THREE-VALUED NEURAL NETWORKS FOR TEST GENERATION

Hideo Fujiwara

Reprinted from PROCEEDINGS OF THE TWENTIETH INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING, Newcastle upon Tyne, U.K., June 26-28, 1990

IEEE Computer Society
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1264

Washington, DC • Los Alamitos • Brussels • Tokyo

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

IEEE

IEEE COMPUTER SOCIETY

Three-Valued Neural Networks for Test Generation

Hideo FUJIWARA

Department of Computer Science
Meiji University
1-1-1 Higashi-mita, Tama-ku
Kawasaki, 214 Japan

Abstract:

An approach to automatic test generation using neural networks was proposed by Chakradhar et al. [1]. They formulated the test generation problem as an optimization problem which can be solved by Hopfield's binary neural networks where neurons take binary values either 0 or 1. In this paper we propose a three-valued (0, 1, and 1/2) neural network which is an extension of the binary Hopfield's model and show that the test generation problem can be solved by the three-valued model more effectively than by the binary one. In the three-valued model, the energy function of networks, hyperplanes of neurons, and update rules of neuron's states are extended so that the third value 1/2 can be treated well. It is proved that the proposed three-valued model always converges. To escape from local minima, an extension of Boltzmann machines is presented where the update rules are modified by introducing probabilities of neuron's states.

Keywords:

Boltzmann Machine, Neural Networks, Optimization Problem, Test Generation, Three-Valued.

I. Introduction

Neural networks have been applied to many different fields. Although there are many neural network models, Hopfield's model [2, 3] is attractive because the computational power and its speed was demonstrated by solving one of the NP-complete problems known as the traveling salesman problem [3].

In the field of test generation, Chakradhar et al. [1] proposed an approach to automatic test generation using neural networks. They formulated the test generation problem as an optimization problem which can be solved by Hopfield's binary neural networks [2] where neurons take binary state values either 0 or 1. Their approach using neural networks is radically different from the conventional algorithms such as the D-algorithm and Podem [4, 5]. Indeed it is difficult to put the approach using neural networks to practical use right away, however when large scale neural networks become a reality with advances in technology, it might provide an advantage over the conventional methods.

The purpose of this paper is to extend the ideas of Chakradhar et al. [1] and to explore new possibilities of solving computationally difficult problems on 3-valued neural networks where neurons take state values from the set {0, 1, 1/2}. We propose a three-valued neural network model which is an extension of the binary Hopfield's model and show that the test generation problem can be solved by the three-valued model more effectively than by the binary one. In this three-valued model, the energy function of the network, hyperplanes of neurons, and update rules of neuron's states are extended so that the third value 1/2 can be treated well. It is proved that the proposed three-valued model always converges. To escape from local minima, an extension of Boltzmann machines is presented where the update rules are modified by introducing probabilities of neuron's states.

II. Chakradhar's Approach

First, we shall introduce briefly the approach of Chakradhar et al. [1] in this section.

2.1 Hopfield's Binary Model

A *neural network* is a collection of *neurons* interacting with each other. The behavior of a neural network is completely determined by the specification of the interaction. Let V_i denote the *state* of neuron i, i.e., $V_i \in \{0, 1\}$ for $i = 1, 2, ..., N$, where N is the number of neurons in the network. Let $V_i(t)$ denote the state of neuron i at moment t, and each neuron *updates* randomly in time its state according to the following equation:

$$V_i(t+1) = \mathrm{sgn}\left(\sum_{j=1}^{N} T_{ij} V_j(t) + I_i\right) \qquad (1)$$

CH 2877-9/90/0000/0064/$01.00 – 1990 IEEE

where sgn is defined as follows:

$$\text{sgn}(x) = \begin{cases} 1 & x > 0 \\ 0 & x \le 0 \end{cases}$$

where T_{ij} is the weight associated with the link between neurons i and j and I_i is the internal parameter of neuron i. Hopfield [2] has shown that if $T_{ij} = T_{ji}$ for all i and j and $T_{ii} = 0$ for all i, neurons always change their states in such a manner that they lead to stable states that do not further change with time and that they locally minimize an *energy function* defined by

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}V_iV_j - \sum_{i=1}^{N} I_iV_i + K \qquad (2)$$

where K is a constant.

2.2 Neural Networks for Logic Circuits

It is shown that an arbitrary logic circuit can be represented by a neural network [1]. Every net (signal line) in the circuit is represented by a neuron and the value on the net is the state value (0 or 1) of the neuron. Neural networks for 2-input AND, OR, NAND, NOR, XOR and XNOR gates and a NOT gate constitute the *basis set* and gates with more than two inputs are constructed from this basis set. A logic circuit is realized by specifying the matrix $T = [T_{ij}]$ and vector $I = [I_i]$ for the neural network. These T and I are determined so that the energy E of Eqn. 2 has global minima only at the neuron states consistent with the functionality of all gates in the circuit and all other inconsistent states have higher energy. In other words, the energy E is a non-negative constant Z for all *consistent* states and $E > Z$ for all *inconsistent* states.

