Implementation of a Gate Neural Network Based on Combinatorial Logic Elements

Taras Mikhailyuk(✉) and Sergey Zhernakov

Ufa State Aviation Technical University, Ufa, Russia
realotoim@mail.ru

Abstract. Generally, math models which use the "continuous mathematics" are dominant in the construction of modern digital devices, while the discrete basis remain without much attention. However, when solving the problem of constructing effective computing devices it is impossible to ignore the compatibility level of the mathematical apparatus and the computer platform used for its implementation. In the field of artificial intelligence, this problem becomes urgent during the development of specialized computers based on the neural network paradigm. In this paper, the disadvantages of the application of existing approaches to the construction of a neural network basis are analyzed. A new method for constructing a neural-like architecture based on discrete trainable structures is proposed to improve the compatibility of artificial neural network models in the digital basis of programmable logic chips and general-purpose processors. A model of a gate neural network using a mathematical apparatus of Boolean algebra is developed. Unlike formal models of neural networks, proposed network operates with the concepts of discrete mathematics. Formal representations of the gate network are derived. The learning algorithm is offered.

Keywords: Boolean algebra · Boolean neural network · Combinatorial logic · Delta rule · Gate neural network · Logical network · Widrow-Hoff rule

1 Introduction

Quite often in practice, there are problems associated with the compatibility of the functional and hardware-software parts of the device. These problems are very complex and require an integrated approach. Their solution leads to a change in qualitative and quantitative characteristics according to specified requirements.

Artificial intelligence algorithms require complex use of hardware and software. Due to specific nature of the research, such basic indicators as productivity, diminutiveness and low economic costs associated with the production and maintenance of the devices being developed remain unchanged. The approach based on modeling of artificial neural networks is versatile and flexible, but has limitations related to the field of their application. Among the disadvantages inherent to the computer of von Neumann architecture, we can distinguish the following:

- virtualization of calculators, architecture, physical processes;
- the dependence of the processing time on the size of the program;

- unjustified growth of hardware costs when increasing productivity;
- low energy efficiency, etc.

At present, there is an increasing number of specialized intellectual architectures aimed at overcoming the described drawbacks [1–8]. Such devices have wide application range and are compatible with the environment of the computer system, but they also have some disadvantages. Generally, math models which use the "continuous mathematics" are dominant in the construction of modern digital devices, while the discrete basis remain without much attention. However, solving the problem of constructing effective computing devices it is impossible to ignore the compatibility level of the mathematical apparatus and the computer platform used for its implementation. In the field of artificial intelligence, this problem becomes urgent during the development of specialized computers based on the neural network paradigm.

Existing mathematical models of a neuron operate with continuous quantities, are realized on the basis of an analog elements, which leads to their poor compatibility with digital equipment. But at the same time, most neural networks use the principles of digital logic [2–4, 6–8]. And as the result, in promising computing devices being developed multi-level systems of models are implemented. These systems introduce certain disadvantages in the final implementation of the solution [9, 10].

In this paper, a method for constructing a neural-like architecture based on discrete trainable structures is proposed to improve the compatibility of artificial neural network models in the digital basis of programmable logic chips and general-purpose processors.

2 Model of the Gate Neural Network

The trainable gate network is representative of Boolean networks [5, 11–16] with the ability to specify the type of mapping of the vector of input signals to the output vector, using the learning algorithm. Such a network can be considered as an attempt to combine certain features of neural network technology and combinational logic gates to achieve a synergistic effect in the implementation of high-performance embedded systems.

We obtain a formalized representation of this type of network. It is known from dicrete mathematics that the full disjunctive normal form (FDNF) can be represented as follows:

$$f(x_1, \ldots, x_P) = \bigvee_{\substack{(\sigma_1, \ldots, \sigma_P) \\ f(\sigma_1, \ldots, \sigma_P) = 1}} x_1^{\sigma_1} \wedge, \ldots, \wedge x_P^{\sigma_P}, \tag{1}$$

while the disjunction of all sets has the form:

$$y = f(\sigma_1, \ldots, \sigma_P) = 1 \tag{2}$$

Rule (2) can be reformulated as a disjunction over all full product terms (FPT) of P variables:

$$\bigvee_{n=1}^{2^P} \psi_n(\mathbf{x}) = 1 \tag{3}$$

Then the minimal term can be written in the following way:

$$\psi_n(\mathbf{x}) = \bigwedge_{p=1}^{P} x_p^{M_p(n-1)} \tag{4}$$

Next, we define the function Mp (α):

$$M_p(\alpha) = \begin{cases} 0, & \alpha \in [i \cdot T;\ (i + 0,5) \cdot T), \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

where the period $T = 2^p$, $i = 0,\ 1,\ 2, \ldots, \frac{2^P}{T} - 1$.

