Minimization Learning of Neural Networks by Adding Hidden Units

Niijima, Koichi
Department of Informatics, Graduate School of Information Science and Electrical Engineering, Kyushu University

Yamada, Masaaki
Department of Control Engineering and Science, Kyushu Institute of Technology : Graduate Student | NEC Corporation

Mohamed, Marghny H.
Department of Informatics, Graduate School of Information Science and Electrical Engineering, Kyushu University : Graduate Student

Akanuma, Taiga
Department of Informatics, Graduate School of Information Science and Electrical Engineering, Kyushu University : Graduate Student

他

https://doi.org/10.15017/1522387

出版情報：九州大学大学院システム情報科学紀要. 2 (2), pp.173-178, 1997-09-26. 九州大学大学院システム情報科学研究科
バージョン：
権利関係：

九州大学大学院
システム情報科学研究科報告
第 2 巻　第 2 号　平成 9 年 9 月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol. 2, No. 2, September 1997

Minimization Learning of Neural Networks by Adding Hidden Units

Koichi NIIJIMA* , Masaaki YAMADA**, Marghny H. MOHAMED***,
Taiga AKANUMA***, Teruya MINAMOTO* and Akito OHKUBO†

(Received June 23, 1997)

Abstract: This paper proposes a minimization learning method for neural networks with one hidden layer. We treat two types of networks without and with thresholds in their output layers. Both of them are learnt by minimizing error functions whenever one unit is added to the hidden layer. Our learning method is applied to design a neural network with thresholds for image recognition.

Keywords: Minimization learning, Hidden unit, Threshold, Error function, Penalty method, Steepest descent method, Lagrange method

1. Introduction

It is well known that the learning ability of neural networks having one hidden layer increases with the number of hidden units. In this paper, we propose a method for learning the network locally by adding hidden units. There are a few articles[1)2)] on minimization learning algorithms for the same type of neural networks. In the paper [1)], a condition for learning a training set exactly was found in relation with a consistency condition for the solutions of linear equations. And from this condition, the paper[1)] derived a minimization problem for deciding connection weights between the input and the hidden layers. An algorithm for solving this problem was also given. However, the cost function to be minimized includes a nonlinear transfer function and hence, we can not avoid the trapping of the solutions in local minima.

In this paper, we present a method for determining connection weights of the network so as to minimize the output errors for training data. This minimization is carried out whenever one unit is added to the hidden layer. First, the weights between the hidden and output layers are determined by minimizing a quadratic functional. Substituting the weights into this functional, we can obtain a relation between the error functions before and after adding one unit to the hidden layer. It is desirable to maximize the difference of both error functions. This maximization problem coincides with that derived in the paper[1)]. In this paper, we separate this problem, under some restriction, into two minimization problems. One is concerned with a minimization of a rational cost function on independent variables, which is derived from the penalty method. The other is related to a process trying to prune the new connections between the input and the hidden layers.

Although the papers [1)2)] deal with only the network without thresholds in the output layer, this paper treats the networks without and with thresholds in the output layer. Our method is applied to construct both types of neural networks.

Finally, the simulations are carried out by using landscapes as a training set with a network having thresholds in the output layer.

2. Minimization Learning Algorithm

2.1 Neural Network without Thresholds in the Output Layer

We consider a neural network which consists of an input layer with $n+1$ nodes, a hidden layer with h units, and an output layer with l units:

$$y_i = g\left(\sum_{j=1}^{h} w_{ij} f\left(\sum_{k=1}^{n+1} v_{jk} x_k\right)\right), i = 1, 2, \ldots, l, \quad (1)$$

where x_k indicates the k-th input value, y_i the i-th output value, v_{jk} a weight connecting the k-th input node with the j-th hidden unit, and w_{ij} a weight between the j-th hidden unit and the i-th output unit. The functions $f(t)$ and $g(t)$ are given by

$$f(t) = \frac{1 - e^{-t}}{1 + e^{-t}} \quad , \quad g(t) = \frac{1}{1 + e^{-t}},$$

