International Journal of Computer Aided VLSI Design 3, 273–290 (1991)

Three-Valued Neural Networks for Test Generation

HIDEO FUJIWARA
Meiji University, Japan

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 that can be solved using Hopfield's binary neural networks. In this paper, we propose a three-valued (0, 1, and $\frac{1}{2}$) neural network as an extension of Hopfield's binary model and show that the test generation problem can be solved more effectively by the three-valued model. In this model, the energy function of the neural networks, the hyperplanes of neurons, and the update rule of neuron are extended. 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 rule is modified by introducing probabilities of a neuron's states. Outputs of neurons in the original Hopfield model take all continuous values between 0 and 1. In this paper, the dynamics of neurons in the modified (binary and three-valued) Hopfield models is discussed in both discrete and continuous domains. Furthermore, a more general three-valued model is introduced in which three arbitrary values V^0, V^1, and V^x can be used as a neuron's state.

Boltzmann machine, neural networks, optimization problem, test generation, three-valued

1 INTRODUCTION

Neural networks have been used in many different fields. Although there are many neural network models, Hopfield's model [1, 2] is attractive because the computational power and its speed are demonstrated by solving one of the NP-complete [4] problems known as the traveling salesman problem [2].

In the field of test generation, Chakradhar et al. [3] proposed an approach to automatic test generation using neural networks. They formulated the test generation problem as an optimization problem that can be solved by Hopfield's binary neural networks [1], where neurons assume binary values (0 or 1). Their approach using neural networks is radically different from the conventional algorithms, such as the D-algorithm [5], Podem [6], Fan [7], and Socrates [8]. 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, this approach may provide an advantage over conventional methods.

The approach of Chakradhar et al. cannot be applied to sequential circuits due to its binary model. To generate a test sequence for a sequential circuit, it is

An earlier version of this paper appeared in *Proceedings of the 20th International Symposium on Fault Tolerant Computing*, June 1990.

Correspondence and requests for reprints should be sent to Hideo Fujiwara, Department of Computer Science, Meiji University, 1-1-1 Higashimita, Tama-ku, Kawasaki 214, Japan.

necessary to deal with three values, 0, 1, and X (don't care or unknown), since the initial state of a sequential circuit is unknown in general. Therefore, in this paper, we extend the ideas of Chakradhar et al. [3] and explore new possibilities of solving computationally difficult problems on three-valued neural networks where neurons assume values from the set $\{0, 1, \frac{1}{2}\}$. We propose a three-valued neural network model that is an extension of Hopfield's binary model, and show that the test generation problem can be solved by this model more effectively than by the binary one. In the proposed model, the energy function of the neural network, the hyperplanes of neurons, and the neuron update rule are extended. It is proved that the proposed three-valued model always converges. To escape from local minima, we present an extension of the Boltzmann machine where the update rule is modified by introducing probabilities of a neuron's states.

Outputs of neurons in the original Hopfield model take all continuous values between 0 and 1. In this paper, the dynamics of neurons in the modified (binary and three-valued) Hopfield models is discussed in both the discrete and continuous domains. We also introduce a more general three-valued model where neurons assume three arbitrary values, V^0, V^1, and V^X ($V^0 < V^X < V^1$). These values, V^0, V^1, and V^X, correspond to the logic values 0, 1, and X, respectively. Both the $(0, \frac{1}{2}, 1)$ and $(-1, 0, 1)$ models are special cases for this general (V^0, V^X, V^1) model.

2 CHAKRADHAR'S APPROACH

First, we shall introduce briefly the approach of Chakradhar et al. [3] 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 determined completely by the specification of the interaction. Let V_i denote the *state* of neuron, i, $V_i \in \{0, 1\}$ for $i = 1, 2, \ldots, N$, where N is the number of neurons in the network. Let $V_i(t)$ denote the state of neuron i at time t, and let each neuron randomly *update* its state according to the following equation:

$$V_i(t + 1) = stp\left(\sum_{j=1}^{N} T_{ij}V_j(t) + I_i\right), \tag{1}$$

where $stp(x)$ is a unit step function, which is 1 for $x \geq 0$ and 0 for $x < 0$; T_{ij} is the weight associated with the link between neurons i and j; and I_i is the threshold of neuron i. Hopfield [1] has shown that if $T_{ij} = T_{ji}$ and $T_{ii} = 0$ for all i and j, neurons always change their states in such a manner that they lead to stable states that do not change again with time and that they locally minimize an *energy function* defined by

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

where K is a constant.

