0,1]$ and $Q=(q_1, q_2)$. Such a spline interpolates the points $Q_{i+1}$ ($u=0$) and $Q_{i+2}$ ($u=1$) and has a continuous first derivative, which is useful for the backpropagation-like learning algorithm. The second derivative is not continuous only at the knots. In
general, equation (10) represents a curve: to obtain a function we have ordered the \( x \)-coordinates according to the rule \( q_{x,1} < q_{x,2} < q_{x,3} < \ldots \).

To find the value of the local parameter \( u \), we have to solve the equation \( F_{x,i}(u) = x_{b0} \), where \( x_{b0} \) is the activation of the neuron: this is a third degree equation, whose solution can make the numerical burden of the learning algorithm even heavier.

The easiest alternative consists in setting the control points uniformly spaced along the \( x \)-axis (\( \Delta x \) is the step); this choice allows to reduce the third degree polynomial \( F_{x,i}(u) \) to a first degree polynomial and the equation for \( u \) becomes linear:

\[
F_{x,i}(u) = u \Delta x + q_{x,i+1}.
\] (11)

Now we can calculate the output of the neuron by \( F_{y,i}(u) \). There is another reason not to make the \( x \)-coordinates of the control points adaptive: in fact, as we have pointed out in the introduction, a too large number of free parameters is the main cause for the overfitting of the training samples: so, if we use many tracts in building the activation function and let them move freely, the neural model will fit also the noise. Then the fixed parameter \( \Delta x \) is the key tool for smoothness control.

As we have decided to adapt only the \( y \)-coordinates of the spline knots, they must be initialized them before starting the backpropagation-style learning: to this aim, we take, along the \( x \)-axis, \( P+1 \) uniformly spaced samples from a sigmoid or from another function assuring universal approximation capability that’s why we use sometimes the acronym GS, standing for Generalized Sigmoid. Outside the sampling interval the neuron’s output will be held constant at the values \( q_{x,1} \), for the negative \( x \)-coordinate, and \( q_{x,P+1} \) for the positive \( x \). In the following, for the sake of simplicity, we’ll indicate the \( y \)-coordinates of the control points without the index \( y \).

### 3.2 The Learning Algorithm for Complex ASNN

The advantage of using complex-valued NNs instead of a real-valued NN counterpart fed with a pair of real values is well known [9], [10]. In complex-valued neural networks one of the main problem to deal with, is the complex domain activation function, whose most suitable features have been suggested in [11]. Let \( F(S) \) be the complex activation function with \( S \in \mathbb{C} \) defined as the complex linear combiner output; the main constraints that \( F(S) \) should satisfy are: 1) \( F(S) \) should be non-linear and bounded; 2) in order to derive the backpropagation algorithm the partial derivatives of \( F(S) \) should exist and be bounded; 3) because of the Liouville’s theorem \( F(S) \) should not be an analytic function.

According to the previous properties, one possible choice, proposed in [12-13], consists on the superposition of real and imaginary activation functions \( F(S) = f_{\text{re}}(\text{Re}(S)) + f_{\text{im}}(\text{Im}(S)) \); where the functions \( f_{\text{re}}(*) \) and \( f_{\text{im}}(*) \), can be simple real-valued sigmoids or more sophisticated adaptive functions.

Using a formalism similar to the one introduced in Widrow and Lehr in [14], and following a development similar to [9], [12-13], for the synaptic weights, the learning algorithm is now extended to the spline’s control points.

Considering \( M \) total layers and indicating each of them with the index \( l \), \( l = 1, \ldots, M \), we can find the span \( a_k \) and the local variable \( u_k \) by

\[
\begin{align*}
\zeta_{k \text{,Re}}^{(i)} &= \frac{\text{Re}\left[S_{k \text{,Re}}^{(i)}\right]}{\Delta x} + \frac{N - 2}{2} \\
\zeta_{k \text{,Im}}^{(i)} &= \frac{\text{Im}\left[S_{k \text{,Im}}^{(i)}\right]}{\Delta x} + \frac{N - 2}{2} \\
a_{k \text{,Re}}^{(i)} &= \lfloor \zeta_{k \text{,Re}}^{(i)} \rfloor \\
a_{k \text{,Im}}^{(i)} &= \lfloor \zeta_{k \text{,Im}}^{(i)} \rfloor \\
{a}^{(i)}_{k \text{,Re}} &= \zeta_{k \text{,Re}}^{(i)} - a_{k \text{,Re}}^{(i)} \\
{a}^{(i)}_{k \text{,Im}} &= \zeta_{k \text{,Im}}^{(i)} - a_{k \text{,Im}}^{(i)}
\end{align*}
\] (12)

where the symbol \( \lfloor \cdot \rfloor \) is the floor operator. We find for each neuron \( j \) the local tract \( a_j \) which \( s_j \) belongs to, and the local coordinate \( u_j \). These expressions lead to neuron structure reported in Figure 2.
to obtain a MSE of approximately -40dB in no more
by networks with 5 sigmoidal hidden units. If we want
represents the limit of the best performance reached
backpropagation. The line at -15dB in Figure 3
five hidden neurons give excellent results both in
1000 samples is the test set. Experiments with just
system and take 1000 samples to be used as the
5] and
15[), made of a IIR linear filter followed by an instantaneous non-linearity.

