

Decomposition of Boolean Function Sets for Boolean Neural Networks


Roman Kohut, Bernd Steinbach

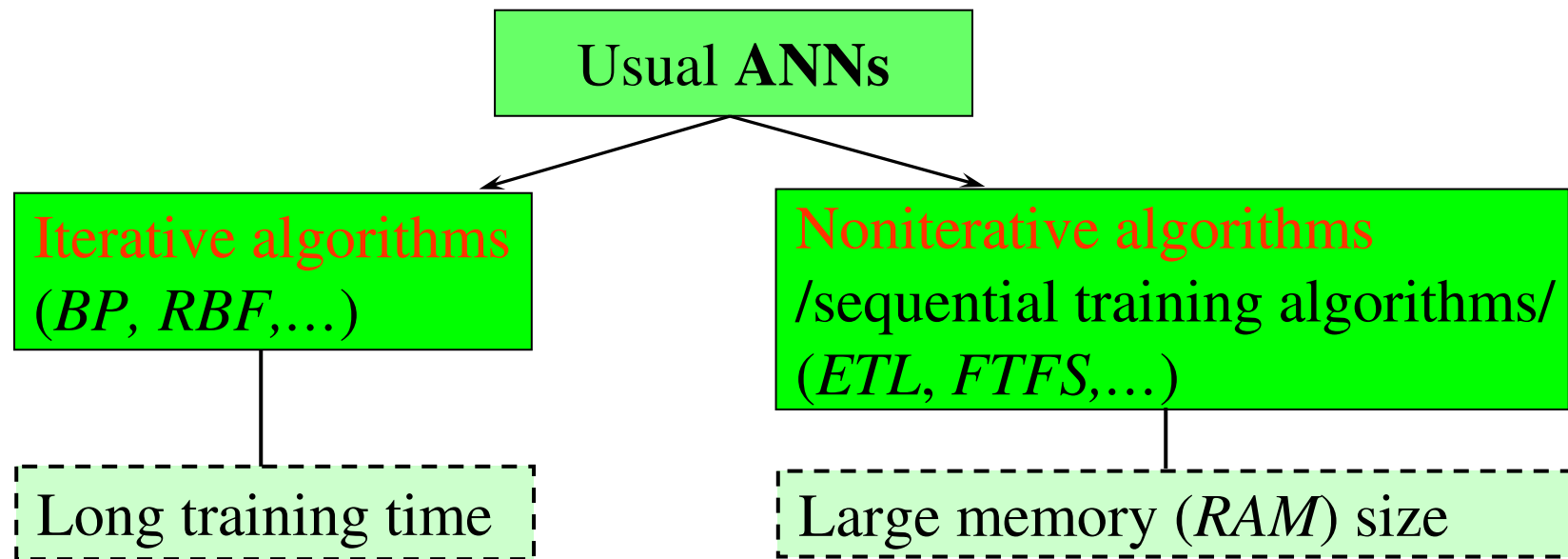
Freiberg University of Mining and Technology
Institute of Computer Science
Freiberg (Sachs), Germany

Outline

- Introduction
- Boolean Neuron
- AND-Decomposition
 - Training of the Boolean Neural Network (BNN)
 - Using the BNN
- Example
- Conclusion

Introduction

- The aims: **compact representation** and **fast computation** of Boolean functions for artificial neural networks
- The Problem: $f: \mathbf{B}^n \rightarrow \mathbf{B}$  2^n different binary vectors



