Learning Algorithms Which Make Multilayer Neural Networks
Multiple-Weight-and-Neuron-Fault Tolerant

Tadayoshi HORITA\textsuperscript{a),} Itsuo TAKANAMI\textsuperscript{b)}, Members, and Masatoshi MORI\textsuperscript{c),} Nonmember

SUMMARY Two simple but useful methods, called the deep learning methods, for making multilayer neural networks tolerant to multiple link-weight and neuron-output faults, are proposed. The methods make the output errors in learning phase smaller than those in practical use. The abilities of fault-tolerance of the multilayer neural networks in practical use, are analyzed in the relationship between the output errors in learning phase and in practical use. The analytical result shows that the multilayer neural networks have complete (100\%) fault-tolerance to multiple weight-and-neuron faults in practical use. The simulation results concerning the rate of successful learnings, the ability of fault-tolerance, and the learning time, are also shown.

key words: multilayer neural network, fault-tolerance, weight fault, neuron fault, multiple fault

1. Introduction

An artificial neural network is an information or signal processing system. It is composed of a large number of simple processing elements, called artificial neurons, and they are interconnected by direct links called connection weights. They cooperate to perform parallel distributed processing, in order to solve a desired computational task. One of the attractive features of artificial neural networks is their capability to adapt themselves to special environment conditions, by “training” their connection strengths (weights). Especially, feed-forward neural networks with neurons arranged in layers, called the multilayer neural networks, are widely used in computational or industrial fields. Furthermore, as VLSI technology has developed, the interest in implementing them in hardware is growing. In this case, there is the possibility of low yield and/or reliability of the system, if there is no strategy for coping with defects or faults.

On the other hand, it may be thought that a multilayer neural network, which is proposed as a model for the cerebral neural network, has a potential ability of fault-tolerance. For this point, Phatak and Koren discussed the fault-tolerance through replications\cite{1}. By the benchmark test for the Sonar nets\cite{2}, they showed that more than 99\% of all possible single weight faults, which are stuck at \(+W\), \(0\), or \(-W\), are tolerated without any additional redundancy, but complete (100\%) fault tolerance is not achieved even at

6 extra replications. Furthermore, Nijhuis et al. showed that fault-tolerance behavior is not self-evident, but it must be activated by an appropriate learning scheme\cite{3}. Since then, many ideas to make the multilayer neural network fault-tolerant have been studied in the literature, e.g., (see \cite{1}\textendash \cite{13}). These works evaluated the fault-tolerance from the viewpoint of stuck-at faults of link weights or outputs of neurons, but not simultaneous faults. Besides these works, Takanami and Oyama proposed a learning algorithm, which makes multilayer neural networks fault-tolerant to multiple weight faults, with their values being arbitrary in an interval which is specified by two extreme values\cite{14}. In learning phase, two extreme ones are intentionally injected into the weights of the selected multiple links. However, we can find no work treating simultaneous faults of weights and neurons.

In this paper, two simple but useful methods, called the deep learning methods, for making multilayer neural networks tolerant to multiple link-weight and neuron-output faults, are proposed. The methods make the output errors in learning phase smaller than those in practical use. The abilities of fault-tolerance of the multilayer neural networks in practical use, are analyzed in the relationship between the output errors in learning phase and in practical use. The analytical result shows that the multilayer neural networks have complete (100\%) fault-tolerance to multiple weight-and-neuron faults in practical use. The simulation results concerning the rate of successful learnings, the ability of fault-tolerance, and the learning time, are also shown. In Sect. 2, the multilayer neural network and the back-propagation algorithm are briefly mentioned. In Sect. 3, the fault model is mentioned. In Sect. 4, the two deep learning methods are proposed, and the theorem concerning the ability of fault-tolerance of the multilayer neural networks obtained by these methods, is shown. In Sect. 5, the simulation results are shown. Finally in Sect. 6, the paper is concluded.

The paper is the extension of\cite{15}.

2. Multilayer Neural Network

Figure 1 shows a multilayer neural network (which is simply denoted as a network in the following). Each neuron in a layer is connected to all neurons in the adjacent layers through uni-directional links (synaptic weights). The first and the last layers are called the input and output layers respectively, and one between them is called a hidden layer.

