Partially weight minimization approach for fault tolerant multi-layer neural networks

Takase Haruhiko, Kita Hidehiko and Hayashi Terumine
Department of Electrical and Electronic Engineering,
Faculty of Engineering, Mie University
1515, Kamihama-cho, Tsu, Mie, 514-8507 Japan
(takase@hayashi.elec.mie-u.ac.jp)

Abstract - We propose a new learning algorithm to enhance fault tolerance of multi-layer neural networks (MLNs). This method is based on the fact that strong weights make MLNs sensitive to faults. To decrease the number of strong connections, we introduce the new evaluation function for the new learning algorithm. The function consists of two terms, one is the output error and the other is the square sum of HO-weights (weights between the hidden layer and the output layer). The second term aims to decrease the value of HO-weights. By decreasing the value of only HO-weights, we enhance fault tolerance against the previous method.

I. Introduction

Artificial neural networks are often assumed that they have inherent fault tolerance, since they are constructed with a large number of units. However, many studies show that multi-layer neural networks (MLNs) with back propagation (BP) algorithm do not make optimal use of its fault tolerance. So, many researchers aim to bring out it.

We show typical three kinds of technique. The first is that the synaptic noise injection during training. It enhances fault tolerance and generalization of MLNs[1]. However it is difficult to decide proper noise distribution. The second is the fault injection method[2]. This method minimizes not only the squared error for the fault free MLN but also the squared error for the faulty MLN. As a result, fault tolerance of the MLN is brought out. Because this method calculates not only the output for the fault free MLN but also the output for faulty MLNs, it requires considerably computation time. The third is the augmentation of hidden units. It improve MLNs fault tolerance[3]. Because the augmentation operation includes duplication of hidden units, the MLN will be constructed with more than enough units.

In our previous papers[4][5], we proposed the new learning algorithm to enhance fault tolerance of MLNs. This method is based on the fact that strong weights make MLNs sensitive to faults. To decrease the number of strong weights, we introduced the evaluation function for the algorithm. It consists of not only the output error but also the sum of square of weights. With the evaluation function the learning algorithm minimizes not only the output error but also the value of weights. As a result, the number of weights which is sensitive to faults is decreasing with progress of the training.

In this paper, we enhance our previous method. To discuss the influence of the fault’s location on the output of MLNs, we divide weights of MLNs into two parts, weights between input layer and hidden layer (IH-weights) and one between hidden layer and output layer (HO-weights). Through the investigation of each part, we show the improved training method for fault tolerant MLNs.

II. Multi-layer neural networks

In this section, we simply explain about the MLN model in this paper and BP algorithm.

A. Network Structure

We consider conventional MLNs with a single hidden layer. MLNs are constructed with an input layer, a hidden layer and an output layer. There are no feedback connections and no connections from any input unit to any output unit.

The output of each input unit is equal to the value of its input. The output of each hidden unit and output unit are described as follows.

\[
\begin{align*}
y_i &= f(x_i) = \frac{1}{1 + \exp(-x_i)}, \\
x_i &= \sum_j w_{ij} y_j
\end{align*}
\]
where \( f(x_i), y_i, \) and \( w_{ij} \) are the output function, the output of the unit \( i, \) and the value of the weight from the unit \( j \) to the unit \( i, \) respectively.

B. BP Algorithm

BP algorithm is a well-known algorithm to realize desired mapping (equation (2)) with MLNs.

\[
(X_1^p, X_2^p, \ldots, X_K^p) \rightarrow (Y_1^p, Y_2^p, \ldots, Y_M^p), (p = 1, 2, \ldots, P)
\]

where we call \((X_1^p, X_2^p, \ldots, X_K^p)\) and \((Y_1^p, Y_2^p, \ldots, Y_M^p)\) as teacher input pattern and teacher output pattern, respectively.

This algorithm is formulated to minimize the evaluation function \( E(w) \) with the idea of the steepest decent method. \( E(w) \) is defined as the squared error between the actual outputs and the desired ones (equation (3)).

\[
E(w) = \sum_{p=1}^{P} \sum_{k \in K} \frac{1}{2} (y_k^p - Y_k^p)^2,
\]

where \( K \) and \( y_k^p \) are the set of output units and the output of unit \( k \) for teacher input pattern \( p, \) respectively.

The modification value of each \( w_{ij} \) at each step of a training is shown in equation (4).

\[
\Delta w_{ij}(n) = -\eta \frac{\partial E(w)}{\partial w_{ij}} + \alpha \Delta w_{ij}(n-1),
\]

where \( \Delta w_{ij}(n), \alpha, \) and \( \eta \) are modification value of \( w_{ij} \) at \( n \)th cycle of training, momentum and learning rate, respectively.

III. Fault tolerant MLN

Many researchers mention that both sufficient hidden units and a proper training algorithm are required to bring out fault tolerance of MLNs.

In this section, we explain the fault model, and the previous method[4][5].

