Enhancing the Generalization Ability of Backpropagation Algorithm through Controlling the Outputs of the Hidden Layers

Weishui Wan*, Kotaro Hirasawa*, Jinglu Hu* and Junichi Murata*

It is well known that backpropagation algorithm is one of the most basic algorithms in neural networks (NNs). Its role cannot be overestimated in the field of neural networks. In this paper we propose a new variant of backpropagation algorithm through controlling the outputs of the hidden layers. The proposed algorithm therefore provides better generalization results than the basic backpropagation algorithm. The added term to the criterion function has the following property: (1) Small added noises in the inputs to the networks will not give evident effects to the outputs of the networks; (2) Small added noises in the weight matrix except the one between the hidden layer and output layer will not give large effects to the outputs of the networks. In addition, simulation comparisons are also made between the new algorithm and some conventional regularization methods, such as Laplace and Gaussian regularizer. Simulation results on the two-spiral problem, function approximation problem and Iris data classification problem confirm this assertion.

Key Words: backpropagation, generalization, regularizer

1. Introduction

The backpropagation algorithm is one of the basic neural network learning algorithms, and many new algorithms are proposed based on this algorithm. Because of the simple property of this algorithm, the backpropagation algorithm is widely used in all applications and is incorporated in the variants of new algorithms. At the same time this algorithm has many shortcomings, typically the neural network trained by this algorithm has a bad property which is called "the distributed weights" 1), that is, the weight matrix of the trained neural network by backpropagation does not tend to be sparse, and this makes the neural networks generalize badly on the new unlearned data points. Another shortcoming is that backpropagation algorithm has no pruning ability, and this is also one of the reasons which produces the above unfavorable phenomena. Due to these reasons, once the initial structure is selected to be too larger for the problems, there will be no method to remedy this bad situation using backpropagation algorithm.

Therefore people have developed many new algorithms to tackle this problem. Among them, one of the often-used algorithms is the regularization method, such as structure learning with forgetting 1) or weight decay etc.. In the paper of reference 2) an extensive review was given, and in the paper of reference 3) the meaning of regularizer terms is explored from the viewpoint of statistical theory and some deep relations among the regularizer terms and parameters were given there, in addition some other interesting exploration of regularizer terms from functional analysis could be found in the paper 4), 5). Please refer to these papers for further knowledge on regularizer terms. Such kinds of algorithms are proposed through adding a regular term to the criterion function with the aim to make the weight distribute unevenly, and make the neural networks have better generalization.

But it should be pointed out that weight matrix is not the only factor which effects the performance of neural networks. This is particularly true for the multilayer neural networks. The other factors which should be considered are the outputs of the hidden layers. These factors should also be controlled in order to get a better generalization performance, especially the outputs of nodes in the layers close to the output layer should be considered, which have a much more direct effect on the total performance than the nodes close to the input layer. With these factors into consideration, in this paper a new kind of regularizer is proposed, the new regularizer has the following property: (1) Small added noises in the inputs to networks will not give evident effects to the outputs of the networks; (2) Small added noises in the weight matrix ex-

* Intelligent Control Laboratory, Graduate School of Information Science and Electrical Engineering, Kyushu University, Hakozaki 6-10-1, Higashi-ku, Fukuoka
(Received November 2, 2001)
(Revised February 14, 2002)

TR 0004/02/3804-0411 © 2001 SICE
cept the one between the hidden layer and output layer will not give large effects to the outputs of the networks. These properties made the network generalize better and converge much more quickly.

This paper is organized as follows, in Section 2 the backpropagation and a new algorithm are introduced, and detailed comparisons are made in this section. Section 3 is about the simulation results, observed phenomena and discussions. The final section of this paper is devoted to conclusions.

2. A New Algorithm for Improving Generalization

In this section we will review the backpropagation and introduce a new algorithm, which adds a term to the backpropagation, and we get enhanced generalization ability. In order to make this point much more clear, detailed comparisons will also be made.

2.1 Backpropagation Algorithm

The basic formula of backpropagation algorithm can be described as follows. Some notations will be first introduced.