In this paper, we deal with only networks which have one
hidden layer. The output of each neuron \( o_i \) is given by

\[
o_i = f(X_i)
\]

\[
X_i = \sum_{j=0}^{N_{pre}} w_{ij} \cdot u_j
\]

where \( w_{ij} \) is the value of the synaptic weight from the \( j \)-th neuron in the preceding layer to the \( i \)-th neuron, \( N_{pre} \) is the number of the neurons in the preceding layer connected to the \( i \)-th neuron, \( u_j \) is the output of the \( j \)-th neuron in the preceding layer, \( w_0 \) is the synaptic weight connected to the input \( u_0 = 1 \) corresponding to the threshold, \( X_i \) is called "inner potential" of the neuron, and \( f \) is the activation function (the sigmoid function) of a neuron defined by

\[
f(x : Temp) = \frac{1}{1 + \exp(-x/T_{emp})}
\]

where \( T_{emp} \) is a constant called "temperature".

The learning process called "back-propagation algorithm" is based on a steepest-descendant gradient rule. Let \( O \) be a set of indices of the neurons in the output layer, and let \( P \) be a set of indices of the learning input examples. The change of each weight for the \( p \)-th learning input example (named \( w^p_{ij} \)) is done as follows:

\[
\Delta w^p_{ij} = -\eta \cdot \frac{\partial E_p}{\partial w_{ij}}
\]

where \( E_p = \sum_{i \in O} (t^p_i - o^p_i)^2 / 2 \) is the learning output error of the \( t \)-th neuron in the output layer. \( t^p_i \) is the output of the \( t \)-th neuron in the output layer for the \( p \)-th learning input example \((i \in O \land p \in P)\), \( o^p_i \) is the output of the \( i \)-th neuron in the output layer for the \( p \)-th learning input example, and \( \eta \) is a parameter of a positive real number.

In addition, the algorithm with momentum is used as follow.

\[
\Delta w(T_{ine}) = \Delta w + p_m \cdot \Delta w(T_{ine} - 1)
\]

where \( \Delta w \) is obtained by the Eq. (4). \( \Delta w(t) \) is the value of \( \Delta w \) at time \( t \), \( p_m \) is the so-called momentum and \( 0 \leq p_m \leq 1 \). Then, the weight modification is repeated until the following condition is satisfied.

\[
\max_{p \in P, i \in O} (o^p_i - o^p_i)^2 < e_{bp}^2
\]

where \( e_{bp} \) is called the output error in learning phase. If a network obtained by a learning with \( P \) and \( e_{bp} \) satisfies this condition, the learning is said to have finished successfully and the network obtained is called to be "successful" in learning phase in terms of \( P \) and \( e_{bp} \) (or "successful" in short if no confusion occurs).

3. Fault Model

**Definition 1:** (Multiple fault) A weight fault is a pair \((i, x)\) of \( i \) and \( x \), where \( i \) and \( x \) denote the index of a weight and the value of a faulty weight, and they are said to be the index and the value of the weight fault, respectively. A neuron fault is a pair \([j, y]\) of \( j \) and \( y \), where \( j \) and \( y \) denote the index of a neuron and the output value of a faulty neuron, and they are said to be the index and the value of the neuron fault, respectively. A successful network is said to have a weight fault \((i, x)\) if the value of the weight with index \( i \) is stuck to \( x \). Similarly, a successful network is said to have a neuron fault \([j, y]\) if the output of the neuron with index \( j \) is stuck to \( y \). A set of faults \( F \) is called a multiple fault if all the faults in \( F \) occur simultaneously. Let \( \hat{N}_F = \{ j | [j, y] \in F \} \) and let \( \hat{W}_F = \{ i | (i, x) \in F \} \). Then, \((\hat{N}_F, \hat{W}_F)\) is called the index set of \( F \).

The three important assumptions concerning faults are denoted as follows.

**Assumption 1:** (the range of faults)
Only neurons in the hidden layer and any weights may be faulty, and other parts (that is, neurons in the input and output layers) are fault-free.

**Assumption 2:** (the value of a weight fault)
The value of each weight fault is assumed to be in the range from -1 to 1.

**Assumption 3:** (the output value of a neuron fault)
The output value of each neuron fault is assumed to be in the range from 0 to 1.

In the following, we mention that these assumptions are reasonable.

1. Concerning Assumption 1, usually, it is assumed that faults occur at neurons themselves, weights, and interconnecting links (see Fig. 2). It can be considered as
a weight or a neuron fault that a link snaps or is stuck to some value. It is natural to assume that neurons in the input layer are fault-free because they are only input terminals, that is, so simple circuits. Next, faults of neurons in the output layer are fatal indeed. But this paper deals with the cases that they are fault-free, from the reason that making each neuron in the output layer stronger (that is, more fault-tolerant) than one in the hidden layer, at the fabrication time, is a practical choice, as the number of neurons in the output layer is small.