2.2 Neural Networks for Logic Circuits

An arbitrary logic circuit can be represented by a neural network [3]. 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 two-input AND, OR, NAND, NOR, XOR, and XNOR gates and the 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. T and I are determined so that energy E [Equation (2)] has global minima only at the neuron states consistent with the function of all gates in the circuit. All other inconsistent states have higher energy. In other words, the energy E is a nonnegative 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 vectors corresponding to consistent states of the network. A vector belongs to P_{i_on} (P_{i_off}) if it corresponds only to one consistent state and neuron i has a state value 1 (0). P_{i_other} consists of all vectors corresponding to consistent states that are not in set P_{i_on} or P_{i_off}.

Definition 2

A hyperplane

$$\sum_{j \neq i} T_{ij} V_j + I_j = 0$$

associated with neuron i is a *decision hyperplane* if the vectors in P_{i_on} and P_{i_off} fall on opposite sides of the hyperplane and all vectors in P_{i_other} lie on the hyperplane.

Theorem 1 [3]

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 Equation (2)] is the existence of a decision hyperplane for each of the n neurons.

Example 1. Figure 1 shows a two-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 $P_{3\_other} = \{\ \}$.

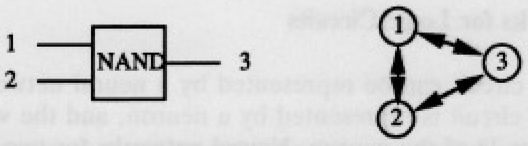

Figure 1. **Two-input NAND and corresponding neural network.**

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 also satisfy the following conditions: $K = T_3 > 0$, $I_3 > I_1 > 0$, $I_3 > I_2 > 0$, $T_{12} < 0$, $T_{13} < 0$, $T_{23} < 0$, $T_{13} + I_1 = 0$, $T_{23} + I_2 = 0$, and $T_{12} + I_1 + I_2 = I_3$. One solution that satisfies all the preceding conditions is $I_1 = I_2 = 2$, $I_3 = 3$, $T_{12} = -1$, and $T_{13} = T_{23} = -2$.

2.3 Test Generation Problem Formulation

Figure 2 illustrates a network that specifies constraints for test generation. This network is constructed by joining the good circuit and a faulty circuit so that the two circuits share the same primary inputs. Their primary outputs are connected through an *output interface* to include the constraint that a 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. If a test exists for a fault, there is a *consistent labeling* of the neurons in the neural network with values from the set {0, 1} that does not violate the function of any gate. In this way, the test generation problem 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.

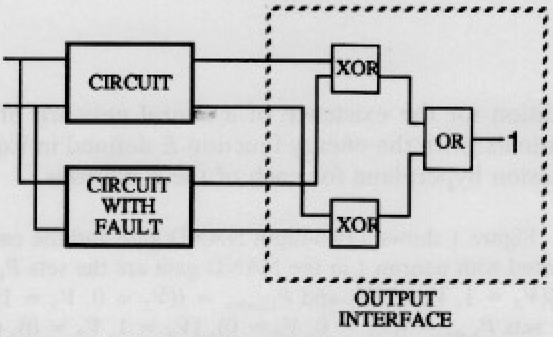

Figure 2. **Constraint newtork for test generation.**

First, a local minimum of E is obtained using a gradient descent approach, where each neuron updates randomly according to the update rule of Equation (1). Although Hopfield's network always converges on the local minima of E, convergence to a global minimum is not guaranteed [1]. Hence, to escape from local minima, the update rule is modified by using a probabilistic hill climbing technique [9, 10]. 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

$$p_k = \frac{1}{1 + e^{-\Delta E_k / T}}. \tag{3}$$

3 THREE-VALUED NEURAL NETWORKS

As mentioned in the previous section, the problem of test generation is to find a consistent labeling of the neurons in the constraint neural network, with values from the set $\{0, 1\}$, that does not violate the function 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 allow signals to assume only values from 0 or 1, we have to select one assignment from three possible input combinations: 00, 01, or 10. Conversely, if signals can assume values from the set $\{0, 1, X\}$, where X denotes *don't care*, we have to select one assignment from two possible input combinations: $0X$ or $X0$. This reduces the search space as shown in Figure 3(a).

In Chakradhar's approach, using the binary model, every neuron is initialized to either 0 or 1. So, it often happens that many unnecessary values are assigned to neurons. In three-valued neural networks, neurons can be initialized to a 0, 1, or X ($\frac{1}{2}$).