Definition 1: Associated with each neuron i is a hyperplane $\sum_{j\neq i} T_{ij}V_j + I_i = 0$ in an n-1 dimensional space. Associated with each neuron i are three sets, P_{i_on}, P_{i_off} and P_{i_other} whose elements are points corresponding to consistent states of the network. A point belongs to P_{i_on} (P_{i_off}) if it corresponds to only one consistent state and neuron i has an state value 1 (0). P_{i_other} consists of all the points corresponding to consistent states but are not in the sets P_{i_on} or P_{i_off}.

Definition 2: A hyperplane $\sum_{j\neq i} T_{ij}V_j + I_i = 0$ associated with neuron i is a *decision hyperplane* if the points in P_{i_on} and P_{i_off} fall on opposite sides of the hyperplane and all points in P_{i_other} lie on the hyperplane.

Theorem 1 [1]: A necessary condition for the existence of a neural network of n neurons for a device with n terminals (with the energy function E defined in Eqn. 2) is the existence of a decision hyperplane for each of the n neurons.

Example 1: Figure 1 shows a 2-input NAND gate and the corresponding neural network. Associated with neuron 1 in the NAND gate are the sets $P_{1\_on} = \{(V_2=1,V_3=0)\}$, $P_{1\_off} = \{(V_2=1,V_3=1)\}$ and $P_{1\_other} = \{(V_2=0,V_3=1)\}$. Associated with neuron 3 are the sets $P_{3\_on} = \{(V_1=0,V_2=0), (V_1=1,V_2=0), (V_1=0,V_2=1)\}$, $P_{3\_off} = \{(V_1=1,V_2=1)\}$ and $P_{3\_other} = \{\}$.

From Theorem 1, the existence of a decision hyperplane for neuron 1 implies that $T_{12} + I_1 > 0$, $T_{12} + T_{13} + I_1 < 0$ and $T_{13} + I_1 = 0$. Similarly, a decision hyperplane for neuron 2 implies that $T_{12} + I_2 > 0$, $T_{12} + T_{23} + I_2 < 0$ and $T_{23} + I_2 = 0$. The decision hyperplane for neuron 3 implies that $I_3 > 0$, $T_{13} + I_3 > 0$, $T_{23} + I_3 > 0$ and $T_{13} + T_{23} + I_3 < 0$. Furthermore, the energy function E should be zero at all four consistent states ($V_1=V_2=0$, $V_3=1$), ($V_1=0$, $V_2=V_3=1$), ($V_1=V_3=1$, $V_2=0$) and ($V_1=V_2=1$, $V_3=0$). Therefore, the neural network model for the NAND gate should satisfy the above equalities and inequalities that are $K = T_3 > 0$, $I_3 > I_1 > 0$, $I_3 > I_2 > 0$, $T_{12} < 0$, $T_{13} < 0$, $T_{23} < 0$, $T_{13} + I_1 = 0$, $T_{23} + I_2 = 0$, and $T_{12} + I_1 + I_2 = I_3$. One solution is that $I_1 = I_2 = 2$, $I_3 = 3$, $T_{12} = -1$ and $T_{13} = T_{23} = -2$.

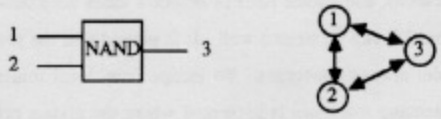


Figure 1. 2-input NAND and the Corresponding Neural Network

2.3 Test Generation Problem Formulation

Figure 2 illustrates a network which specifies constraints for test generation. This network is constructed by joining the good circuit and a faulty circuit in such a manner that the primary inputs of the two circuits are connected directly and that the primary outputs are connected through an *output interface* to include the constraint that at least one of the primary outputs of the faulty circuit will differ from the corresponding good circuit output. The neural network corresponding to this constraint network (the good circuit, the faulty circuit and the output interface) is used for generating a test vector for the fault. In this neural network, the output interface incorporates the constraint that at least one of the primary outputs of the faulty circuit will differ from the corresponding good circuit output. Therefore, if a test exists for a fault, there exists a *consistent labeling* of the neurons in the neural network with values from the set {0,1} that does not violate the functionality of any gate. In this way, test generation problem

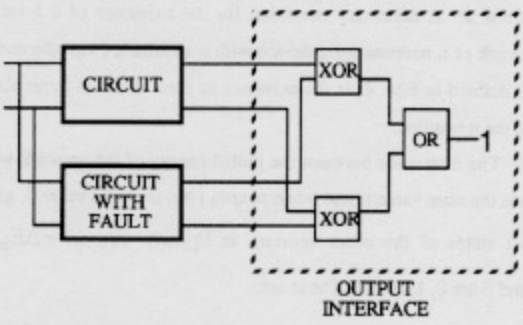


Figure 2. Constraint Network for Test Generation

can be formulated as an optimization problem such that the desired optima in the constraint neural network are the test vectors for a given fault.

