2004 International Symposium on Nonlinear
Theory and its Applications (NOLTA2004)
Fukuoka, Japan, Nov. 29 - Dec. 3, 2004

A GA-based Fault-Containment Learning Algorithm for Binary Neural Networks

Takashi Yamamichi†, Toshimichi Saito† and Hiroyuki Torikai†

†EECE Dept, Hosei University, kajino-cho, Koganei, Tokyo,184-8584 Japan
Email: yamamichi@nonlinear.k.hosei.ac.jp, {tsaito, torikai}@k.hosei.ac.jp

Abstract—This paper presents a novel learning algorithm for the binary neural network that has the unit step activation function. The remarkable properties are binary weight vectors in the hidden layer and fault-containment. As teacher signals are given from a Boolean function, our algorithm determines hidden neurons based on the genetic algorithm where the chromosomes correspond to the binary weight vectors of hidden layer and the fitness corresponds to the number of separated teacher signals. The fault-containment is effective to reduce the rate of hidden neurons and smart BNNs can be synthesized.

1. Introduction

The binary neural network (ab. BNN) is a simple feed-forward network whose activation is represented by the unit step function [1]-[4]. The BNN can approximate a desired Boolean function and has many applications including pattern classification and error correcting codes [5] [6]. In order to synthesize the BNN, there exit several supervised learning algorithms including expand-and-truncate learning (ab. ETL) based on geometrical structure of teacher signal space [2]. The ETL and its improved version [?] are much simpler than the back-propagation algorithm that is not suitable for the BNN but for smooth feed forward networks.

This paper presents a novel learning algorithm (ab. FCLA) for the BNN: the remarkable properties are binary weight vectors in the hidden layer and fault-containment. As teacher signals are given from a Boolean function, the FCLA determines hidden neurons based on the genetic algorithm (ab. GA), where the chromosomes correspond to the binary weight vectors and the fitness corresponds to the rate of separated teacher signals into all signals. We then introduce a fault-containment parameter E in order to remove a few noisy teacher signals by permitting false separation. Even though the weight vector is binary, the fault-containment is effective to reduce the number of hidden neurons and the FCLA can synthesize smart BNNs. The efficiency of the FCLA is confirmed by basic numerical experiments. Note that our previous GA-based algorithm [7] has neither the binary hidden weighs vectors nor the fault-containment parameter.

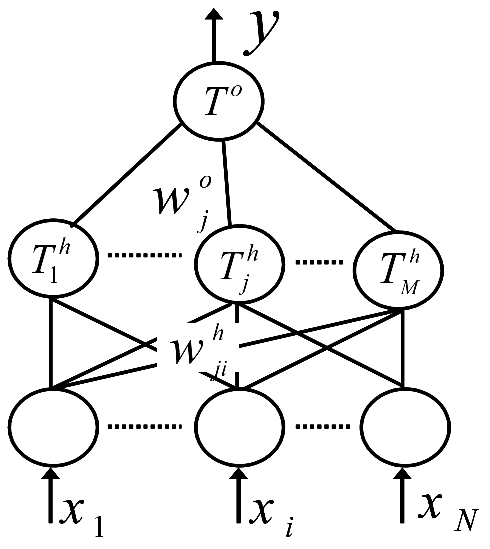

Figure 1: Three-Layer Binary Neural Networks

2. Binary Neural Networks

Fig. 1 shows the three-layer binary neural networks (ab. BNN) that can approximate a desired Boolean function F_B from N-dimensional binary space $\boldsymbol{B}^N \equiv \{0,1\}^N$ to 1-dimensional binary space $\boldsymbol{B} \equiv \{0,1\}$. The BNN is described by Equation (1).