* Department of Informatics
** Department of Control Engineering and Science, Kyushu Institute of Technology, Graduate Student. At present, NEC Corporation
*** Department of Informatics, Graduate Student
† Fukuoka Institute of Health and Environmental Sciences

respectively. We write (1) as

$$y = g\left(Wf\left(Vx\right)\right),$$

where we set $x = (x_1, x_2, \ldots, x_n, x_{n+1})$ with $x_{n+1} = -1$, $y = (y_1, y_2, \ldots, y_l)$, $V = (v_{jk})$ and $W = (w_{ij})$. Moreover, $f(Vx)$ means $(f(V_1x), f(V_2x), \ldots, f(V_hx))$ and $g(Wf(Vx))$ indicates $(g(W_1f(Vx)), g(W_2f(Vx)), \ldots, g(W_lf(Vx)))$, where $V_j = (v_{j1}, v_{j2}, \ldots, v_{j,n+1})$ and $W_i = (w_{i1}, w_{i2}, \ldots, w_{ih})$. This network is shown in **Fig.1**.

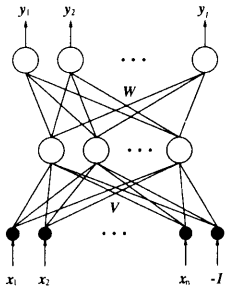

Fig.1 Neural network without thresholds in its output layer

Let $(x^\nu, y^\nu), \nu = 1, 2, ..., m$, be training data for the network. We define an output error between the outputs of the network for the inputs x^ν and the relevant outputs y^ν by

$$J(V, W) = \sum_{\nu=1}^{m} \|g^{-1}(y^\nu) - Wf(Vx^\nu)\|^2 \tag{2}$$

where $g^{-1}(y^\nu) = (g^{-1}(y_1^\nu), g^{-1}(y_2^\nu), \ldots, g^{-1}(y_l^\nu))$ with the inverse function $g^{-1}(s)$ of $s = g(t)$, and $\|\cdot\|$ stands for the Euclidean norm. To determine V and W, we need to minimize the error function (2). In this paper, we present a technique for determining the weights V and W successively. We suppose that the weights V and W have already been learnt. We add one unit to the hidden layer of this network, which is called an $(h+1)$-th hidden unit. Let $\mathbf{v}$ denote a connection weight vector between the $(h+1)$-th hidden unit and the input layer, and let $\mathbf{w}$ be a weight vector connecting the $(h+1)$-th hidden unit with the output layer. The neural network after adding the $(h+1)$-th hidden unit is shown in **Fig.2**.

We denote the weight matrices $(V, \mathbf{v})$ and $(W, \mathbf{w})$ by $\widetilde{V}$ and $\widetilde{W}$, respectively. Then a new error function can be written as

$$J(\widetilde{V}, \widetilde{W}) = \sum_{\nu=1}^{m} \|g^{-1}(y^\nu) - \widetilde{W}f(\widetilde{V}x^\nu)\|^2.$$

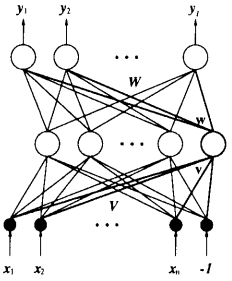

Fig.2 Neural network after adding one unit in the hidden layer in **Fig.1**

We describe how to determine the added weight vectors $\mathbf{w}$ and $\mathbf{v}$. Since it holds that $\widetilde{W}f(\widetilde{V}x^\nu) = Wf(Vx^\nu) + \mathbf{w}f(\mathbf{v}x^\nu)$, we have

$$J(\widetilde{V}, \widetilde{W}) = J(V, W) - 2\langle d, \mathbf{w}\rangle + a\|\mathbf{w}\|^2 \tag{3}$$