- $x^l_i$: output of neuron $i$ in the $l$th layer
- $d_j$: desired output of neuron $j$ in the output layer
- $o_j^l = f(a_j^l)$: actual output of neuron $j$ in the $l$th layer
- $a_j^l = f(\sum_{i} W_{ji}^l * x_{l-1}^i)$: activation of neuron $j$ in the $l$th layer
- $W_{ji}^l$: weights connecting neuron $i$ in the $(l-1)$th layer to neuron $j$ in the $l$th layer
- $\Delta W_{ji}^l$: weight update value
- $\eta$: learning rate parameter
- $E$: global error
- $M$: number of layers in the neural network

Here the input layer is not included in the number of layers of the neural network. The global error can be expressed as

$$E = \frac{1}{2} \sum_j (d_j - o_j^M)^2$$

Then the basic formula of backpropagation is:

for the output layer

$$W_{ji}^M(t + 1) = W_{ji}^M(t) + \eta * \frac{\partial f(a_j^M)}{\partial a_j^M} * (d_j - o_j^M) * x_{i}^{M-1}$$

$$= W_{ji}^M(t) + \eta * \delta_j^M * x_{i}^{M-1}, \quad (2)$$

where $\delta_j^M$ is defined to be

$$\delta_j^M = \frac{\partial f(a_j^M)}{\partial a_j^M} * (d_j - o_j^M), \quad (3)$$

while for the hidden layer, the weight iteration formula is:

$$W_{ji}^l(t + 1) = W_{ji}^l(t) + \eta * \delta_j^l * x_{i}^{l-1}, \quad (4)$$

where

$$\delta_j^l = \sum_k \delta_{jk}^{l+1} * W_{kj}^{l+1}.$$  

(5)

Through these formulae the weight changes for the hidden nodes are carried out.

2.2 A New Algorithm

In the new algorithm a new term is added to the criterion function (1) used in the basic backpropagation. The flowchart of the new algorithm is the same as that of backpropagation algorithm. In the new algorithm, the criterion function is described as follows:

$$E_{new} = E + \alpha * \sum_{l \in \{k, l+1, \ldots, M-1\}} \sum_j (f(a_j^l)),$$  

(6)

or

$$E_{new} = E + \alpha * \sum_{l \in \{k, l+1, \ldots, M-1\}} \sum_j (-f(a_j^l)),$$  

(7)

where $j$ runs through all nodes in $l$th layer and $l$ runs through all hidden layers which are situated between $k$th layer and the layer in front of the output layer. $\alpha$ is a predefined parameter. Eq.(6) and (7) mean that parameters of the networks are trained to minimize or maximize the outputs of the hidden layers as well as to minimize the criterion function $E$. Also please note that here the summation is not made on all the hidden layers, because to our understanding the hidden layers between the input layer and $(k-1)$th layer play a little role than those near the output layer.

The weight iteration formula is changed as follows: for the output layer the formular is the same as eq.(2); while for the hidden layer,

$$W_{ji}^l(t + 1) = W_{ji}^l(t) + \eta * \delta_j^l * x_{i}^{l-1} - \alpha * \eta * v_j^l * x_{i}^{l-1}, \quad (8)$$

where

$$v_j^l = 0, \quad (9)$$

$$v_j^l = \frac{\partial f(a_j^l)}{\partial a_j^l} * \sum_k \sum_{l+1} W_{kj}^{l+1} + \frac{\partial f(a_j^l)}{\partial a_j^l}$$  

(corresponding to (6))

$$v_j^l = \frac{\partial f(a_j^l)}{\partial a_j^l} * \sum_k \sum_{l+1} W_{kj}^{l+1} - \frac{\partial f(a_j^l)}{\partial a_j^l}.$$  

(corresponding to (7))

where the subscript $k$ runs through all the nodes in the $(l+1)$th layer which are directly connected from the $j$th node in the $l$th layer. From the above formula it is
clear that computational burden in the new algorithm is the same as the backpropagation algorithm in one epoch, and only a term which controls the outputs of hidden layers is added. Therefore the new algorithm is easy to program. Furthermore, when the outputs of some hidden nodes are negligibly small, then these nodes can be temporarily deleted from the network, because we used eq.(6) and (0,1)-valued sigmoidal function. More concretely, we just set all weights connecting from these nodes to be zeros, therefore, the convergence of the networks can be accelerated.