Boolean Neuron

- Definition of a Boolean Neuron

$$Out = f(Inp, w)$$



$$Out_B = f_B(Inp_B, w_B)$$

$$Inp_B = \{x_1, x_2, \dots, x_{N_x}\} \quad x_i \in \{0,1\}$$

$$w_B = \{w_1, w_2, \dots, w_{N_x}\} \quad w_i \in \{0,1\}$$

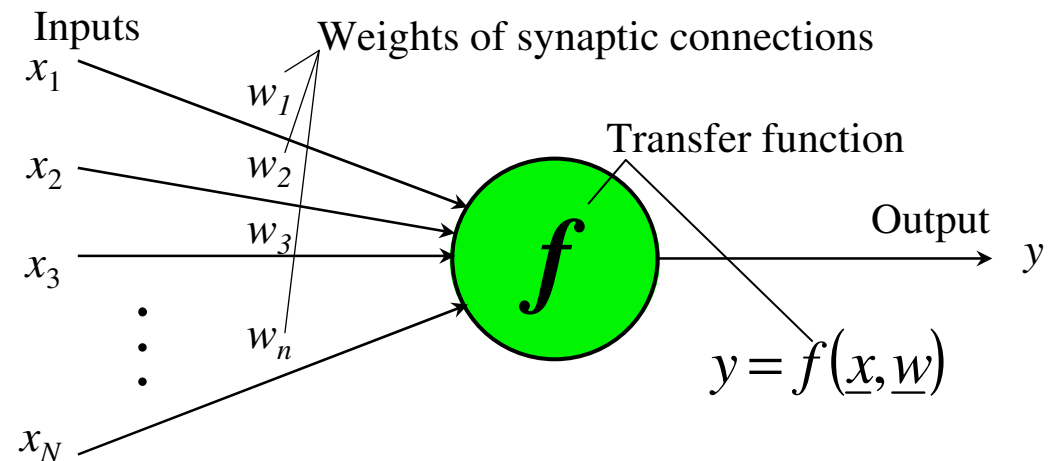
f_B - Boolean transfer function

Out_B - output signal

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

Advantages of the BN:

- speeding up of calculation significantly,
- reduction of necessary memory size,
- possibility to map the BN into CLB of FPGAs.



General structure of Boolean neuron

Boolean Neuron

- hidden neuron

$$Out^{[z]} = f^{[z]} \left(Inp_i \wedge w_i^{[z]} \right)$$

$Out^{[z]}$ – output signal of the neuron with number z ,

$f^{[z]}$ – transfer function of the neuron with number z ,

$[z]$ – index $z = 1, \dots, Z_N$,

Z_N – number of neurons on the hidden layer,

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

- output neuron

$$Out^{[j]} = f \left(Inp_i \wedge w_i^{[j]} \vee \overline{w_i^{[j]}} \right)_{i=1}^{Z_n}$$

$f \in \{\text{AND, "equivalence"}\}$

$$Out^{[j]} = f \left(Inp_i \wedge w_i^{[j]} \right)_{i=1}^{Z_n}$$

$f \in \{\text{OR, EXOR}\}$

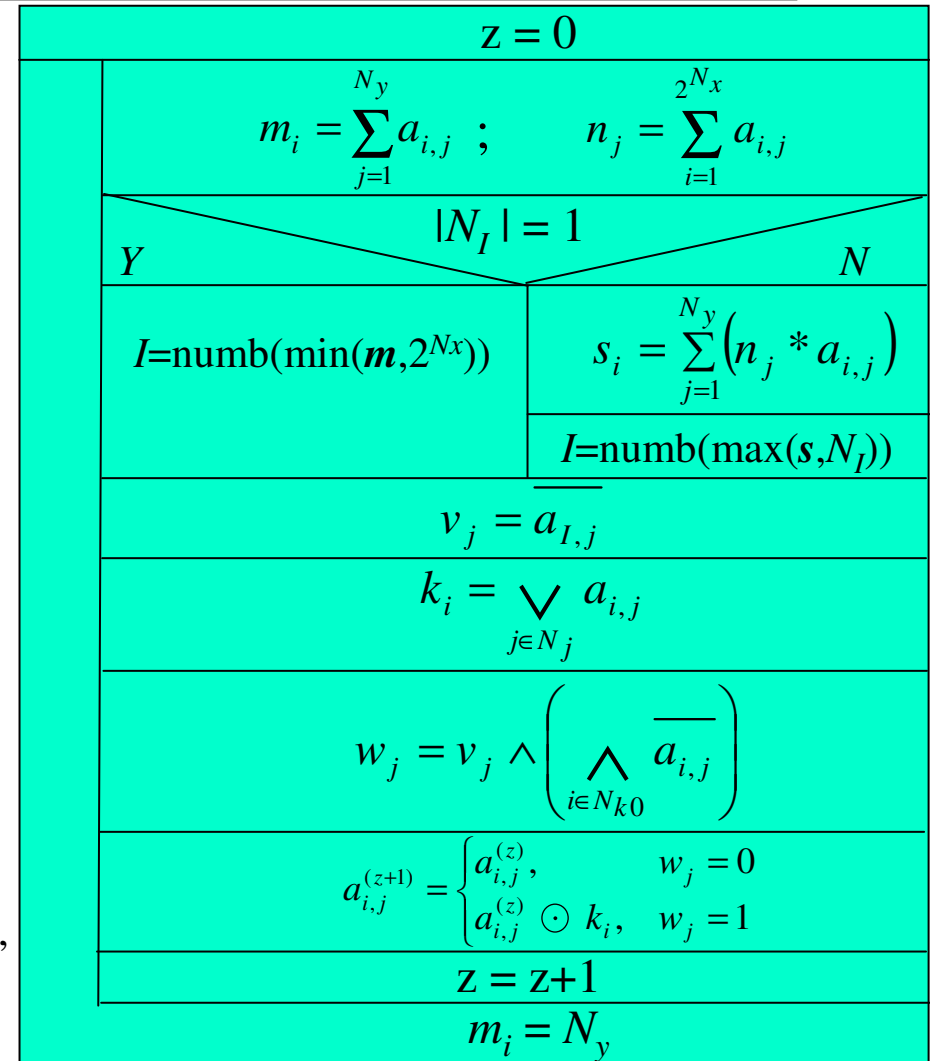
$[j]$ – number of neuron on the output layer of the BNN