2. Concerning Assumption 2, if a network is realized with hardware and the value of a weight is assumed to be in a range specified by two finite values \(-a\) and \(+a\), we have only to normalize it by dividing it by \(+a\). If it happens to become so large, the value through it is saturated and hence, bounded by the maximum (minimum) voltage or current in the circuit.

3. The validation of Assumption 3 is like that of Assumption 2.

The concept of fault-tolerance in a network is defined as follows.

**Definition 2: (Fault-tolerance)** If a successful network which has a multiple fault \(F\) satisfies the equation,

\[
\max_{p \in F, j \in \mathcal{O}} (t_{pj}^p - o_j^p)^2 < e_f^2,
\]

the network is called to be fault-tolerant to \(F\) within the output error of \(e_f\) in practical use. \(F\) which does not satisfy the Eq. (7) is called to be dangerous.

\[\square\]

4. **Proposed Methods**

Two deep learning methods (denoted as “Deep-LM-A” and “Deep-LM-B” for short in the following) to make a network fault-tolerant are proposed. Both methods are based on the normal back-propagation algorithm mentioned in Sect. 2, but \(T_{emp}\) in the activation function of neurons in the output layer or the output error in practical use is different from those in learning phase. Actually, \(T_{emp}\) and the output error are set to 1 and \(e_f\) in practical use, respectively. Furthermore, they modify weights in learning phase in the different way from that in the normal back-propagation algorithm, and this modification is called \(W|1|-process\). The detail is as follows. Note that \(T_{emp}\) in the activation function of each neuron in the hidden layer is set to 1 in both learning phase and practical use.

- **Deep-LM-A**
  
  In learning phase,
  
  - \(T_{emp}\) in the activation function of each neuron in the output layer is set to 1.
  - The output error \(e_p\) is set to \(e_A = \frac{1}{(e_f^2)^{N_d-1}}\exp(N_d)+1\), where \(N_d\) is a positive integer parameter representing the degree of fault-tolerance as shown in Theorem 1, which is mentioned later in this section. \(e_A\) is derived from the equation \(e_A = 1 - f(x_e + N_d : 1)\) (that is, \(T_{emp} = 1\)), where \(X_e\) is the value of \(x\) which satisfies \(f(x : 1) = (1 - e_f)\) and hence, \(X_e = \ln(e_f^{-1})\).
  
  - Each weight is modified according to the normal back-propagation algorithm, but is set to 1 (-1) if it is greater (less) than 1 (-1). This process is called “\(W|1|-process\)”. The purpose of this execution is to make each weight value in the range from \(-1\) to 1.

  In practical use,
  
  - \(T_{emp}\) in the activation function of each neuron in the output layer is set to 1.
  - The output error is set to \(e_f\).

- **Deep-LM-B**

  In learning phase,
  
  - \(T_{emp}\) in the activation function of each neuron in the output layer is set to \(T_{emp}(B) = (1 + N_d/X_e)\), which is the value of \(T_{emp}\) which satisfies \(f(x_e + N_d : T_{emp}) = (1 - e_f)\).
  
  - Each weight is modified as in the Deep-LM-A.

  In practical use,
  
  - \(T_{emp}\) and the output error are set as in the Deep-LM-A.

Figure 3 shows the activation function in the Deep-LM-A with \(N_d\), \(X_e\), \(e_f\), and \(e_A\), and that in the Deep-LM-B. A network in practical use via a successful learning, which is obtained by the Deep-LM-A (or Deep-LM-B) with the degree of fault-tolerance \(N_d\), is denoted as “MLN(\(N_d\))-A” (or “MLN(\(N_d\))-B”) for short.

According to these notations, note that both MLN(0)-A and -B are equal to a network obtained by a normal back-propagation algorithm with the execution of \(W|1|-process\),
because $e_A = e_f$, and $T_o(B) = 1$ when $N_d = 0$.

The following theorem concerns the ability of fault tolerance of MLN($N_d$)-As and MLN($N_d$)-Bs.