2.3 Analysis of a Newly Added Term

Without loss of generality, a simple $N_0 - N_1 - 1$ network with two layers will be used in this section. Then using the above notations, this network can be expressed as follows:

$$o_k^2 = \sum_{k=0}^{N_1} W_{ik}^2 \ast f(a_k^1)$$

$$= \sum_{k=1}^{N_1} W_{ik}^2 \ast f(\sum_{j=1}^{N_0} W_{kj}^1 \ast x_0^0)$$

where $x_0^j$ is the j-th input to network. Now the meaning of a new added term in (6) or (7) is discussed. When the added term is minimized, the derivative of this term, for example, $\sum_k f(a_k^1)$ with respect to the $W_{kj}^1$ should be zero, that is

$$\frac{\partial f(a_k^1)}{\partial W_{kj}^1} = 0$$

Eq. (13) will be used in the following Taylor expression.

**Theorem** Under the minimization of a newly added term, i.e., under (13), the following conclusions are valid:

1. Small added noises in the inputs to networks will not give evident effects to the outputs of networks.
2. Small added noises in the weight matrix except the one between the hidden layer and output layer will not give large effects to the outputs of networks.

**Proof** Only the proof of the first conclusion is given here. The proof of the second conclusion is similar. Suppose the noise added to the input is $\epsilon_j$. The new output after using Taylor expansion formula is

$$o_k^2 + y = \sum_{k=1}^{N_1} W_{ik}^2 \ast f(\sum_{j=1}^{N_0} W_{kj}^1 \ast (x_0^0 + \epsilon_j))$$

$$= \sum_{k=1}^{N_1} W_{ik}^2 \ast f(a_k^1 + (\sum_{j=1}^{N_0} W_{kj}^1 \ast \epsilon_j))$$

$$= \sum_{k=1}^{N_1} W_{ik}^2 \ast f(a_k^1 + O((\sum_{j=1}^{N_0} W_{kj}^1 \ast \epsilon_j)^2))$$

To derive (14), terms $f(a_k^1 + (\sum_{j=1}^{N_0} W_{kj}^1 \ast \epsilon_j))$ for all $k$ are expanded by using Taylor expansion formula for $N_0$ variables up to the second order derivatives. From eq.(13) all the terms containing the first order derivatives in the expansion are zeros, so eq.(14) is obtained. Therefore

$$y = O((\sum_{j=1}^{N_0} W_{kj}^1 \ast \epsilon_j)^2)$$

Where the meaning of $O(x)$ is when $x$ tends to zero $O(x)$ also tends to zero.

This says that there is almost no effect when a small noise is added to the inputs of the networks. That is to say, after adding a new term in (6) or (7), the trained network dose not become sensitive to the changes of the input. Through this, the generalization of the network is enhanced. Second property says that the network is also not sensitive to small changes of weights except the ones between the hidden layer and output layer. Therefore large weight changes are favored than little changes in weight iteration.

2.4 Difference between the Proposed Method and Backpropagation Algorithm with Regularizer

It is well known that one can always get better generalization results than the ones without the regularizer. After adding a regularizer, the criterion becomes

$$E_{\text{new}} = E + \alpha \ast \text{Regularizer term}$$

where $\alpha$ is a regularization weight. The following forms of regularizers are often used:

$$\text{form 1} = \sum_{ij} |W_{ij}^l|,$$

$$\text{form 2} = \sum_{ij} (W_{ij}^l)^2,$$

$$\text{form 3} = \sum_{ij} \frac{(W_{ij}^l)^2}{1 + (W_{ij}^l)^2},$$

where $i, j$ is the index of nodes in neural networks, $l$ is the index of layers. $W_{ij}^l$ is the connection weights, $\alpha$ is the predefined parameter. Regularizer form 1 is Laplace regularizer, regularizer form 2 is Gaussian regularizer, and regularizer form 3 is the weight elimination regularizer. Relations between (6) and (17) ~ (19) are summarized as follows:

1. The first conclusion of Theorem is also valid for the regularizers, such as (17) ~ (19).
2. When computing the weight changes during the network training of the proposed methods, the relations with other weights are also considered, while there is no such consideration in conventional regularizer methods,
which just prune the network through direct control on the weight. This is one of the defaults pointed out in the review paper.