The function (5) is square wave logical basis, similar to the Rademacher function [17]. Figure 1 shows the form of this function for p $\leq$ 3.

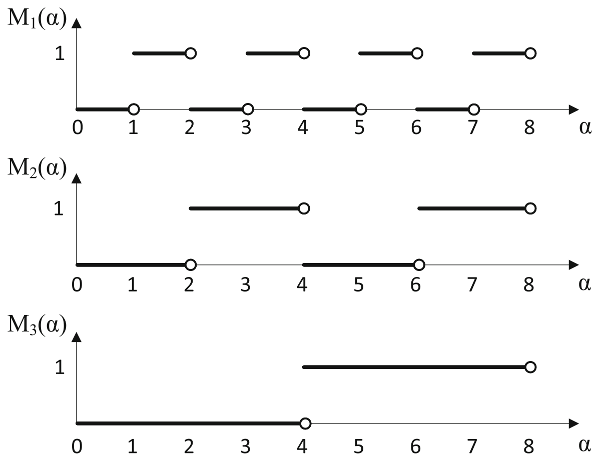

Fig. 1. View of the square wave function for p $\leq$ 3

The square wave function masks each variable included in Eq. (4) with the goal of specifying all FPTs. Next, we represent the FPT (3) in vector form:

$$\boldsymbol{\psi} = [\boldsymbol{\psi}_1(\mathbf{x}),\ \boldsymbol{\psi}_2(\mathbf{x}),\ \ldots,\ \boldsymbol{\psi}_N(\mathbf{x})], \tag{6}$$

where $\mathbf{x}$—the column vector of input signals:

$$\mathbf{x} = [x_1,\ x_2,\ \ldots,\ x_P]^T. \tag{7}$$

Next, we weigh functions of input signals in vector form, which is known from the theory of neural networks [1, 18, 19]:

$$\mathbf{w}^T \wedge \boldsymbol{\psi} = \mathbf{y}, \tag{8}$$

where $\mathbf{w}$—the column vector (9), and $\mathbf{y}$—the column vector (10):

$$\mathbf{w} = [w_1,\ w_2,\ \ldots,\ w_N]^T, \tag{9}$$

$$\mathbf{y} = [y_1,\ y_2,\ \ldots,\ y_S]^T. \tag{10}$$

The matrix Eq. (8) has a similar form with the equation describing the formal neuron, radial basis function network [18, 19] and also the sigma-pi network [20], but in this case the multiplication operation is replaced by the conjunction operation, since the matrices have a binary form.

For a network containing one element in the output layer, we get the following expression:

$$\mathbf{w}^T \wedge \boldsymbol{\psi} = [w_1,\ w_2,\ \ldots,\ w_N] \wedge \begin{bmatrix} \boldsymbol{\psi}_1(\mathbf{x}) \\ \boldsymbol{\psi}_2(\mathbf{x}) \\ \cdots \\ \boldsymbol{\psi}_N(\mathbf{x}) \end{bmatrix} = \bigvee_{n=1}^{N} w_n \wedge \psi_n(\mathbf{x}). \tag{11}$$

Next, we substitute (4) into (11), and obtain the following relation in the general form:

$$y = \bigvee_{n=1}^{N} w_n \wedge \bigwedge_{p=1}^{P} x_p^{M_p(n-1)} \tag{12}$$

The Eq. (12) is the model of a Boolean (gate) trainable network. It follows from expression (12) that in such model there are no operators inherent to neural networks, since they are bit-oriented. Weights are Boolean variables there, and not real numbers. This model describes a two-layer network in which the first layer is represented by a set of N constituent units (4), besides this layer does not require training. The output layer is represented by one disjunctive element, which summarizes the minterms, enabled by means of weight coefficients.