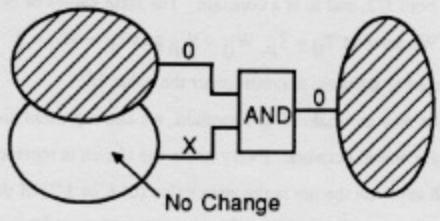
A local minimum of E is first obtained using a gradient descent approach where each neuron updates randomly in time its state according to the update rule of Eqn. 1. Although it is guaranteed that Hopfield's network always converges on the local minima of E, it is not guaranteed that it stabilizes at a global minimum [2]. Hence, to escape from these local minima, we modify the update rule through a probabilistic hill climbing technique [6, 7]. If the energy gap between the 0 and 1 states of the kth neuron is ΔE_k then the state value of the neuron is set to 1 with probability

$$p_k = \frac{1}{1 + e^{-\Delta E_k/T}} \qquad (3)$$

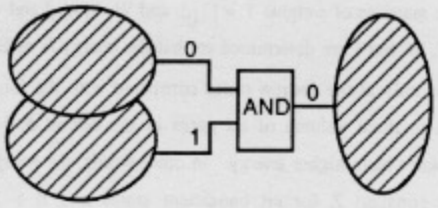
III. Three-Valued Neural Networks

As mentioned in the previous section, the problem of test generation is to find out a consistent labeling of the neurons in the constraint neural network with values from the set {0,1} that does not violate the functionality of any gate. However, searching with binary values involves a lot of wasteful assignments. For example, suppose that we have to set the value 0 on the output of an AND gate in Figure 3. If we search with values only from 0 or 1, we have to select one from three cases of the input combination, 00, 01, or 10. On the other hand, if we search with values from the set {0, 1, X} where X denotes *don't care*, we can select one from two cases, 0X or X0, which reduces the search space as shown in Figure 3 (a).

In Chakradhar's approach using the binary model, initial conditions of neural networks for test generation are determined so that every neuron has to be either 0 or 1. So, it often happens that many unnecessary values are assigned to neurons. In three-valued neural networks with values 0, 1, and X



(a) 3-valued model



(b) Binary model

Figure 3. Comparison of 3-valued and binary models

(or 1/2), initial conditions can be selected so that all neurons have the value 1/2 (don't care) except the fault site.

The purpose of introducing three values (0, 1, and 1/2) is to avoid unnecessary assignment of values 0 and 1 (pruning the search space), to obtain necessary and sufficient values to detect a given fault (minimal test vectors), and to speed up the convergence to the global minima.

3.1 Energy Function and Hyperplanes for 3-Valued Model

The *energy* function for three-valued neural networks is defined by the form:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}V_iV_j - \sum_{i=1}^{N} I_iV_i - \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}V_i(1 - V_i)V_j(1 - V_j) + K \qquad (4)$$

where N is the number of neurons in the neural network, T_{ij} is the weight associated with the link between neurons i and j, V_i is the state value of neuron i, I_i is the internal parameter of neuron i, W_{ij} is the weight

associated with the link between neurons i and j which is effective only when V_i and V_j are both 1/2, and K is a constant. The state values of neurons are 0, 1 and 1/2. We assume $T_{ij} = T_{ji}$, $W_{ij} = W_{ji}$ and $T_{ii} = W_{ii} = 0$. The third term is introduced to stabilize neurons under the value 1/2.

In the same way as Chakradhar's method, we can represent an arbitrary logic circuit by a neural network. Every net in the circuit is represented by a neuron and the value on the net is the state value (0, 1, or 1/2) of the neuron. Neural networks for 2-input AND, OR, NAND, NOR, XOR and XNOR gates and a NOT gate constitute the basis set and gates with more than two inputs are constructed from this basis set. A logic circuit is realized by specifying the matrices of weights $T = [T_{ij}]$ and $W = [W_{ij}]$ and vector $I = [I_i]$. These T, W and I are determined so that the energy E of Eqn. 4 has global minima only at the neuron states consistent with the functionality (with respect to three values) of all gates in the circuit and all other inconsistent states have higher energy. In other words, the energy E is a non-negative constant Z for all consistent states and E > Z for all inconsistent states.

Definition 3: Associated with each neuron i are three hyperplanes

(1) $E_{(V_i=0)} - E_{(V_i=1)} = \sum_{j\neq i} T_{ij}V_j + I_i = 0$

(2) $E_{(V_i=0)} - E_{\left(V_i=\frac{1}{2}\right)} = \frac{1}{2}\left(\sum_{j\neq i} T_{ij}V_j + I_i + \sum_{j\neq i} W_{ij}V_j(1 - V_j)\right) = 0$

(3) $E_{\left(V_i=\frac{1}{2}\right)} - E_{(V_i=1)} = \frac{1}{2}\left(\sum_{j\neq i} T_{ij}V_j + I_i - \sum_{j\neq i} W_{ij}V_j(1 - V_j)\right) = 0$

in an n-1 dimensional space. Associated with each neuron i are nine sets, $P_{i(0>1)}$, $P_{i(0<1)}$, $P_{i(0=1)}$, $P_{i(0>1/2)}$, $P_{i(0<1/2)}$, $P_{i(0=1/2)}$, $P_{i(1/2>1)}$, $P_{i(1/2<1)}$, and $P_{i(1/2=1)}$, whose elements are points corresponding to consistent states of the network. A point belongs to $P_{i(a>b)}$ ($P_{i(a<b)}$) if it corresponds to a consistent state when V_i=b (V_i=a) and an inconsistent state when V_i=a (V_i=b). A point belongs to $P_{i(a=b)}$ if it corresponds to a consistent state both when V_i=a and V_i=b.