(3) The second conclusion of Theorem is not shared by regularizer (17) ~ (19).

The similar consideration is found in the paper in reference 6). From equation(3) in 6), one can see that when making pruning, the relations with the other weights are also considered. But in equation(3) of that paper, only counter effect (called a parametric lateral inhibition in that paper) is considered. Although actually the relations among nodes are very complicated, the shortcoming of that paper is that a concrete form of the relation is presupposed, while in our paper much more general relations are considered. (Refer to paper 6) for details)

3. Simulations of the Proposed Method

In the simulations, as was stated before, it is supposed that eq. (6) and (0, 1)-valued sigmoidal function are used. Furthermore, two kinds of data samples are used. The first one is the famous two-spiral problem. This problem is to try to classify two kinds of different data as shown in the Fig.1.

![Fig. 1 Two spiral problem](image)

The two-spiral problem is a very difficult problem to solve using the basic Backpropagation in a reasonable time. Please refer to other papers which try to solve this problem using backpropagation algorithm. The second problem is a function approximation problem. The third one is Iris data classification problem, which is a higher dimensional problem from machine learning databases. In these simulations we test the generalization ability of neural networks by using the new algorithm. Also different initial structures of neural networks are set and used in the simulations. This is to test the pruning ability of the new algorithm. The meaning of different initial structures here is that the various numbers of hidden layers and various numbers of nodes in each layer are used.

3.1 Two-spiral Problem

The sample data of this problem are produced by the following formula (pseudo code)

```plaintext
t=8;
r=t*3.1415926; //start radius
for (all sample i)
    x=r*sin(t)/(t*3.1415926);
y=r*cos(t)/(t*3.1415926);
    point(-x,y)--->class one;
    point(x,-y)--->class two;
r=r-(3.1415926/16); //radius reduce
```

In our simulation we set i = 192, not the commonly used i = 96, the latter case is relatively simple. The task to solve this problem is to try to classify the two interwined spirals. A successful solution means that the generalization results on the unknown points should have a boundary curve (spiral-like curve) between the separate domains which include the separate learning samples. A generalization criterion is generally set as follows: if a network get a successful solution in the above meaning, then this network is said to have good generalization ability, otherwise this network has no generalization ability. If the boundary of final generalization results approximates the spiral curve much more accurately, this network can have better generalization ability.

The following basic feed-forward neural network structure is adopted: 3 layers with two inputs and output: 2 : N1 : N2 : 1, hidden node function is sigmoidal, output function is linear function, where N1, N2 varies between the interval [13, 40]. The learning rate \( \eta \) is set to 0.0031, and the parameter \( \alpha \) is set to 0.004. The meaning of \( \alpha \) can be explained similarly as the regularizer term used in the functional analysis. \( \alpha \to 0 \) implies the final solution is completely determined by the given examples; while \( \alpha \to \infty \) says the examples are completely unreliable. The middle value says that a compromise is made between these two extreme cases, that is to say some noises are contained in the learning data. Also with the large value of \( \alpha \), the generalization of neural networks largely depends on the complex relation between the structure of networks and regularizer term \( \alpha \). Based on the above reason and considering the simplicity of backpropagation algorithm, we set \( \alpha \) to a small value, such as 0.004.

Some explanations for the following listed figures are made as follows. Fig.2 ~ Fig.7 are the learning curves and generalization results when the network structure is...
set to $2:18:13:1, 2:25:15:1, 2:40:20:1$, respectively. The training error is computed by using mean square error. The same simulation setting is used 3 times, and the average result is listed here. The size of weight matrix of the initial network structure used is different. The size of weight matrix used in Fig.2 and Fig.3 is small, while the one used in Fig.6 and Fig.7 is large.