$$\begin{aligned} &y = f_B(\boldsymbol{x}), \;\; F_B = \boldsymbol{B}^N \rightarrow \boldsymbol{B} \\ &y = U\left(\sum_{j=1}^{M} W^o_j z_j - T^o\right) \\ &z_j = U\left(\sum_{i=1}^{N} W^h_{ji} x_i - T^h_j\right) \end{aligned} \tag{1}$$

where $\boldsymbol{x} = (x_1, \cdots, x_N) \in \boldsymbol{B}^N$ is the input vector, $\boldsymbol{z} = (z_1, \cdots, z_M) \in \boldsymbol{B}^M$ is the hidden layer output vector and $y \in \boldsymbol{B}$ is the output, where M is the number of hidden layer neurons. U is the unit step function

$$U(X) = \begin{cases} 1 & \text{for } X \geq 0 \\ 0 & \text{for } X < 0 \end{cases}$$

$\boldsymbol{W}^o = (W_1^o, \cdots, W_M^o)$ is the output weight vector, and $\boldsymbol{W}_j^h = \left(W_{j1}^h, \cdots, W_{jN}^h\right)$ is the weight vector of the hidden layer. In this paper, let $W_{ji}^h \in \{-1, 1\}$. This binary setting of W_{ji}^h is much simpler than usual integer setting. $T^o \in \boldsymbol{Z}$ is the threshold of output layer and $T_j^h \in \boldsymbol{Z}$ is the threshold of jth hidden neuron where $\boldsymbol{Z}$ denotes integers.

3. Learning Algorithm

Our leaning algorithm FCLA determines the hidden layer parameters $\boldsymbol{W}_j^h$, T_j^h and the output layer parameters $\boldsymbol{W}^0$, T^0, which based on teacher signals from a desired Boolean function F_B. Let $\boldsymbol{U} = \{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_{N_u}\}$ be a subset of teacher signals such that $f_B(\boldsymbol{u}_j) = 1$. Let $\boldsymbol{V} = \{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_{N_v}\}$ be a subset of teacher signals such that $f_B(\boldsymbol{v}_j) = 0$. Let $\boldsymbol{u}_j = \{\boldsymbol{u}_{j1}, \cdots, \boldsymbol{u}_{jN}\}$ and $\boldsymbol{v}_j = \{\boldsymbol{v}_{j1}, \cdots, \boldsymbol{v}_{jN}\}$, where $u_{ji} \in \boldsymbol{B}$ and $v_{ji} \in \boldsymbol{B}$. Since a teacher signal correspond to a vertex of an N-dimensional hypercube, we then refer to an element of $\boldsymbol{U}$ (respectively, $\boldsymbol{V}$) as true vertex (respectively, false vertex). Our algorithm consists of the following 5 steps and one subroutine.

Main routine

STEP 1 : Initialization.
Let j be an index of a hidden neuron and let $j = 1$.

STEP 2 : Determination of separated hyperplane
Using a GA-based subroutine shown afterward, we determine j-th separated hyperplane (ab. SHP) corresponding to the j-th hidden neuron. The j-th SHP is represented by weight vector $\boldsymbol{W}_j^h$ and threshold of the hidden neuron T_j^h. In the GA-based subroutine, the chromosome corresponds to the weight vector $\boldsymbol{W}_j^h$ and the fitness L_j corresponds to the rate of true vertices separated by the jth SHP over all the 2^N vertices. The threshold T_j^h is determined with the fitness as shown in the GA-based subroutine. The SHP can separate true vertices and our algorithm adopts a fault-containment principle: the SHP also separate a few noisy false vertices as shown in Fig. 2. Let $E \cdot 2^N$ be the upper limit number of the false separation. We refer to E as the fault-containment parameter.

STEP 3 : Inspection of the j-th SHP
If $L_j > E$ then true (and false) vertices separated by the j-th SHP are declared as "*don't care*", the jth SHP is adopted and goto STEP 4. If $L_j \leq E$ then true vertices separated by the j-th SHP are declared as "*don't care*", the jth SHP is not adopted and go to STEP 4.