Backpropagation Learning
Algorithm
As in [9], [12-13], for the synaptic weights, the learning algorithm is
now extended to the spline’s
control points.

\[
E^{(i)}_k = \begin{cases} 
D_k - X^{(i)}_k & l = M \\
\sum_{l=1}^{M} A^{(i)}_k W^{(i)}_{k,l} & l = M-1,...,1 
\end{cases}
\]

\[
A^{(i)}_k = \text{Re} \left\{ E^{(i)}_k \left( \frac{d \text{Re} \left\{ F^{(i)}_{k,m} \right\} (u)}{du} \right) \right\} + j \text{Im} \left\{ E^{(i)}_k \left( \frac{d \text{Im} \left\{ F^{(i)}_{k,m} \right\} (u)}{du} \right) \right\} \frac{1}{\Delta \xi},
\]

\[
W^{(i)}_k[p+1] = W^{(i)}_k[p] + 2 \mu_e A^{(i)}_k X^{(i-1)}
\]

with 0 ≤ k ≤ N_i and 0 ≤ j ≤ N_{j+1}.

The adaptation of the control points is ruled by

\[
q^{(i)}_{x(m-1),m} [p+1] = q^{(i)}_{x(m-1),m} [p] + 2 \mu_e \text{Re} \left[ E^{(i)}_k \left( \frac{\partial \text{Re} \left\{ F^{(i)}_{k,m} \right\}}{\partial q^{(i)}_{x(m-1),m}} \right) \right] + 2 \mu_e \text{Re} \left[ E^{(i)}_k \left( \frac{\partial \text{Im} \left\{ F^{(i)}_{k,m} \right\}}{\partial q^{(i)}_{x(m-1),m}} \right) \right] \frac{1}{\Delta \xi},
\]

\[
q^{(i)}_{y(m-1),m} [p+1] = q^{(i)}_{y(m-1),m} [p] + 2 \mu_e \text{Im} \left[ E^{(i)}_k \left( \frac{\partial \text{Re} \left\{ F^{(i)}_{k,m} \right\}}{\partial q^{(i)}_{y(m-1),m}} \right) \right] + 2 \mu_e \text{Im} \left[ E^{(i)}_k \left( \frac{\partial \text{Im} \left\{ F^{(i)}_{k,m} \right\}}{\partial q^{(i)}_{y(m-1),m}} \right) \right] \frac{1}{\Delta \xi},
\]

with the patch index m=0, ..., 3. The adaptation rates are \( \mu_e \) for the connection weights and biases and \( \mu_q \) for
the control points. The control point with index 0, 1, (N-1) and N are fixed.

4. Experimental Results

4.1 Real Case: Back & Tsoi System identification

Now we are going to test the proposed architecture on the identification of a system proposed by Back & Tsoi
[15], made of a IIR linear filter followed by an instantaneous non-linearity.

\[
z[t] = 0.0154u[t] + 0.0462u[t - 1] + 0.0462u[t - 2] + 0.0152u[t - 3] + 1.99z[t - 1] - 1.572z[t - 2] + 0.4583z[t - 3]
\]

\[
y[t] = \sin \left( \frac{\pi}{2} z[t] \right)
\]

where the \( u(t) \) and \( y(t) \) are the input and output respectively. The temporal dynamic is obtained externally feeding into the network \( u[t], u[t-1], ..., u[t-5] \) and \( y[t-1], ..., y[t-5] \). Fixing arbitrary initial conditions for \( u \) and \( y \), we track the behavior of the system and take 1000 samples to be used as the training set; a second group of independently obtained
1000 samples is the test set. Experiments with just five hidden neurons give excellent results both in
terms of training error and of generalization error with a total number of 96 coefficients updated by
backpropagation. The line at -15dB in Figure 3 represents the limit of the best performance reached
by networks with 5 sigmoidal hidden units. If we want to obtain a MSE of approximately -40dB in no more

Figure 2. The complex-valued generalized sigmoid (GS) neuron structure (the structure of the imaginary part is omitted because it is identical to that of the real part).

Networks with 5 sigmoidal neurons

Networks with 5 GS neurons

Figure 3. Training error for the Back & Tsoi system: comparison of architectures with the same number of hidden neurons.
then 15000 epochs, we are forced to use 20 sigmoidal hidden neurons, more or less: this means an architecture involving 261 free parameters all of which must be adapted leading to a very long training time and to a more dangerous situation as far as generalization is concerned; besides the architecture is quite expensive if compared to the solution using 5 spline based neurons.