A similar dependence can be obtained for a network with several elements in the output layer:

$$\mathbf{w}^T \wedge \boldsymbol{\psi} = \begin{bmatrix} w_{11}, \ w_{12}, \ \ldots, \ w_{1N} \\ w_{21}, \ w_{22}, \ \ldots, \ w_{2N} \\ \ldots \\ w_{S1}, \ w_{S2}, \ \ldots, \ w_{SN} \end{bmatrix} \wedge \begin{bmatrix} \boldsymbol{\psi}_1(\mathbf{x}) \\ \boldsymbol{\psi}_2(\mathbf{x}) \\ \ldots \\ \boldsymbol{\psi}_N(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \bigvee_{n=1}^{N} w_{1n} \wedge \boldsymbol{\psi}_n(\mathbf{x}) \\ \bigvee_{n=1}^{N} w_{2n} \wedge \boldsymbol{\psi}_n(\mathbf{x}) \\ \ldots \\ \bigvee_{n=1}^{N} w_{sn} \wedge \boldsymbol{\psi}_n(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_S \end{bmatrix}. \tag{13}$$

Then the Eq. (6) for each output can be written in a general form:

$$y_s = \bigvee_{n=1}^{N} w_{sn} \wedge \bigwedge_{p=1}^{P} x_p^{M_p(n-1)} \tag{14}$$

The analysis of dependences (13) and (14) shows that it is possible to synthesize on their basis an arbitrary combination device with P inputs and S outputs, which has two levels of gates and has an increased speed in hardware implementation. These formulas represent a trainable logical basis. Figure 2 shows a graph of the network.

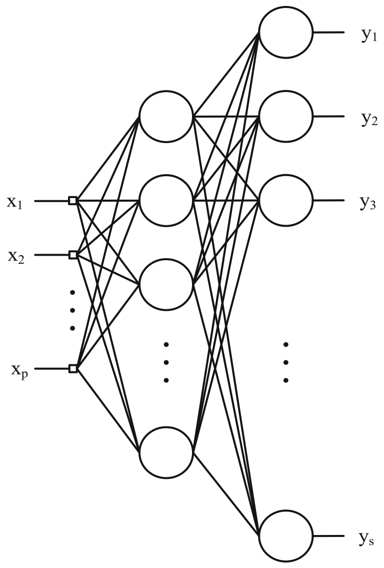

Fig. 2. Trainable gate neural network

It is known that the maximum number of combinations of P variables is equal to 2^P, and the number of functions is 2^{2^P}. It follows that the number of neurons of the first layer is not more than 2^P:

$$N \leq 2^P. \tag{15}$$

In turn, the number of neurons in the output layer is less than 2^{2^P}:

$$S \leq 2^{2^P}. \tag{16}$$

Thus, the maximum sum from (15) and (16) describes the largest network without repeating elements. However, duplication of elements can be aimed to increasing the reliability of the network.

It is not difficult to show that the obtained model can be realized in the form of a full conjunctive normal form (FCNF). On the basis of de Morgan's laws for several variables [21], we can show:

$$\bigvee_{n=1}^{N} a_n = \overline{\bigwedge_{n=1}^{N} \bar{a}_n}. \tag{17}$$

Applying the rule (17) to expression (12) we obtain:

$$y_s = \overline{\bigwedge_{n=1}^{N} \left(\bar{w}_{sn} \vee \bigvee_{p=1}^{P} x_p^{\bar{M}_p(n-1)} \right)}. \tag{18}$$

Next, replacing the variables, we get the FCNF:

$$\lambda_s = \bigwedge_{n=1}^{N} \left(m_{sn} \vee \bigvee_{p=1}^{P} x_p^{W_p(n-1)} \right). \tag{19}$$

Equations (12) and (19) are equivalent in essence like the FCNF and the FDNF are equivalent. It is seen from (19) that the weighing is performed by the disjunction operation, in contrast to (12).

3 Network Learning Algorithm

The learning algorithm of the perceptron according to the Widrow-Hoff rule is known from the theory of neural networks, [18, 19]:

$$w_{sn}(t+1) = w_{sn}(t) + \Delta w_{sn}(t), \tag{20}$$

$$\Delta w_{sn}(t) = x_n(t) \cdot (d_s - y_s(t)), \tag{21}$$

On the basis of (20) and (21), it is easy to see the following:

- weight w_{sn} can increase or decrease depending on the sign of the increment of weight Δw_{sn};
- weight change occurs when the output signal y_s deviates from the reference d_s only for the input x_n which causes this influence.

