Approximation of Sigmoid Function and the Derivative for Artificial Neurons

KOLDO BASTERREXTEA
Elektronika eta Telekomunikazio Saila
University of the Basque Country (UPV/EHU)
EUITI Bilbao. La Casilla plaza 3. 48012 Bilbao
SPAIN

JOSÉ MANUEL TARELA
and INÉS DEL CAMPO
Elektrizitate eta Elektronika Saila
University of the Basque Country (UPV/EHU)
SPAIN

Abstract: - A piecewise linear recursive approximation scheme is applied to the computation of the sigmoid function and its derivative in artificial neurons with learning capability. The scheme provides high approximation accuracy with very low memory requirements. The recursive nature of this method allows for the control of the rate accuracy/computation delay just by modifying one parameter with no impact on the occupied area. The error analysis shows an accuracy comparable to or better than other reported piecewise linear approximation schemes. No multiplier is needed for a digital implementation of the sigmoid generator and only one multiplier for its derivative.

Key-Words: - sigmoid function, neural networks, hardware design, piecewise linear approximation, recursive centered interpolation.

1 Introduction

In the hardware implementation of artificial neural networks (ANN), the non-linearity of the activation function is one of the factors that constrains either the occupied area or the computation time of the system. The most popular activation function is the sigmoid, often used with gradient-descendent type learning algorithms. There are different possibilities for evaluating this function, such as a truncated series expansion, look-up tables, or piecewise approximation. Piece Wise Linear (PWL) approximation schemes appear to be a good alternative [1,2] that save silicon area and show short latency times [3]. Moreover, when gradient descendent algorithms are used, they require the existence of the first derivative of the activation function. In consequence, if the learning is going to be carried out on-line, it is also necessary to compute efficiently the first derivative.

The solutions we propose in this paper for an efficient and cost effective implementation of the sigmoid function is based on a PWL approximation scheme that can be applied to the approximation of any non-linear function. The method has been optimized to approximating both the sigmoid function and its first derivative for future ANN hardware implementations. This procedure is well suited for VLSI circuit design, is computationally efficient and allows for a certain control of the rate accuracy/computation delay as it is based on a recursive algorithm.

2 PWL Optimized Approximation of the Sigmoid Function

In this section we will describe the computational scheme for the approximation of a sigmoid function given by

$$f(x) = \frac{1}{1+e^{-x}} \tag{1}$$

We will apply a Centered Linear Approximation (CRI) to the efficient generation of an optimized approximation to the above-defined function. CRI is a recursive computational scheme for the generation of PWL functions with successive vertex smoothing. This method has been described with detail in [4] and is a natural quadratic approximation with lattice structure. The main advantage of the CRI is its simplicity and recursively-improved accuracy. Moreover, CRI generates accurate approximations to non-linear functions from a very simple PWL starting structure and requieres the storage of very few parameters.

The initial structure, which must be generated as the starting PWL approximation, is shown in Fig.1 and described bellow

$$\begin{aligned} y_1(x) &= 0 \\ y_2(x) &= \frac{1}{2}\left(1+\frac{x}{2}\right) \\ y_3(x) &= 1 \end{aligned} \tag{2}$$

where $y_2(x)$ is the tangent to the reference sigmoid function in $x = 0$. Notice that we only need an adder and a shift register in the case of a digital design. This initial structure assures null error at x=0 and the squashing property of the function.

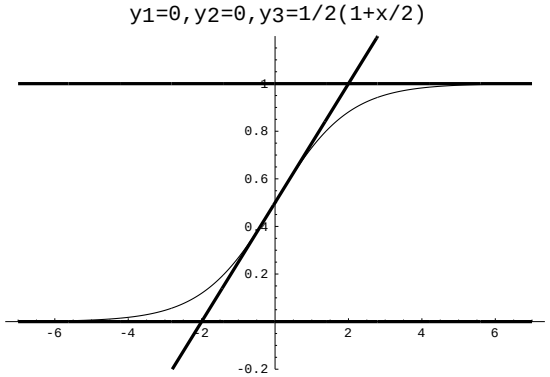

Figure 1: Initial structure for the CRI approximation of the sigmoid function.