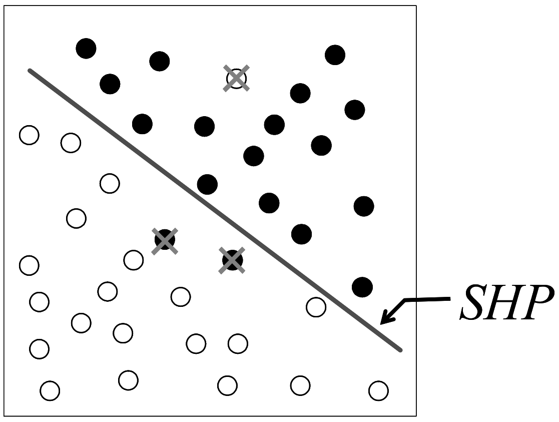

Figure 2: Separated Hyper Plane (SHP)

STEP 4 :
Let γ be the rate of true vertices over the number of all vertices 2^N. If $\gamma \leq E$ then goto STEP 5. Otherwise, j=j+1 and goto STEP 2.

STEP 5 : Learning the output layer
The output neuron is determined by $T^o = 1$ and $W_j^o = 1$:

$$y = U\left(\sum_{j=1}^{M} z_j - 1\right) = \begin{cases} 0 & \text{if } z_j = 0 \text{ for all j} \\ 1 & \text{otherwise} \end{cases}$$

That is, the output neuron can approximate the teacher signals by "OR" operation of the hidden neuron outputs. If $E = 0$, the exact realization of F_B is possible.

GA-based Subroutine

We use a set of chromosomes $\{\boldsymbol{C}_1, \cdots, \boldsymbol{C}_K\}$ where $\boldsymbol{C}_l$=$(C_{l1}, \cdots, C_{lN})$, $C_{li} \in \{-1, 1\}$. The l-th chromosome $\boldsymbol{C}_l$ corresponds to the lth hidden weight vector $\boldsymbol{W}_l^h$. For l-th chromosome $\boldsymbol{C}_l$, we define the following function to estimate the fitness.

$$h(\boldsymbol{x}) = U\left(\sum_{i=1}^{N} \boldsymbol{C}_{li}\boldsymbol{x}_i - D\right)$$

where D is an integer parameter corresponding to the threshold T_j^h. The fitness L_j is the rate of number for true vertices separated by the hyper plane $\boldsymbol{C}_j \cdot \boldsymbol{x} - D$ with fault-containment parameter E:

$$L_j = \frac{\#\{\boldsymbol{u}|h(\boldsymbol{u}) = 1\}}{2^N} \quad \text{where} \quad \frac{\#\{\boldsymbol{v}|h(\boldsymbol{v}) = 1\}}{2^N} \leq E.$$

Table 1: teacher signals

42 true vertices
0000000, 0000011, 0000101, 0000110, 0001001, 0001010
0001100, 0010100, 0011001, 0011010, 0011100, 0100111
0101011, 0101101, 0101110, 0101111, 0110101, 0110110
0110111, 0111011, 0111101, 0111110, 1000111, 1001011
1001101, 1001110, 1001111, 1010011, 1010101, 1010110
1010111, 1011000, 1011011, 1011101, 1101000, 1101010
1110000, 1110111, 1111000, 1111011, 1111101, 1111111
42 true vertices
0000001, 0000010, 0000100, 0000111, 0001000, 0001011
0001101, 0001110, 0010000, 0010001, 0010010, 0010101
0010110, 0011000, 0011011, 0011101, 0011110, 0100000
0100011, 0100101, 0100110, 0101001, 0101010, 0101100
0110100, 0111000, 0111001, 0111010, 0111100, 1000000
1000011, 1000101, 1000110, 1001001, 1001010, 1001100
1010100, 1011001, 1011010, 1011100, 1111001, 1111010

Changing D from $-N$ to N monotonically, we find a value of D that gives the maximum value of L_j. If there exist plural values of D which give the maximum value of L_j we adopt the middle of the maximum and minimum values: $D = \frac{1}{2}(D_{max} + D_{min})$. After Each chromosomes are evaluated by the fitness L_j, we apply the Elite strategy, roulette selection, single point crossover with probability Pc and normal mutation with probability Pm. We repeat such operations until the maximum generation T.

