A low-bit learning algorithm for digital multilayer neural networks applied to pattern recognition

A LOW-BIT LEARNING ALGORITHM FOR DIGITAL MULTILAYER NEURAL NETWORKS APPLIED TO PATTERN RECOGNITION

Kenji NAKAYAMA **Hiroshi KATAYAMA**
Dept. of Electrical and Computer Eng., Faculty of Tech., Kanazawa Univ.
2-40-20, Kodatsuno, Kanazawa, 920 JAPAN

ABSTRACT For integrating a large scale digital neural networks, reductions in the number of binary bits are very important. A low-bit learning algorithm has been proposed [2], which gradually reduces the number of bits during a training process. In some applications, however, it is desired to use the same neural network in both training and information processing. This paper presents a new low-bit learning algorithm, by which the training can be carried out with a small number of bits. A set of orthogonal vectors is employed for output layer targets. Pattern recognition is , therefore, to transform arbitrary pattern into an orthogonal vector. In this case, hidden layer outputs are desired to be linearly independent. Furthermore, in order to gain noise margin, Hamming distances among them should be as large as possible. The optimum targets, satisfying these conditions, are assigned to hidden layers. Adjusting weights is independently carried out in both hidden and output layers. Distribution of weights is optimized by using variable hysteresis threshold. Computer simulation demonstrated efficiency of the proposed method.

I INTRODUCTION

There are several approaches to artificial neural network realization. A digital hardware realization is one hopeful approach. Variable connection weights, complicated learning algorithms and a large memory capacity can be easily realized, with desired accuracy.

Digital hardware, however, requires a plural number of bits for expressing weights and unit outputs. In addition, neural networks usually consist of a large number of units, and massive connections among them. Therefore, it is difficult to integrate a large scale digital neural network on a single chip.

One way to solve this problem is to decrease the number of bits, maintaining the desired performance. For this purpose, a low-bit learning algorithm, based on the back-propagation (BP) algorithm [1], has been proposed [2],[3]. In this method, the number of bits is gradually decreased during a training process. This method, however, cannot simplify the digital hardware in the training process. In actual applications, however, it is sometime required to use the same neural network in both training and information processing.

This paper proposes a new low-bit learning algorithm for multilayer neural networks applied to pattern recognition. The training can be carried out with a small number of bits. In order to make the neural network insensitive to noisy patterns, conditions to be satisfied by hidden layer outputs are first discussed. Based on this, optimum targets are assigned to the hidden layer. Adjusting weights is independently carried out in each layer. Furthermore, distribution of weights is optimized so as to make noise margin at the unit inputs by employing variable hysteresis threshold.

II MULTILAYER NEURAL NETWORKS

Figure 1 shows a block diagram of two-layer neural network. Connection weights, and inputs and outputs of units are denoted by

Connection weights: Input layer → Hidden layer w_{IJ}
Hidden layer → Output layer w_{JK}
Input layer: Output= y_I, $i=1,2,\dots,N_I$
Hidden layer: Input = x_J, Output = y_J, $j=1,2,\dots,N_H$,
Output layer: Input = x_K, Output = y_K, $k=1,2,\dots,N_O$,

They are related by

$$x_J = \sum_{I=1}^{NI} w_{IJ} y_I \quad (1a)$$

$$y_J = f(x_J) \quad (1b)$$

$$x_K = \sum_{J=1}^{NH} w_{JK} y_J \quad (2a)$$

$$y_K = f(x_K) \quad (2b)$$

f() is an activaion function.

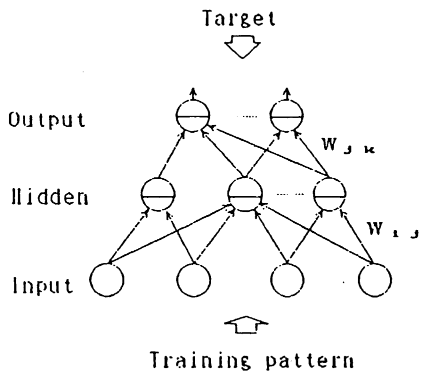

Fig.1 Multilayer neural network.