Nevertheless, it is known that once computed the sigmoid function for negative inputs (x^-) the computation of the same function for positive inputs (x^+) is straightforward and is given by

$$f(x^+) = 1 - f(x^-) \tag{3}$$

Consequently, we will only consider de negative semi axis. Therefore, the definition of $y_3(x)$ and the computation of the minimum operations are avoided in the computation algorithm, although a sign detector must be implemented for the inputs. The CRI algorithm for negative inputs is as follows:

$$g(x) = y_1(x) = 0\,;\, h(x) = y_2(x) = 1/2\,(1 + x/2);$$
$$\text{for}\,(\mathrm{i} = 0;\mathrm{i} = q;\mathrm{i}++)$$
$$\{g'(x) = \mathrm{Max}[g(x), h(x)];$$
$$h(x) = 1/2\,(g(x) + h(x) + \Delta);$$
$$g(x) = g'(x);$$
$$\Delta = \Delta/4;\}$$
$$g(x) = \mathrm{Max}[g(x), h(x)];$$

where Δ is the *depth parameter*, that must be stored in memory for each q, q is *the interpolation level*, h is the *linear interpolation function*, and $g(x)$ is the obtained approximated function. $g'(x)$ is only needed to write a sequential algorithm, but it would disappear from the computation scheme in a physical circuit implementation as $g(x)$ and $h(x)$ would be computed in parallel.

For an optimal approximation, we must choose the value of Δ that minimizes the error of the approximation for each q ($\Delta_{q,opt}$). In this case we have optimized Δ to achieve the minimum of the maximum error for any input for every interpolation level, that is

$$\min\left\{E^q_{\max}(\mathbf{P})\right\} \tag{4}$$

where $\boldsymbol{P}$=[Δ] is a one-dimensional vector and

$$E^q_{\max}(\mathbf{P}) = Max|\varepsilon| = Max\left| f(x) - g_q(x,\Delta)\right| \tag{5}$$

As the value of the input x the maximum error occurs for is a function of Δ, the error surface $|\varepsilon(x,\Delta)|=|f(x)-g_q(x,\Delta)|$ must be computed, the maximum values for each Δ obtained and finally the minimum value of such maximums identified to select $\Delta_{q,opt}$. The values of $\Delta_{q,opt}$ for each q are:

$\Delta_{1,opt}$ = 0.30895 $\Delta_{2,opt}$ = 0.28094
$\Delta_{3,opt}$ = 0.26588 $\Delta_{4,opt}$ = 0.26380

These values have been obtained by calculating numerically the error surface with an input resolution of 10^{-5} in Δ and 10^{-2} in x.

The generated optimized approximations through CRI, for negative and positive inputs, and the error curves magnified by ten are shown in Fig.2. The initial structure has only 3 segments, comprising $y_1(x)$=0 and $y_3(x)$=1. This quantity increases in each recursion, and can be calculated for each interpolation level as follows:

$$number\ of\ segments\,(ns) = 2^{q+1} + 1 \tag{6}$$

So the number of segments increases to 5, 9, 17 and 33 for q=1, 2, 3 and 4 respectively.

As mentioned above, note that, being $y_2(x)$ the tangent to $f(x)$ in x=0, the error is zero for x = 0 and very small in its surroundings. This fact enhances the training when off-line simulation packages based on gradient descendent algorithms are used [1], as the derivative of $f(x)$ and $g(x)$ are very close.

3 Approximation of the First Derivative

The first derivative of the sigmoid function given in (1) is

$$f'(x) = \frac{e^{-x}}{\left(1 + e^{-x}\right)^2} \tag{7}$$

Online training, if based on gradient descendent methods, demands on-chip computation of the first derivative that can be implemented in various forms.

