Boolean Neural Networks

ROMAN KOHUT, BERND STEINBACH
Institute of Computer Science
Freiberg University of Mining and Technology
Bernhard-von-Cotta-Straße 2, D-09596 Freiberg,
GERMANY

Abstract: - In this paper, we present a new type of neuron, called Boolean neuron. Further, we suggest the general structure of a neural network that includes only Boolean neurons and may realize several sets of Boolean functions. The advantages of these neural networks consist in the reduction of memory space and computation time in comparison to the representation of Boolean functions by usual neural networks. The Boolean neural network may be mapped to a FPGA so that our new training algorithms substitute classical design methods of these circuits. In the example, we show the decomposition of a set of Boolean functions into common basic functions and their mapping to the general Boolean neural network.

Key-Words: - Boolean neuron, BNN, data representation, FTFS, functions set

1 Introduction

There are very strong developments of information technologies in the field of artificial intelligence, presently. Recent researches in neurology and neurobiology produced convincing results, which prove perfection of natural processes and living beings, humans particularly [10]. That also impels neural mathematics and neural computing to use principles of the behavior of human memory, mind and central neural system.

The range of use of neural computers, genetic algorithms or technique of chaos theory is not limited to data mining, quantum computing or production of robots. Having more than half a century of history behind, artificial neural networks, referred here as neural networks, initially was created to simulate Boolean functions [5] and first conclusions about the future of a new branch of science based on results of those simulations. Some times later, intensive research produced incredible variety of neural architectures and neural paradigms for different purposes. The range of usage of neural networks became much wider than just Boolean functions.

In this paper, we return to the root and apply neural network technique for compact representation and fast computation of Boolean functions. Currently, Di Wang and Narendra S. Chaudhari suggest new approaches for Boolean neural networks. Unfortunately, their methods for construction of Boolean neural networks, in detail Multi-Core Learning (MCL), Multi-Core Expand-and-Truncate Learning (MCETL) [2], and Fast Covering Learning Algorithm (FCLA) [3] are unsuitable for sets of Boolean functions. Moreover, their theory does not restrict to Boolean values and Boolean operations, and therefore it is not well suitable for FPGA circuits.

Vinay Deolalikar showed in [12] the ability of a Boolean neural network comprising only neurons with zero thresholds and Boolean weights to map given samples of a Boolean function. The author developed a mathematical model describing a network. It was shown that proposed model is quite amenable to algebraic manipulation. A key feature of the model is that it replaces the two input and output variables with a single "normalized" variable. This approach maps a Boolean function to a Boolean neural network using the back propagation, which needs very long time for the training process. For that, reason is searching for more an efficient solution.

Our previously suggested approaches [9] are too complicated for large Boolean functions, as well.

The rest of the paper has the following structure. Section 2 describes the problem of unacceptably large size of operating memory required for neural networks working with Boolean data. A new type of neural element, so called Boolean neuron is introduced in section 3. Section 4 describes general algorithms for training and using of a Boolean neural network. Section 5 shows in an example how a set of Boolean functions can be implemented as Boolean neural network. Section 6 summarizes the paper.

2 The Problem

There are many representations of Boolean functions. The simplest representations of a Boolean function are a formula or a table that defines the function value for each of the 2^n input vectors. However, these representations of Boolean functions have several shortcomings and do not satisfy all requirements of practical tasks. As result, a number of alternative representations of Boolean functions were proposed, which eliminate some of the disadvantages. Widely used are Karnaugh-Planes, Ternary Vector Lists - TVL, Binary Decision Diagrams - BDD etc. However, aside from advantages even these representations of Boolean functions have their own disadvantages as well. After analyzing their disadvantages, we propose the use of neural networks for representation of Boolean functions.

Some of the advantageous properties of neural networks are the common structure for different behaviors and the fast calculation of the implemented functions. A very important approach is the implicit decomposition of functions modeled by the neurons.

As well known, McCulloch and Pitts introduced neural networks in their seminal work [5]. This first type of a neural network models the function of a nervous system. Later on, it was called Boolean network. A node, together with its output channel, was bi-stable, i.e., it was either in the state zero or in the state one [4]. We emphasize that the root of the scientific field of neural networks is closed connected to the Boolean domain. Unfortunately, the Boolean neural networks suggested by McCulloch and Pitts introduced only a new theory but was not used for practical applications.