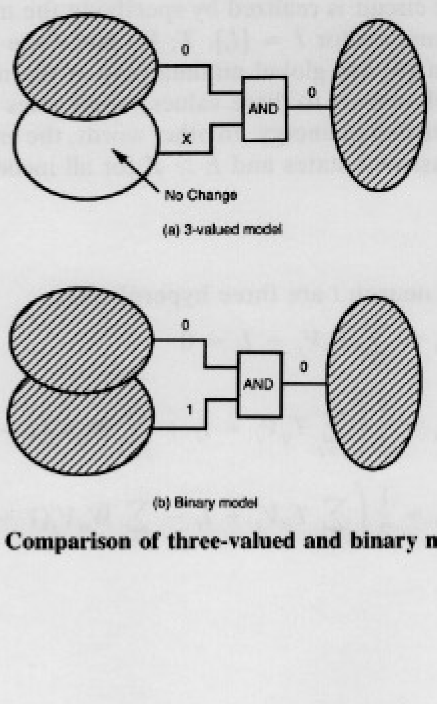

Figure 3. Comparison of three-valued and binary models.

The purpose of allowing neurons to assume a third value ($\frac{1}{2}$) is to avoid unnecessary assignment of 0 or 1 to signals (pruning the search space), to obtain necessary and sufficient signal values to detect a given fault (minimal test vectors), and to speed up the convergence to a global minimum.

Another important reason why a third value is needed, is to deal with test generation for sequential circuits. The approach of Chakradhar et al. cannot be applied to sequential circuits due to its binary model. To generate a test sequence for a sequential circuit, it is necessary to deal with the three values 0, 1, and X (don't care or unknown), since the initial state of a sequential circuit is unknown in general.

3.1 Energy Function and Hyperplanes for Three-Valued Model

The *energy* function for three-valued neural networks is of the following form:

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

where W_{ij} is another weight associated with the link between neurons i and j. The state values of the neurons are 0, 1, and $\frac{1}{2}$. 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 of $\frac{1}{2}$.

Following Chakradhar's method, we can represent an arbitrary logic circuit by using 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 $\frac{1}{2}$) 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 $I = [I_i]$. T, W, and I are determined so that the energy E of Equation (4) has global minima only at the neuron states consistent with the function (with respect to three values) of all gates in the circuit. All other inconsistent states have higher energy. In other words, the energy E is a nonnegative 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. $E_{(V_i=0)} - E_{(V_i=1/2)} = \frac{1}{2}\left(\sum_{j\neq i} T_{ij}V_j + I_i + \sum_{j\neq i} W_{ij}V_j(1-V_j)\right) = 0$

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

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 $n - 1$-dimensional vectors corresponding to consistent states of the network. A vector 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 vector 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 vectors in $P_{i(0>1)}$ and $P_{i(0<1)}$ fall on opposite sides of the hyperplane and all vectors 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 $(0, \frac{1}{2})$-*decision hyperplane* if the vectors in $P_{i(0>1/2)}$ and $P_{i(0<1/2)}$ fall on opposite sides of the hyperplane and all vectors 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 $(\frac{1}{2}, 1)$-*decision hyperplane* if the vectors in $P_{i(1/2>1)}$ and $P_{i(1/2<1)}$ fall on opposite sides of the hyperplane and all vectors in $P_{i(1/2=1)}$ lie on the hyperplane.

Theorem 2

A necessary condition for the existence of a three-valued neural network of n neurons for a device with n terminals [with the energy function E defined in Equation (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 $\frac{1}{2}$ These are:

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

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

$$E_{(V_i=1/2)} - E_{(V_i=1)} = \Delta E_{i1/2-1} = \frac{1}{2}\left(\sum_{j\neq i} T_{ij}V_j + I_j - \sum_{j\neq i} W_{ij}V_j(1 - V_j)\right).$$

For an arbitrary vector $p \in P_{i(\alpha>\beta)}$ $(P_{i(\alpha<\beta)})$, neuron i assumes the 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 values 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 lets vector p lie in the region $R1$ $(R2)$. Therefore, for E to exist, all vectors in $P_{i(\alpha>\beta)}$ $(P_{i(\alpha<\beta)})$ must lie in region $R1$ $(R2)$. For an arbitrary vector $p \in P_{i(\alpha=\beta)}$, 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 states, it is mandatory that $\Delta E_{i\alpha-\beta}$ be zero. Hence, p must lie on the hyperplane. Hence, for any given unit i, the existence of an (α, β)-decision hyperplane is a necessary condition for the existence of E. **Q.E.D.**

Example 2. Consider the two-input NAND gate and its neural network shown in Figure 1. Associated with neuron 1 are nine sets:

