PHASE-BASED NEURON AS A BIOLOGICALLY PLAUSIBLE NEURON MODEL

IONEL-BUJOREL PĂVĂLOIU[1], DUMITRU-IULIAN NĂSTAC[2], PAUL DAN CRISTEA[1]

Key words: Artificial neural networks, Complex-valued neural networks, Universal binary neuron, Phase-based neuron.

Complex-valued neural networks (CVNNs) are extensions of the classical neural networks, which have complex-valued weights, accept complex inputs and have more computational power than the classical ones. We present the structure and function of the phase-based neurons (PBNs), a simple type of CVNNs that uses as weights and biases complex numbers with magnitude 1, the phase being the only tunable parameter. We provide an adaptation method for PBNs and show that PBN based CVNNs have more computational power than classical artificial neural networks (ANNs), being able to implement some non-linearly-separable mappings, such as the XOR logical function. We show that the PBS has many features found in a class of biologically plausible neural networks called the phase based spiking neuron.

1. **INTRODUCTION**

1.1. COMPLEX VALUED NEURAL NETWORKS

The complex-valued neural networks (CVNNs) are artificial neural networks that deal with complex valued information by using complex-valued parameters and variables. The history of the CVNNs started back in 1971, with [1], written by N.N. Aizenberg and his team in the former Soviet Union. The paper was published during the so called "Artificial intelligence winter", a few years after the Minsky and Papert famous report [2] that showed the limitations of single-layer perceptrons. These ANNs binary classifiers, introduced by Rosenblat [3], cannot learn an exclusive disjunction (exclusive OR/XOR) function, because of their limitation as linearly-separable domain classifiers. Introducing additional hidden layers is not the only solution to this problem, another one being the use of a single neuron with complex valued weights [4].

[1] "Politehnica" University of Bucharest, BioMedical Engineering Center, 313 Spl. Independenței, 060042 Bucharest, Romania, E-mail: bujor.pavaloiu@ing.pub.ro, pcristea@dsp.pub.ro.

[2] "Politehnica" University of Bucharest, Dept. of Electronics, Telecom. and IT, O.P. 16, C.P. 77, 061112 Bucharest, Romania, nastac@ieee.org.

1.2. THE UNIVERSAL BINARY NEURON

The universal binary neuron (UBN) and the multi-valued neurons (MVN) are the component of the research field on CVNNs started by N.N. Aizenberg and continued by I.N. Aizenberg [1, 4–6].

An UBN operates with complex-valued weights and bias, and has a complex-valued activation function, which depends only on the argument of the weighted inputs. The activation function is a P-realizable Boolean function over the field of complex numbers for an input vector x with n elements and an associated weight vector w (including a supplementary parameter w_0 as bias), both with complex entries. We will use [6] to introduce the theoretical framework of the UBN.

$$y = f(x_1, \ldots, x_n) = P_B(z) = P_B(w_0 + w_1 x_1 + \ldots w_n x_n), \tag{1}$$

where

$$P_B(z) = (-1)^h, \quad \text{if} \quad \frac{2\pi h}{m} \le \arg(z) < \frac{2\pi(h+1)}{m}. \tag{2}$$

The UBN computes the output using just the phase of z – the net input (inputs weighted sum) and its position in the complex plane divided into an even number m of equal sectors, where h is the sector index. The output $y = f(x_1,\ldots, x_n)$ in (1) is +1 in the even sectors and –1 in the odd ones. Only the phase of the neuron inputs weighted sum z is significant for the output, its amplitude having no influence on it.

In Fig. 1 we show the behavior of P_B function for $m = 4$. The output is 1 for the net input in quadrants I and III, and –1 in quadrants II and IV.