Definition 4: A hyperplane $\sum_{j\neq i} T_{ij}V_j + I_i = 0$ associated with neuron i is a (0,1)-*decision hyperplane* if the points in $P_{i(0>1)}$ and $P_{i(0<1)}$ fall on opposite sides of the hyperplane and all points in $P_{i(0=1)}$ lie on the hyperplane.

A hyperplane $\sum_{j\neq i} T_{ij}V_j + I_i + \sum_{j\neq i} W_{ij}V_j(1 - V_j) = 0$ associated with neuron i is a (0,1/2)-*decision hyperplane* if the points in $P_{i(0>1/2)}$ and $P_{i(0<1/2)}$ fall on opposite sides of the hyperplane and all points in $P_{i(0=1/2)}$ lie on the hyperplane.

A hyperplane $\sum_{j\neq i} T_{ij}V_j + I_i - \sum_{j\neq i} W_{ij}V_j(1 - V_j) = 0$ associated with neuron i is a (1/2,1)-*decision hyperplane* if the points in $P_{i(1/2>1)}$ and $P_{i(1/2<1)}$ fall on opposite sides of the hyperplane and all points in $P_{i(1/2=1)}$ lie on the hyperplane.

Theorem 2: A necessary condition for the existence of a 3-valued neural network of n neurons for a device with n terminals (with the energy function E defined in Eqn. 4) is the existence of three decision hyperplanes for each of the n neurons.

Proof: The difference between the global energy of the network when neuron i has the state value α and when neuron i has the state value β, given the current states of the other neurons, is $E_{(V_i=\alpha)} - E_{(V_i=\beta)} = \Delta E_{i\alpha-\beta}$, where α and β are 0, 1, or 1/2. These are:

$$E_{(V_i=0)} - E_{(V_i=1)} = \Delta E_{i0\text{-}1} = \sum_{j\neq i} T_{ij}V_j + I_i$$

$$E_{(V_i=0)} - E_{\left(V_i=\frac{1}{2}\right)} = \Delta E_{i0\text{-}\frac{1}{2}} = \frac{1}{2}\left(\sum_{j\neq i} T_{ij}V_j + I_i + \sum_{j\neq i} W_{ij}V_j(1 - V_j)\right)$$

and

$$E_{\left(V_i=\frac{1}{2}\right)} - E_{(V_i=1)} = \Delta E_{i\frac{1}{2}\text{-}1} = \frac{1}{2}\left(\sum_{j\neq i} T_{ij}V_j + I_i - \sum_{j\neq i} W_{ij}V_j(1 - V_j)\right)$$

For an arbitrary point $p \in P_{i(\alpha>\beta)}$ ($P_{i(\alpha<\beta)}$) neuron i has an state value β (α) in the consistent state S_1 and α (β) in the inconsistent state S_2. The energy function E should have lower value of energy for the consistent state S_1 as compared to the inconsistent state S_2. Therefore, $\Delta E_{i\alpha-\beta}$ should necessarily be positive (negative). The hyperplane divides the n-1 dimensional space into two regions R1 and R2, and let this point p lie in the region R1 (R2). Therefore, for an E to exist all points in $P_{i(\alpha>\beta)}$ ($P_{i(\alpha<\beta)}$) must lie in region R1 (R2). For an arbitrary point $p \in P_{i(\alpha=\beta)}$, there correspond two consistent states S_1 and S_2 with neuron i having state values α and β, respectively. Since E should attain its minimum value, Z, in both the states, it is mandatory that $\Delta E_{i\alpha-\beta}$ be zero. Hence p must lie on the hyperplane for E to exist. Hence, for any given unit i, the existence of a (α,β)-decision hyperplane is a necessary condition for the existence of E.

Q.E.D.

Example 2: Let us consider again a 2-input NAND gate and the corresponding neural network shown in Figure 1. Associated with neuron 1 in the NAND gate are nine sets:

$P_{1(0>1)} = \{(V_2=1,V_3=0), (V_2=1/2,V_3=1/2)\}$

$P_{1(0<1)} = \{(V_2=1,V_3=1), (V_2=1/2,V_3=1)\}$

$P_{1(0=1)} = \{(V_2=0,V_3=1)\}$

$P_{1(0>1/2)} = \{(V_2=1,V_3=1/2), (V_2=1/2,V_3=1/2)\}$

$P_{1(0<1/2)} = \{(V_2=1,V_3=1), (V_2=1/2,V_3=1)\}$

$P_{1(0=1/2)} = \{(V_2=0,V_3=1)\}$

$P_{1(1/2>1)} = \{(V_2=1,V_3=0)\}$

$P_{1(1/2<1)} = \{(V_2=1,V_3=1/2)\}$ and

$P_{1(1/2=1)} = \{(V_2=0,V_3=1), (V_2=1/2,V_3=1/2)\}$.