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

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

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

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

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

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

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

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

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

From Theorem 2, the existence of three decision hyperplanes for each neuron is necessary for the existence of a three-value neural network for the NAND gate. Let us derive the conditions of those decision hyperplanes. From the 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

$$T_{12} + I_1 > 0$$

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

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

$$\tfrac{1}{2}T_{12} + T_{13} + I_1 < 0 \quad \text{and}$$

$$T_{13} + I_1 = 0.$$

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

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

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

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

$$\tfrac{1}{2}T_{12} + T_{13} + I_1 + \tfrac{1}{4}W_{12} < 0 \quad \text{and}$$

$$T_{13} + I_1 = 0.$$

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

$$T_{12} + I_1 > 0$$
$$T_{12} + \tfrac{1}{2}T_{13} + I_1 - \tfrac{1}{4}W_{13} < 0$$
$$T_{13} + I_1 = 0 \quad \text{and}$$
$$\tfrac{1}{2}(T_{12} + T_{13}) + I_1 - \tfrac{1}{4}(W_{12} + W_{13}) = 0.$$

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

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

Similarly, from the 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

$$I_3 > 0$$
$$T_{13} + I_3 > 0$$
$$T_{23} + I_3 > 0$$
$$\tfrac{1}{2}T_{13} + I_3 > 0$$
$$\tfrac{1}{2}T_{23} + I_3 > 0 \quad \text{and}$$
$$T_{13} + T_{23} + I_3 < 0.$$

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

$$\tfrac{1}{2}(T_{13} + T_{23}) + I_3 + \tfrac{1}{4}(W_{13} + W_{23}) > 0 \quad \text{and}$$
$$T_{13} + T_{23} + I_3 < 0.$$

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

$$I_3 > 0$$

$$T_{13} + I_3 > 0$$

$$T_{23} + I_3 > 0$$

$$\tfrac{1}{2}T_{13} + I_3 > 0$$

$$\tfrac{1}{2}T_{23} + I_3 > 0 \quad \text{and}$$

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

Furthermore, the energy function E should be zero at all nine consistent states ($V_1 = V_2 = 0, V_3 = 1$, ($V_1 = 0, V_2 = V_3 = 1$), ($V_1 = V_3 = 1, V_2 = 0$), ($V_1 = V_2 = 1, V_3 = 0$), ($V_1 = 0, V_2 = \tfrac{1}{2}, V_3 = 1$), ($V_1 = \tfrac{1}{2}, V_2 = 0, V_3 = 1$), ($V_1 = \tfrac{1}{2}, V_2 = \tfrac{1}{2}, V_3 = \tfrac{1}{2}$), ($V_1 = 1, V_2 = \tfrac{1}{2}, V_3 = \tfrac{1}{2}$), and ($V_1 = \tfrac{1}{2}, V_2 = 1, V_3 = \tfrac{1}{2}$). Therefore,

$$K - I_3 = 0$$

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

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

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

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

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

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

$$K - (\tfrac{1}{2}T_{12} + \tfrac{1}{2}T_{13} + \tfrac{1}{4}T_{23}) - (I_1 + \tfrac{1}{2}I_2 + \tfrac{1}{2}I_3) - \tfrac{1}{8}W_{23} = 0 \quad \text{and}$$

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

Summarizing the preceding equalities and inequalities, we have

$$K = I_3 > 0$$

$$I_3 > I_1 > 0$$

$$I_3 > I_2 > 0$$

$$T_{12} < 0$$

$$T_{13} < 0$$

$$T_{23} < 0$$

$$T_{23} + I_2 = 0$$

$$T_{13} + I_1 = 0$$

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

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

$$W_{13} = -2T_{13} \quad \text{and}$$

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

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

$$E(V_1, V_2, V_3 = E(0, 0, 0) = K = I_3 > 0$$
$$E(0, 1, 0) = K - I_2 = I_3 - I_2 > 0$$
$$E(1, 0, 0) = K - I_1 = I_3 - I_1 > 0$$
$$E(1, 1, 1) = K - (T_{12} + T_{13} + T_{23}) - (I_1 + I_2 + I_3) = -T_{12} > 0$$
$$E(0, x, 0) = K - \tfrac{1}{2}I_2 > 0$$
$$E(x, 0, 0) = K - \tfrac{1}{2}I_1 > 0$$
$$E(0, x, x) = K - \tfrac{1}{4}T_{23} - \tfrac{1}{2}(I_2 + I_3) - \tfrac{1}{8}W_{23} > 0$$
$$E(x, 0, x) = K - \tfrac{1}{4}T_{13} - \tfrac{1}{2}(I_1 - I_3) - \tfrac{1}{8}W_{13} > 0$$
$$E(x, x, 0) = K - \tfrac{1}{4}T_{12} - \tfrac{1}{2}(I_1 + I_2) - \tfrac{1}{8}W_{12} = \tfrac{1}{2}I_3 > 0$$
$$E(x, x, 1) = K - (\tfrac{1}{4}T_{12} + \tfrac{1}{2}T_{13} + \tfrac{1}{2}T_{23}) - (\tfrac{1}{2}I_1 + \tfrac{1}{2}I_2 + I_3) - \tfrac{1}{8}W_{12} > 0$$
$$E(1, x, 0) = K - \tfrac{1}{2}T_{12} - (I_1 + \tfrac{1}{2}I_2) > 0$$
$$E(1, x, 1) = K - (\tfrac{1}{2}T_{12} + T_{13} + \tfrac{1}{2}T_{23}) - (I_1 + \tfrac{1}{2}I_2 + I_3) = -T_{12} > 0$$
$$E(x, 1, 0) = K - \tfrac{1}{2}T_{12} - (\tfrac{1}{2}I_1 + I_2) > 0$$
$$E(x, 1, 1) = K - (\tfrac{1}{2}T_{12} + \tfrac{1}{2}T_{13} + T_{23}) - (\tfrac{1}{2}I_1 + I_2 + I_3) > 0.$$