III OPTIMUM HIDDEN LAYER OUTPUTS

In the case of pattern recognition, a set of orthogonal vectors, consisting of 1 or 0, is useful for the output layer targets. One unit in the output layer is uniquely corresponds to one of the input patterns. It can guarantee high separability. In this case, pattern recognition can be considered as orthogonal mapping. Therefore, the hidden layer outputs should be linearly independent to each other. This condition could be somewhat relaxed by using a logistic function.

Next, noise sensitivity is discussed in the following. Let the mth input pattern be $Y_{I,m}$, $m=1\sim N_P$, and the corresponding hidden output be $Y_{H,m}$. Their elements are assumed to be 1 or 0. If $Y_{H,m}$ is changed into $Y'_{H,m}$ due to noises added to $Y_{I,m}$, and the following inequality is held,

$$|Y_{H,m}-Y'_{H,m}| > |Y_{H,n}-Y'_{H,m}|,\ m\neq n \quad (3)$$

then the network tends to recognize the input pattern as $Y_{I,n}$. $|Y_1-Y_2|$ means the Hamming distance between vectors Y_1 and Y_2.

The above situation easily occurs if the Hamming distance between $Y_{H,m}$ and $Y_{H,n}$, $m\neq n$, is small. Therefore, the second condition for the hidden layer outputs is to increase the Hamming distance between the hidden layer outputs for all possible combinations of two input patterns.

In the proposed algorithm, the optimum targets, satisfying the above conditions, are assigned to the hidden layer. The targets are generated using a binary Hadamard matrix, which is obtained by replacing +1 with 0, and -1 with 1. One example is given by Eq.(4). The 1st row and 1st column are removed. The remaining rows are used as the hidden layer targets. The Hamming distance of all pairs of two

$$H_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad (4)$$

vectors is about $N_H/2$.

IV DISTRIBUTION OF CONNECTION WEIGHTS

The unit inputs are expressed by

$$x_j = \sum_{i=1}^{N_I} w_{ij} y_i \tag{5}$$

Suppose one of w_{ij}, $i=1\sim N_I$, is very large compared with the rest. If the corresponding y_i is changed by noise, then large change occurs in x_j, and the output y_j could be easily changed. This also occurs in the output layer.

Thus, the distribution of the weights should be minimized. For this purpose, we employ hysteresis threshold in both layers. For example, a threshold function in the hidden layer is given by

$$y_j = f(x_j) = \begin{cases} 1, & x_j \geqq \alpha_H \\ 0, & x_j \leqq -\alpha_H \end{cases} \tag{6}$$

In the training process, the threshold α_H is first set to be small, and is gradually increased. In this process, absolute values of weights are increased. On the other hand, by assuming finite length binary expression, their maximum value is limited. So, the distribution of the weights can be suppressed. At the same time, noise margin for the unit inputs can be gained.

V LOW-BIT LEARNING ALGORITHM

In a low-bit expression, the minimum change of weights is determined by a quantization step. Furthermore, the target is assigned to each layer. Therefore, supervised learning for a single-layer perceptron [4] can be basically applied. First, the weights w_{ij} from the input to the hidden layers are adjusted. After that, w_{jk} from the hidden to the output layers are adjusted using the adjusted w_{ij} as constant.

A concrete learning procedure is descried in the following.

(1) Initial guess for weights:

All weights are initially set to be zero.

(2) Targets for hidden and output layers:

A set of orthogonal vectors consisting of the same number of units as that of the input patterns are used. Only one unit is activated for one input pattern. The hidden layer targets are determined using the binary Hadamard matrix, as described in Sec.IV.