The main stream of the development of neural networks moved then outside of the Boolean domain. An overview of existing paradigms and approaches and their disadvantages are described in [2]. Here, we only repeat that all training algorithms can be classified into two categories based on their training process. The first category fixes the network structure, in detail the number of hidden layers and the number of neurons in each of them, and adjusts connection weights and thresholds in the parameter space by decreasing errors between model outputs and desired outputs. Examples of such methods are BackPropagation (BP) and Radial Basis Function (RBF). These algorithms cannot guarantee fast convergence and need more training time. The second category, called sequential training algorithms, adds hidden layers and hidden neurons in the training process. Examples of such methods are Expand-and-Truncate Learning (ETL) algorithm and Constructive Set Covering Learning Algorithm (CSCLA). Sequential training algorithms are promising because they guarantee faster convergence and need less training time [2].

Among existing types of neural networks, we chose the neural network model, called “Functional on the tabular functions set” (FTFS). Their advantages are higher precision and shorter time for learning in comparison with other models of neural networks.

In order to model Boolean functions efficiently, we suggest functional changes of the neural element and adapt this new type of neuron to the algorithms for training and using of the neural network FTFS.

3 Boolean Neuron

The basic element in modern neural networks, including the model “Functional on the tabular functions set”, is the programmable neuron, which defines relation between vector of input signals $\underline{Inp}$ and output signal Out.

$$Out = f\left(\underline{Inp}, \underline{w}, x_0\right) \tag{1}$$

Where $\underline{Inp} = \{x_1, x_2, ..., x_{N_x}\}$ – input vector of the neuron,

N_x – number of inputs,

$\underline{w} = \{w_1, w_2, ..., w_{N_x}\}$ – vector of synaptic weights,

x_0 – bias,

Out – output of the neuron,

f – Dependency, which includes nonlinear transformation (transfer function, activation function).

In general, input signals, weighting coefficients and bias are real numbers, less often – integers. The output Out is determined by an activation function, and can be a real or integer number. Using such values for Boolean data, which requires only one bit for each value, is clearly inefficient because of unnecessary memory expenses. To resolve this inefficiency we propose a Boolean neuron.

A Boolean neuron (or Boolean neural element) operates with Boolean signals and uses only Boolean operations. According to definition [6] and (1), the output signal of the Boolean neuron is defined as:

$$Out_B = f_B\left(\underline{Inp_B}, \underline{w_B}\right) \qquad (2)$$

Where index B indicates Boolean values and

$\underline{Inp_B} = \{x_1, x_2, ..., x_{N_x}\},\ x_i \in \{0,1\}$,

$\underline{w_B} = \{w_1, w_2, ..., w_{N_x}\},\ w_i \in \{0,1\}$,

f_B - Boolean transfer function,

$f_B, Out_B \in \{0,1\}$.

In the following, we omit the index B because only Boolean neurons are considered.

The structure of such a Boolean neuron is unchanged comparing to regular neuron. The general computational process and the structure of the neural network are unchanged, too [9 and 13]. The structure of Boolean neuron is shown in figure 1.

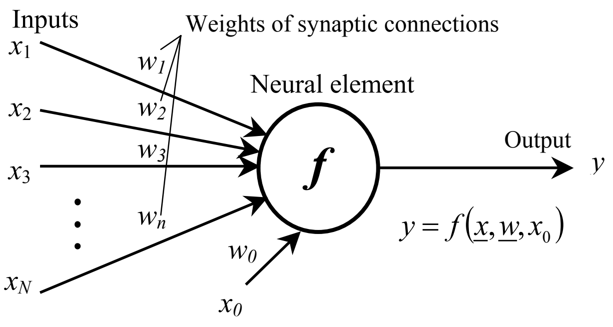

Fig. 1. General structure of Boolean neuron

The limitation to Boolean operations reduces time for converting input vector into output signal significantly. An additional advantage of the Boolean neuron consists in the reduction of necessary memory size.

Replacing normal neural elements by Boolean neurons in FTFS neural network we create a new type of neural network, called Boolean neural network (BNN). Boolean neural network belong like FTFS networks to the class of feed forward neural networks.