AND-Decomposition - Training of the BNN

• Training algorithm

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

- z – number of hidden layers,
- $a_{i,j}$ – coefficient of matrix A ,
- N_x – number of Boolean variables,
- N_y – number of Boolean functions,
- m_i, n_j – weights values
- I - index of base row of matrix A ,
- N_I – set of numbers of minimal values of the vector m ,
- s_j – element of auxiliary vector s ,
- v_j – element of base row v of matrix A ,
- k_i – value of transfer function k ,
- N_j – set of column numbers j with $v_j = 1, n_j = \max(v, N_y)$,
- w_i – synaptic weight,
- N_{k0} – set of row numbers i with $k_i = 0$.



AND-Decomposition - Using of the BNN

Results of the training:

- unique transfer functions of the hidden neurons: $k_z = f(\mathbf{x})$,
- weights $w_{z,j}$ of the output neurons,
- the structure of BNN:
 - *three-layer architecture* (fix),
 - N_x *input neurons*,
 - N_y *output neurons*,
 - Z_N *hidden neurons*.

Type of decomposition: “AND”



$$y(\mathbf{x}) = \bigwedge_{z=1}^{Z_N} \left(w_{z,j} \wedge k_z(\mathbf{x}) \vee \overline{w_{z,j}} \right)$$

Example

- Training process

y_1, y_2, \dots, y_{10} - Boolean functions, $y_i = f(x_1, x_2, x_3)$

Primary matrix A

x_1	x_2	x_3	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	...
0	0	0	1	1	0	0	1	1	0	0	0	1	5
0	0	1	0	0	0	0	1	0	0	0	0	1	2
0	1	0	1	1	0	1	0	0	1	1	0	0	3
0	1	1	0	0	1	0	1	0	0	1	1	1	5
1	0	0	1	0	1	1	0	1	0	0	0	1	5
1	0	1	0	0	0	0	0	0	0	1	0	0	1
1	1	0	0	0	1	0	1	0	0	1	1	1	5
1	1	1	1	1	0	1	0	0	1	1	0	0	1

$$a_{i,j}^{(z+1)} = \begin{cases} a_{i,j}^{(z)} & w_j = 0 \\ a_{i,j}^{(z)} \cdot k_i & w_j = 1 \end{cases}$$

$$k_i = \bigvee_{j \in N} a_{i,j}$$

Matrix A after the first cycle of training

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	m
1	1	0	0	1	1	0	0	0	1	5
0	0	0	0	1	0	0	0	0	1	2
1	1	1	1	1	1	1	1	1	1	10
0	0	1	0	1	0	0	1	1	1	5
1	0	1	1	0	1	0	0	1	1	5
0	0	1	0	0	1	1	1	1	1	6
0	0	1	0	1	0	1	1	1	1	5
1	1	1	1	1	1	1	1	1	1	10