4. Numerical Experiments

We apply the proposed FCLA to some numerical experiments, and compare the results to the ETL. In the numerical experiments, we fix the GA-parameters

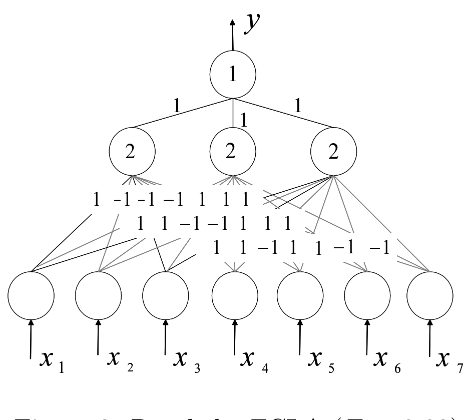

Figure 3: Result by FCLA ($E = 0.03$)

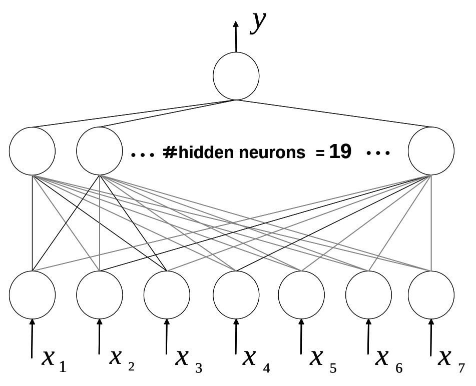

Figure 4: Result by ETL

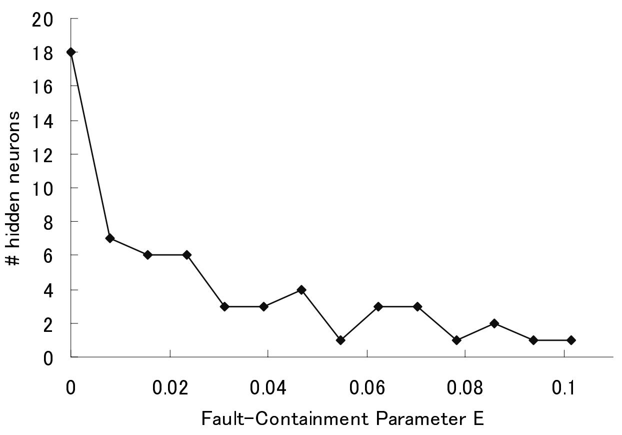

Figure 5: Role of fault-containment parameter E

as the following after trials and errors.

$$(Mg, Pc, Pm, T) = (7, 0.7, 0.09, 10)$$

Experiment 1: A 7-Bit Function.
We apply the FCLA to a 7-bit function F_7. Table 1 shows the teacher signals: the number of true vertices is 42, the number of false vertices is 42, and the remained 44 vertices are "don't care".

Fig. 3 shows an example of the results for $E = 0.03$: F_7 can be approximated by the BNN having three hidden neurons and each weight parameters W^h_{ji} is 1 or -1. Fig. 4 shows the result by the ETL: the BNN has 19 hidden neurons where the weight parameters W^h_{ji} take various integer values. Table 2 shows the results by FCLA for $E \in \{0.00, 0.01\}$ and the result by ETL. In the table, column AV and SD shows the average value and standard deviation of the parameters: the hidden weights W^h_{ji}, output weights W^o_j and hidden thresholds T^h_j. The output threshold $T^o = 1$ for all the algorithms. In this table, we can see that the FCLA can provide much smaller number of hidden neurons and much lower SD of parameters than

Table 2: Comparison of FCLA and ETL

	Hidden Layer						Output Layer		
	E	**N**	W_{ji}^h		T_j^h		W_j^o		T^o
			AV	**SD**	**AV**	**SD**	**AV**	**SD**	
ETL		**19**	**0.26**	**8.79**	**9.47**	**16.8**	**136**	**1212**	**1**
FCLA	**0.00**	**18**	**- 0.11**	**0.99**	**2.67**	**1.20**	**1**	**0**	**1**
FCLA	**0.01**	**7**	**0.27**	**0.97**	**3.14**	**1.73**	**1**	**0**	**1**