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

One solution that satisfies the preceding 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 Neuron Update Rule

The state of an individual neuron i is updated as follows. Let V_i denote the state of neuron i, i.e., $V_i \in \{0, 1, \tfrac{1}{2}\}$ for $i = 1, 2, \ldots, N$, 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 the case of $\theta_i(t) > 0$,

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

In the 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

$$U_i(t) = \sum_{j \neq i} T_{ij} V_j(t) + I_i \quad \text{and} \quad \theta_i(t) = \sum_{j \neq i} W_{ij} V_j(t)(1 - V_j(t)). \qquad (7)$$

Theorem 3

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

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

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

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 $\frac{1}{2}$ These are:

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

$$E_{(V_i=0)} - E_{(V_i=1/2)} = \Delta E_{i0-1/2}$$

$$= \frac{1}{2}\left(\sum_{j \neq i} T_{ij} V_j + I_i + \sum_{j \neq i} W_{ij} V_j (1 - V_j)\right) = \frac{1}{2}(U_i + \theta_i) \qquad (10)$$

$$E_{(V_i=1/2)} - E_{(V_i=1)} = \Delta E_{i1/2-1}$$

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

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 $E_{(V_i=1/2)} - E_{(V_i=1)} = \Delta E_{i1/2-1} \geq 0$. According to 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 $E_{(V_i=0)} - E_{(V_i=1/2)} = \Delta E_{i0-1/2} > 0$ and $E_{(V_i=1/2)} - E_{(V_i=1)} = \Delta E_{i1/2-1} < 0$. Similarly, according to update rule (5), neuron i takes the state value $\frac{1}{2}$ 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 $E_{(V_i=0)} - E_{(V_i=1/2)} = \Delta E_{i0-1/2} \leq 0$. From 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 $U_i > 0 \geq \theta_i$, $E_{(V_i=1/2)} - E_{(V_i=1)} = \Delta E_{i1/2-1} > 0$. According to 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 $E_{(V_i=0)} - E_{(V_i=1)} = \Delta E_{i0-1} < 0$. Furthermore, $U_i < 0 \leq -\theta_i$, i.e., $E_{(V_i=0)} - E_{(V_i=1/2)} = \Delta E_{i0-1/2} < 0$. According to 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 $E_{(V_i=0)} - E_{(V_i=1)} = \Delta E_{i0-1} = 0$. Furthermore, $U_i = 0 \geq \theta_i$, i.e., $E_{(V_i=1/2)} - E_{(V_i=1)} = \Delta E_{i1/2-1} > 0$. $U_i = 0 \leq -\theta_i$, i.e., $E_{(V_i=0)} - E_{(V_i=1/2)} = \Delta E_{i0-1/2} < 0$. Therefore, $E_{(V_i=0)} = E_{(V_i=1)} < E_{(V_i=1/2)}$. According to update rule (6), neuron i retains its state value.

Q.E.D.

3.3 Three-Valued Boltzmann Machines

In the previous section, we describe the gradient descent method. However, for some problems, this method converges to a *local* minimum. To escape from local minima, we modify the update rule in the same way as a probalilistic hill climbing technique [5, 6] 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. Such a model is called the Boltzmann machine. Here, we shall extend the binary Boltzmann machine to a three-valued model.