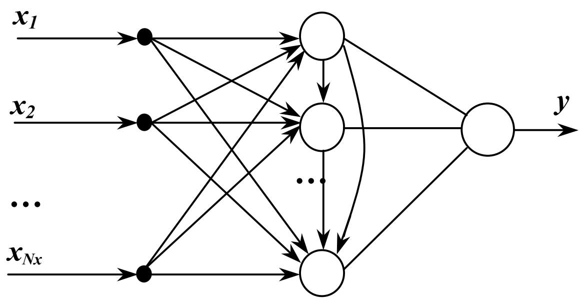

Fig. 2. General structure of BNN

As shown in figure 2 the basic structure of the Boolean neural network has additional to usual projective connections between neurons of the neighbor layers also lateral connections between neurons of the same layer. This feature of the structure results from the common paradigm of training neural networks FTFS.

Note, there are distinctions between Boolean neurons on the hidden layer and Boolean neurons on the output layer of the BNN. In general, each neuron on the hidden layer has a special Boolean transfer function, which is different form the Boolean transfer function of all other neurons in this layer. The mathematical description of the neuron with number z on the hidden layer is:

$$Out^{[z]} = f^{[z]}\left(\underline{Inp}, \underline{w}\right) \qquad (3)$$

where

$Out^{[z]}$	- output signal of the neuron with number z,
$f^{[z]}$	- transfer function of the neuron with number z,
z	- index $z = 1, ..., Z_N$
Z_N	- number of neurons on the hidden layer,

$f^{[i]} \neq f^{[j]}:\ \forall\ i \neq j;\ i, j \in [1, Z_N]$.

All neurons of the output layer have a fixed Boolean transfer function. This Boolean function connects the weighted inputs by exclusive OR (XOR) operations.

$$Out^{[j]} = \bigoplus_{i=0}^{Z_N}\left(Inp_i \wedge w_i^{[j]}\right) \qquad (4)$$

Where

j	– number of neuron on the output layer of the Boolean neural network,
$\oplus$	– Boolean operation „XOR”,
$\wedge$	– Boolean operation „AND”.

The following description of the algorithms of training and using shows the main difference between BNN and common FTFS network.

4 Training and using of the Boolean neural network

Boolean neural networks need like all feed forward neural networks two basic procedures, training and using. Frequently, the training process is called learning. Correspondingly, the using process solves the tasks and describes how the BNN works.

The starting point of the learning algorithm of BNN is the function table of the set of Boolean functions. This table is also called matrix A that covers the input and output signals as completes training set of the neural network [11]. The matrix A has 2^{N_x} rows and $N_x + N_y$ columns, where N_x is the number of Boolean variables and N_y is the number of Boolean functions. Input signals of the network are argument values and output signals are the values of the Boolean functions.

$$A = \left\| \begin{array}{ccc:ccc} x_{1,1} & x_{1,2} & x_{1,N_x} & y_{1,1} & y_{1,2} & y_{1,N_y} \\ x_{2,1} & x_{2,2} & x_{2,N_x} & y_{2,1} & y_{2,2} & y_{2,N_y} \\ \hdashline x_{i,1} & x_{i,2} & x_{i,N_x} & y_{i,1} & y_{i,2} & y_{i,N_y} \\ \hdashline x_{2^{N_x},1} & x_{2^{N_x},2} & x_{2^{N_x},N_x} & y_{2^{N_x},1} & y_{2^{N_x},2} & y_{2^{N_x},N_y} \end{array} \right\| =$$

$$= \left\| \begin{array}{ccc:cc} a_{1,1} & a_{1,2} & a_{1,N_x} & a_{1,N_x+1} & a_{1,N_x+N_y} \\ a_{2,1} & a_{2,2} & a_{2,N_x} & a_{2,N_x+1} & a_{2,N_x+N_y} \\ \hdashline a_{i,1} & a_{i,2} & a_{i,N_x} & a_{i,N_x+1} & a_{i,N_x+N_y} \\ \hdashline a_{2^{N_x},1} & a_{2^{N_x},2} & a_{2^{N_x},N_x} & a_{2^{N_x},N_x+1} & a_{2^{N_x},N_x+N_y} \end{array} \right\| \tag{5}$$

The base idea of the suggested training algorithm of the neural net is representation of Boolean functions by final polynomials of firstly unknown, unique Boolean base functions. The algorithm describes special decomposition of a set of Boolean functions into the set simpler, firstly unknown, unique Boolean base functions. Analog to the training algorithm of usual FTFS neural networks [11], our algorithm bases on a geometrical aim that should reduce the space dimension. The training algorithm consists of the following steps.