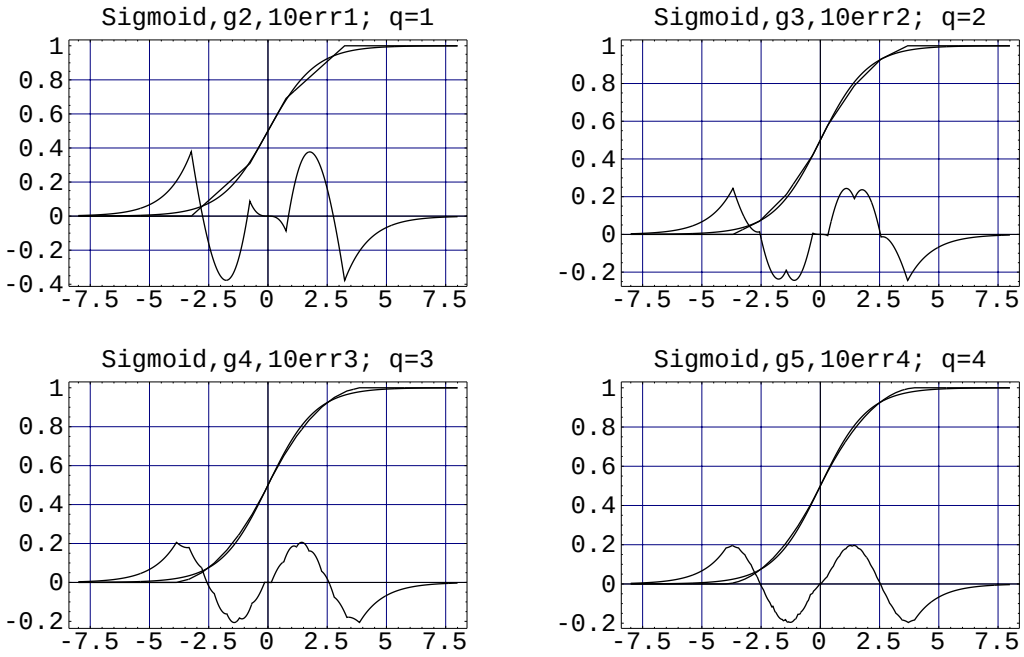

Figure 2: Reference function (sigmoid), approximated function and error curve magnified by ten for the first four interpolation levels. Number of segments: 5 for q=1, 9 for q=2, 17 for q=3 and 33 for q=4.

The first approach consists of a straightforward differentiation of the PWL approximation of the sigmoid that results in a non-continuous step function. This function must be computed efficiently, for example by choosing power of two values for each step in digital designs [3]. Alternatively

$$f'(x) = f(x)(1 - f(x)) \tag{8}$$

can be computed. In our case, we would use g(x), i.e. the CRI approximation of the sigmoid, for the calculation of (8), but the obtained approximations are not very accurate.

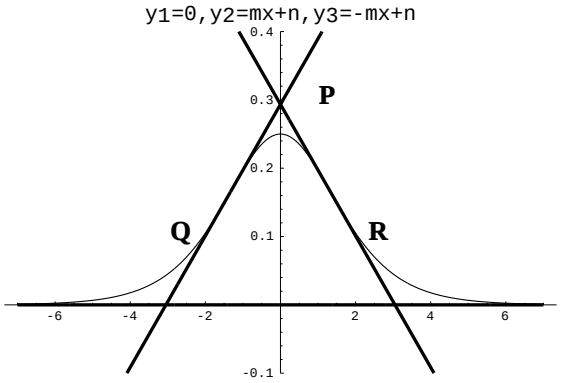

Figure 3: Initial structure for the CRI approximation of the first derivative.

A better approximation can be achieved if CRI is used again to generate a PWL approximation of the derivative function. Now, three straight lines form the initial structure shown in Fig.3:

$$\begin{aligned} y_1(x) &= 0 \\ y_2(x) &= mx + n \\ y_3(x) &= -mx + n \end{aligned} \tag{9}$$

where

$$\begin{aligned} m &= \frac{2}{\left(2-\sqrt{3}\right)\left(1+\frac{1}{2-\sqrt{3}}\right)^2} \\ &\quad - \frac{1}{\left(2-\sqrt{3}\right)\left(1+\frac{1}{2-\sqrt{3}}\right)^3} = 0.096225 \\ n &= \frac{1}{\left(2-\sqrt{3}\right)\left(1+\frac{1}{2-\sqrt{3}}\right)^2} \\ &\quad - mLn\left(2-\sqrt{3}\right) = 0.293391 \end{aligned} \tag{10}$$

so being $y_2(x)$ and $y_3(x)$ the tangents to $f'(x)$ in its points of inflection. The CRI algorithm is very similar to the one given for the approximation of the sigmoid function. Now we need two different

interpolation depths for accurately approximate de reference function $f'(x)$: one for vertex P ($\Delta_{a,q}$) and another one for vertexes Q and R ($\Delta_{b,q}$). The recursive algorithm is as follows:

$h(x) = y_1(x) = 0\,;\, \mathrm{d}g(x) = y_2(x) = mx + n\,;$
$h'(x) = y_3(x) = -mx + n;$
$\text{for (i = 0;i = q;i++)}$
$\{\mathrm{d}g'(x) = \mathrm{Min}[g(x), h'(x)];$
$h'(x) = 1/2\,(dg(x) + h'(x) - \Delta_a);$
$\mathrm{d}g(x) = \mathrm{Max}[dg'(x), h(x)];$
$h(x) = 1/2\,(dg'(x) + h(x) + \Delta_b);$
$\Delta_a = \Delta_a/4; \Delta_b = \Delta_b/4;\}$
$dg'(x) = \mathrm{Min}[dg(x), h'(x)];$
$dg(x) = \mathrm{Max}[dg'(x), h(x)];$

where $dg(x)$ is the approximated first derivative of the sigmoid function.