**Theorem 1:** Let an MLN($N_d$)-A or MLN($N_d$)-B have only one neuron in the output layer. Then, it is fault-tolerant to any multiple fault $F$ whose index set is $(\tilde{N}_F, \tilde{W}_F)$ which satisfies

$$|\tilde{N}_F| + 2|\tilde{W}_F| - |\tilde{N}_F \cap \tilde{W}_F| \leq N_d$$  \(8\)

where $\tilde{N}_F$ is a set of indices of neuron faults in the hidden layer, $\tilde{W}_F$ is a set of indices of weight faults between the hidden and the output layers, and the neuron and weight faults may take any values in Assumptions 2 and 3, respectively. $|\text{a set}|$ denotes the number of elements of the set. Note that a weight fault between the input and the hidden layers is treated as a neuron fault in the hidden layer, to which the line with the faulty weight is connected, and the index of the neuron is included in $\tilde{N}_F$, because the fault has a direct influence only on the output of that neuron.

**Proof:** Let $X$ and $\hat{X}$ be the inner potentials of the neuron in the output layer, when the network is fault-free, and when it has a multiple fault $F$, respectively. $X$ and $\hat{X}$ are given as follows.

$$X = \sum_{i \in I_H} w_i \cdot h_i$$  \(9\)

where $I_H$ is the set of indices of neurons in the hidden layer, $h_i$ is the output of the neuron with index $i$, and $w_i$ is the weight of the link from the neuron with index $i$.

$$\hat{X} = \sum_{k \in \tilde{W}_F, k \in \tilde{N}_F} w_k \cdot h_k + \sum_{k \in \tilde{W}_F, k \in \tilde{N}_F} w_k' \cdot h_k'$$

$$+ \sum_{k \in \tilde{W}_F, k \in \tilde{N}_F} w_k \cdot h'_k + \sum_{k \in \tilde{W}_F, k \in \tilde{N}_F} w_k' \cdot h_k$$  \(10\)

where $w_i$ ($w'_i$) is the value of the $i$-th weight when it is healthy (faulty), and $h_i$ ($h'_i$) is the value of the $i$-th neuron in the hidden layer when it is healthy (faulty).

Now, if the network is an MLN($N_d$)-A, when $t_p = 1$, $(1 - f(X : 1)) < e_A = (1 - f(X_e + N_d : 1))$ in learning phase. Thus, the next equation is true because $f(x : 1)$ is a monotonically increasing function.

$$X > X_e + N_d$$  \(11\)

When $t_p = 0$, the following equation is true because $f(X : 1) < e_A = (1 - f(X_e + N_d : 1)) = f(- (X_e + N_d : 1))$ in learning phase.

$$X < -(X_e + N_d)$$  \(12\)

Similarly, the Eqs. (11) and (12) are also true in an MLN($N_d$)-B, because $(1 - f(X : T_o(B))) < e_f = (1 - f(X_e + N_d : T_o(B)))$ in learning phase when $t_p = 1$ and $T_o(B) = 1 + \frac{N_d}{N_e}$, and $f(X : T_o(B)) < e_f = (1 - f(X_e + N_d : T_o(B))) = f(- (X_e + N_d : T_o(B)))$ when $t_p = 0$.

From the Eqs. (9) and (10),

$$\hat{X} - X = \sum_{k \in \tilde{W}_F, k \in \tilde{N}_F} (w_k' - w_k) \cdot h_k + \sum_{k \in \tilde{W}_F, k \in \tilde{N}_F} w_k \cdot (h'_k - h_k) + \sum_{k \in \tilde{W}_F, k \in \tilde{N}_F} (w_k' \cdot h'_k - w_k \cdot h_k)$$  \(13\)

In addition, the following equations are true because of the $W_{11}$-process, Assumptions 2 and 3, and $h_k$ being in the range from 0 to 1.

$$-2 \leq (w_k' - w_k) \cdot h_k \leq 2$$

$$-1 \leq w_k \cdot (h'_k - h_k) \leq 1$$

$$-2 \leq (w_k' \cdot h'_k - w_k \cdot h_k) \leq 2$$

Furthermore,

$$N_{13} + N_{15} = |\tilde{W}_F|,$$

$$N_{14} + N_{15} = |\tilde{N}_F|,$$

and

$$N_{15} = |\tilde{N}_F \cap \tilde{W}_F|,$$

where $N_{13}$, $N_{14}$, and $N_{15}$ are the number of the additional terms in Eqs. (13), (14), and (15), respectively. Therefore,

$$2N_{13} + 2N_{14} + 2N_{15} = |\tilde{N}_F| + 2|\tilde{W}_F| - |\tilde{N}_F \cap \tilde{W}_F|$$

Thus,

$$-(|\tilde{N}_F| + 2|\tilde{W}_F| - |\tilde{N}_F \cap \tilde{W}_F|) \leq (\hat{X} - X) \leq (|\tilde{N}_F| + 2|\tilde{W}_F| - |\tilde{N}_F \cap \tilde{W}_F|).$$