From the simulation results one can easily observe that the new algorithm can solve this problem successfully under the different initial size of weight matrix of the networks, whether it is small size like in Fig.2 and Fig.3 or it is large size as in Fig.6 and Fig.7. This says that the new algorithm has the ability to prune the unnecessary connection in the network during the learning process. Therefore the new algorithm is not so much effected by the size of initial weight matrix as that of the backpropagation.

Under the same above simulation conditions, Gaussian regularizer method is also used to solve the same problem, because Gauss regularizer(16) has been used more often than other regularizers. The reason of not using regularizer(17) is that the regularizer(17) sometimes performs poorly for some parameter values of $\alpha^{(3,5,11)}$. In all cases Gaussian regularizer also solves successfully this problem. The total time (in mean hours) needed for solving this problem is listed in Table 1 for comparison. The mean square error for stopping the program is preset at 0.0001.

From Table 1 one can easily see that the new algorithm converges much more quickly than Gaussian regularizer method, especially when the initial size of weight matrix is set to be large. It is known that the backpropagation algorithm with regularizer converges much more quickly and generalizes much better than that of the backpropagation algorithm with no regularizer, so also in this sense the new algorithm is much more powerful than the backpropagation. For detailed simulations using the regularizers (17)(18)(19), please see the paper of reference11). From simulations in the paper of reference 11), one can easily observe that the generalization results by these regularizers are not better than the results by the proposed method.

Next, a comparison is made to observe the pruning ability of the new algorithm. Conditions of simulation are: network structure is set to $[2:18:13:1]$, the number of iteration steps is set to 160,000, initial network is fully connected. The connection ratio used in the figure is defined as follows:

$$Connection\ Ratio = \frac{N_{el}}{N_{e2}}$$  \hspace{1cm} (20)

where $N_{e1}$ is the number of connections of the networks at a certain specific iteration step, $N_{e2}$ is the number of connection of the initial network, in this example $N_{e1} = 2 \times 18 + 18 \times 13 + 13 = 283$. Gaussian regularizer is used for comparison under the same condition. A result is shown in Fig.8. From Fig.8, one can easily note that the new algorithm reduces the size of the network more quickly in the first 60,000 iteration steps, after this step the structure of the network almost remain unchanged with only a small variation in the connection ratio of the network observed. On the other hands, with Gaussian regularizer, the same connection ratio 0.265 is reached only after 120,000 iteration steps.

### 3.2 Function Approximation

The function to be used is:

$$F(X,Y) = \exp(-\frac{(X^2 + Y^2)}{0.5}) \ast \left(\cos(2 \times \pi \ast (X + Y^2)) + \exp(X)\right)$$  \hspace{1cm} (21)

where intervals of $X$ and $Y$ are all set to $[-0.5:0.5]$. This function is devised after making revision to Gabor function\(^{(10)}\). Gabor functions play an important role in physics.

Learning points are selected from the domain of $X$ and $Y$ evenly distributed with an interval of 0.08 respectively, while the generalization points are selected from the domain of $X$ and $Y$ with an interval of 0.02. The reason for selecting so few number of learning data compared to so many generalizing data is that the generalization ability of the new algorithm is to be demonstrated more clearly. The learning data and their true function are shown in Fig.9 and Fig.10.

The learning rate $\eta$ is set to 0.0031, and the parameter $\alpha$ is set to 0.004 as the former example. The learning curve and generalization results are listed in Fig.11 ~ Fig.16. Some explanations for the listed figures are made as follows. Fig.11 and Fig.12 are the learning curve and generalization results respectively, when the network structure is set to $2:20:15:1$; Fig.13 and Fig.14 show the results of the network with $2:30:16:1$; and Fig.15 and Fig.16 are the learning curve and generalization results of the networks with to $2:35:20:1$.