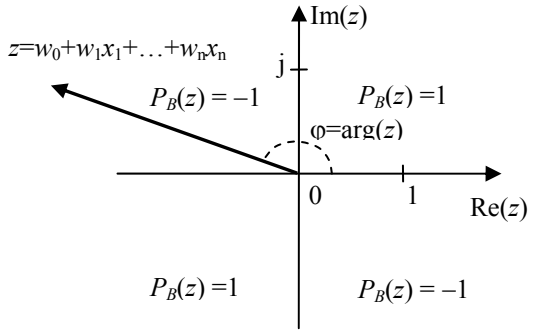

Fig. 1 – Activation function using a P-realizable Boolean function with $m = 4$ sectors.

The UBN and MVN have confirmed learning algorithms, which modify the weights as a function of the error (net input not being in the correct sector) and the

complex conjugate of the input, in order to move the net input along the unit circle. They give excellent results in solving various recognition and classification problems, including some with rather complicated decision rules [4–6].

2. THE PHASE-BASED NEURON

2.1. THE PHASE-BASED NEURON MODEL

The UBN uses two parameters for each complex weight, the real and the imaginary part (or the magnitude and the phase). This approach is essentially different from the one used in classic neural networks, where weights are single real numbers. The fact that the output of the UBN is given by the phase of z – the net input, provides the idea to work only with phases as parameters of the network. In [7] we proposed the phase-based neuron (PBN), a simpler CVNN for which every weight is a complex number with a fixed modulus 1 and a phase φ – the tunable parameter of the weight:

$$w_k = 1\angle\varphi_k . \tag{3}$$

The output of the network is computed the same way as in the case of UBN, by using the function P_B defined in (2). In PBN, a weight just rotates the corresponding input according to its phase and the challenge is to find the appropriate phases that will rotate the inputs such that the net input, their sum, will be placed in the right type of sector for any given pattern.

2.2. IMPLEMENTATION OF THE XOR FUNCTION

In the following, we present the potential of the PBN to solve the exclusive OR (XOR) problem.

We chose the weights using a heuristic search:

$$\begin{aligned} w_0 &= \frac{1}{\sqrt{2}}(1-\mathrm{j}) = 1\angle -\frac{\pi}{4}, \\ w_1 &= \frac{1}{\sqrt{2}}(1+\mathrm{j}) = 1\angle \frac{\pi}{4}, \\ w_2 &= \frac{1}{\sqrt{2}}(-1-\mathrm{j}) = 1\angle -\frac{3\pi}{4}. \end{aligned} \tag{4}$$

The values of the inputs, activation and of the outputs for the four patterns are shown in Table 1 and the corresponding mapping in the complex plane is represented in Fig. 2.

Table 1

Heuristic solution of the XOR problem using PBN

No.	x_1	x_2	z	y
1	–1	–1	$\frac{1}{\sqrt{2}}(1-\mathrm{j})$	–1
2	–1	1	$\frac{1}{\sqrt{2}}(-1-3\mathrm{j})$	1
3	1	–1	$\frac{1}{\sqrt{2}}(3+\mathrm{j})$	1
4	1	1	$\frac{1}{\sqrt{2}}(1-\mathrm{j})$	–1

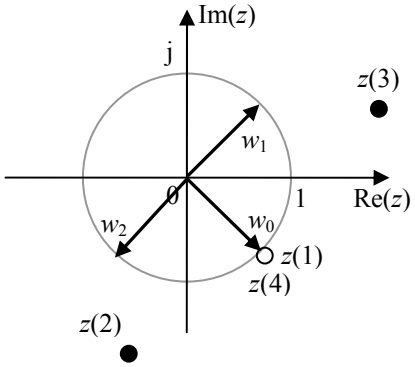

Fig. 2 – Weights w_1, w_2, and w_0 and net inputs $z(1)$, $z(2)$, $z(3)$, and $z(4)$ for data defined in *Table 1*. Filled circles correspond to positive outputs, empty circle to a negative output.

2.3. TRAINING FOR PBN

As shown on Fig. 3, we denote by ψ the net error – the phase difference between the target t and the net input z, and by ψ_k the k^{th} input error – phase difference between the target t and the weighted input phase $w_k x_k$.