From Theorem 2, the existence of three decision hyperplanes for each neuron are necessary for the existence of a 3-valued neural network for the NAND gate. Let us derive the conditions of those decision hyperplanes. From three sets, $P_{1(0>1)}$, $P_{1(0<1)}$ and $P_{1(0=1)}$, the existence of a (0,1)-decision hyperplane for neuron 1 implies that

$T_{12} + I_1 > 0$

$1/2(T_{12} + T_{13}) + I_1 > 0$

$T_{12} + T_{13} + I_1 < 0$

$1/2T_{12} + T_{13} + I_1 < 0$ and

$T_{13} + I_1 = 0.$

Similarly, from three sets, $P_{1(0>1/2)}$, $P_{1(0<1/2)}$ and $P_{1(0=1/2)}$, the existence of (0,1/2)-decision hyperplanes for neuron 1 implies that

$T_{12} + 1/2T_{13} + I_1 + 1/4W_{13} > 0$

$1/2(T_{12}+T_{13}) + I_1 + 1/4(W_{12}+W_{13}) > 0$

$T_{12} + T_{13} + I_1 < 0$

$1/2T_{12} + T_{13} + I_1 + 1/4W_{12} < 0$ and

$T_{13} + I_1 = 0.$

From three sets, $P_{1(1/2>1)}$, $P_{1(1/2<1)}$ and $P_{1(1/2=1)}$, the existence of (1/2,1)-decision hyperplanes for neuron 1 implies that

$T_{12} + I_1 > 0$

$T_{12} + 1/2T_{13} + I_1 - 1/4W_{13} < 0$

$T_{13} + I_1 = 0$ and

$1/2(T_{12}+T_{13}) + I_1 - 1/4(W_{12}+W_{13}) = 0.$

Associated with neuron 3 in the NAND gate are nine sets:

$P_{3(0>1)} = \{(V_1=0,V_2=0), (V_1=1,V_2=0), (V_1=0,V_2=1), (V_1=1/2,V_2=0), (V_1=0,V_2=1/2)\}$

$P_{3(0<1)} = \{(V_1=1,V_2=1)\}$

$P_{3(0=1)} = \{\}$

$P_{3(0>1/2)} = \{(V_1=1/2,V_2=1/2)\}$

$P_{3(0<1/2)} = \{(V_1=1,V_2=1)\}$

$P_{3(0=1/2)} = \{\}$

$P_{3(1/2>1)} = \{(V_1=0,V_2=0), (V_1=1,V_2=0), (V_1=0,V_2=1), (V_1=1/2,V_2=0), (V_1=0,V_2=1/2)\}$

$P_{3(1/2<1)} = \{(V_1=1/2,V_2=1/2)\}$

$P_{3(1/2=1)} = \{\}$

Similarly from three sets, $P_{3(0>1)}$, $P_{3(0<1)}$ and $P_{3(0=1)}$, the existence of a (0,1)-decision hyperplane for neuron 3 implies that

$I_3 > 0$

$T_{13} + I_3 > 0$

$T_{23} + I_3 > 0$

$1/2T_{13} + I_3 > 0$

$1/2T_{23} + I_3 > 0$ and

$T_{13} + T_{23} + I_3 < 0$

From three sets, $P_{3(0>1/2)}$, $P_{3(0<1/2)}$ and $P_{3(0=1/2)}$, the existence of (0,1/2)-decision hyperplanes for neuron 3 implies that

$1/2(T_{13} + T_{23}) + I_3 + 1/4(W_{13}+W_{23}) > 0$ and

$T_{13} + T_{23} + I_3 < 0$

From three sets, $P_{3(1/2>1)}$, $P_{3(1/2<1)}$ and $P_{3(1/2=1)}$, the existence of (1/2,1)-decision hyperplanes for neuron 3 implies that

$I_3 > 0$

$T_{13} + I_3 > 0$

$T_{23} + I_3 > 0$

$1/2T_{13} + I_3 > 0$

$1/2T_{23} + I_3 > 0$ and

$1/2(T_{13} + T_{23}) + I_3 - 1/4(W_{13}+W_{23}) < 0$

Furthermore, the energy function E should be zero at all nine consistent states ($V_1=V_2=0, V_3=1$), ($V_1=0, V_2=V_3=1$), ($V_1=V_3=1, V_2=0$), ($V_1=V_2=1, V_3=0$), ($V_1=0, V_2=1/2, V_3=1$), ($V_1=1/2, V_2=0, V_3=1$), ($V_1=1/2, V_2=1/2, V_3=1/2$), ($V_1=1, V_2=1/2, V_3=1/2$), and ($V_1=1/2, V_2=1, V_3=1/2$). These are