A. Fault model

There are various candidates for faults in MLNs[6]. Since specific hardware implementations for the MLNs have not been discussed here, it is impossible to define the type of faults which might occur in MLNs. To remain generality, we adopt the fault which is loss of connection between two units, because it is a physically plausible type of faults [2][3]. We treat this type of faults as fixing the value of faulty connection’s weight to zero.

B. Learning algorithm for fault tolerant MLNs

BP algorithm only aims to reduce the difference between the actual outputs of the MLN and the desired ones as small as possible. So, the MLN which trained with BP algorithm does not always bring out its fault tolerance. In this section, we discuss what makes MLNs sensitive to faults and how to improve their fault tolerance.

First, we discuss about the MLN which has one faulty weight \( w_{ij}. \) The difference of \( y_i \) between the fault-free MLN and faulty one is shown as \( \Delta y_i \) in equation (5).

\[
\Delta y_i = |f(x_i) - f(x_i - w_{ij}y_j)|.
\]

Our purpose is to reduce the difference \( \Delta y_i \) as small as possible for all units and all input patterns. Because \( f() \) is monotone increasing function and \( y_i \) is changed with input patterns of the MLN, the strong connection brings significantly difference \( \Delta y_i \) on some units. Thus, we should make the absolute value of all weights as small as possible.

In the previous method, we defined the evaluation function \( E'(w) \) to evaluate not only the output error but also the square sum of all weights. BP algorithm with the evaluation function \( E'(w) \) should make MLNs insensitive to faults and bring out fault tolerance, because the algorithm minimize not only \( E(w) \) but also the square sum of all weights simultaneously.

\[
E'(w) = E(w) + \frac{\lambda}{2} \sum_{i,j} w_{ij}^2,
\]

where \( \sum_{i,j} \) means sum for all weights and \( \lambda \) is parameter to determine the influence of the additional term.

The modification value of each weight is shown in equation (7).

\[
\frac{\partial E'(w)}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left\{ E(w) + \frac{\lambda}{2} \sum_{i,j} w_{ij}^2 \right\}
= \frac{\partial E}{\partial w_{ij}} + \lambda w_{ij},
\]

\[
\Delta w_{ij}(n) = -\eta \frac{\partial E'(w)}{\partial w_{ij}} + \alpha \Delta w_{ij}(n-1).
\]

The parameter \( \lambda \) is defined as equation (8)[5].

\[
\lambda = \lambda' \frac{\text{grad}(E)}{\text{grad}(W)}, \quad W = \frac{1}{2} \sum_{i,j} w_{ij}^2,
\]

where \( \lambda' \) is a fixed parameter. With proper \( \lambda', \) our algorithm brings not only desired mapping but also fault tolerance. Otherwise, if \( \lambda' \) is too large, it should bring all weights to be zero. If \( \lambda' \) is too small, it should be no difference with BP algorithm.

0-7803-7278-6/02/$10.00 ©2002 IEEE
TABLE I

| Method | Target | \(\text{ave}(|w|)\) | \(\text{max}(|w|)\) | \(O_c\) |
|--------|--------|----------------|----------------|--------|
| BP     | \(w_{all}\) | 0.573 | 2.255 | 0.394 |
|        | \(w_{IH}\)  | 0.536 | 1.689 | 0.085 |
|        | \(w_{HO}\)  | 0.705 | 2.255 | 0.394 |
| Previous | \(w_{all}\) | 0.378 | 1.998 | 0.319 |
|        | \(w_{IH}\)  | 0.304 | 1.240 | 0.049 |
|        | \(w_{HO}\)  | 0.635 | 1.998 | 0.319 |

IV. Improvement of the previous method

In this section, we show our new method. First, we discuss the influence of the fault’s location on the output of the MLN. Then, we propose the new method for fault tolerant MLNs.

A. Discussion of the previous method

The purpose of the previous method is to decrease both the output error and the absolute value of all weights during the training. We discuss about the effect of each purpose.

To discuss the effect of the proposed method, we investigate the value of weights of MLNs trained with BP algorithm or the previous method. We divide weights into two parts which are weights between the input layer and the hidden layer (IH-weights), and one between the hidden layer and the output layer (HO-weights).

Figure 1 shows the histogram of the absolute value of a weight among IH-weights (solid line) and HO-weights (dotted line). Figure (a) and (b) are results of MLNs trained with BP algorithm and the previous method respectively. The simulation condition is the same as shown in section V.

Table I shows the average of the absolute value of weights (\(\text{ave}(|w|)\)), maximum (\(\text{max}(|w|)\)) and maximum output change (\(O_c\)) among faults on each part.

The output change \(O_c\) means the maximum change of an output of the MLN among unit brought by the fault of one connection, defined in equation (9).

\[
O_c = \max_{f \in F, p \in P, i \in O} (|y_i^p - y_i^{p^f}|),
\]

where \(F\), \(P\), \(O\), and \(y_i^{p^f}\) mean the set of supposed faults, the input pattern set, the set of units belongs to output layer, and faulty output of unit \(i\) for the input pattern \(p\) (fault \(f\) is occurred). Because the output change means the maximum change of output brought by the fault set, the small \(O_c\) means the fault tolerance of the MLN. In other words, we must minimize both the error \(E\) and the output change \(O_c\) among all faults for the fault tolerance of MLNs.