Then

$$X - (|\tilde{N}_F| + 2|\tilde{W}_F| - |\tilde{N}_F \cap \tilde{W}_F|) \leq \hat{X} \leq X + (|\tilde{N}_F| + 2|\tilde{W}_F| - |\tilde{N}_F \cap \tilde{W}_F|).$$

From this equation and the Eqs. (11) and (12),

- $\hat{X} \geq X_e + (N_d - (|\tilde{N}_F| + 2|\tilde{W}_F| - |\tilde{N}_F \cap \tilde{W}_F|))$ (if $t_p = 1$),
- $\hat{X} \leq -(X_e + (N_d - (|\tilde{N}_F| + 2|\tilde{W}_F| - |\tilde{N}_F \cap \tilde{W}_F|)))$ (if $t_p = 0$).

From these equations and the Eq. (8), if $t_p = 1$, $f(\hat{X} : 1) \geq f(X_e : 1)$. Since $f(X_e : 1) = 1 - e_A$, $(t_p - f(\hat{X} : 1))^2 \leq (t_p - f(X_e : 1))^2 = e_A^2$. Similarly, if $t_p = 0$, $f(\hat{X} : 1) \leq f(X_e : 1)$. Since $f(X_e : 1) = 1 - f(X_e : 1)$, $(t_p - f(\hat{X} : 1))^2 \leq (t_p - f(X_e : 1))^2 = (1 - f(X_e : 1))^2 = e_A^2$. Therefore, the theorem is proved.

**Corollary:** Let an MLN($N_d$)-A or MLN($N_d$)-B have $m$ neurons in the output layer. Let $F$ be any multiple fault whose index set is $(\tilde{N}_F, \bigcup_{k=1}^m \tilde{W}_F^k)$, where $\tilde{W}_F^k$ is the set of indices of faulty weights which are connected to the $k$-th neuron in the output layer. Then, it is fault-tolerant to $F$ if the following equations are satisfied for all $k$ ($1 \leq k \leq m$), where we consider $\tilde{W}_F^k$ like $\tilde{W}_F$ in Theorem 1 when we focus on the $k$-th neuron in the output layer.

$$|\tilde{N}_F| + 2|\tilde{W}_F^k| - |\tilde{N}_F \cap \tilde{W}_F^k| \leq N_d$$  \(16\)

**5. Simulation Results**

The experiments are executed in a PC/AT machine with
a Pentium 4 (3.2 GHz), 512 MB RAM, and Fedora Core 5 Linux OS. Simulation programs are described in C language, and the gcc version 4.1.0 compiler is used.

Figure 4 and Table 1 show the learning input and output examples, respectively. The input examples are called P₁, P₂, …, and P₂₀, respectively, and each of them consists of 100 bit signals (black=1 and white=0). Two sets of learning input examples denoted as {IEx-10} and {IEx-20} are used in the simulation, where {IEx-10}={Pᵢ|₁≤ᵢ≤₁₀} and {IEx-20}={Pᵢ|₁≤ᵢ≤₂₀}. Note that when {IEx-10} is used, the number of neurons in the output layer is four, and tₚ₅ in Table 1 is ignored.

Other parameters are as follows.

- The number of neurons in the input layer is 100.
- The number of neurons in the hidden layer is 50 or 100.
- eᵣₛ = 0.10
- η = 0.10
- pₘ = 0.50

Each weight is initialized by a pseudo-random number (MT19937 [16]), whose range is from -0.1 to 0.1.

Concerning this case, the data of the rate of successful learnings, the learning times, and the ability of fault-tolerance, which are obtained by the simulations, are given as follows, including the considerations and the comparison between the Deep-LM-A and -B.

(1) Rate of successful learnings

It depends on Nₜ and an initialization of weights as well as the above parameters whether a learning finishes successfully. As Nₜ increases, it becomes difficult that a learning finishes successfully.

Figure 5 shows the rate of successful learnings (denoted as “Success”) in 100 trials for each Nₜ, where each trial begins with randomly chosen initial weight values. The labels “A” and “B” indicate MLN(Nₜ)-A and MLN(Nₜ)-B, respectively. The first number sandwiched by “(” and “)” is the number of neurons in the hidden layer. The second is {IEx-10} or {IEx-20}.

