tained by carrying out 25 runs per trial, each run being 200 iterations of the EM algorithm.

In the second experiment the true matrix Φ was

$$\Phi = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -0.87 & 1.096 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -0.870 & 0.899 \end{pmatrix}.$$

The complex envelope's power spectra of the two signals is depicted in Fig. 2. The sources power ratio was 1 : 1. Table II summarizes the results of this experiment.

Finally, in Fig. 3 we provide an example of the convergence rates of the proposed algorithm for three typical runs of the trial corresponding to SNR = 15 dB.

Remark: Note that experiment 1's results are markedly different from the results of the ad hoc method in [3] which requires orders of magnitude more data points for the same setting.

References

[1] A. D. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc.*, vol. B-39, pp. 1-37, 1977.

[2] R. H. Shumway and D. S. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *J. Time Series Anal.*, vol. 3, no. 4, pp. 253-264, 1982.

[3] I. Ziskind and Y. Bar-Ness, "Direction finding of narrow-band autoregressive sources by antenna arrays," *IEEE Trans. Signal Processing*, vol. 40, no. 2, pp. 484-487, Feb. 1992.

Multilayer Feedforward Neural Networks with Single Powers-of-Two Weights

Chuan Zhang Tang and Hon Keung Kwan

***Abstract*—A new algorithm for designing multilayer feedforward neural networks with single powers-of-two weights is presented in this correspondence. By applying this algorithm, the digital hardware implementation of such networks becomes easier as a result of the elimination of multipliers. This proposed algorithm consists of two stages. First, the network is trained by using the standard backpropagation algorithm. Weights are then quantized to single powers-of-two values, and weights and slopes of activation functions are adjusted adaptively to reduce sum of squared output errors to a specified level. Simulation results indicate that the multilayer feedforward neural networks with single powers-of-two weights obtained using the proposed algorithm have similar generalization performance as the original networks with continuous weights.**

I. Introduction

Multilayer feedforward structure trained by the backpropagation (BP) algorithm [1] has been one of the most widely used artificial neural network models. However, the digital implementation of such networks is still an important issue. Among the computations involved in feedforward networks, it is the multiplication that makes it difficult to implement the network with digital hardware [2], [3]. One possible way to solve this problem is to use powers-of-two weights such that multipliers are replaced by shift registers. Such techniques have been applied successfully to digital filter designs [4]-[7]. The application of the powers-of-two technique in multilayer feedforward neural networks had also been studied by Marchesi *et al.* [8], [9], in which adaptive biases and automatic learning rate control were employed in order to compensate the quantization error. The quantization scheme they adopted appears to be fairly complicated. Moreover, minimization of the sum of squared weight quantization error as adopted in [8], [9] does not necessarily reduce the sum of squared output error of the network because this is a nonlinear system. It will be much simpler to adopt direct quantization of weights to their nearest single powers-of-two values which we will introduce in the following.

In the current correspondence, a new algorithm for designing multilayer feedforward neural networks with single powers-of-two weights will be presented. This algorithm consists of two stages. First, the backpropagation algorithm is used to train the network repeatedly with continuous weights until converging to a predetermined error level. The obtained weights will then be quantized to values of single powers-of-two and the slopes of activation functions will be adjusted adaptively to reduce the sum of squared output errors to the same level as in the first stage while weights will still be adjusted slightly under the constraint of single powers-of-two values.

II. The Algorithm

A. The Backpropagation [1]

Consider the multilayer feedforward neural network shown in Fig. 1. The output of a neuron j at a layer h is computed by

$$y_{jk}^{[h]} = F\left(\sum_{i=1}^{N_{h-1}} w_{ij}^{[h]} y_{ik}^{[h-1]} + \theta_j^{[h]}\right) \quad \text{for } h = 1, 2, \cdots, L \tag{1}$$

where $F(\,)$ is a nonlinear activation function; $y_{ik}^{[h-1]}$ is the activation of a neuron i at the layer $(h-1)$; $w_{ij}^{[h]}$ is the connection weight between a neuron i at the layer $(h-1)$ and a neuron j at the layer h; $\theta_j^{[h]}$ is the threshold value of a neuron j at the layer h; and N_{h-1} is the number of neurons at the layer $h-1$. For $h = 1$, $y_{ik}^{[h-1]} = x_{ik}$, where x_{ik} is the ith element of an input pattern k. This operation is illustrated by Fig. 2. The sum of squared output errors (SSE) of the network related to a pattern k is

$$e_k = \sum_{j=1}^{N_L} (t_{jk} - y_{jk}^{[L]})^2. \tag{2}$$

Here t_{jk} represents the jth element of target pattern k and L refers to the output layer. The change in weight $w_{ij}^{[h]}$ is defined as

$$\Delta_k w_{ij}^{[h]} = -\epsilon \frac{\partial e_k}{\partial w_{ij}^{[h]}} \quad \text{for } 1 \le h \le L \tag{3}$$

where ϵ is a learning rate parameter for weights. After some derivations, the following relations can be obtained:

$$\Delta_k w_{ij}^{[h]} = \epsilon\, \delta_{jk}^{[h]} y_{ik}^{[h-1]} \quad \text{for } i = 1, 2, \cdots, N_{h-1}, \; j = 1, 2, \cdots, N_h \tag{4}$$

Manuscript received June 18, 1992; revised September 22, 1992. The associate editor coordinating the review of this correspondence and approving it for publication was Prof. J. N. Hwang. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and in part by the Micronet.

The authors are with the Department of Electrical Engineering, University of Windsor, Windsor, Ontario, Canada N9B 3P4.