E : Fault - Containment Parameter
N : hidden neuron's number

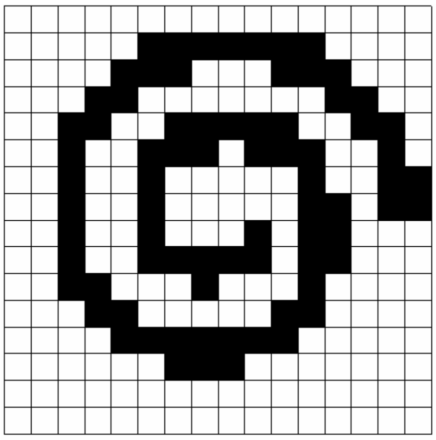
Figure 6: two spirals

the ETL. Fig. 5 shows characteristic of the number of hidden layer neurons for fault-containment parameter E: we can see that E is a suitable selection in this experiment.

Experiment 2: Two Spirals problem.
The two spirals problems require a highly nonlinear classification. It is an extremely hard problem [8]. We then apply the FCLA to teacher signals of two spirals in Fig. 6. The number of true vertices is 180 and the number of false vertices is 76. Fig.7 shows the number of hidden neurons for the fault-containment parameter E.

5. Conclusions

We have considered a simple learning algorithm for the BNN. The algorithm has fault-containment and determines hidden neurons based on GA . Even though the weight vector is binary, the fault-containment is effective to reduce the number of hidden neurons and smart BNNs can be constructed. Future problems include analysis of learning process, automatic setting of learning and GA parameters, comparison with other digital systems and applications to practical problems.

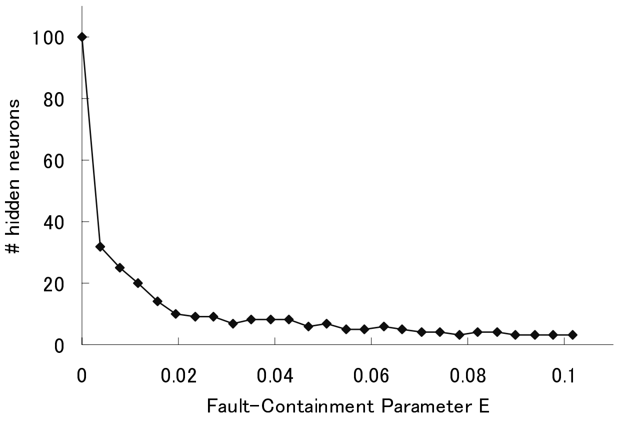

Figure 7: Role of fault-containment parameter E

References

[1] D. L. Gray and A. N. Michel, A training algorithm for binary feed forward neural networks, IEEE Trans. Neural Networks, vol.3, no.2, pp.176-194, March 1992.

[2] J. H. Kim and S. K. Park, The geometrical learning of binary neural networks, IEEE Trans. Neural Networks, vol.6, no.1, pp.237-247, Jan. 1995.

[3] M.Muselli, On sequential construction of binary neural networks, IEEE Trans. Neural Networks, vol.6, no.3, pp.678-690, May 1995.

[4] A.Yamamoto and T.Saito, A flexible learning algorithm for binary neural networks, IEICE Fundamentals, vol.E81-A, no.9, pp. 1925-1930, Sep. 1998.

[5] S. Gazula and M. R. Kabuka, Design of supervised classifiers using Boolean neural networks, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 17, no. 12, pp. 1239 - 1246, Dec. 1995.

[6] X-A. Wang and S. B. Wicker, An artificial neural net Viterbi decoder, IEEE Trans. Commun. vol. 44, no. 2, pp. 165-171, Feb. 1996.

[7] M. Shimada and T. Saito, A GA-based learning algorithm for binary neural networks, IEICE Trans. Fundamentals, E85-A, pp. 2544-2546, 2002

[8] F.M. Frattale Maxcioli and G. Martinelli, A Constructive Algorithm for Binary Neural Networks: The Oil-Spot Algorithm, IEEE Trans. Neural Networks, vol.6, NO3, pp. 794-797, MAY. 1995