IFAC
Publications

OPTIMAL IMPLEMENTATION OF NEURAL ACTIVATION FUNCTIONS IN PROGRAMMABLE LOGIC USING FUZZY LOGIC

E. Soria-Olivas, A. Rosado-Muñoz, L. Gómez-Chova, R. Magdalena-Benedito, J. Muñoz-Marí

G.P.D.S, Digital Signal Processing Group, Dpt. Electronic Engineering,
E.T.S.E, Universitat de València. Dr. Moliner, 50. 46100 Burjassot. Valencia. SPAIN.
E-mail: alfredo.rosado@uv.es

Abstract: This work presents a methodology for implementing neural activation function in programmable logic using tools from fuzzy logic. This methodology will allow implementing these intrinsic non-linear functions using comparators and simple linear modellers, easily implemented in programmable logic. This work is particularized to the case of a hyperbolic tangent, the most common function in neural models, showing the excellent results yielded with the proposed approximation. *Copyright © 2005 IFAC*

Keywords: neural networks, hardware, embedded systems, hardware programming, circuit design, fuzzy logic.

1. INTRODUCTION

The field of study in Neural Networks has experimented an exponential increase on applications, algorithms and hardware implementation during last years (Arbib, 03), (Haykin, 99). Most of this success is due to its character as universal modeller, and its flexibility working as non-linear classifiers (Haykin, 99). There are a big amount of neural models, increasing continuously, as can be seen in specialized literature. There is an element giving the name to the model and playing an essential role; it is the minimum calculation element: the neuron. A general neuron can be described by the structure shown in Fig. 1.

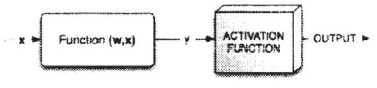

Fig. 1. General structure of a neuron.

Two key elements can be enumerated inside a neuron. The first block is a function of the synaptic weights and the inputs to the neuron. Amongst the most common functions, it is used the scalar product of the vectors of the synaptic weights and the inputs (equation 1), namely, $\mathbf{w}_n = [w_n(1)\ldots. w_n(L)]$ and $\mathbf{x}_n = [x_n(1)\ldots. x_n(L)]$.

$$y_n = \mathbf{w}_n \cdot \mathbf{x}_n^t = \sum_{i=1}^{L} w_n(i) \cdot x_n(i) \qquad (1)$$

This function is the most usual in the multilayer perceptron, one of the most popular neural models used in a great amount of applications (Haykin, 99). Another function, also quite habitual, is the one that calculates the distance between these two vectors, namely:

$$y_n = \left\| \mathbf{w}_n - \mathbf{x}_n \right\|^2 = \sum_{i=1}^{L} \left[w_n(i) - x_n(i) \right]^2 \qquad (2)$$

This kind of function is used in widely spread neural models, as radial basis functions (RBF) and self organizing maps (SOM) (Haykin, 99), (Arbib, 03), widely used in several fields of knowledge and research, as digital signal processing. Moreover, some of them have been implemented in programmable logic (Rosado, 98), (Omondi, 05).

The second block inside a neuron is the activation function, it usually is a non-linear function and the element where the power of calculation of different neural models resides (Haykin, 99). Next table shows the most common activation functions.

Table 1 Most common activation functions: β, K_1 and K_2 are constant parameters

Name	Expression.
Sign	$f(y)=\begin{cases}-1 & if \quad y<0\\ 0 & if \quad y=0\\ 1 & if \quad y>0\end{cases}$
Sigmoid	$f(y)=\frac{1}{1+e^{-\beta\cdot y}}$
Hyperbolic tangent	$f(y)=\frac{1-e^{-\beta\cdot y}}{1+e^{-\beta\cdot y}}$
Piece-Linear Function	$f(y)=\begin{cases}-1 & if \quad y\le 0.5\\ x & if \quad \lvert y\rvert\le 0.5\\ 1 & if \quad y\ge 0.5\end{cases}$
Gaussian function	$f(y)=K_1\cdot e^{-K_2\cdot y^2}$

As can be seen, all activation functions are non-linear functions; a linear function will be pointless, mainly because the performance of these methods lays on their non-linear behaviour.

It is easy to infer that a key problem in the hardware implementation of neural models is the activation function. There exist several methods of implementing it, as look-up table or Taylor series, external memory, and others, usually requiring a lot of hardware resources.

This work proposes an approximation based on fuzzy logic that allows a simple implementation on programmable logic using comparators and linear models.

In the next section, the proposed method is developed, and then particularized for a hyperbolic tangent, testing later the approximation in a hardware implementation using programmable logic. An IP core is proposed for easy customizing of the neural structure.

2. FUZZY LOGIC APPROACH

The proposed method of implementing this activation function relies on using fuzzy logic as the basic element to model it. The procedure consists on splitting the function to be modelled in linear pieces, assigning each one to a linguistic variable (in our case Low, Medium and High) with a membership function (Klir, 97). In Fig. 2 the hyperbolic tangent can be seen with its corresponding linear approximation.

For every one of the pieces a membership function used in fuzzy logic (Klir, 1997) is used. Simplicity is the main target in the hardware implementation of selected functions, triangular functions will be chosen (although trapezoidal would also be a simple candidate).

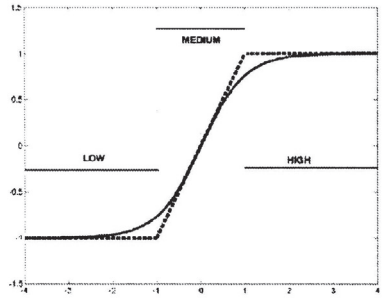

Fig. 2. Structure of the proposed approximation for the hyperbolic tangent. Black line: real function. Dotted line: linear pieces proposed.

Fig.3 depicts the membership functions defined in order to model the hyperbolic tangent in linear pieces.

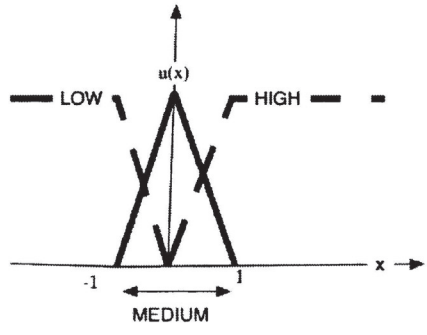

Fig. 3. Low, Medium and High Membership Functions.

The following element to be defined that has already been introduced in Fig. 2 is the mathematical expression of the linear pieces set in the model. In our case, that is:

$$\textbf{if}\begin{cases} y & low & f(y)=-1\\ y & medium & f(y)=a\cdot y\\ y & high & f(y)=1\end{cases} \qquad (3)$$

Where a is the slope of the hyperbolic tangent in the origin (that will be different, varying as a function of the value of β). The value of the function at a specific point, x_0, is given by:

$$f(y_0)=(-1)\mu_1(y_0)+\mu_2(y_0)y_0+(+1)\mu_3(y_0) \qquad (4)$$

Where μ_1, μ_2 and μ_3 are the membership functions of Low, Medium and High respectively. In Fig. 4, the proposed approximation is drawn together with the original function.