1. Compute the coefficient D_i for each row of the matrix A:

$$D_i = \bigvee_{j=1}^{N_x+N_y} a_{i,j} \,. \tag{6}$$

2. If all coefficients D_i equals to zero, then the algorithm stops. Otherwise, take a row vector $\underline{am_z}$ from the matrix A, where the coefficients $D_i = 1$ and the index z is number of training cycle.
3. Based on $\underline{am_z}$ compute the coefficients $k_{i,z}$ for each row of the matrix A:

$$k_{i,z} = \bigvee_{j=1}^{N_x+N_y} \left(a_{i,j} \wedge am_{z,j}\right) \wedge D_i \,. \tag{7}$$

4. Calculate a new matrix A by (8):

$$a_{i,j}^{(z+1)} = a_{i,j}^{(z)} \oplus (am_{z,j} \wedge k_{i,z}) \,. \tag{8}$$

$a_{i,j}^{(z+1)}$ and $a_{i,j}^{(z)}$ are coefficients of the matrix A after training cycle number $z+1$ and z corresponding.

If coefficients $k_{i,z} = 0$, no calculations are necessary for the row i.

A similar learning algorithm for general neural net FTFS is the orthogonalization procedure by Gramm-Shmidt [7, 8 and 11]. Our learning procedure for BNN includes successive transformation of the matrix A until (9) is valid.

$$a_{i,j}^{(2^{N_x})} = 0 \tag{9}$$

The results of the training algorithm are the unique Boolean base functions k_z , $z = 1,\ldots,Z_N$ and the weight vectors $\underline{am_j}$, $j = 1,\ldots,N_y$ of the output neurons. Z_N is the number of neurons on the hidden layer that is equal to number of training cycles.

The unique Boolean base functions k_z , $\forall \;\; z \in [1, Z_N]$ depend on input signals $\underline{x}$:

$$k_z = f(\underline{x}) \,. \tag{10}$$

and are realized by the neurons of hidden layer of Boolean neural network.

Using these data, the primary implementations matrix A can represent as shown in (11) that defines the using procedure of the BNN.

$$a_{i,j}^{(0)} = \bigoplus_{i=1}^{Z_N} \left(am_{z,i} \wedge k_{i,z}\right) \tag{11}$$

$a_{i,j}^{(0)}$ are coefficients of primary matrix A.

The Boolean neural network creates in the using regime the Boolean functions (12) depending on the vector of input signals $\underline{x}$.

$$y_s(\underline{x}) = \bigoplus_{i=1}^{Z_N} \left(am_{z,s+N_x} \wedge k_{s,z}(\underline{x}) \right) \qquad (12)$$

5 Example

For the better understanding and verifying of the obtained results, we show an example. Given is the set of simple Boolean functions $y_1, y_2, ..., y_7$; $y_i = f(x_1, x_2)$ in the table 1. From (6) follows that $\underline{D} = 1$. According step 2 of the above algorithm we have to select the row am where $D_i = 1$; in this case we have a free choice and take the last row for $\underline{am_1}$. The Boolean function k_1 of table 1 is calculated by (7).

Table 1: Primary implementation matrix A

x_1	x_2	y_1	y_2	y_3	y_4	y_5	y_6	y_7		D		k_1
0	0	0	0	0	1	1	1	1		1		1
0	1	0	1	1	0	0	0	1	⇒	1	⇒	1
1	0	1	0	1	0	1	1	0		1		1
1	1	0	0	0	1	0	1	1		1		1

am_1	am_2	am_3	am_4	am_5	am_6	am_7	am_8	am_9
1	1	0	0	0	1	0	1	1

Using formula (8) and the vectors $\underline{k_1}$ and $\underline{am_1}$ we calculate the new matrix A in table 2, whereby the values of the last row of matrix became zero.
In the next cycles, the same steps are repeated. We selected the rows am in the order of 2, 1 and 3.