$K - I_3 = 0$

$K - T_{23} - (I_2+I_3) = 0$

$K - T_{13} - (I_1+I_3) = 0$

$K - T_{12} - (I_1+I_2) = 0$

$K - 1/2T_{23} - (1/2I_2 + I_3) = 0$

$K - 1/2T_{13} - (1/2I_1 + I_3) = 0$

$K - 1/4(T_{12} + T_{13} + T_{23}) - 1/2(I_1+I_2+I_3) - 1/8(W_{12}+W_{13}+W_{23}) = 0$

$K - (1/2T_{12} + 1/2T_{13} + 1/4T_{23}) - (I_1+1/2I_2+1/2I_3) - 1/8W_{23} = 0$

and

$K - (1/2T_{12} + 1/4T_{13} + 1/2T_{23}) - (1/2I_1+I_2+1/2I_3) - 1/8W_{13} = 0.$

Summarizing the above equalities and in equalities, we have that

$K = I_3 > 0$

$I_3 > I_1 > 0$

$I_3 > I_2 > 0$

$T_{12} < 0$

$T_{13} < 0$

$T_{23} < 0$

$T_{23} + I_2 = 0$

$T_{13} + I_1 = 0$

$T_{12} + I_1 + I_2 = I_3$

$W_{12} = 2T_{12}$

$W_{13} = -2T_{13}$ and

$W_{23} = -2T_{23}.$

Under these conditions, let us compute the energy for all inconsistent states.

$E(V_1, V_2, V_3) = E(0, 0, 0) = K = I_3 > 0$

$E(0, 1, 0) = K - I_2 = I_3 - I_2 > 0$

$E(1, 0, 0) = K - I_1 = I_3 - I_1 > 0$

$E(1, 1, 1) = K - (T_{12}+T13+T23) - (I1+I2+I3) = -T12 > 0$

$E(0, x, 0) = K - 1/2I_2 > 0$

$E(x, 0, 0) = K - 1/2I_1 > 0$

$E(0, x, x) = K - 1/4T_{23} - 1/2(I_2 + I_3) - 1/8W_{23} > 0$

$E(x, 0, x) = K - 1/4T_{13} - 1/2(I_1 + I_3) - 1/8W_{13} > 0$

$E(x, x, 0) = K - 1/4T_{12} - 1/2(I_1 + I_2) - 1/8W_{12} = 1/2I_3 > 0$

$E(x, x, 1) = K - (1/4T_{12} + 1/2T_{13} + 1/2T_{23}) - (1/2I_1 + 1/2I_2 + I_3) - 1/8W_{12} > 0$

$E(1, x, 0) = K - 1/2T_{12} - (I_1 + 1/2I_2) > 0$

$E(1, x, 1) = K - (1/2T_{12} + T_{13} + 1/2T_{23}) - (I_1 + 1/2I_2 + I_3) = -T_{12} > 0$

$E(x, 1, 0) = K - 1/2T_{12} - (1/2I_1 + I_2) > 0$

$E(x, 1, 1) = K - (1/2T_{12} + 1/2T_{13} + T_{23}) - (1/2I_1 + I_2 + I_3) > 0$

As seen from these computation, the energy E is positive for any inconsistent state.

One solution which satisfies the above equalities and inequalities is that $I_1 = I_2 = 2$, $I_3 = 3$, $T_{12} = -1$, $T_{13} = T_{23} = -2$, $W_{12} = -2$, and $W_{13} = W_{23} = 4$.

3.2 State Transition of Neurons

The state of an individual neuron i is updated as follows. Let V_i denote the state of neuron i, i.e., $V_i \in \{0, 1, 1/2\}$ for i = 1, 2, ..., N, where N is the number of neurons in the network. Let $V_i(t)$ denote the state of neuron i at moment t, and each neuron updates randomly in time its state according to the following rule:

State Update Rule:

In case of $\theta_i(t) > 0$

$$\begin{aligned} V_i(t+1) &= 1 && \text{if } U_i(t) \geq \theta_i(t) \\ &= 1/2 && \text{if } -\theta_i(t) < U_i(t) < \theta_i(t) \\ &= 0 && \text{if } U_i(t) \leq -\theta_i(t) \end{aligned} \tag{5}$$

In case of $\theta_i(t) \leq 0$

$$\begin{aligned} V_i(t+1) &= 1 && \text{if } U_i(t) > 0 \\ &= 0 && \text{if } U_i(t) < 0 \\ &= V_i(t) && \text{otherwise.} \end{aligned} \tag{6}$$

where

$$\begin{aligned} U_i(t) &= \sum_{j \neq i} T_{ij} V_j(t) + I_i \\ \theta_i(t) &= \sum_{j \neq i} W_{ij} V_j(t)(1 - V_j(t)) \end{aligned} \tag{7}$$

Theorem 3: The algorithm based on the update rules (5)-(7) converges on stable states provided that $T_{ij} = T_{ji}$, $W_{ij} = W_{ji}$ and $T_{ii} = W_{ii} = 0$.