in which d and a represent

$$d = \sum_{\nu=1}^{m} f(\mathbf{v}x^\nu)c^\nu \tag{4}$$

and

$$a = \sum_{\nu=1}^{m} f^2(\mathbf{v}x^\nu), \tag{5}$$

where $c^\nu = g^{-1}(y^\nu) - Wf(Vx^\nu)$ and the symbol $\langle\cdot,\cdot\rangle$ indicates an inner product in R^l. When the vector $\mathbf{v}$ is fixed, the vector $\mathbf{w}$ which minimizes the error function (3) is given by

$$\mathbf{w} = \frac{d}{a}. \tag{6}$$

Substituting this vector into (3) yields

$$J(\widetilde{V}, \widetilde{W}) = J(V, W) - I(\mathbf{v}),$$

where

$$I(\mathbf{v}) = \frac{\langle d, d\rangle}{a}. \tag{7}$$

Next, we must determine the weight vector $\mathbf{v}$. In the paper [1], the vector $\mathbf{v}$ was determined so as to minimize $-I(\mathbf{v})$ by applying directly the steepest descent method. However, this method shows a poor convergence behavior and has a disadvantage that solutions are trapped in local minima, because $I(\mathbf{v})$ contains the nonlinear function f. We employ another approach to determine the vector $\mathbf{v}$.

Let us assume that the dimension n of input pattern vectors is larger than the number m of training patterns. This case occurs frequently. For example, the size n of image patterns is sufficiently large in comparison with m. Under this condition, we maximize the function $I(\mathbf{v})$, namely, minimize $-I(\mathbf{v})$. As is easily seen from (4), (5) and (7), $-I(\mathbf{v})$ can be written as

$$-I(\mathbf{v}) = -\left\langle \sum_{\nu=1}^{m} s_\nu c^\nu, \sum_{\nu=1}^{m} s_\nu c^\nu \right\rangle / \sum_{\nu=1}^{m} s_\nu^2, \tag{8}$$

where we have put

$$s_\nu = f(\mathbf{v}x^\nu). \tag{9}$$

So we regard (8) as a function of the variables s_ν and minimize it with respect to s_ν first. This minimization problem can be solved by the penalty method. Indeed, we can formulate the function with a penalty term as

$$-\left\langle \sum_{\nu=1}^{m} s_\nu c^\nu, \sum_{\nu=1}^{m} s_\nu c^\nu \right\rangle + P\,(\sum_{\nu=1}^{m} s_\nu^2 - 1)^2,$$

where P is a penalty constant. The unknown parameters s_ν can be sought by minimizing this functional using, for instance, the steepest descent method. Substituting the solutions into (9) and using the inverse function $f^{-1}(s)$ give

$$\mathbf{v}x^\nu = \ln \frac{1+s_\nu}{1-s_\nu}, \qquad \nu = 1, 2, \ldots, m \tag{10}$$

which are simultaneous equations with respect to $\mathbf{v}$. By assumption $n > m$, the number of unknown variables $\mathbf{v} = (v_{h+1,1}, v_{h+1,2}, \ldots, v_{h+1,n+1})$ is more than the number of the equations. So we impose on (10) a restrictive condition

$$\|\mathbf{v}\|^2 \quad \rightarrow \quad \min. \tag{11}$$

which has a role to improve the behavior of these networks. The minimization problem (10) and (11) can be solved by Lagrange method as follows:

$$\mathbf{v} = -\frac{1}{2} \sum_{\mu=1}^{m} \lambda_\mu x^\mu,$$

where λ_μ are solutions of

$$-\frac{1}{2} \sum_{\mu=1}^{m} (x^\nu, x^\mu)\, \lambda_\mu = \ln \frac{1+s_\nu}{1-s_\nu}, \quad \nu = 1, 2, \ldots, m.$$

Substituting the solution $\mathbf{v}$ into a and d defined by (5) and (4), respectively, we can obtain $\mathbf{w}$ of (6). If we go through this procedure from one unit in the hidden layer, we can compose a minimum scale of network in a sense.

2.2 Neural Network with Thresholds in the Output Layer

Next we will consider the network with thresholds $\theta_i, i = 1, 2..., l$, in its output layer as shown in **Fig.3**.