Several adaptation methods for a given pattern are given in [9]. They can minimize ψ directly or by minimizing the phase differences ψ_k. For more than one pattern, the updates are computed for each pattern and batch learning is used.

In PBN, a weight just rotates the corresponding input such that the net input is placed in the correct type of sector for any given pattern. Increasing the number of sectors, we boost the representation power of the neuron, *i.e.*, nonlinearities it can cope with, but we also increase the overfitting risk.

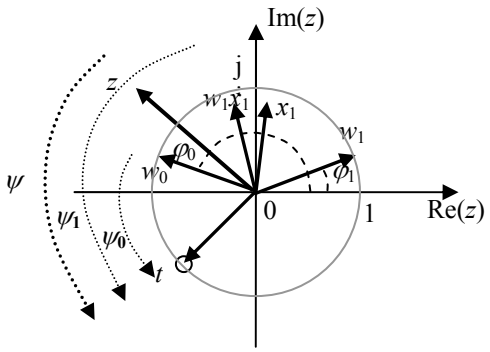

Fig. 3 – Training for PBN.

3. **BIOLOGICALLY PLAUSIBLE NEURONS**

3.1. SPIKING NEURON MODELS

A biological neuron receives action potential (spikes) from the afferent neurons and, after their processing as inputs, can issue a spike to participate at the activation of the efferent neurons. Even if there are several thousands of neuron types, their structure is essentially similar. A neuron adds-up in soma the input information from other neurons, whose axons are connected to its dendrites through synapses. When certain conditions are fulfilled, an action potential is triggered and travels down the axon to synapses connecting with other neurons.

The leaky integrate-and-fire model reproduces electrophysiological data with good accuracy. It consists of a capacitor C in parallel with a resistor R driven by a current $I(t)$, as in Fig. 4. A presynaptic spike $\delta(t)$ is low-pass filtered at the synapse (Fig. 4a) and generates an input current pulse $\varepsilon(t)$ that sums up in the soma of the neuron. When the membrane potential $u(t)$ exceeds a given threshold ϑ, it fires, sending a spike to the efferent neurons, like in Fig. 4b.

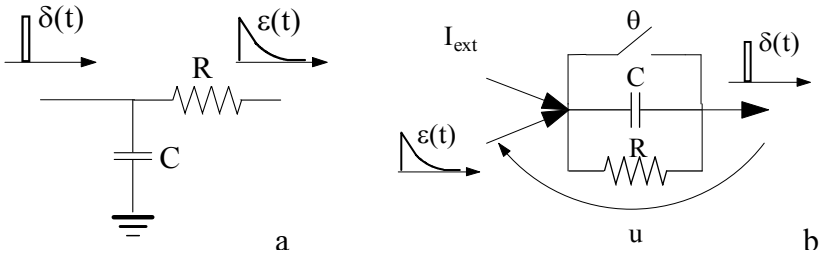

Fig. 4 – Equivalent electrical circuit for synapse (a) and for soma (b).

The spiking response model defined in [10] describes the state of a neuron by a single variable u, using kernel functions for different aspects of its behavior. Its state is the summation of presynaptic pulse-response functions, a self-spike response function and an external input function. The neuron fires if u reaches a threshold ϑ from below. The firing time $t(f)$ is defined as the moment when

$$u(t) = \vartheta \text{ and } \frac{\mathrm{d}}{\mathrm{d}t}u > 0\,. \tag{5}$$

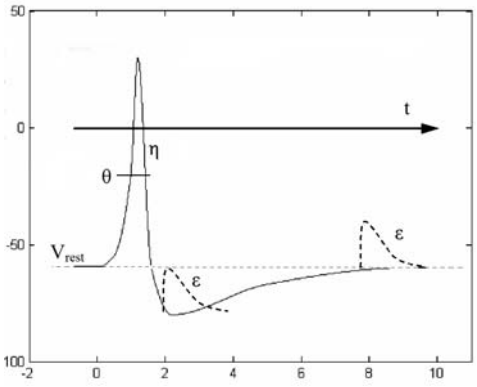

Fig. 5 – Spike response model, from [10].