From the simulation results one can see clearly that the main part of the original true function is reconstructed by the trained network under different initial network structures, although there are some deformed part in the reconstructed figures. Considering the number of learning data, i.e., 224 and number of generalization data, i.e., 2276, complete elimination of the local deformation is cer-
Table 1  Comparison of computation time between Gaussian and proposed algorithm (Two-spiral problem)

<table>
<thead>
<tr>
<th>size of initial NN</th>
<th>Gaussian regularizer</th>
<th>New algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 : 18 : 13 : 1</td>
<td>5.47</td>
<td>5.35</td>
</tr>
<tr>
<td>2 : 25 : 15 : 1</td>
<td>5.89</td>
<td>5.24</td>
</tr>
<tr>
<td>2 : 40 : 20 : 1</td>
<td>7.74</td>
<td>6.97</td>
</tr>
</tbody>
</table>
Fig. 9 Learning data for function approximation

Fig. 10 Testing function for function approximation

Fig. 11 Learning curve of [2:20:15:1] network

Fig. 12 Generalization results of [2:20:15:1] network

Fig. 13 Learning curve of [2:30:16:1] network

Fig. 14 Generalization results of [2:30:16:1] network

Fig. 15 Learning curve of [2:35:20:1] network

Fig. 16 Generalization results of [2:35:20:1] network
tamently impossible in the reconstructed figures. Also note that the simulation results do not vary with the changes in the network structure, while this property does not hold for the basic backpropagation.

Under the same simulation conditions, Gaussian regularizer method is also used to solve the same problem. The total times in mean hours needed for solving this problem, that is to say, the algorithm is stopped once the preset error limit 0.0001 is obtained, are listed for comparison in Table 2, while the generalization errors (sums of square errors) are listed in Table 3.

From Table 2 one can see the new algorithm converges more quickly than Gaussian regularizer method, while Table 3 shows the new algorithm behaves better in terms of the errors of generalization.

Next, a comparison is made to observe the pruning ability of the new algorithm. Conditions of simulation are: network structure is set to \([2:35:20:1]\), the number of iteration steps is set to 400,000, initial network is fully connected. In this example \(N_0 = 2 \times 35 + 35 \times 20 + 20 = 790\). Gaussian regularizer is used under the same condition for comparison. A result is listed as Fig. 17.

From Fig. 17, one can easily note that the new algorithm reduces the size of network more quickly in the first 100,000 iteration steps, after the 200,000 iteration steps the structure of the network almost remain unchanged with only a small connection ratio reduction of the network observed. The same connection ratio 0.325 is obtained only after 270,000 iteration steps in Gaussian regularizer.

### 3.3 Iris Data Problem

The classification of Iris data problem is well known as a higher dimensional data classification problem with continuous inputs and binary outputs. Irises are classified into three categories: setosa, versicolor and virginica. Each category has 50 samples. Each sample possesses 4 attributes: sepal length, sepal width, petal width and petal length. The characteristics of this data samples are that one of the 3 classes is well separated from the other two, which are not easily separable due to the overlapping of their convex hulls.

Two network structures used here are a three-layer network: an input layer with 4 nodes, hidden layer with 10 or 15 nodes and an output layer with 3 nodes. A subset of data is randomly chosen for training, and data are normalized before training. The generalization ability of the resulting network is evaluated by the percentage of classification errors on the average of 5 trials on the testing data. The assignment of input vectors to classes was based on a winner-takes-all strategy. The proposed algorithm and Gaussian Regularizer are used to solved this problem. The results are listed in Table 4 and Table 5.

From Table 4 and Table 5 one can easily observe that under two different sizes of initial network structures, using various number of training data, the new algorithm generalizes better than that of Gaussian regularizer, because the error of misclassification on the testing data

---

**Table 2** Comparison of computation time between Gaussian and proposed algorithm (Function approximation)

<table>
<thead>
<tr>
<th>size of initial NN</th>
<th>Gaussian regularizer</th>
<th>New algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:20:15:1</td>
<td>3.85</td>
<td>3.51</td>
</tr>
<tr>
<td>2:25:10:1</td>
<td>4.35</td>
<td>4.06</td>
</tr>
<tr>
<td>2:30:16:1</td>
<td>4.27</td>
<td>4.24</td>
</tr>
<tr>
<td>2:35:20:1</td>
<td>4.93</td>
<td>4.46</td>
</tr>
</tbody>
</table>