The figure shows that the maximum values of Nₜs, for which learnings finish successfully by the Deep-LM-A and -B, are 3 for A(50, IEx-20), 4 for A(100, IEx-10), 5 for B(50, IEx-20), 15 for B(100, IEx-10), respectively. Since Nₜ indicates the degree of fault-tolerance from Theorem 1, this implies that the Deep-LM-B can make a network more fault-tolerant than the Deep-LM-A if the amount of resources such as neurons and weights is the same. This is remarkable between B(100, IEx-10) and A(100, IEx-10), but not so between B(50, IEx-20) and A(50, IEx-20). This difference should be explained as follows.

In the learning methods proposed here, it can be assumed that the ability that a network gets by learning is divided into that of memorizing the learning examples and that of fault-tolerance. Then, in case of B(100, IEx-10) and A(100, IEx-10), since the numbers of neurons and weights are large and conversely the number of the learning examples is small, the part used for memorizing the learning examples is small, and hence the part used for fault-tolerance becomes large. Therefore, it can be thought that it is easy to activate the ability of fault-tolerance, and hence the difference in activating the ability of fault-tolerance appears remarkably between the Deep-LM-B and -A. On the other hand, in case of B(50, IEx-20) and A(50, IEx-20), since the numbers of neurons and weights are small and many of them...
are used for memorizing the learning examples, the part used for fault-tolerance becomes small. Therefore, it can be thought that it is not easy to activate the ability of fault-tolerance, and hence the difference in activating the ability of fault-tolerance does not appear remarkably between the Deep-LM-B and -A. From the discussion above, it can be thought that the Deep-LM-B is superior to the Deep-LM-A, concerning easiness of activating the ability of fault-tolerance, especially in case that there is a lot of resources.

(2) Learning time

Figure 6 shows that the relation between the learning time and \( N_d \) for each case shown in Fig. 5, when \( \text{Success} \) is not 0. The learning time is the average value for 100 learning trials.

From the figure, it can be seen that the Deep-LM-B is also superior to the Deep-LM-A in terms of learning time. This difference should be explained as follows. From the Eq. (4), \( \Delta w_{ij}^p = \eta \cdot (t_i^p - o_i^p) \cdot \frac{\partial f(x : T_{emp})}{\partial x_i} \cdot h_j \), that is, \( \Delta w_{ij}^p \) is proportional to the output error \( (t_i^p - o_i^p) \). As shown in Fig. 3, when a learning almost finishes, that is, when an inner potential \( X \) almost reaches \( \sum (X_e + N_d) \) (or \( -\sum (X_e + N_d) \)), the absolute value of \( \Delta w_{ij}^p \) for the Deep-LM-A is much smaller than that for the Deep-LM-B. Thus, as shown in Fig. 5, the rate of successful learnings for the Deep-LM-A is worse than that for the Deep-LM-B, because the Deep-LM-A requires more times of weight modifications than the Deep-LM-B.

In addition, the figure shows that as \( N_d \) becomes larger, the learning time becomes exponentially longer. Especially, it increases rapidly when \( N_d \) is almost the limit to which the learning finishes successfully. From the figure, it can be seen that at the limit to which the learning finishes successfully by the Deep-LM-B (and -A), the learning time takes several hundred times that when \( N_d = 0 \) (which is taken by a standard back-propagation algorithm). In general, the learning time and the degree of fault-tolerance are in the relation of the trade-off, and the Deep-LM-B and -A takes the learning time more to make a network more fault-tolerant.

(3) Ability of fault-tolerance

We check the abilities of MLN-As and MLN-Bs by injecting faults into them.

For a network, let \( I_H \) be the set of indices of neurons in the hidden layer. In order to make the discussion simple, we focus on a neuron \( o \) in the output layer, and suppose that the index of the weight, between the \( i \)-th neuron in the hidden layer and the neuron \( o \), is also \( i \). Let \( \bar{N} \) and \( \bar{W} \) be subsets of \( I_H \). We define multiple faults \( F(N, W, 1) \) and \( F(N, W, 0) \) as follows.

\[
F(N, W, 1) = \{(j, 0) \mid j \in \bar{N} - \bar{W}, w_j \geq 0\} \quad \cup \quad \{(j, 1) \mid j \in \bar{N} - \bar{W}, w_j < 0\} \quad \cup \quad \{(j, -1) \mid j \in \bar{W} - \bar{N}, w_j \geq 0\} \quad \cup \quad \{(j, 1) \mid j \in \bar{W} - \bar{N}, w_j < 0\}
\]

\[
F(N, W, 0) = \{(j, 1) \mid j \in \bar{N} - \bar{W}, w_j \geq 0\} \quad \cup \quad \{(j, 0) \mid j \in \bar{N} - \bar{W}, w_j < 0\} \quad \cup \quad \{(j, 1) \mid j \in \bar{W} - \bar{N}, w_j \geq 0\} \quad \cup \quad \{(j, 0) \mid j \in \bar{W} - \bar{N}, w_j < 0\}
\]