$$w_j = \bigvee_{i \in N_k} a_{i,j}$$

$$w_1 = 0$$

$$w_2 = 0$$

$$w_3 = 1$$

$$w_4 = 1$$

$$w_5 = 1$$

$$w_6 = 1$$

$$w_7 = 1$$

$$w_8 = 1$$

$$w_9 = 1$$

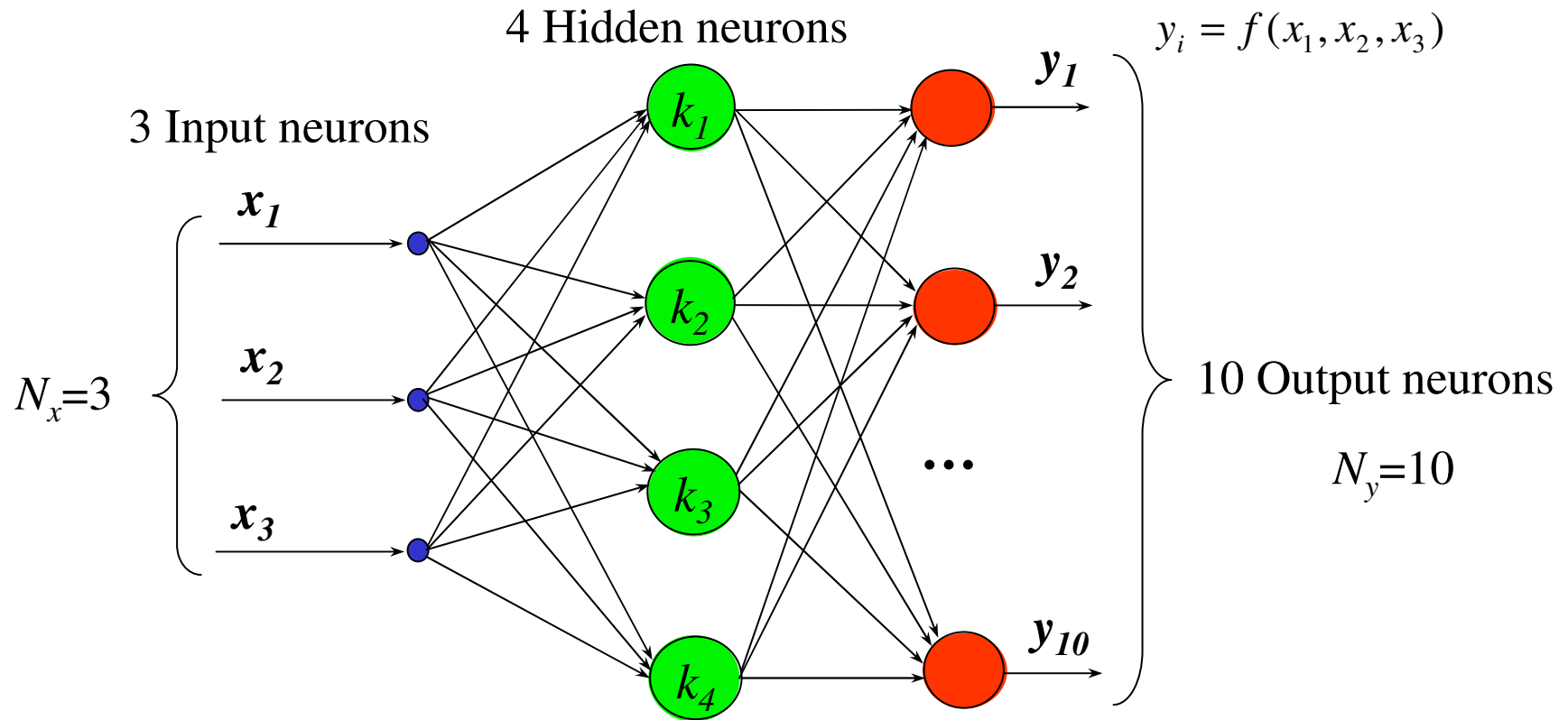
$$w_{10} = 1$$

$$v_j = a_{1,j}$$

$$A = \sum_{i=1}^n a_i$$

Example

- Structure of BNN



BNN to represent the set of Boolean function y_1, y_2, \dots, y_{10}

Example

- Reconstruction of the set of Boolean functions

Out: **1 2 3 4 5 6 7 8 9 10**

w_1	0	0	1	0	1	1	0	0	1	1
w_2	0	0	1	1	0	0	1	1	1	0
w_3	0	1	0	0	1	0	1	1	1	0
w_4	1	1	0	1	0	1	1	0	0	0

$$y(\mathbf{x}) = \bigwedge_{z=1}^{z_N} (w_{z,j} \wedge k_z(\mathbf{x}) \vee \overline{w_{z,j}})$$

x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

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

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}
1	1	0	0	1	1	0	0	0	1
0	0	0	0	1	0	0	0	0	1
1	1	0	1	0	0	1	1	0	0
0	0	1	0	1	0	0	1	1	1
1	0	1	1	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0
0	0	1	0	1	0	0	1	1	1
1	1	0	1	0	0	1	1	0	0

Conclusion

Results:

- (1) formal definition of the Boolean neuron
- (2) simple mapping to the CLBs of FPGAs
- (3) application of Boolean neurons in Boolean neural networks (BNN), that realize a set of Boolean functions
- (4) decomposition of a set of Boolean functions into common basic functions
- (5) structures of the BNN for AND, OR, XOR and equivalence-decomposition of a set of Boolean functions

Further work:

- optimization of the BNN using mixed types of output neurons
- developing a program that uses Boolean neural networks for compact presentation and fast calculation of Boolean functions