Proof: This can be proved by showing that the energy function E of Eqn. 4 is always decreased by any state change produced by the algorithm based on the update rules (5)-(7). From Eqn. 4, the energy function E is

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij} V_i V_j - \sum_{i=1}^{N} I_i V_i - \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij} V_i (1 - V_i) V_j (1 - V_j) + K \tag{8}$$

The difference between the global energy of the network when neuron i has the state value α and when neuron i has the state value β, given the current states of the other neurons, is $E_{(V_i=\alpha)} - E_{(V_i=\beta)} = \Delta E_{i\alpha-\beta}$, where α and β are 0, 1, or 1/2. These are:

$$E_{(V_i=0)} - E_{(V_i=1)} = \Delta E_{i0-1} = \sum_{j \neq i} T_{ij} V_j + I_i = U_i \tag{9}$$

$$E_{(V_i=0)} - E_{(V_i=\frac{1}{2})} = \Delta E_{i0-\frac{1}{2}} = \frac{1}{2}\left(\sum_{j \neq i} T_{ij} V_j + I_i + \sum_{j \neq i} W_{ij} V_j (1 - V_j)\right) = \frac{1}{2}(U_i + \theta_i) \tag{10}$$

$$E_{(V_i=\frac{1}{2})} - E_{(V_i=1)} = \Delta E_{i\frac{1}{2}-1} = \frac{1}{2}\left(\sum_{j \neq i} T_{ij} V_j + I_i - \sum_{j \neq i} W_{ij} V_j (1 - V_j)\right) = \frac{1}{2}(U_i - \theta_i) \tag{11}$$

Let us first consider the case of $\theta_i > 0$. If $U_i \geq \theta_i$ then $E_{(V_i=0)} - E_{(V_i=1)} = \Delta E_{i0-1} > 0$ and $E_{(V_i=\frac{1}{2})} - E_{(V_i=1)} = \Delta E_{i\frac{1}{2}-1} \geq 0$. According to the update rule (6), neuron i takes the state value 1 only when $U_i \geq \theta_i$, and hence this state change decreases the energy. If $-\theta_i < U_i < \theta_i$ then $E_{(V_i=0)} - E_{(V_i=\frac{1}{2})} = \Delta E_{i0-\frac{1}{2}} > 0$ and $E_{(V_i=\frac{1}{2})} - E_{(V_i=1)} = \Delta E_{i\frac{1}{2}-1} < 0$. Similarly, according to the update rule (5), neuron i takes the state value 1/2 only when $-\theta_i < U_i < \theta_i$, and hence the energy is decreased. If $U_i \leq -\theta_i$ then $E_{(V_i=0)} - E_{(V_i=1)} = \Delta E_{i0-1} < 0$ and $E_{(V_i=0)} - E_{(V_i=\frac{1}{2})} = \Delta E_{i0-\frac{1}{2}} \leq 0$. From the update rule (5), neuron i takes the state value 0 only when $U_i \leq -\theta_i$, and hence the energy is decreased.

Next, consider the case of $\theta_i \leq 0$. If $U_i > 0$ then $E_{(V_i=0)} - E_{(V_i=1)} = \Delta E_{i0-1} > 0$. Since $U_i > 0 \geq \theta_i$, $E_{(V_i=\frac{1}{2})} - E_{(V_i=1)} = \Delta E_{i\frac{1}{2}-1} > 0$. According to the update rule (6), neuron i changes the state value to 1 only when $U_i > \theta_i$, and hence the energy is decreased. If $U_i < 0$ then $E_{(V_i=0)} - E_{(V_i=1)} = \Delta E_{i0-1} < 0$. Further $U_i < 0 \leq -\theta_i$, i.e., $E_{(V_i=0)} - E_{(V_i=\frac{1}{2})} = \Delta E_{i0-\frac{1}{2}} < 0$. According to the update rule (6), neuron i changes the state value to 0 only when $U_i < \theta_i$, and hence the energy is decreased. If $U_i = 0$ then $E_{(V_i=0)} - E_{(V_i=1)} = \Delta E_{i0-1} = 0$. Further $U_i = 0 \geq \theta_i$, i.e., $E_{(V_i=\frac{1}{2})} - E_{(V_i=1)} = \Delta E_{i\frac{1}{2}-1} > 0$. $U_i = 0 \leq -\theta_i$, i.e., $E_{(V_i=0)} - E_{(V_i=\frac{1}{2})} = \Delta E_{i0-\frac{1}{2}} < 0$. Therefore, $E_{(V_i=0)} = E_{(V_i=1)} < E_{(V_i=\frac{1}{2})}$ According to the update rule (6), neuron i retains its state value.

Q.E.D.