Lemma 1: Let the \( p \)-th learning input example be input to an MLN(\( N_d \))-A (or MLN(\( N_d \))-B), and let \( t_P(=0 \text{ or } 1) \) be the \( p \)-th learning output example of the neuron \( o \). We suppose that a multiple fault \( F \) whose index set is \( (\bar{N}_F, \bar{W}_F) \) occurs in it, and the output of \( o \) is \( o_F^p \). Then, if \( (t^p - o_F^p)^2 \geq e_{f_1}^2 \), \( (t^p - o_F^p)^2 \geq e_{f_1}^2 \), where \( o_F^p(\bar{N}_F, \bar{W}_F, t_P) \) is the output of the neuron \( o \), when the multiple fault \( F(\bar{N}_F, \bar{W}_F, t_P) \) occurs.

Proof: From the definition of \( F(\bar{N}_F, \bar{W}_F, t_P) \), it is easily proved that if \( t^p = 1 \), \( o_F(\bar{N}_F, \bar{W}_F) \leq o_F^p \), and if \( t^p = 0 \), \( o_F(\bar{N}_F, \bar{W}_F) \geq o_F^p \). Hence, in either case, if \( (t^p - o_F^p)^2 \geq e_{f_1}^2 \), \( (t^p - o_F^p)^2 \geq e_{f_1}^2 \).

From the above lemma, we have the following property.

Property 1: In any fixed neuron in the output layer, if a multiple fault \( F \) whose index set is \( (\bar{N}_F, \bar{W}_F) \) is dangerous for the \( p \)-th learning input example, so is the multiple fault \( F(\bar{N}_F, \bar{W}_F, t_P) \).

As a general case, if a multiple fault \( F_M \) is dangerous for the \( p \)-th learning example, there is a neuron \( o \) with index \( i \) in the output layer, and a learning example \( p \) such that \( (t_i^p - o_i^p)^2 \geq e_{f_1}^2 \). Let \( F \) be the multiple fault, consisting of the neuron faults, and the weight faults in \( F_M \) which are between the hidden layer and \( o \). Then, the multiple fault \( F(\bar{N}_F, \bar{W}_F, t_P) \) made as above is also dangerous. In other words, if \( F(\bar{N}_F, \bar{W}_F, t_P) \) is not dangerous for the \( p \)-th learning example, no multiple fault whose index set is \( (\bar{N}_F, \bar{W}_F) \) is dangerous for the same example. In this meaning, we say that \( F(\bar{N}_F, \bar{W}_F, t_P) \) is an essential multiple fault among multiple faults, whose index set is \( (\bar{N}_F, \bar{W}_F) \) for the \( p \)-th learning example. From the consideration above, we perform simulations to check whether an essential multiple fault is dangerous for MLN(\( N_d \))-As and MLN(\( N_d \))-Bs as follows.

1. Set \( k = 0 \) and \( N_e(k) = 0 \)
2. Do the following by trial times where trial is a positive integer.

(i) Choose $i \in O$ and $\bar{N}$ and $\bar{W}$ from $I_H$ at random which satisfy $|\bar{N}| + 2|\bar{W}| - |\bar{N} \cap \bar{W}| = k$.

(ii) For each $p \in P$, do the following;

(a) input the $p$-th learning input example,

(b) inject the multiple fault $F(\bar{N}, \bar{W}, t^p)$ and compute the output $o^p_i$ of the neuron with index $i$.

(c) If $(t^p_i - o^p_i)^2 \geq e^2_i$, $F(\bar{N}, \bar{W}, t^p)$ is dangerous, increase $N_e(k)$ by 1.

3. Set the rate of dangerous essential multiple faults, in the essential multiple faults in terms of $k$, to $N_e(k)/\text{trial}/|P|$, which is denoted as $dF(k)$ in short, increase $k$ by 1 and go to 2.

Figures 7 and 8 show $dF(k)$s for MLN($N_d$)-A and MLN($N_d$)-B with |Ex-10| or |Ex-20|, where the numbers of neurons in the hidden layer are 100 and 50, respectively. Each $dF(k)$ is the average when a million tested ones per each $k$ are used, that is, $(\text{trial} \cdot |P|)$ is equal to a million. The labels “A($N_d$)” and “B($N_d$)” indicate MLN($N_d$)-A and MLN($N_d$)-B, respectively. The data for several $N_d$ values are omitted in these figures.