For three-valued neural networks, we must consider three energy gaps between 0 and 1, 0 and $\frac{1}{2}$, and $\frac{1}{2}$ and 1. These energy gaps are, respectively,

$$E_{(V_i=0)} - E_{(V_i=1)} = \sum_{j \neq i} T_{ij}V_j + I_i \tag{12}$$

$$E_{(V_i=0)} - E_{(V_i=1/2)} = \frac{1}{2}\left(\sum_{j \neq i} T_{ij}V_j + I_i + \sum_{j \neq i} W_{ij}V_j(1 - V_j)\right) \tag{13}$$

$$E_{(V_i=1/2)} - E_{(V_i=1)} = \frac{1}{2}\left(\sum_{j \neq i} T_{ij}V_j + I_i - \sum_{j \neq i} W_{ij}V_j(1 - V_j)\right). \tag{14}$$

As the state probability of Equation (3) was derived from the update rule of Equation (1), so we can define the probabilities that neuron i takes each state value, 0, 1, or $\frac{1}{2}$, from the update rule of Equations (5) and (6). These state probabilities are as follows:

In the case of $\theta_i > 0$,

$$p_{i1} = p_{(V_i=1)} = \frac{1}{1 + e^{-(U_i-\theta_i)/2T}} \tag{15}$$

$$p_{i0} = p_{(V_i=0)} = 1 - \frac{1}{1 + e^{-(U_i+\theta_i)/2T}} \tag{16}$$

$$p_{i1/2} = p_{(V_i=1/2)} = 1 - (p_{i1} + p_{i0}) = \frac{1}{1 + e^{-(U_i+\theta_i)/2T}} - \frac{1}{1 + e^{-(U_i-\theta_i)/2T}}. \tag{17}$$

In the case of $\theta_i \leq 0$,

$$p_{i1} = p_{(V_i=1)} = \frac{1}{1 + e^{-U_i/T}} \tag{18}$$

$$p_{i0} = p_{(V_i=0)} = 1 - p_{i1} - \frac{1}{1 + e^{-U_i/T}}, \tag{19}$$

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

4 DISCRETE VERSUS CONTINUOUS MODELS

The neural network models we have considered so far have used a stochastic algorithm involving sudden changes of a neuron's states among 0, 1, and $\frac{1}{2}$ at random

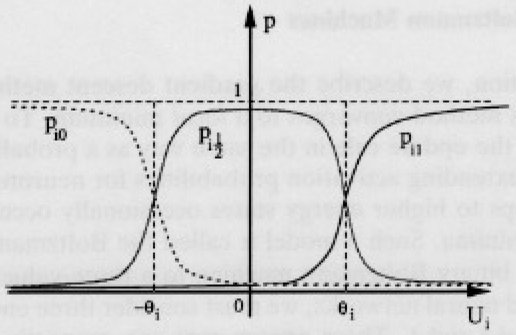

Figure 4. State probabilities when $\theta_i > 0$.

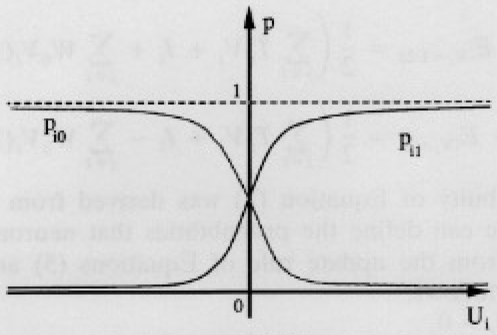

Figure 5. State probabilities when $\theta_i \leq 0$.

time. However, since real neurons have continuous input/output relations, we need to analyze the neural networks in continuous space as well as in discrete space. In this section, we shall describe briefly an extension of the three-valued Hopfield machines with discrete input/output relations into those with continuous input/output relations.

In the discrete binary Hopfield model, every neuron assumes only 0 or 1 values. Neuron i receives inputs $T_{ij}V_j$ from every other neuron j and a bias input I_i. The net input to a neuron is as follows:

$$U_i(t) = \sum_{j \neq i} T_{ij}V_j(t) + I_i. \tag{20}$$

Each neuron randomly updates its state according to the following equation:

$$V_i(t+1) = stp\left(\sum_{j=1}^{N} T_{ij}V_j(t) + I_i\right), \tag{21}$$

where $stp(x)$ is a unit step function that is 1 for $x \geq 0$ and 0 for $x < 0$.

In the continuous Hopfield model, neurons change their state according to the following equations of dynamics [1]:

$$\frac{dU_i}{dt} = \sum_{j=1}^{N} T_{ij}V_j(t) + I_i \tag{22}$$

$$V_i = g(U_i) = \frac{1}{1 + e^{-U_i/T}} \tag{23}$$

where i is time, and $g(x)$ is sigmoid that approaches a unit step function $stp(x)$ as T tends to zero.

Similarly, we can derive the continuous Hopfield model based on the three-state model from state probability functions Equations (15)–(19) of the three-valued Boltzmann machine.