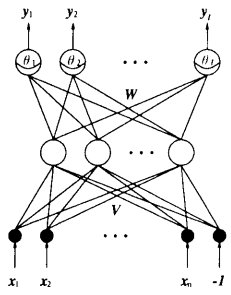

Fig.3 Neural network with thresholds in the output layer

In this case, we can write

$$y_i = g\left(\sum_{j=1}^{h} w_{ij} f\left(\sum_{k=1}^{n+1} v_{jk} x_k \right) - \theta_i \right), \qquad i = 1, 2, \ldots, l$$

whose simple form is

$$y = g\left(W f\left(Vx\right) - \theta\right),$$

where $\theta = (\theta_1, \theta_2, \ldots, \theta_l)$ and $g(Wf(Vx) - \theta) = (g(W_1 f(Vx) - \theta_1), g(W_2 f(Vx) - \theta_2), \ldots, g(W_l f(Vx) - \theta_l))$. The error function related to the present network takes the following form

$$J(V, W, \theta) = \sum_{\nu=1}^{m} \|g^{-1}(y^\nu) - Wf(Vx^\nu) + \theta\|^2.$$

We add one unit to the hidden layer and represent new weight vectors again by $\mathbf{v}$ and $\mathbf{w}$. By adding one hidden unit, the threshold vector θ must be changed. We denote a new threshold by $\widetilde{\theta}$ and write as $\widetilde{\theta} = \theta + \triangle\theta$. The network after adding one hidden unit is shown in **Fig.4**: The error function related

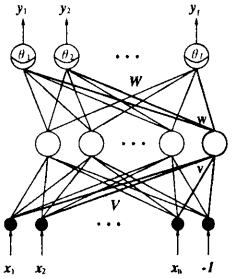

Fig.4 Neural network with thresholds after adding one unit in the hidden layer in **Fig.3**

to this network can be expressed as

$$J(\widetilde{V},\widetilde{W},\widetilde{\theta})=\sum_{\nu=1}^{m}\|g^{-1}(y^{\nu})-\widetilde{W}f(\widetilde{V}x^{\nu})+\widetilde{\theta}\|^2,$$

where we have used again the symbols $\widetilde{V}=(V,\mathbf{v})$ and $\widetilde{W}=(W,\mathbf{w})$. The same procedure as in the previous subsection will be applied in order to determine $\mathbf{v}$, $\mathbf{w}$ and $\triangle\theta$ so that $J(\widetilde{V},\widetilde{W},\widetilde{\theta})$ is minimum. Since $\widetilde{W}f(\widetilde{V}x^{\nu})=Wf(Vx^{\nu})+\mathbf{w}f(\mathbf{v}x^{\nu})$ and $\widetilde{\theta}=\theta+\triangle\theta$, we can decompose the error function $J(\widetilde{V},\widetilde{W},\widetilde{\theta})$ as

$$\begin{aligned}J(\widetilde{V},\widetilde{W},\widetilde{\theta})=J(V,W,\theta)&-2\sum_{\nu=1}^{m}f(\mathbf{v}x^{\nu})\langle c^{\nu}+\triangle\theta,\mathbf{w}\rangle\\&+\sum_{\nu=1}^{m}f^2(\mathbf{v}x^{\nu})\|\mathbf{w}\|^2+2\sum_{\nu=1}^{m}\langle\triangle\theta,c^{\nu}\rangle\\&+\sum_{\nu=1}^{m}\|\triangle\theta\|^2,\end{aligned}\tag{12}$$

where we put $c^{\nu}=g^{-1}(y^{\nu})-Wf(Vx^{\nu})+\theta$. We fix $\mathbf{v}$ and determine $\mathbf{w}$ and $\triangle\theta$ so that $J(\widetilde{V},\widetilde{W},\widetilde{\theta})$ is minimum. The weight vector $\mathbf{w}$ and the correction $\triangle\theta$ are obtained by solving the equations