Due to its time-only dependence, it is used to avoid integration methods and time-driven simulation. Determining the next threshold-crossing point, *i.e.*, the next firing time, an event driven approach to the simulation can be applied.

Different functions are considered for ε, from very simple ones to some more plausible. In Fig. 6 there are shown some of these functions.

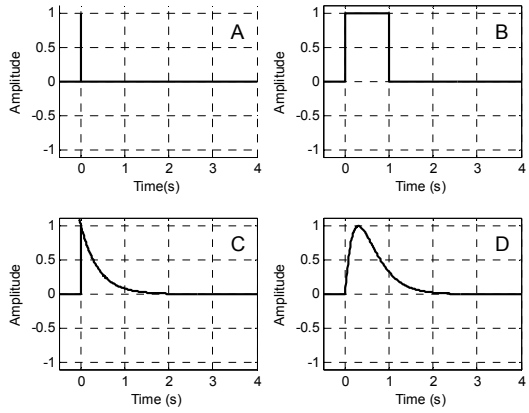

Fig. 6 – Simulation PSP waveforms: A – δ(t) , B – square, C – exponential $e^{-\frac{t}{\tau}}$, D – alpha $te^{-\frac{t}{\tau}}$.

3.2. INFORMATION MAPPING

The way the data is represented in ANNs is one of the fields where ANNs challenge the biological systems [11]. Starting from the inputs, there were tried several methods for the neural system models, that encode real, integer or binary values to the properties of the PSPs that enter the network. They include rate code, count code, binary code, timing code, synchrony based code, rank order code and delay coding [12]. These encodings are more or less natural, but mechanisms that drive them are the spike timing, phase, or correlations [10].

Because time is continuous and it is included in the spike equation, we can encode with a single spike any analogue number x as a spiking time $x+T$, where T is a given reference time. This version of timing coding, named in many papers as "delay coding", is used for problems addressed formerly by the classical ANNs.

4. **PHASE-BASED SPIKING NEURON**

4.1. PARAMETERS

We will propose here the Phase-Based Spiking Neuron (PBSN), a spiking neuron that is inspired by the PBN. The PBSN has the following features:

• The PSP is made by a finite sequence of sine waves. The number of waves is given by the maximum time lag induced by input encoding plus one period, pledging the proper summation of PSPs. For convenience, the frequency of the sine is taken 1 Hz. In Fig. 7 it is shown a PSP for two binary inputs with delay coding.

• The input encoding method is the delay coding, as described in 3.2. An input will be the PSP delayed by a time equal with the value of the input expressed in seconds. For binary inputs, the input 1 is coded with no delay and the input –1 is coded with a delay of 0.5 s, corresponding to a phase delay of π radians.

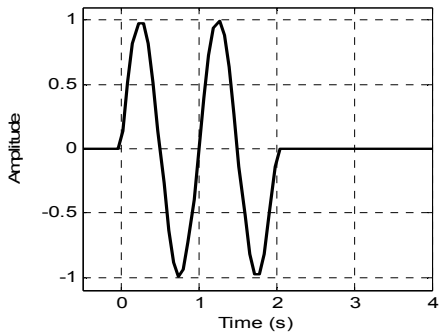

Fig. 7 – Phase-based spiking neuron PSP for the XOR problem.

• The weights are represented by delays of the inputs. For causality reasons, they are positive numbers from 0 to 1s, corresponding to phase delays from 0 to 2π.

• The output is given by the time interval when the net input reaches the threshold $\vartheta = 0$ from below. The time window where u is evaluated starts 1 second after the maximum delay given by the input encoding. The interval is divided into 4 periods of 0.25 seconds (corresponding to the quadrants in the complex plane). If the threshold is reached in the odd periods, the output is 1 and if it reached in the even periods, the output is –1.