Let us compute the average value of the state of neuron i from Equations (15)–(19). In the case of $\theta_i > 0$, we have

$$\begin{aligned} V_i &= 1 \times p_{i1} + \frac{1}{2} \times p_{i1/2} + 0 \times P_{i0} \\ &= \frac{1}{2}\left(\frac{1}{1 + e^{-(U_i+\theta_i)/2T}} + \frac{1}{1 + e^{-(U_i-\theta_i)/2T}}\right). \end{aligned} \tag{24}$$

Similarly, in the case of $\theta_i \leq 0$, we have

$$\begin{aligned} V_i &= 1 \times p_{i1} + 0 \times p_{i0} \\ &= \frac{1}{1 + e^{-U_i/T}}. \end{aligned} \tag{25}$$

The state of each neuron i is determined by Equations (24) and (25), which approach Equations (5) and (6), respectively, as T tends to zero.

5 GENERAL (V^0, V^X, V^1) MODEL

So far, we have considered a three-valued model where neurons assume a value of 0, $\frac{1}{2}$, or 1. However, we can consider another three-valued model where neurons assume a value of -1, 0, or 1. In this section, we introduce briefly a more general three-valued neural network model that includes these two models as special cases.

Let V^0, V^X, and V^1 be the state values of neurons that correspond to the logic values 0, X, and 1, respectively. Note that the three values V^0, V^X, and V^1 satisfying $V^0 < V^X < V^1$ are chosen arbitrarily. This three-valued model is called the (V^0, V^X, V^1) *model*.

We can define the energy function for a three-valued neural network of the (V^0, V^X, V^1) model in the following:

$$\begin{aligned} E = &-\frac{1}{2}\sum_i \sum_j T_{ij}V_iV_j - \sum_i I_iV_i \\ &- \sum_i \sum_j W_{ij}(V_i^0 - V_i)(V_i^1 - V_i)(V_j^0 - V_j)(V_j^1 - V_j) + K. \end{aligned} \tag{26}$$

In the same way as Section 3.1, associated with each neuron i are the following three hyperplanes:

$$E_{(V_i=V^0)} - E_{(V_i=V^1)} = \left(\sum_{j \neq i} T_{ij}V_j + I_i\right)(V^1 - V^0) = 0 \tag{27}$$

$$E_{(V_i=V^0)} - E_{(V_i=V^X)} = \left(\sum_{j \neq i} T_{ij}V_j + I_i \right) (V^X - V^0)$$
$$- 2 \left(\sum_{j \neq i} W_{ij}(V^0 - V_j)(V^1 - V_j) \right) (V^X - V^0)(V^1 - V^X) = 0 \quad (28)$$

$$E_{(V_i=V^X)} - E_{(V_i=V^1)} = \left(\sum_{j \neq i} T_{ij}V_j + I_i \right) (V^1 - V^X)$$
$$+ 2 \left(\sum_{j \neq i} W_{ij}(V^0 - V_j)(V^1 - V_j) \right) (V^X - V^0)(V^1 - V^X) = 0. \quad (29)$$

Using the preceding energy function and three hyperplanes, we can represent an arbitrary logic circuit by a three-valued neural network of the (V^0, V^X, V^1) model in the same way as the $(0, \frac{1}{2}, 1)$ model. Since this is straightforward, we shall omit here the detailed discussion.

Next, let us consider the $(0, \frac{1}{2}, 1)$ model as a special case of the (V^0, V^X, V^1) model. By substituting 0, $\frac{1}{2}$, and 1 for V^0, V^X, and V^1, respectively, in Equations (26)–(29), we have

$$E = -\frac{1}{2} \sum_i \sum_j T_{ij}V_iV_j - \sum_i I_iV_i - \sum_i \sum_j W_{ij}V_i(1 - V_i)V_j(1 - V_j) + K \quad (30)$$

$$E_{(V_i=0)} - E_{(V_i=1)} = \sum_{j \neq i} T_{ij}V_j + I_i = 0 \quad (31)$$

$$E_{(V_i=0)} - E_{(V_i=1/2)} = \frac{1}{2} \left(\sum_{j \neq i} T_{ij}V_j + I_i + \sum_{j \neq i} W_{ij}V_j(1 - V_j) \right) = 0 \quad (32)$$

$$E_{(V_i=1/2)} - E_{(V_i=1)} = \frac{1}{2} \left(\sum_{j \neq i} T_{ij}V_j + I_i - \sum_{j \neq i} W_{ij}V_j(1 - V_j) \right) = 0. \quad (33)$$

Equation (30) coincides with (4) of the energy function of the $(0, \frac{1}{2}, 1)$ model. Equations (31), (32), and (33) coincide with those of Definition 3, which are hyperplanes for the $(0, \frac{1}{2}, 1)$ model.