$$\frac{\partial J(\widetilde{V},\widetilde{W},\widetilde{\theta})}{\partial w_{i,h+1}}=0,\qquad\frac{\partial J(\widetilde{V},\widetilde{W},\widetilde{\theta})}{\partial\triangle\theta_i}=0$$

as follows:

$$\mathbf{w}=\frac{md_1-a_2d_2}{b},\tag{13}$$

$$\triangle\theta=\frac{a_2d_1-a_1d_2}{b},\tag{14}$$

where $a_1=\sum_{\nu=1}^{m}f^2(\mathbf{v}x^{\nu})$, $a_2=\sum_{\nu=1}^{m}f(\mathbf{v}x^{\nu})$, $d_1=\sum_{\nu=1}^{m}f(\mathbf{v}x^{\nu})c^{\nu}$, $d_2=\sum_{\nu=1}^{m}c^{\nu}$, and $b=ma_1-a_2^2$. Substituting (13) and (14) into (12), we get

$$J(\widetilde{V},\widetilde{W},\widetilde{\theta})=J(V,W,\theta)-K(\mathbf{v})$$

in which $K(\mathbf{v})$ denotes

$$K(\mathbf{v})=K_0(\mathbf{v})+\frac{1}{m}\langle d_2,d_2\rangle,$$

where $K_0(\mathbf{v})$ is given by

$$K_0(\mathbf{v})=\frac{m}{b}\left\langle d_1-\frac{a_2}{m}d_2,d_1-\frac{a_2}{m}d_2\right\rangle.$$

We want to determine the weight vector $\mathbf{v}$ so that $K(\mathbf{v})$ is maximum. The term $\frac{1}{m}\langle d_2,d_2\rangle$ is independent of $\mathbf{v}$. Therefore, it suffices to determine $\mathbf{v}$ so that $K_0(\mathbf{v})$ is maximum, namely, the term $-K_0(\mathbf{v})$ is minimum. The term $K_0(\mathbf{v})$ can be transformed into

$$\begin{aligned}K_0(\mathbf{v})&=\frac{1}{m}\frac{\langle md_1-a_2d_2,md_1-a_2d_2\rangle}{ma_1-a_2^2}\\&=\frac{1}{m}\frac{\left\langle\sum_{\nu=1}^{m}t_\nu C^{\nu},\sum_{\nu=1}^{m}t_\nu C^{\nu}\right\rangle}{m\sum_{\nu=1}^{m}t_\nu^2-(\sum_{\nu=1}^{m}t_\nu)^2},\end{aligned}$$

where $t_\nu=f(\mathbf{v}x^{\nu})$ and $C^{\nu}=mc^{\nu}-\sum_{\mu=1}^{m}c^{\mu}$. We first determine $t_\nu,\nu=1,2,...,m$, such that $-K_0(\mathbf{v})$ is minimum. This is carried out using the penalty method:

$$-\frac{\left\langle\sum_{\nu=1}^{m}t_\nu C^{\nu},\sum_{\nu=1}^{m}t_\nu C^{\nu}\right\rangle}{m-\left(\sum_{\nu=1}^{m}t_\nu\right)^2}+P\,(\sum_{\nu=1}^{m}t_\nu^2-1)^2\to\min.$$

This minimization problem can be solved by the steepest descent method. Using t_ν thus obtained, the weight vector $\mathbf{v}$ is sought in the same way as in the previous subsection and, as a result, $\mathbf{w}$ and $\triangle\theta$ can be obtained by (13) and (14), respectively.

3. Simulations

As regard for the simulations of image recognition, we construct a neural network with thresholds. To memorize the training patterns x^{ν}, we consider 40 landscapes, i.e. $m=40$, with gray scale, each of which has the size 128×128. Some of them are shown in **Fig.5**. The number l of output units was taken as $l=40$, and the output y^{ν} for the pattern x^{ν} was chosen as $y_i^{\nu}=\delta_{\nu,i}$, where $\delta_{\nu,i}$ indicates the Kronecker's delta symbol.