Table 2: Matrix A after the first step of training

x_1	x_2	y_1	y_2	y_3	y_4	y_5	y_6	y_7		D		k_2
1	1	0	0	0	0	1	0	0		1		1
1	0	0	1	1	1	0	1	0	⇒	1	⇒	1
0	1	1	0	1	1	1	0	1		1		1
0	0	0	0	0	0	0	0	0		0		0

am_1	am_2	am_3	am_4	am_5	am_6	am_7	am_8	am_9
1	0	0	1	1	1	0	1	0

Table 3: Matrix A after the step 2 of training

x_1	x_2	y_1	y_2	y_3	y_4	y_5	y_6	y_7		D		k_3
0	1	0	1	1	1	1	1	0		1		1
0	0	0	0	0	0	0	0	0	⇒	0	⇒	0
1	1	1	1	0	0	1	1	1		1		1
0	0	0	0	0	0	0	0	0		0		0

am_1	am_2	am_3	am_4	am_5	am_6	am_7	am_8	am_9
0	1	0	1	1	1	1	1	0

Table 4: Matrix A after the step 3 of training

x_1	x_2	y_1	y_2	y_3	y_4	y_5	y_6	y_7		D		k_4
0	0	0	0	0	0	0	0	0		0		0
0	0	0	0	0	0	0	0	0	⇒	0	⇒	0
1	0	1	0	1	1	0	0	1		1		1
0	0	0	0	0	0	0	0	0		0		0

am_1	am_2	am_3	am_4	am_5	am_6	am_7	am_8	am_9
1	0	1	0	1	1	0	0	1

Table 5: Matrix A after the training

x_1	x_2	y_1	y_2	y_3	y_4	y_5	y_6	y_7		D
0	0	0	0	0	0	0	0	0		0
0	0	0	0	0	0	0	0	0	⇒	0
0	0	0	0	0	0	0	0	0		0
0	0	0	0	0	0	0	0	0		0

At the end of the fourth cycle is $\underline{D} = 0$. Therefore the computation stops. It means that artificial neural network is learned and ready to run in using regime. The architecture of the associated Boolean neural network is shown in figure 3.

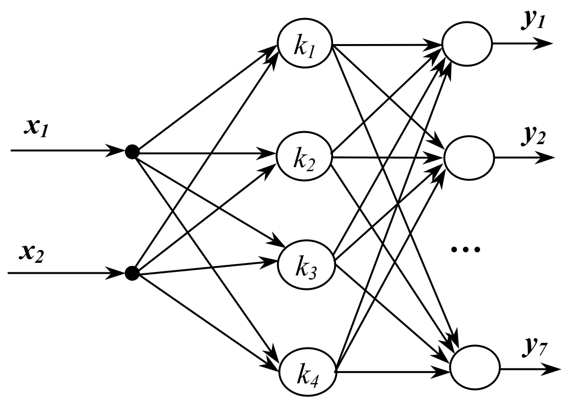

Fig. 3. Structure of Boolean neural network to represent set of Boolean functions.

The number of neurons on the hidden layer is equal to number of executed cycles in training algorithm. On the output layer we have 7 neurons. Each of them represents one Boolean functions in the set.
In consequence of learning procedure, we have obtained the matrix K and matrix AM . The columns 1 and 2 of matrix AM are unnecessary because the neural network will represent in using regime only output values of the Boolean functions $y_1, y_2, ..., y_7$. Thus, storage of these vectors is senseless.

For the verification of using algorithm of neural network, calculate the output signals of the initial set of Boolean functions $y_1, y_2, ..., y_7$ by the equation (11). The result is shown in table 6.

For example the value $y_3(x_1 = 0, x_2 = 1) =$

$$= \left(k_1(0,1) \wedge am_5(1)\right) \oplus \left(k_2(0,1) \wedge am_5(2)\right) \oplus$$
$$\oplus \left(k_3(0,1) \wedge am_5(3)\right) \oplus \left(k_4(0,1) \wedge am_5(4)\right) = (1 \wedge 0) \oplus$$
$$\oplus (1 \wedge 1) \oplus (0 \wedge 1) \oplus (0 \wedge 1) = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

The calculation of $y_3(0,1)$ is highlighted in the table 6 an explicitly, give in (11).