4.2. XOR PROBLEM USING PBSN

We will use the XOR problem to illustrate the effectiveness of the PBSN and to emphasize the similarities with the PBN. The nonlinear problem and its heuristic solution from Table 1 will be transferred in the time domain.

In Fig. 8 you can see the PBSN carrying out the XOR function.

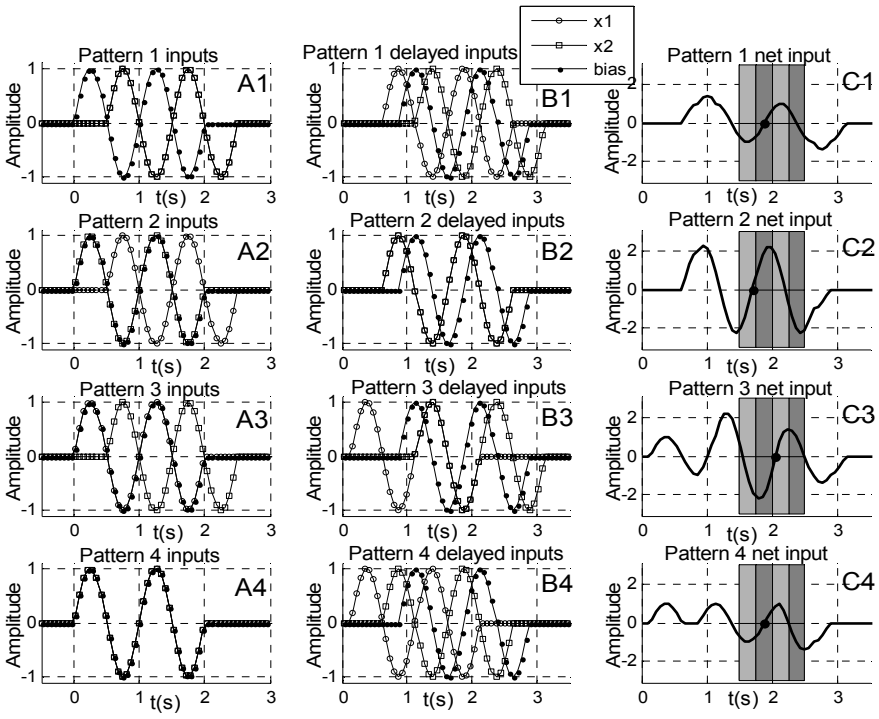

Fig. 8 – XOR problem implementation with PBSN.

The waveforms corresponding to the inputs for all the 4 patterns from Table 1 are shown in the plots A1–A4. They are sine waveforms with two periods, like the

PSP from Fig. 7. For the input value being 1 (including the value 1 associated to the bias), the delay is 0. When the input is –1, the waveform is delayed with 0.5 seconds (the equivalent of π). The synapses will influence these inputs with postponements of 0.875 s, 0.125 s, 0.625 s, corresponding to phase delays of $7\pi/4$, $\pi/4$, $5\pi/4$, as shown in the plots B1–B4. The phase delays are equivalent with the heuristic phases from (4.), having positive values for causality reasons. In the plots C1–C4 from Fig. 8, we have the sum (integration) of the delayed PSPs from the plots B1–B4, respectively. The expectation time sectors are represented with colored bars and the firing time is highlighted as a small disk. It is established corresponding to (5), being the time when the potential u reaches the threshold υ equal to 0 V from below. The firing is placed respectively in the time sectors 2, 1, 3, 2, resulting in the outputs –1, 1, 1, –1, equal with the targets for the XOR problem.

The summation of presinaptic pulse-response function is approximately a sine-wave for the evalution time-window, because in it we add just full waves sine waveforms. This implies that inside this window the sum of kernel functions can be done in the simpler phasors domain.

4.3. PBSN TRAINING

Because the PBSN has the phasor representation identical with PBN for the expectation windows, the training methods using error minimization from [9] are valid as well. The most natural one is the direct minimization for the PBN, where the weighted inputs rotate in order to minimize their phase error ψ_k (Fig. 3). The update of a weight for a given pattern is proportional with the phase difference from the target to the weighted input. The update for all the targets is computed as the sum of updates, using the batch learning.