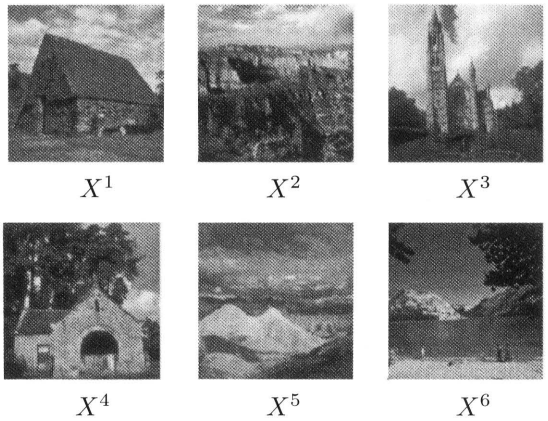

Fig.5 Part of the memorized patterns

Using these pictures, we determined the weights and thresholds of the network based on the proposed learning algorithm. The error function was sufficiently small when adding 30 units to the hidden layer.

Noisy patterns for recognition were created first by adding 30% random noise to all the memorized patterns. Some of them are shown in **Fig. 6**.

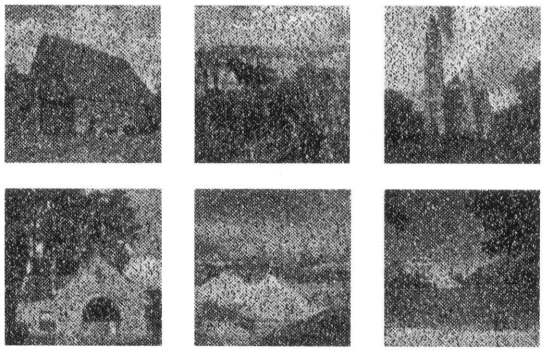

Fig.6 Part of the noisy patterns with 30 % random noise

Table 1 shows that 10 units in the hidden layer were needed to recognize all the noisy patterns. We can notice from **Table 1** that some of those could be recognized by a smaller size of network.

In each box of **Tables 1-3**, we listed the number of hidden units necessary for recognition, the maximum value of outputs and recognized patterns.

Table 1. Recognition results for noisy patterns with 30 % added noise.

6 0.142 X^1	**10** 0.188 X^2	4 0.174 X^3	7 0.144 X^4	6 0.159 X^5
3 0.149 X^6	6 0.157 X^7	3 0.161 X^8	8 0.171 X^9	8 0.159 X^{10}
5 0.168 X^{11}	4 0.165 X^{12}	5 0.147 X^{13}	5 0.153 X^{14}	1 0.135 X^{15}
4 0.149 X^{16}	5 0.151 X^{17}	4 0.179 X^{18}	8 0.145 X^{19}	3 0.135 X^{20}
4 0.150 X^{21}	4 0.146 X^{22}	7 0.176 X^{23}	6 0.164 X^{24}	4 0.145 X^{25}
3 0.143 X^{26}	5 0.150 X^{27}	4 0.141 X^{28}	6 0.148 X^{29}	5 0.158 X^{30}
2 0.174 X^{31}	7 0.165 X^{32}	4 0.155 X^{33}	**10** 0.222 X^{34}	3 0.142 X^{35}
5 0.149 X^{36}	6 0.144 X^{37}	9 0.184 X^{38}	2 0.155 X^{39}	7 0.204 X^{40}

Next, we created noisy patterns by adding 50 % random noise to all the memorized patterns, some of which are shown in **Fig. 7**.

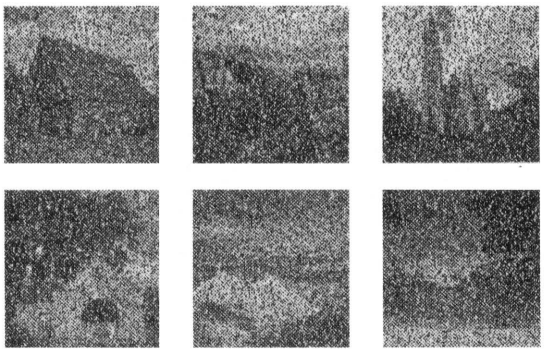

Fig.7 Part of the noisy patterns with 50 % random noise

Table 2 illustrates that only 10 hidden units of the network were needed to recognize all the noisy patterns.