(3) Adjusting weights w_{ij}:

Calculate the input of the jth hidden unit for the mth input pattern by

$$x_{m,j} = \sum_{i=1}^{N_I+1} w_{ij} y_{m,i} \tag{7}$$

Supposing $y_{m,i}$ takes 1 or 0, $x_{m,j}$ can be calculated through summing the weights w_{ij}. y_{m,N_I+1} takes always 1, and $w_{N_I+1,j}$ is used to control bias. Connection weights are adjusted as follows:

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij} \tag{8}$$

Letting $d_{m,j}$ be the target for $y_{m,j}$, correction Δw_{ij} is determined by

· $d_{m,j}=1$ If $x_{m,j} \geqq \alpha_H$ then $\Delta w_{ij}=0$ (9a)

If $x_{m,j} < \alpha_H$ then $\Delta w_{ij}=\mu q$ (9b)

· $d_{m,j}=0$ If $x_{m,j} \leqq -\alpha_H$ then $\Delta w_{ij}=0$ (10a)

If $x_{m,j} > -\alpha_H$ then $\Delta w_{ij}=-\mu q$ (10b)

In the above equations, $y_{m,i}$ are assumed to be 1. If $y_{m,i}=0$, then $\Delta w_{ij}=0$. μ is a learning rate, which takes an integer number, and q is a quantization step.

The above process is repeated for all patterns until Δw_{ij} becomes zero. In this process, convergence is not always guaranteed due to a single layer. However, the weights can approach to a near optimum solution.

The initial value of α_H is set to be small. After the training converges, α_H is further increased. Using the updated α_H, the above adjusting is repeated. α_H is gradually increased just before the training fails.

At this step, adjusting w_{ij} is completed.

(4) Calculate output of hidden units:

The inputs are calculated by Eq.(7). The outputs are obtained by

$$y_{m,j} = \begin{cases} 1, & x_{m,j} \geqq 0 \\ 0, & x_{m,j} < 0 \end{cases} \tag{11}$$

The adjusted weights w_{ij} are fixed in this process.

(5) Adjusting weight w_{jk}:

The inputs of the output layer, for the mth input pattern, are calculated by

$$x_{m,k} = \sum_{j-1}^{NH+1} w_{jk} y_{m,j} \tag{12}$$

$y_{m,j}$ are fixed in this step. Since $y_{m,j}$ takes 1 or 0, the above equation also requires only summation of weights w_{jk}. $y_{m,NH+1}$ and $w_{NH+1,k}$ are also used to control bias. Adjusting w_{jk} is carried out in the same manner as in step (3). The threshold, α_O, is also gradually increased during the training process.

VI PATTERN RECOGNITION

After the training was completed, pattern recognition is carried out using the adjusted weights. The input pattern is recognized by comparing the targets $d_{m,k}$ and the actual outputs $y_{m,k}$ of the output layer. Threshold levels for both hidden and output layers are set to be zero.

Noisy Pattern Recognition:

When noises are added to the input patterns, the unit inputs are decreased to some extent. Therefore, the recognition process is modified as follows:

(1)The number of hidden units, whose output will be 1, is fixed. The corresponding hidden units are selected in order of magnitude of their input. Since, in the training process, the number of the hidden units, whose output is 1, is specified to almost $N_H/2$, the above constraint is useful in the recognition process. The actual number of the hidden units is chosen to be slightly less than $N_H/2$, due to the noises.

(2)The threshold level in the output layer is slightly decreased from zero. Because the inputs tend to decrease due to the noises. The former modification is valid for only the proposed method. The latter modification is, however, valid for not only the proposed but also the conventional methods.

Ⅶ SIMULATION

A. Neural Network and Targets

Input, hidden and output layers include 16x16=256, 63 and 62 units, respectively. Alphabet letters A~Z, a~z and numbers 0~9 are applied. So, the total number of patterns is 62. Some examples are shown in Fig.2(a). The targets for the hidden layer are obtained from a 64x64 binary Hadamard matrix. The unit outputs are always expressed with 0, 1 in both layers. A fixed point binary number system is employed. The number of bits for weights are examined using 7 bits and 4 bits.