As we can see the curves of “A & B(0)”, especially note that there is a single fault, to which the multilayer neural network, obtained by the normal back-propagation algorithm with the execution of $W_{ij}$-process, is not fault tolerant, because $dF(k)$ of “A & B(0)” > 0 when $k = 1$. In addition, these figures show that $dF(k)$ is 0 when $k \leq N_d$ for both MLN($N_d$)-A and MLN($N_d$)-B, and it has been confirmed that no dangerous essential multiple fault is found if $k \leq N_d$ as proved in Theorem 1. Furthermore, it can be seen that the numbers of dangerous essential multiple faults increase as $k$ becomes greater than $N_d$.

Comparing the curves for MLN($N_d$)-B with those for MLN($N_d$)-A, the Deep-LM-B is superior to the Deep-LM-A in terms of fault-tolerance, because the curve of an MLN($N_d$)-B is under that of an MLN($N_d$)-A in each $N_d$. Especially in Fig. 8, the curves for MLN($N_d$)-B increase much slower than those for MLN($N_d$)-A. To know the reason, we have checked the averages of absolute values of the inner potentials in MLN($N_d$)-A and -B, and it has been found that the latter is about 8 to 15 % bigger than the former (the detail data is omitted), and this is caused by the difference concerning $\Delta \omega_{ij}^p$ when a learning almost finishes. This should cause the difference of fault-tolerance between two methods as shown in Figs. 7 and 8.

6. Conclusions

Two extended back-propagation algorithms, called the Deep-LM-A and -B, which make multilayer neural networks fault-tolerant to multiple faults of weights and neurons in the hidden layer, are proposed. The theorem concerning their fault-tolerances has been given.

The Deep-LM-A is algorithmically simpler compared to the Deep-LM-B, but results of simulations show that the latter is superior to the former in terms of the successful learning, learning time, and fault-tolerance.

In this paper, we have considered multilayer neural networks whose input and output are digital signals. If we need to consider multilayer neural networks whose input and output are analog signals, it will be in future work. However, for this case, if we digitize analog input and output signals using A/D or D/A converters, they can be processed in the multilayer neural networks considered here. From the viewpoint of engineering, the digitization is essential to realize fault-tolerance, thus we think that our methods can be practically applied to the general cases.

In addition, as the future work, another novel extended back-propagation algorithms using fault injection
techniques, which realize the fault-tolerance like the DeepLM-B, will be proposed, and comparisons among them, the DeepLM-A and -B, will be given. Further, an implementation method of digital fault-tolerant multilayer neural networks, using FPGAs (Field Programmable Gate Arrays), HDL (Hardware Description Language), and those algorithms, will be proposed. The case that the number of learning examples increases is also in the future work.

Acknowledgments

The author would like to thank the reviewers for useful comments and suggestions.

References


Tadayoshi Horita received his B.E. degree in electronic engineering, and M.E. degree in computer science and systems engineering, from from Yamaguchi university, in 1993 and 1995, respectively. He received his D.E. degree in engineering from Iwate university in 2000. He has been an assistant professor of Iwate university, from 1995 to Sep. 2001. He has been a lecturer of polytechnic university, Sagamihara, 229–1196 Japan, since Oct. 2001. He received two encouragement awards from the Tohoku branch of information processing society of Japan, in 1996 and 2000, respectively. His current interests are fault tolerant array processors, reconfigurable architectures, and artificial neural networks.

Itsuo Takanami received his M.E. degrees in electrical and communication engineering from Tohoku university, Sendai, Japan, in 1967 and 1970 respectively. In April 1970 he joined Tohoku university, where from 1970 to 1971 and from 1971 to 1974 he was an assistant professor and an associate professor, respectively. From 1974 to 1994 he was a professor at Yamaguchi University. From 1994 to 2000 he was a professor at Iwate University. In 2000 he joined Ichinoseki national college of technology where he was the president and in 2003 he retired from the college. He is an emeritus professor of Iwate University and of Ichinoseki national college of technology. His current interest is fault tolerant computing system, including artificial neural networks.

Masatoshi Mori has been a student of polytechnic university from 2003 to 2007. He is currently a vocational trainer of employment and human resources development organization of Japan. His current interest is fault-tolerant artificial neural networks.