Using these statements, we can show the training algorithm for a binary network. We convert these formulas into a system of residual classes. It is known that additive operations and multiplication will look like the following [22]:

$$(a \pm b) \bmod c = ((a \bmod c) \pm (b \bmod c)) \bmod c, \tag{22}$$

$$(a \cdot b) \bmod c = ((a \bmod c) \cdot (b \bmod c)) \bmod c. \tag{23}$$

We describe (20) and (21), using (22) and (23). Then the Widrow-Hoff rules will take the form which is typical for operations performed by digital devices:

$$\begin{aligned} w_{sn}(t+1) \bmod q &= (w_{sn}(t) + \Delta w_{sn}(t)) \bmod q \\ &= (w_{sn}(t) \bmod q + \Delta w_{sn}(t) \bmod q) \bmod q, \end{aligned} \tag{24}$$

$$\begin{aligned} \Delta w_{sn}(t) \bmod q &= (x_n(t) \cdot (d_s - y_s(t))) \bmod q \\ = ((x_n(t) \bmod q) &\cdot ((d_s) \bmod q - y_s(t) \bmod q)) \bmod q, \end{aligned} \tag{25}$$

where q is a positive integer.

It is required that all variables (24) and (25) could accept only two states, or that the modulo is equal 2. Considering that additive operations can be replaced by the exclusive-OR operation and multiplication—by conjunctions, the Widrow-Hoff rule will be written in the following form:

$$w_{sn}(t+1) = w_{sn}(t) \oplus x_n(t) \wedge (d_s \oplus y_s(t)). \tag{26}$$

We apply rule (26) to the received network model (12). Taking into account the influence of minterms (4) on the learning element, we obtain the learning rule for the Boolean network:

$$w_{sn}(t+1) = w_{sn}(t) \oplus (d_s \oplus y_s(t)) \wedge \bigwedge_{p=1}^{P} \left(x_p(t)\right)^{M_p(n-1)}. \tag{27}$$

4 Analysis of the Results

On the basis of the dependence (12), the following features of the model can be noted:

- the model is a network;
- first and second layer have specialization;
- signals can be either excitatory or inhibitory;
- the type of generalization is different for FDNF and FCNF networks;

- there is no influence of minterms (maxterms) on each other.

Unlike formal models of neural networks, the Boolean network operates with the concepts of discrete mathematics. From the point of view of an intelligent approach, only binary input signals processing may seem insufficient when working with higher-order sets, but the feature of the obtained formulas (12), (19) is in the possibility of applying them as a logical basis controlled by weight coefficients. It is known that on the basis of a Boolean basis arbitrary combinational devices are constructed. Furthermore, with the actual implementation of the trainable gate network, it is characterized by greater performance and reliability associated with the fixed depth of the gates and the simplicity of the individual handlers. For solving more complicated tasks it is possible to use the series of gate networks. In this case, the topology of the device is more homogeneous, which leads to the interchangeability of its individual elements.

The developed network can be considered as a basis for constructing feedforward neural networks with a flexible topology that can be adapted to a specific task, up to the level of logical elements.

The proposed approach has the following advantages:

1. Greater homogeneity of the topology of the device, in contrast to the formal neuron, which contains adders, multipliers, activation functions.
2. Increase of the applied component on the hardware level to solve specific problems.
3. Reduction of the occupied area of the crystal, which is required for the hardware implementation of the network.
4. Parallelizing of the processing and learning of the network at the level of logical elements.
5. Flexible learning architecture of a formal neuron.

5 Conclusion

The work in the field of creating discrete learning networks is aimed to solve the problems of optimizing hardware and software costs in the construction of neural networks and digital equipment in general. The trainable gate network is not intended to replace a feedforward neural network, but it can be considered as a basis for constructing any digital network. The possibilities of gate networks are quite various. They can find the application for the creation of associative memory devices, cryptography, high performance combinational devices, solvers of Boolean functions and in other applications.