3.3 State Probability Functions

The algorithm mentioned in the previous section is to use gradient descent, i.e., the values of the variables in the problem are modified in whatever direction reduces the energy. However, for hard problems, gradient descent gets stuck at *local* minima that are not globally optimal. To escape from local minima, we modify the update rule in the same way as a probabilistic hill climbing technique [6, 7] by extending activation probabilities for neurons. This is based on the idea that if jumps to higher energy states occasionally occur, it is possible to break out of local minima.

For 3-valued neural networks, we have to consider three energy gaps between 0 and 1, 0 and 1/2, and 1/2 and 1. These energy gaps are respectively

$$E(v_i=0) - E(v_i=1) = \sum_{j\neq i} T_{ij} V_j + I_i \tag{8}$$

$$E(v_i=0) - E(v_i=\tfrac{1}{2}) = \frac{1}{2}\left(\sum_{j\neq i} T_{ij} V_j + I_i + \sum_{j\neq i} W_{ij} V_j (1 - V_j)\right) \tag{9}$$

$$E(v_i=\tfrac{1}{2}) - E(v_i=1) = \frac{1}{2}\left(\sum_{j\neq i} T_{ij} V_j + I_i - \sum_{j\neq i} W_{ij} V_j (1 - V_j)\right) \tag{10}$$

As the state probability of Eqn. (3) was derived from the update rule of Eqn. (1), so we can define the probabilities that neuron i takes each state value, 0, 1 or 1/2, from the update rule of Eqns. (5) and (6). These state probabilities are as follows:

In case of $\theta_i > 0$

$$p_{i1} = p(v_i=1) = \frac{1}{1 + e^{-(U_i - \theta_i)/2T}} \tag{11}$$

$$p_{i0} = p(v_i=0) = 1 - \frac{1}{1 + e^{-(U_i + \theta_i)/2T}} \tag{12}$$

$$p_{i\frac{1}{2}} = p(v_i=\tfrac{1}{2}) = 1 - (p_{i1} + p_{i0}) = \frac{1}{1 + e^{-(U_i + \theta_i)/2T}} - \frac{1}{1 + e^{-(U_i - \theta_i)/2T}} \tag{13}$$

In case of $\theta_i \leq 0$

$$p_{i1} = p(v_i=1) = \frac{1}{1 + e^{-U_i/T}} \tag{14}$$

$$p_{i0} = p(v_i=0) = 1 - p_{i1} = 1 - \frac{1}{1 + e^{-U_i/T}} \tag{15}$$

where T is a parameter which acts like the temperature of a physical system. These probabilities are illustrated in Figures 4 and 5.

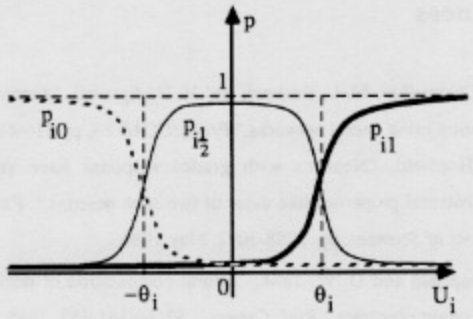


Figure 4. State Probabilities when $\theta_i > 0$

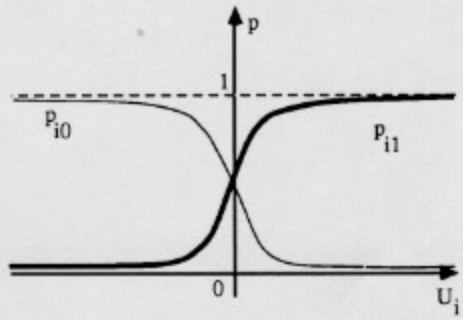


Figure 5. State Probabilities when $\theta_i \leq 0$

IV. Conclusions

This paper proposed a three-valued (0, 1, and 1/2) neural network which is an extension of the binary Hopfield's model and showed that the test generation problem can be solved by the three-valued model more effectively than by the binary one. In the three-valued model, the energy function of networks, hyperplanes of neurons, and update rules of neuron's states were extended naturally so that the third value 1/2 (don't care) can be treated well. It was proved that the proposed three-valued model converges to local minima as Hopfield's model does. To escape from local minima, an extension of Boltzmann machines was presented where the update rules were modified by introducing state probabilities which are functions of a temperature.

References

[1] S. T. Chakradhar, M. L. Bushnell and V. D. Agrawal, "Automatic test generation using neural networks," *Proc. ICCAD'88*, pp.416-419, 1988.

[2] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons," *Proc. Nat'l Academy of Sciences*, pp. 3088-3092, May 1984.

[3] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.* , 52, pp.141-152, 1985.

[4] H. Fujiwara, *Logic Testing and Design for Testability*, MIT Press, 1985.

[5] V. D. Agrawal and S. C. Seth, *Test Generation for VLSI Chips*, IEEE Computer Society Press, 1988

[6] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp.721-741, 1984.

[7] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, "Boltzmann machines: Constraint satisfaction networks that learn," Tech Report CMU-CS-84-119, Carnegie-Mellon U., May 1984.