In the table, \(w_{all}\), \(w_{IH}\), and \(w_{HO}\) mean all weights, IH-weights, and HO-weights respectively. For example, the output change \(O_c\) for \(w_{IH}\) (or \(w_{HO}\)) means the output change among all faults on IH-weights (or HO-weights).

Table I and figure 1 show that the previous method decrease the value of IH-weights rather than the value of HO-weights. It indicates that the effect of decreasing the value of weights is strongly appeared on IH-weights, though the output change among faults on IH-weights is smaller than the one for HO-weights.

In table I, the output change among faults on IH-weights of BP algorithm is smaller than the output change among faults on HO-weights of previous method.
Since we must minimize the output change for fault tolerance of MLNs, we may not need to decrease the value of IH-weights, and should decrease the value of HO-weights rather than IH-weights to improve fault tolerance of the MLN.

B. Proposed method

In the section IV-A, we pointed out that the effect of the previous method works mostly on IH-weights. In this section, we improve the previous method by transforming its evaluation function.

To decrease error $E$ and the value of only HO-weights, we transform the equation (6) into the equation (10).

$$ E''(w) = E(w) + \frac{\lambda}{2} \sum_{i,j} w_{ij}^2, $$

where $\sum_{i,j}$ means sum for all HO-weights. The difference from the equation (6) is the target of the second term (all weights for the previous method). The modification value of each weights (equation (7) and (8)) is transformed in the same manner.

V. Experimental result

In this section, we show the effectiveness of the proposed method against BP algorithm and the previous method.

We use 5×7 binary images of 10 characters (0, 1, · · · , 9) as teacher input patterns (figure 2) and 10 output units corresponding to each character (table II). The number of input units, hidden units and output units are 35, 20, and 10, respectively. We train MLNs 50,000 cycles with $\eta = 0.05$, $\alpha = 0.9$, and $\lambda' = 0.7$.

A. The value of weights

In this section, we evaluate the value of weights.

Table II

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;0&quot;</td>
<td>0.9 0.1 0.1 · · · 0.1 0.1 0.1</td>
</tr>
<tr>
<td>&quot;1&quot;</td>
<td>0.1 0.9 0.1 · · · 0.1 0.1 0.1</td>
</tr>
<tr>
<td>&quot;2&quot;</td>
<td>0.1 0.1 0.9 · · · 0.1 0.1 0.1</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>&quot;7&quot;</td>
<td>0.1 0.1 0.1 · · · 0.9 0.1 0.1</td>
</tr>
<tr>
<td>&quot;8&quot;</td>
<td>0.1 0.1 0.1 · · · 0.1 0.9 0.1</td>
</tr>
<tr>
<td>&quot;9&quot;</td>
<td>0.1 0.1 0.1 · · · 0.1 0.1 0.9</td>
</tr>
</tbody>
</table>

Table III

| Method | Target | ave(|w|) | max(|w|) | $O_e$ |
|--------|--------|---------|---------|-------|
| Proposed | $w_{all}$ | 0.635 | 2.387 | 0.231 |
|         | $w_{IH}$  | 0.685 | 2.387 | 0.060 |
|         | $w_{HO}$  | 0.458 | 1.499 | 0.231 |

Figure 3 and table III show the same information for the MLN trained with the proposed method as figure 1 and table I respectively.

Though the value of IH-weights for the proposed method is greater than the value for BP algorithm, the value of HO-weights for the proposed method is less than the value for the previous method.

Compared with the MLN trained with the previous method, the output change among faults on IH-weights is increased, and the output among faults on HO-weights is decreased. Since the output change among faults on HO-weights is greater than the output change among faults on IH-weights, the output change among faults on all weights decreased.
This result shows that the proposed method decreases the sensitivity to faults of MLNs by decreasing the value of only HO-weights.

B. Degradation of the performance

In this section, we evaluate fault tolerance of MLNs by the degradation of the recognition rate against the increase of faults. For MLNs adapted to pattern recognition, the degradation of recognition rate against the increase of fault connections is a basic measurement for fault tolerance. The graceful degradation means fault tolerance of the MLN. Because fault tolerance means that the system with a few faults works well, it is suitable measurement.

The degradation of recognition rate is shown by figure 4. It contains 3 curves. They are results of MLNs which are trained by BP algorithm (solid curve), the previous method (dotted curve) and the proposed method (broken curve).

To evaluate these recognition rate, we use all teacher patterns and all disturbed patterns which are reversed only one bit from each teacher input pattern. Each output of MLNs is digitized with threshold 0.5 and compared with desired one. The horizontal axis means a rate of lost connections and the vertical one means a rate of miss recognition.

The result shows that the MLN trained with the proposed method is enhanced fault tolerance against one trained with other methods.

VI. Conclusion

We improved the evaluation function of the previous method [4][5] to enhance fault tolerance of MLNs. This function is consist of two terms which are output error and the square sum of weights between hidden layer and output layer (HO-weights). By decreasing the value of only HO-weights, we enhance the fault tolerance against the previous method.

References