Table 6: Reconstruction of the set of Boolean functions

K

k_1	k_2	k_3	k_4
1	1	1	0
1	1	0	0
1	1	1	1
1	0	0	0

AM

	am_3	am_4	am_5	am_6	am_7	am_8	am_9
1	0	0	0	1	0	1	1
2	0	1	1	1	0	1	0
3	0	1	1	1	1	1	0
4	1	0	1	1	0	0	1

Eq. (11)

x_1	x_2	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	0	0	1	1	1	1
0	1	0	1	1	0	0	0	1
1	0	1	0	1	0	1	1	0
1	1	0	0	0	1	0	1	1

The Boolean neural network is now ready for work.

6 Conclusion and Future Work

Neural network models are widely used to solve tasks on real and rarely integer numbers. Due to smaller errors multi-byte data representation are preferred. Even the smallest computing error is illegal for Boolean logic. The direct using of the typical types of the neural networks for the tasks based on Boolean logic needs inadmissible size of memory. As result of this analysis, we found, that the handling of Boolean function needs special Boolean neural networks.

We introduced Boolean neuron and its application in Boolean neural networks (BNN). In detail, we described the mathematic definition of the Boolean neuron, the structure of Boolean neural network, the training algorithm for such a neural network and the general using algorithm of BNNs. The main benefits of such Boolean neurons are minimizing of the necessary memory, the decreasing of the calculation time and their simple possible mapping to the basic elements of FPGAs.

Suggested algorithms use common learning and using principles of the neural networks and especially non-iteration learning of feed-forward neural networks. Our approach can be used for other structures of neural network, too.

In the future, we are going to develop the neural net structure for clusterization-classification Boolean functions on the base proposed Boolean neuron. Further, we continue our research for optimal Boolean functions presentation and calculation by using neural nets. Based on the results described in this paper we will design and develop a program, which used Boolean neural networks for compact presentation, decomposition and fast calculation of Boolean functions.

References:

[1] Dayhoff, Judith E. Neural networks architectures: an introduction. - Van Nostrand Reinhold, New York, 1990.- 260 p.

[2] Di Wang and Narendra S. Chaudhari, Binary Neural Network Training Algorithms Based On Linear Sequential Learning, Intarnational Journal of Neural Systems, 13(5) Oct. 2003

[3] Di Wang and Narendra S. Chaudhari, An Approach for Construction of Boolean Neural Networks Based on Geometrical Expansion. (www.ntu.edu.sg/home/asnarendra)

[4] Lauria, F. E., M. Sette, S. Visco Adaptable Boolean neural networks - Consorzio Editoriale Fridericiana, Liguori ed. Napoli -1997 212 ISBN 88-207-2676-9

[5] McCulloch W.S., Pitts W. A logical calculus of ideas immanent in nervous activity, Bull. Mathematical Biophysics. - 1943. - vol. 5. - pp. 115-133.

[6] Minsky M. and Papert S. Perceptrons: An Introduction to Computational Geometry. MIT Press, Cambridge, MA, 1969.

[7] Novikov, L., Obschta, A.: Numerical and approximate methods of applied mathematics. (in ukr.) Script of lectures. - Lviv: NU “Lviv politechnic”, 1998.

[8] Sigorsky, V.: Mathematical engineer device.. (in rus) - Kiev, 1975.

[9] Steinbach, B. Kohut, R.: Neural Networks – A Model of Boolean Functions. Proceedings of 5th International Workshop on Boolean Problems, Freiberg, Germany, September 19-20, 2002;

[10] Tkachenko, R.: Kohut, R.: Feed forward neural networks: the problems of synthesis and using. (In Ukrainian), Bulletin of Lviv Polytechnic National University: Computer Engineering and Information Technologies, № 433, Lviv, pp. 166-171, 2001.

[11] Tkachenko R.: Feed forward neural networks with non-iteration learning procedure. Dissertation thesis, Lviv, 2000.

[12] Vinay D.: Mapping Boolean Functions with Neural Networks having Binary Weights and Zero Thresholds, HP Laboratories Palo Alto, HPL-2001-64 (R.1), Reprinted, with permission, from IEEE Transactions on Neural Networks, Copyright 2001 IEEE, July, 2001

[13] Wasserman, P.: Neural Computing Theory and Practice. Van Nostrand Reinhold, New York, 1989.