

#104

BICA+AI
2020

Logical circuits of a $\Sigma\P$ -neuron model.

Kazakov Mukhamed¹

kasakow.muchamed@gmail.com

¹Institute of Applied Mathematics and Automation of Kabardin-Balkar Scientific Centre of RAS (IAMA KBSC RAS), Nalchik.



Author



SUMMARY

The paper proposes a schematic representation for implementing a $\Sigma\P$ -neuron model. Two types of digital and hybrid $\Sigma\P$ -neurons are presented with two layers that include multiplication and summing operations. The networks are built using digital and analog electronics. The multiplication operations in both networks are carried out identically. Summing operations in the digital network is implemented through a digital circuit - an adder, while in the hybrid framework it is implemented via an analogue-to-digital conversion and analogue voltage adder. To set up the threshold voltage in the hybrid and digital networks the hybrid and logic comparators are employed correspondingly. The results obtained can be used to construct $\Sigma\P$ -neural networks that could be embedded into hardware/software for controlling mechatronic systems.

INTRODUCTION

Neural networks are successfully trained to solve problems of pattern recognition, search of associative memory, function approximation, optimization, and others. Usually learning process involves adjusting the weights of connections between neurons. A classic training scenario includes an input image sequence [1].

The $\Sigma\P$ -neuron and $\Sigma\P$ -neural network potential has been actively explored and applied since the 90s. The $\Sigma\P$ -neuron model is a generalization of the classical model of a formal neuron with a linear function. This model reflects better properties of its natural prototype and covers a much wider class of transformations [1, 2]. The article proposes functional and logical $\Sigma\P$ -neural models involving logical and digital elements.

1. Shibzukhov Z.M., Constructive Learning Methods for $\Sigma\P$ Neuron Networks. Nauka: Mos-cow. 159 p. (2006)

2. Shibzukhov Z.M. About $\Sigma\P$ -neuron models of aggregating type. - In: Lecture Notes in Computer Sciences 9719. Advances in Neural Networks - ISNN 2016. Eds: L.Cheng, Q.Liu, A.Ronzlin. Springer. PP.657-664. (2016)

STATEMENT OF THE PROBLEM

In practice neural networks are rather widely simulated on digital computers. The training procedures of the neural networks are quite convenient, as well as their application in solving certain problems. In a distributed system all network elements and operations are carried out programmatically. Consequently, we have parallel implementation of the artificial neural network training and sequential implementation of the digital computers. Therefore, this approach is not well-suited for full revealing of capabilities of the neural networks. Thus a need arises for a new approach to implementing artificial neural networks. This approach exists in practice [4] although its realization entails some technological difficulties while the software implementation of digital computers, in most cases, is lack of similar problems

$\Sigma\P$ -neuron model

A $\Sigma\P$ -neuron is a type of artificial neuron. The summation function for a discrete $\Sigma\P$ -neuron is a multilinear function:

$$s(\mathbf{x}) = \theta + \sum_{k=1}^N \omega_k \prod_{i \in i_k} x_i$$

where:

$\mathbf{x} = (x_1, \dots, x_n)$ is a input vector;

ω_k are weights;

$\emptyset \neq i_k \subseteq \{1, 2, \dots, n\}$ is a synaptic cluster;

θ is the bias.

The $\Sigma\P$ -neuron can be represented as a two-layer network with an intermediate layer consisting of Π -elements and the Σ -neuron on the output layer:

$$y = \text{out} \left(\theta + \sum \omega_k s_k \right)$$

$$s_k = \prod_{i \in i_k} x_i$$

The threshold function can be used as the activation function y of a $\Sigma\P$ -neuron:

$$\text{out}(s) = \begin{cases} 1, & \text{if } s \geq T, \\ 0, & \text{if } s < T, \end{cases}$$

where T – is the threshold value.

RESULTS

This section discusses two logical circuits of $\Sigma\P$ -neuron model: the circuit with an analog adder and the circuit with digital adder. Each circuit can be conventionally divided into two layers - Π -layer (conjunction) and Σ -layer (summation). The circuits basically differ from each other only in Σ -layer. The functional elements are described in detail in [6].

First we describe the Π -layer shared by the two circuits. This layer includes registers, 2-input AND gates and M -input AND gates (where M – number of neuron inputs). The digital signal enters Π -layer in the form of a binary M -dimensional vector, with respect to the number of inputs. The number of M -input AND gates corresponds to the number of synaptic clusters (see Fig. 1). In the most general case, to simultaneously implement all possible clusters, it is necessary to employ $(2M-2)$ gates. However, for practical purposes fewer gates might be more appropriate. For each M -input gate there is an M -bit register whose value determines what input signals participate to form the cluster (one input per register). The 2-input AND gate and the register determine which signals in the conjunction are important. Thus each M -input AND gate is cluster.

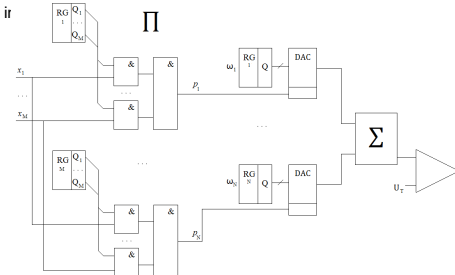


Fig. 1. Diagram of a $\Sigma\P$ -neuron and analog adder.

Now consider the Σ -layer and analog adder circuit (see Fig. 2). The synaptic function is realized through a set of register and DAC. The synaptic weights are stored in registers. DAC converts the M -input AND gate digital signal into the voltage level X_k with respect to the synaptic weight. An analog adder produces an output equal to the sum of the input voltages of all DACs:

$$U_s = k \sum_{k=1}^N X_k$$

In order the input voltage U_s does not exceed the operating voltage U_0 , the condition $k = 1/N$, should be met, where N is the number of synaptic clusters. A comparator performing a threshold-based activation function outputs a digital signal.

RESULTS

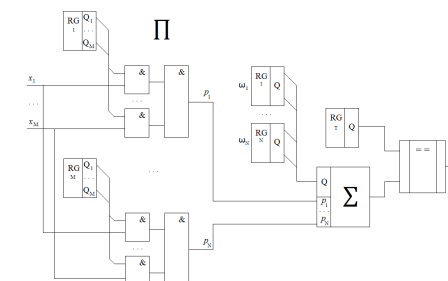


Fig. 2. $\Sigma\P$ -neuron with digital adder.

It is assumed that there are several learning methods for the register values corrections and one of them is training the artificial neural network computer model. Thus we get values for synaptic weights and clusters combined for the input signal. The results obtained are loaded into the corresponding registers and in this case parallel loaded registers are used.

Another method is in interactive training the artificial neural network employing a brain-computer interface. Each step of training involves the input-output analysis with software changing the register values in real time.

At last the method which requires an additional circuit that capable of directly training the network, including modifying weights and the clusters rebuilding for each learning stage. This method suggests storing the weight values in reversible counters. While training, synaptic weight increases or decreases with respect to the incoming signal to the counter.

CONCLUSIONS

The analog processing elements for the summation function compared with digital adder result in the speed benefit, but loses in accuracy. When building neural networks digital and analog components can be combined, according to accuracy or speed priority. Since the weight coefficients and synaptic configuration are determined by the registers values, the neural model can be trained quite simply, either using parallel data loading or modifying the set values.