**Table 3** Comparison of generalization errors between gaussian and proposed algorithm (Function approximation)

<table>
<thead>
<tr>
<th>size of initial NN</th>
<th>Gaussian regularizer</th>
<th>New algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:20:15:1</td>
<td>0.08</td>
<td>0.07</td>
</tr>
<tr>
<td>2:25:10:1</td>
<td>0.23</td>
<td>0.19</td>
</tr>
<tr>
<td>2:30:16:1</td>
<td>0.37</td>
<td>0.23</td>
</tr>
<tr>
<td>2:35:20:1</td>
<td>0.19</td>
<td>0.12</td>
</tr>
</tbody>
</table>

**Table 4** Generalization ability in the classification of Iris data with 10 hidden nodes

<table>
<thead>
<tr>
<th>NO. of Data</th>
<th>Percentage of Misclassification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>Testing</td>
</tr>
<tr>
<td>21</td>
<td>129</td>
</tr>
<tr>
<td>30</td>
<td>120</td>
</tr>
<tr>
<td>60</td>
<td>90</td>
</tr>
</tbody>
</table>

**Table 5** Generalization ability in the classification of Iris data with 15 hidden nodes

<table>
<thead>
<tr>
<th>NO. of Data</th>
<th>Percentage of Misclassification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>Testing</td>
</tr>
<tr>
<td>21</td>
<td>129</td>
</tr>
<tr>
<td>30</td>
<td>120</td>
</tr>
<tr>
<td>60</td>
<td>90</td>
</tr>
</tbody>
</table>
is smaller than that of Gaussian regularizer. Concerning the pruning ability of the new algorithm, a phenomenon similar to Fig.8 and Fig.17 can also be observed.

4. Conclusions

In this paper a new variant of backpropagation is proposed with the aim to improve its generalization ability. The difference between the proposed method and the other methods is that the new proposed method in this paper accomplishes the aim through controlling the outputs of hidden layers, while the conventional regularizer methods through directly controlling the weight matrix. This results in the great differences between these two kinds of methods. The new regularizer has the following property: (1) Small added noises in the inputs to the networks will not give evident effects to the outputs of the networks; (2) Small added noises in the weight matrix except the one between the hidden layer and output layer will not give large effects to the outputs of the networks. These properties made the network generalize better and converge much more quickly.

References

1) M. Ishikawa: Structure learning with forgetting, Neural Networks, 9, 509/521 (1996)
12) P.M. Murphy and D.W. Aha: UCI repository of machine learning databases, Department of Information and Computer science, University of California, Irvine (1992)

Weishui Wan

He received the M.Sci. degree in 1990 from Jilin University, China. From August 1990 to December 1998, he was a Research Associate and then a lecture in Northeast University, China. Now he is a second-year Dr. candidate at Graduate School of Information Science and Electrical Engineering, Kyushu University. His current research interest lies in designing the efficient neural network algorithms.

Kotaro Hirasawa (Member)

He received the M.S. degree in Electrical Engineering from Kyushu University in 1966. From April 1966, he served in Hitachi Lab. of Hitachi Ltd., and in 1989 he was a vice president of Hitachi Lab. From August 1991 to November 1992, he served in Omika Factory of Hitachi Ltd. Since December 1992, he has been a professor in the faculty of Engineering, Kyushu University. Now he belongs to the Graduate School of Information Science and Electrical Engineering, Kyushu University. Dr. Hirasawa is a member of the Institute of Electrical Engineers of Japan.

Jinglu Hu (Member)

He received the M.Sc. degree in 1986 from Zhongshan University, China and the Ph.D degree in 1997 from Kyushu Institute of Technology, Japan. From 1986 to 1993, he was a Research Associate and then a lecturer in Zhongshan University. Since 1997, he has been a Research Associate at Kyushu University. His current research interests are learning network theory, system identification and their applications. Dr. Hu is a member of the Institute of Electrical Engineers of Japan.

Junichi Murata (Member)

He received the D.Eng. degree in Electrical Engineering from Kyushu University in 1986. From 1986 to March 1988, he was a Research Associate at Kyushu University. From 1988, he has been an Associate Professor at Kyushu University. Now he belongs to the Graduate School of Information Science and Electrical Engineering, Kyushu University. He is a member of the Institute of Electrical Engineers of Japan, a member of ISCIE and a member of IEEE.