Similarly, let us consider the $(-1, 0, 1)$ model as a special case of the (V^0, V^X, V^1) model. By substituting -1, 0, and 1, respectively for V^0, V^X, and V^1 in Equations (26)–(29), we have

$$E = \frac{1}{2} \sum_i \sum_j T_{ij}V_iV_j - \sum_i I_iV_i$$
$$- \sum_i \sum_j W_{ij}(1 + V_i)(1 - V_i)(1 + V_j)(1 - V_j) + K \quad (34)$$

$$E_{(V_i=-1)} - E_{(V_i=1)} = 2\left(\sum_{j \neq i} T_{ij}V_j + I_i \right) = 0 \quad (35)$$

$$E_{(V_i=-1)} - E_{(V_i=0)} = \sum_{j \neq i} T_{ij}V_j + I_i + \sum_{j \neq i} W_{ij}(1 + V_j)(1 - V_j) = 0 \quad (36)$$

$$E_{(V_i=0)} - E_{(V_i=1)} = \sum_{j \neq i} T_{ij} V_j + I_i - \sum_{j \neq i} W_{ij}(1 + V_j)(1 - V_j) = 0. \quad (37)$$

Equation (34) is the energy function of a three-valued neural network for the $(-1, 0, 1)$ model. Equations (35), (36), and (37) are three hyperplanes for the $(-1, 0, 1)$ model.

6 CONCLUSIONS

This paper proposed a three-valued (0, 1, and $\frac{1}{2}$ neural network that 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 concepts of energy function, hyperplanes of neurons, and update rules of a neuron's states were extended naturally. 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 the Boltzmann machines was presented, where the update rules were modified by introducing state probabilities that are functions of a temperature.

We have also introduced a more general three-valued model where neurons assume three arbitrary values of V^0, V^1, and V^X ($V^0 < V^X < V^1$). These values, V^0, V^1, and V^X, correspond to the logic values 0, 1, and X, respectively. Outputs of neurons in the original Hopfield model take all continuous values between 0 and 1. We have discussed the dynamics of neurons in both the discrete and continuous domains.

REFERENCES

[1] J.J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two State Neurons," *Proc. of National Academy of Sciences*, May 1984, pp. 3088–3092.

[2] J.J. Hopfield, and D.W.Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, Vol. 52, pp. 141–152, 1985.

[3] S.T. Chakradhar, M.L. Bushnell, and V.D. Agrawal, "Automatic Test Generation Using Neural Networks," *Proc. of ICCAD'88*, 1988, pp. 416–419.

[4] M.R. Garey, and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W.H. Freeman and Company, 1979.

[5] J.P. Roth, "Diagnosis of Automata Failures: A Calculus and a Method," *IBM Journal of Research and Development*, Vol. 10, pp. 278–291, July 1966.

[6] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Transactions on Computers*, Vol. C-30, No. 3, pp. 215–222, March 1981.

[7] H. Fujiwara, and T. Shimono, "On the Acceleration of Test Pattern Generation Algorithms," *IEEE Transactions on Computers*, Vol. C-32, No. 12, pp. 1137–1144, December 1983.

[8] M.H. Schulz, and E. Auth, "Advanced Automatic Test Pattern Generation and Redundancy Identification Techniques," *Digest of Papers, FTCS-18*, June 1988, pp. 30–35.

[9] S. Geman, and D. Geman, "Stochastic Relaxation, Gibbs Distributions and the Baye-

sian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 721–741, 1984.

[10] G.E. Hinton, T.J. Sejnowski, and D.H. Ackley, "Boltzmann Machines: Constraint Satisfaction Networks that Learn," Technical Report CMU-CS-84-119, Carnegie-Mellon University, May 1984.

Hideo Fujiwara received his B.E., M.E., and Ph.D. degrees in Electronic Engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively.

He is currently a Professor at the Department of Computer Science, Meiji University, Kawasaki, Japan. In 1981 he was a Visiting Research Assistant Professor at the University of Waterloo, and in 1984 he was a Visiting Associate Professor at McGill University, Canada.

His research interests include logic design, design for testability, test pattern generation, fault simulation, built-in self-test, computational complexity, parallel processing, and neural networks for design and test. He is the author of *Logic Testing and Design for Testability* (MIT Press, 1985). He was the Far East International Editor of IEEE Design and Test of Computers and is now the Editor of IEICE Transactions on Information and Systems. He was a member of program committees such as ICCAD and FTCS, and now the Vice-Chair of Asian Subcommittee of IEEE International Test Conference.

He is a fellow of the IEEE, a member of the IEICE and the IPS of Japan.