The optimization problem has been solved as stated before in (4) and (5), i.e. by searching for the minimum value of the maximum error at any input x. In particular, the optimization problem for $\Delta_{a,q}$ has been solved with the constraint of $\varepsilon(x=0) = 0$. The solution is straightforward as CRI assures that the height of the function is set in the first interpolation level:

$$\Delta_a = 2\left(n - \frac{1}{4}\right) \approx 0.0868 \quad \text{for any } q \qquad (11)$$

From the resolution of (5) we obtain for $\Delta_{b,q}$:

$\Delta_{b,1} = 0.11266$ $\quad$ $\Delta_{b,2} = 0.10150$
$\Delta_{b,3} = 0.09634$ $\quad$ $\Delta_{b,4} = 0.09547$

The approximations obtained for the first four interpolation levels are depicted in Fig.4, where error curves have been magnified by ten.

The approximation error for each q has been evaluated for 10^6 input data uniformly spaced in the domain [−8,8] as made in [5]. The numerical values are summed up in Table 1 and Table 2 for the sigmoid and the first derivative respectively. These values of the maximum and average errors are comparable in order of magnitude to those reported in [1, 2, 5] for specific digital designs of PWL sigmoid function approximating circuits.

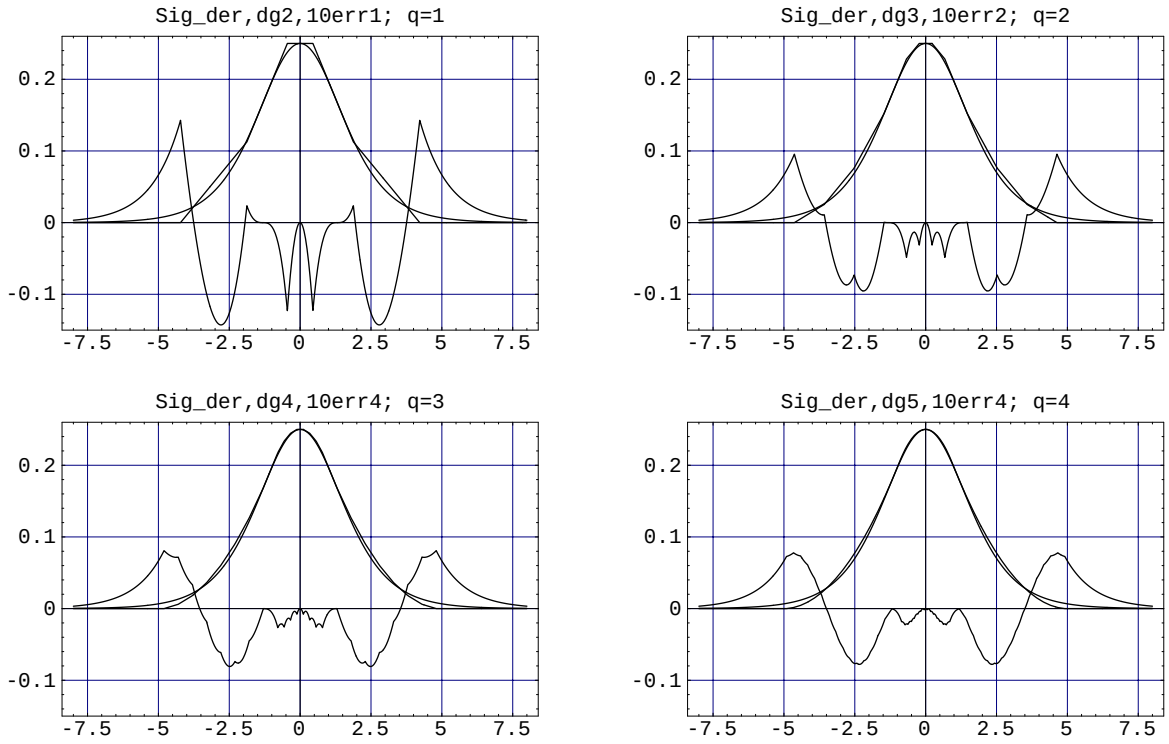

Figure 4: Reference function (first derivative), CRI approximation ($dg(x)$) and error curve magnified by ten for the first four recursion levels. Number of segments: 7 for q=1, 15 for q=2, 27 for q=3 and 41 for q=4.