This is translated in the following training algorithm for PBSN:

1. Take random delays for each input;

2. For each pattern m, compute the net input u (integration of the delayed PSPs), the firing time and find out the appropriate target (the center of the closest time sector). If all the firing times are in the right time sectors, stop the algorithm.

3. For the given pattern, compute the update for the input delay, proportional with the difference from the target to the delayed input.

4. Calculate the update of each input delay as the algebraic sum of that input delay updates computed for each pattern.

5. Update the time delays.

6. Go to step 2.

This novel spiking neuron has an adaptation method that is equivalent in the phasors domain with the training method for the PBN that was presented in [7] and [8]. That method is shown to give good results for two inputs binary problems.

The training for PBSN is extremely plausible, having the meaning that the synapses adapt by delaying the inputs towards the target, taking into account the whole set of patterns to be learned.

5. **CONCLUSIONS**

We presented in this paper the Phase-Based Neuron, a simplified version of the Universal Binary Neuron. The PBN uses just a single parameter per complex weight – the phase. Gradient descent methods are currently being applied for the implementation of the parity function, a generalization of the XOR function.

The PBN has a biologically plausible counterpart, the PBSN, an integrate-and-fire neuron with a sine-type PSP. The PBSN has a spiking response model reduced to phasor addition due to the inheritance of the properties from the PBN, making it simpler to be implemented.

ACKNOWLEDGEMENTS

The work was supported by the Sectorial Operational Programme Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the contract number SOP HRD/89/1.5/S/59758.

Receiced on 20 November 2012

REFERENCES

1. N. Aizenberg, Y.L. Ivaskiv, D.A. Pospelov, *A certain generalization of threshold functions,* Doklady Akademii Nauk SSSR, 196, 1971, pp. 1287–1290.
2. M. Minsky, S. Papert, *Perceptrons: An Introduction to Computational Geometry*, The MIT Press, Cambridge, 1969.
3. F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan, New York, 1962.
4. I. Aizenberg, *Solving the XOR and Parity N Problems Using a Single Universal Binary Neuron*, Soft Computing, **12**, *3*, pp. 215–222, 2008.
5. I. Aizenberg, N. Aizenberg, *Universal binary and multi-valued neuronsparadigm: Conception, Learning, Applications*, Lect. Not. in Cmp. Sc., Vol. 1240, Springer-Verlag, 1997, pp. 463–472.
6. I. Aizenberg, N. Aizenberg, J. Vandewalle, *Multivalued and universal binary neurons: theory, learning, applications*, Kluwer Academic Publishers, Boston, Dordrecht, London, 2000.
7. B. Păvăloiu, P.D. Cristea, *Phase-Based Neural Networks*, GSP 2011 – 2nd International Workshop on Genomic Signal Processing, UPB, Bucharest, 26–28 June 2011, pp. 95–98.
8. B.R. Păvăloiu, A. Vasile, P.D. Cristea, *Training for Phase-Based Neurons*, IWSSIP 2012 Conference Proceedings, Vienna, Austria, 11–13 April 2012, pp. 540–543.

9. B. Păvăloiu, P.D. Cristea, *Error minimization in Phase-Based Neurons*, 11th Symposium on *Neural Networks in Electrical Engineering*, Neurel 2012 Conference proceedings, pp. 155–161.
10. W. Gerstner, W.M. Kistler, *Spiking Neuron Models, Single Neurons, Populations, Plasticity*, Cambridge University Press, 2002.
11. P. D. Cristea, A. Cristea, T. Okamoto, *Knowledge Representation and Conversion*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **45**, *2*, pp. 157–173, 2000.
12. S. Thorpe, A. Delorme, R. VanRullen, *Spike-based strategies for rapid processing*, Neural Networks, **14**, *6–7*, pp. 715–726, 2001.