Table 2. Recognition results for noisy patterns with 50 % added noise.

6	**10**	2	7	6
0.133	0.164	0.137	0.131	0.146
X^1	X^2	X^3	X^4	X^5
3	6	3	8	9
0.142	0.153	0.144	0.157	0.162
X^6	X^7	X^8	X^9	X^{10}
1	4	5	5	1
0.130	0.150	0.134	0.143	0.130
X^{11}	X^{12}	X^{13}	X^{14}	X^{15}
4	5	4	**10**	3
0.141	0.139	0.172	0.162	0.124
X^{16}	X^{17}	X^{18}	X^{19}	X^{20}
3	4	7	6	4
0.134	0.141	0.158	0.149	0.137
X^{21}	X^{22}	X^{23}	X^{24}	X^{25}
3	5	4	**10**	5
0.137	0.141	0.136	0.165	0.150
X^{26}	X^{27}	X^{28}	X^{29}	X^{30}
2	8	4	**10**	3
0.164	0.156	0.145	0.195	0.136
X^{31}	X^{32}	X^{33}	X^{34}	X^{35}
5	**10**	9	2	7
0.142	0.153	0.167	0.148	0.171
X^{36}	X^{37}	X^{38}	X^{39}	X^{40}

Finally, we present the result of simulation for five noisy patterns as shown in **Fig.8**.

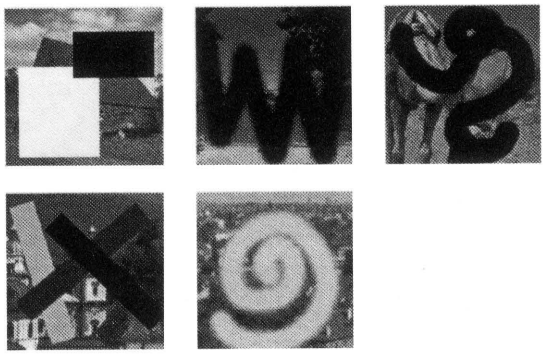

Fig.8 Patterns which are lacking part of the memorized patterns

Table 3 shows that the first three patterns could be recognized by the network with 18 hidden units, but the last pattern failed to be recognized even if 30 units are added in the hidden layer.

Table 3. Recognition results for the patterns shown in **Fig.8**.

18	4	3	7	29
0.264	0.178	0.137	0.237	0.354
X^1	X^6	X^{18}	X^{25}	X^5

4. Conclusion

In this paper, we proposed a method for learning three-layered neural networks by adding hidden units successively. For two types of neural networks without and with thresholds in the output layers, their connection weights were determined locally whenever hidden units are added. Our learning method can update the model structure when our application requires it. To avoid the trapping of solutions in local minima, we devised two steps of algorithms in determining the weights between the first and the second layers. Such a device realizes fast learning of neural networks. We designed a neural network with thresholds by using landscape training data based upon our method. By the use of this network, we carried out the simulations for recognition of some noisy patterns. Patterns produced by adding random noise to the memorized pictures were recognized very well. The network, however, couldn't recognize the picture which is lacking a large part of the original one.

There are some remaining problems to be solved in the future. One problem is to analyze a detailed mechanism of our minimization learning procedure. Simulations were carried out using limited number of pictures, but must be done for more pictures in order to examine the relation between the recognition ability and the size of neural networks. It is also important to examine the recognition for shifted patterns.

References

1) F.Bärmann, and F.Biegler-König, On a class of efficient learning algorithms for neural networks, Neural Networks, 5, 139-144, 1992.

2) O.Fujita, Optimization of the hidden unit function in feedforward neural networks, Neural Networks, 5, 755-764, 1992.