4.2 Complex domain: non linear QAM channel equalization

The block diagram of the radio link used in our experiments is depicted in Figure 4. The complex input data sequence \( D(k) \), represents the points of the QAM constellation. The function \( g(t) \) represents the modulator filter impulse response: in the simulation we use a square-root of a raised-cosine having a roll-off factor \( \alpha \) equal to 0.5, and the over-sampling factor \( M \) is chosen equal to 3. The \( g(t) \) filter length is equal to 5\( M \) taps.

The model for the HPA, described in [17], is characterized by the expression:

\[
W_{\text{in}} = |u(t)|^2
\]

The input-output HPA (memoryless) response is described by the AM-AM response, represented by the module, and the AM-PM response represented by the phase. This description assumes, for convenience, that the maximum possible HPA input power \( W_{\text{in}} = |u(t)|^2 \) is equal to 1W, and the maximum shift is \( \Phi_0 = \frac{\pi}{6} \), which are typical values [16].

In the simulations the transmission of 16-QAM signals are considered. The maximum input power to the HPA \( W_{\text{in,max}} \) is very close to the saturation point, and fixed equal to -1 dB.

Three complex-valued equalizers have been tested: 1) complex-valued linear combiner (\( A_{15\_1} \)); 2) complex-value standard multilayer neural network with one hidden layer composed by 10 sigmoidal neurons and linear output (\( N_{15\_10\_1} \)); 3) complex-valued ASNN composed by only one complex GS neuron (\( S_{15\_1} \)).

The training set consists of 6144 (2048x3) input samples, \( v(t)+n(t) \) in the scheme of Figure 4, corresponding to 2048 target QAM symbols. Since the neural network output are the QAM complex constellation points, the network performs also the down-sampling conversion. For each epoch, a different realization of white zero-mean Gaussian noise \( n(t) \) is added, to obtain a S/N equal to 20 dB.

During the learning phase, the \( S_{15\_1} \) and \( A_{15\_1} \) networks are trained for 100 epochs, while, in order to reach a suitable convergence, the standard MLP \( N_{15\_10\_1} \) is trained for \( 10^4 \) epochs. Both adaptation rates \( \mu_0 \) and \( \mu_\infty \) for the \( S_{15\_1} \) are chosen to be equal to 0.0001; the same value is used for the MLP \( N_{15\_10\_1} \), while for the adaptive linear combiner \( A_{15\_1} \) \( \mu_\infty \) is equal to 0.0001.

Several tests using different networks initialization weights have been carried out. In order to evaluate in a realistic way the radio link performances, the symbol error probability (Pe) is computed. Figure 5 reports the Pe values vs. the S/N, both expressed in dB. From this figure we can observe that the proposed approach leads to significant improvements not only with respect to the classical linear adaptive filter, but also with respect to the already known sigmoid MLP based equalization technique.

5. Conclusions

This paper has presented some theoretical properties of a new architecture based on adaptive activation functions, realized through Catmull–Rom cubic splines.

The use of cubic splines in single hidden layer structures can be justified in the framework of regularization theory when introducing additive models. We also pointed out that the mechanism of adaptation of the activation functions works only on local tracts, so that if a segment on the x-axis is never involved by any input data, its 4 control points will never be updated. This fact, together with the reduction in the network’s size due to the augmented expressive power of each neuron, is the reason for a rather low number of free parameters with respect to sigmoidal networks (especially for input spaces with high dimensionality). Besides, the local
character of the adaptation is an application of the principle of minimal disturbance, since a new input data
does not modify the whole shape of the neuron’s activation function, but only a well-defined portion.

It is also important to notice that the way the non-linearity tracks the target improves learning speed and
convergence properties, which means that we need fewer learning epochs and that it’s more difficult to end the
training in very bad local minima: in general, networks with GS neurons having a good behavior on test sets
are easier to find, as if the error surface had a more regular aspect.

For what concerns the QAM equalization problem, this advantage is even more evident, since the network
reduces to a single complex neuron. Moreover, the reduced complexity is responsible for the shorter adaptation
phase in terms of training epochs, as experimentally observed. Comparing our technique with classical linear
approaches, we can notice that there is a low implementation overhead with respect to the adaptive linear filter,
but with a significant improvement in the performance, both in terms of MSE and symbol error probability.

References

Networks, Vol.9, No 4, pp. 627-641, June 1996.
1481-1497, 1990.
Activation Functions” In Proceedings of the International Neural Network Society Annual Meeting
Neural Networks in Communication Applications”, Proc. of Intern. Conference on Neural Networks,
ICANN91, Espoo, Finland, June 1991.
Blind Neural Networks”, in Proceedings of IEEE International Conference on Acoustic Speech and
Amplifiers”, IEEE Journ. on Selected Areas in Communications, Vol. SAC-5, No 3, pp 534-456, April
1987.