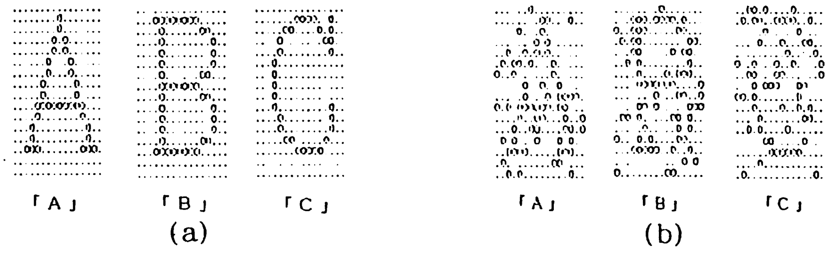

Fig.2 Examples of patterns. (a) Training patterns. (b) Noisy patterns.

B. Simulation Results

Figures 3(a) and 3(b) show the learning curves in adjusting w_{ij} and w_{jk}, respectively. The horizontal axis indicates the number of iteration. The vertical axis in Fig.3(a) indicates error in the hidden layer, defined by

$$\mathrm{Error} = \frac{1}{N_P}\sum_{m=1}^{NP}\left\{\frac{1}{N_H}\sum_{j=1}^{NH}\left[d_{m,j}T_+(\alpha_H, x_{m,j}) + (1-d_{m,j})T_-(\alpha_H, x_{m,j})\right]\right\} \qquad (13)$$

$$T_+(\alpha_H, x_{m,j}) = \begin{cases} 0, & x_{m,j} \geqq \alpha_H, \\ \alpha_H - x_{m,j}, & x_{m,j} < \alpha_H \end{cases} \qquad (14a)$$

$$T_-(\alpha_H, x_{m,j}) = \begin{cases} 0, & x_{m,j} \leqq -\alpha_H, \\ x_{m,j} + \alpha_H, & x_{m,j} > -\alpha_H \end{cases} \qquad (14b)$$

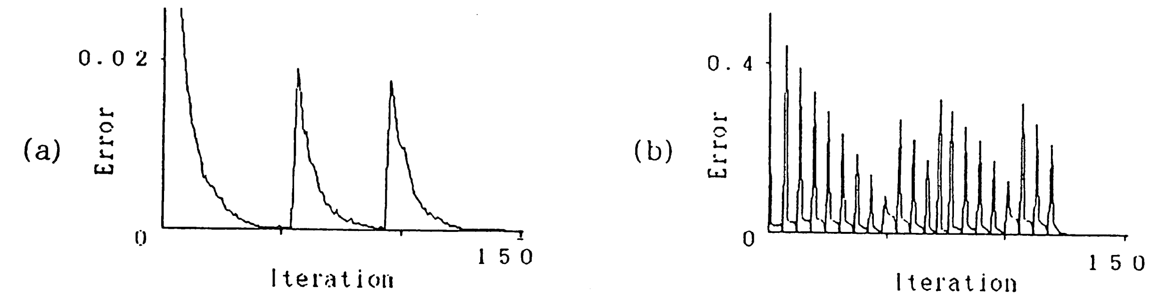

Fig.3 Learning curves. (a) Hidden layer. (b) Output layer.

Error in the output layer, shown in Fig.3(b), is calculated in the same manner. The weights are expressed with 7 bits. α_H and α_O start with 1/8 and 1/2, and are increased by 1/8, 1/2, respectively. Their final values are 3/8 and 21/2, respectively. The errors are periodically increased, due to changing values of α_H and α_O. The learning rate μ is chosen to be 1. Thus, weights are changed by a quantization step.