References

1. Aljautdinov, M.A., Galushkin, A.I., Kazancev, P.A., Ostapenko, G.P.: Neurocomputers: from software to hardware implementation, p. 152. Gorjachaja linija - Telekom, Moscow (2008). (in Russian)
2. Mezenceva, O.S., Mezencev, D.V., Lagunov, N.A., Savchenko, N.S.: Implementations of non-standard models of neuron using Neuromatrix. Izvestija JuFU. Tehnicheskie nauki **131** (6), 178–182 (2012). (in Russian)
3. Adetiba, E., Ibikunle, F.A., Daramola, S.A., Olajide, A.T.: Implementation of efficient multilayer perceptron ANN neurons on field programmable gate array chip. Int. J. Eng. Technol. **14**(1), 151–159 (2014)
4. Manchev, O., Donchev, B., Pavlitov, K.: FPGA implementation of artificial neurons. Electronics: An Open Access Journal, Sozopol, Bulgaria, 22–24 September (2004). https://www.researchgate.net/publication/251757109_FPGA_IMPLEMENTATION_OF_ARTIFICIAL_NEURONS. Accessed 28 Jan 2017
5. Kohut R., Steinbach B.: The Structure of Boolean Neuron for the Optimal Mapping to FPGAs. http://www.informatik.tu-freiberg.de/prof2/publikationen/CADSM2005_BN_FPGA.pdf. Accessed 1 Feb 2017
6. Korani, R., Hajera, H., Imthiazunnisa, B., Chandra Sekhar, R.: FPGA modelling of neuron for future artificial intelligence applications. Int. J. Adv. Res. Comput. Commun. Eng. **2**(12), 4763–4768 (2013)
7. Omondi, A. R., Rajapakse, J. C.: FPGA Implementations of Neural Networks. Springer (2006). http://lab.fs.uni-lj.si/lasin/wp/IMIT_files/neural/doc/Omondi2006.pdf. Accessed 28 Jan 2017
8. Gribachev, V.: Element base of hardware implementations of neural networks (in Russian). http://kit-e.ru/articles/elcomp/2006_8_100.php. Accessed 30 June 2016
9. Mikhailyuk, T. E., Zhernakov, S. V.: Increasing efficiency of using FPGA resources for implementation neural networks. In: Nejrokomp'jutery: razrabotka, primenenie, vol. 11, pp. 30–39 (2016). (in Russian)
10. Mikhailyuk, T.E., Zhernakov, S.V.: On an approach to the selection of the optimal FPGA architecture in neural network logical basis. Informacionnye tehnologii **23**(3), 233–240 (2017). (in Russian)
11. Kohut, R., Steinbach, B.: Decomposition of Boolean Function Sets for Boolean Neural Networks. https://www.researchgate.net/publication/228865096_Decomposition_of_Boolean_Function_Sets_for_Boolean_Neural_Networks. Accessed 1 Feb 2017
12. Anthony, M.: Boolean Functions and Artificial Neural Networks. http://www.cdam.lse.ac.uk/Reports/Files/cdam-2003–01.pdf. Accessed 29 Jan 2017
13. Kohut, R., Steinbach, B.: Boolean neural networks. WSEAS Trans. Syst. **3**(2), 420–425 (2004)
14. Steinbach, B., Kohut, R.: Neural Networks – A Model of Boolean Functions. https://www.researchgate.net/publication/246931125_Neural_Networks_-_A_Model_of_Boolean_Functions. Accessed 1 Feb 2017
15. Vinay, D.: Mapping boolean functions with neural networks having binary weights and zero thresholds. IEEE Trans. Neural Netw. **12**(3), 639–642 (2001)
16. Zhang, C., Yang, J., Wu, W.: Binary higher order neural networks for realizing boolean functions. IEEE Trans. Neural Netw. **22**(5), 701–713 (2011)
17. Rademacher, H.: Einige Sätze über Reihen von allgemeinen Orthogonalfunktionen. Math. Ann. **87**(1–2), 112–138 (1922)

18. Hajkin, S.: Neural networks: a comprehensive foundation. Vil'jams, Moscow (2008). (in Russian)
19. Osovskij, S.: Neural networks for information processing, p. 344. Finansy i statistika, Moskow (2002). (in Russian)
20. Shin, Y., Ghosh, J.: Efficient higher-order neural networks for classification and function approximation. The University of Texas at Austin (1995). https://www.researchgate.net/publication/2793545_Efficient_Higher-order_Neural_Networks_for_Classification_and_Function_Approximation. Accessed 28 Jan 2017
21. Shevelev Ju, P.: Discrete mathematics. Part 1: The theory of sets. Boolean algebra (Automated learning technology “Symbol”), p. 118. TUSUR University, Tomsk (2003). (in Russian)
22. Omondi, A., Premkumar, B.: Residue number systems: theory and implementation, p. 312. Imperial College Press, London (2007)