Interpolation level	Optimum depth parameter	Average error	Maximum error	No. segments
q = 1	$\Delta_{1,opt} = 0.30895$	1.20×10^{-2}	3.78×10^{-2}	3 (5)
q = 2	$\Delta_{2,opt} = 0.28094$	9.21×10^{-3}	2.45×10^{-2}	7 (9)
q = 3	$\Delta_{3,opt} = 0.26588$	8.48×10^{-3}	2.06×10^{-2}	15 (17)
q = 4	$\Delta_{3,opt} = 0.26380$	8.41×10^{-3}	1.97×10^{-2}	31 (33)

Table 1: Sigmoid approximation. Optimization of the maximum error. In brackets the number of segments excluding the constant functions y=1 and y=0.

Interpolation level	Optimum depth parameter	Average error	Maximum error	No. segments
q = 1	$\Delta_{b,1} = 0.11266$	4.91×10^{-3}	1.43×10^{-2}	5 (7)
q = 2	$\Delta_{b,2} = 0.10150$	3.67×10^{-3}	9.56×10^{-3}	13 (15)
q = 3	$\Delta_{b,3} = 0.09634$	3.40×10^{-3}	8.09×10^{-3}	25 (27)
q = 4	$\Delta_{b,4} = 0.09547$	3.31×10^{-3}	7.79×10^{-3}	39 (41)

Table 2: Approximation of the first derivative. Optimization of the maximum error. In brackets the number of segments excluding de constant functions y=1 and y=0

4 Concluding Remarks

We have presented a new scheme for the generation of a sigmoid function and its first derivative in order to achieve hardware design of self contained neurons with on-line learning capability. This scheme is recursive and provides enhancing accuracy and function smoothing for each epoch. For digital designs, the proposed method requires no multiplication for the sigmoid generation. Another mayor advantage of the method lies in the fact that only one number, the optimized value of Δ, must be stored in memory for each value of q, while the number of segments increases by powers of two as shown in (6). The approximation of the first derivative is also possible through the same method. In this case, only one multiplication is required for the generation of each tangent to provide the initial PWL structure.

The calculated error both for the sigmoid and the first derivative approximation depends of the parameter optimization priorities, i.e. maximum or average error optimization, and the interpolation level. We have chosen to assure a null error in x=0 and to optimize the maximum deviation for any input. The obtained errors are comparable to the better error ranges reported for specific sigmoid function PWL approximation schemes. Although general properties of this scheme are promising, the occupied area and the computation times can not be estimated until a specific technology is selected for the physical circuit implementation.

Acknowledgments: This work has been partially supported by UPV (Project 224.310-EA105/99) and DGES (Project TIC99-0357).

References:

[1] D.J. Myers and R.A. Hutchinson, Efficient implementation of piecewise linear activation function for digital VLSI neural networks, *Electronic Letters*, 25(24), 1989, 1662-1663.

[2] C. Alippi and G. Storti-Gajani, Simple approximation of sigmoidal functions: realistic design of digital neural networks capable of learning, *in* "Proc. IEEE International Symposium on Circuits and Systems", 1991, 1505-1508.

[3] P. Murtagh and A.C. Tsoi, Implementation issues of sigmoid function and its derivative for VLSI digital neural networks, *IEE Proceedings-E*, 139(3), 1992, 207-214.

[4] K. Basterretxea, E. Alonso, J. M. Tarela, I. del Campo, PWL Approximation of Non-linear, Functions for the Implementation of Neuro-Fuzzy Systems, Proceedings of the IMACS/IEEE CSCC'99, Athens 1999.

[5] M. Zhang, S. Vassiliadis and J.G. Delgado-Frias, Sigmoid Generators for Neural Computing Using Piecewise Approximations, *IEEE Transactions on Computers*, 45(9), 1996, 1045-1049.