Recognition of noisy patterns were simulated. Examples of noisy patterns are illustrated in Fig.2(b). Figure 4 shows the results. The vertical axis indicates the total number of patterns, which cannot be recognized under the corresponding number of noises on the horizontal axis. In order to statistically evaluate recognition rates, ten kinds of noise patterns are used, and the average number of noises is shown.

Figure 4 includes the following algorithms:

① BP algorithm, with a sufficient number of bits.
② Conventional low-bit learning algorithm [2],[3].
③ Proposed low-bit learning algorithm.
④Proposed low-bit learning without the hidden layer.

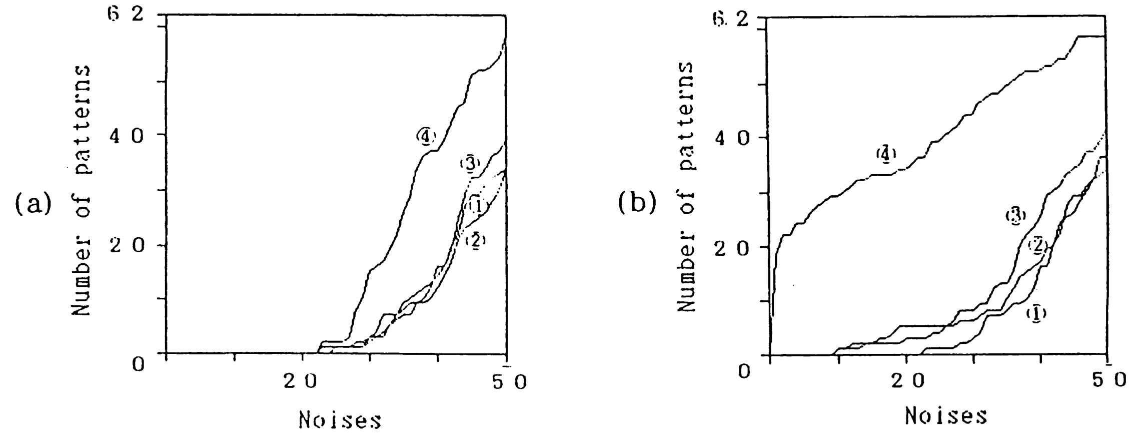

Fig.4 Number of patterns, cannot be recognized under the corresponding number of noises. The number of bits for weights is 7 bits (a), and 4 bits (b). Unit outputs are expressed with two-level values (1, 0) in both cases.

The weights are expressed with 7 bits and 4 bits in Figs.4(a) and 4(b), respectively. The unit outputs are always expressed with two-level values (1,0). The proposed algorithm ③ can provide almost the same recognition rates compared with ① and ②. Thus, feature of the proposed method is to drastically simplify digital hardware in the training process, while maintaining almost the same performance as in the conventional methods ① and ②.

The curves ④, in Figs.4(a) and 4(b), show the case, where the hidden layer is not used. Procedures of adjusting weights are the same as in ③. From these results, a single-layer neural networks is not useful for low-bit learning.

VI CONCLUSIONS

A new low-bit learning algorithm for digital multilayer neural networks has been proposed. The optimum target for the hidden layer has been developed. The training is carried out in each layer, using variable hysteresis threshold. Computer simulation demonstrated that the proposed algorithm can carried out the training, with a small number of bits, maintaining high recognition rates.

REFERENCE

[1]D.E.Rumelhart and J.T.McClelland, Parallel Distributed Processing, MIT Press, 1986.

[2]K.Nakayama, S.Inomata and Y.Takeuchi,"A digital multilayer neural network with limited binary expressions", IJCNN'90, San Diego, pp. II-587-592, June 1990.

[3]K.Nakayama, et al,"Reductions in number of bits for digital realization of multilayer neural networks (in Japanese)", IEICE Trans. vol.J73-D-II, No.8, pp.1336-1345, August 1990.

[4]F.Rosenblatt, Principles of Neurodynamics, New York, Spartan Book 1962.