IEEE Log Number 9209388.

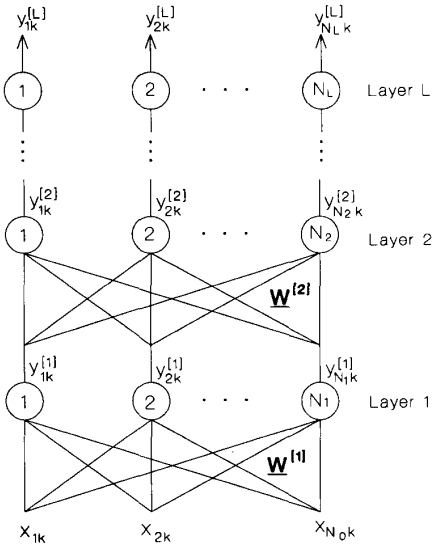

Fig. 1. Multilayer feedforward neural network.

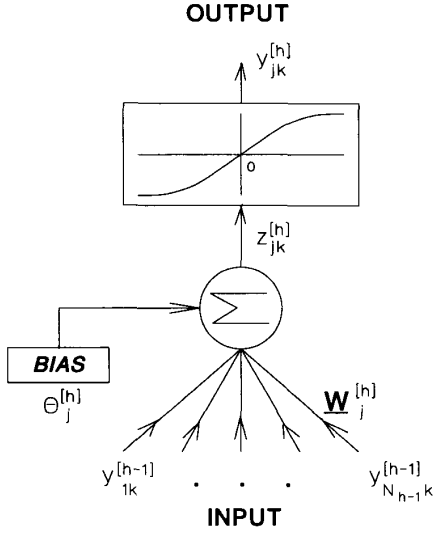

Fig. 2. Structure of neuron j at layer h.

where

$$\delta_{jk}^{[h]} = F'(z_{jk}^{[h]}) \sum_{i=1}^{N_{h+1}} \delta_{ik}^{[h+1]} w_{ji}^{[h+1]} \qquad \text{for } h < L \tag{5}$$

and

$$\delta_{jk}^{[L]} = 2(t_{jk} - y_{jk}^{[L]})\, F'(z_{jk}^{[L]}) \tag{6}$$

where $F'(z_{jk}^{[h]})$ represents the partial derivative of $F(z_{jk}^{[h]})$ with respect to $z_{jk}^{[h]}$ for $h = 1$ to L. We update weights only at the end of each epoch. Thus we have

$$w_{ij}^{[h]}(t+1) = w_{ij}^{[h]}(t) + \sum_{k=1}^{K} \Delta_k w_{ij}^{[h]} \tag{7}$$

where K represents the total number of patterns used for training.

The main drawback of BP algorithm is the slow convergence speed of the learning process. Several modifications have been proposed to improve it. One of the most significant improvements was achieved by adaptively adjusting the slope of nonlinear functions besides updating weights, which will be described below.

B. Adaptive Slope of Activation Functions [10]

The activation function under consideration is given by

$$F(z) = \frac{1 - e^{-\alpha z}}{1 + e^{-\alpha z}}. \tag{8}$$

From (5) and (6), we can see that the slope of the activation functions, $F'(z)$, plays an important role in weights update procedure. In addition to other factors, a proper choice of the slope of activation functions can result in a better adjustment of weights. Furthermore, the slope of sigmoidal functions also affects the activation of each neuron which is the input to other neurons and in this way it can affect the input–output mapping. Obviously, by adjusting α, we can adjust the slope of sigmoidal functions and further control the amount of change in weights and value of activations. The idea of gradient descent can also be applied to adapt the parameter α, i.e.,

$$\Delta_k \alpha_j^{[h]} = -\epsilon_\alpha \frac{\partial e_k}{\partial \alpha_j^{[h]}} \qquad \text{for } 1 \le h \le L \tag{9}$$

where ϵ_α is a step size for α. Similar to (5) and (6), we have

$$\Delta_k \alpha_j^{[h]} = \epsilon_\alpha F'_\alpha(z_{jk}^{[h]}, \alpha_j^{[h]}) \sum_{i=1}^{N_{h+1}} \delta_{ik}^{[h+1]} w_{ji}^{[h+1]} \qquad \text{for } h < L \tag{10}$$

and

$$\Delta_k \alpha_j^{[L]} = 2\epsilon_\alpha (t_{jk} - y_{jk}^{[L]}) F'_\alpha(z_{jk}^{[L]}, \alpha_j^{[L]}) \tag{11}$$

where $F'_\alpha(z, \alpha)$ is the partial derivative of the nonlinear function with respect to α. The formula for updating α is

$$\alpha_j^{[h]}(t+1) = \alpha_j^{[h]}(t) + \sum_{k=1}^{K} \Delta_k \alpha_j^{[h]} + \mu_\alpha \{\alpha_j^{[h]}(t) - \alpha_j^{[h]}(t-1)\} \tag{12}$$

where μ_α is the momentum for α.

III. Feedforward Neural Networks with Powers-of-Two Weights

Consider a multilayer feedforward neural network where the nonlinear function applied at the output of each neuron is the sigmoidal function as given by (8), our objective is to realize the network with powers-of-two weights, i.e., all the weights can only take values on

$$\{\pm 1, \pm 2^{-1}, \pm 2^{-2}, \cdots, \pm 2^{-M}, 0\} \tag{13}$$

where M determines the number of quantization levels.

As stated in Section I, our algorithm consists of two basic stages: standard backpropagation and slope adjustment with powers-of-two weights. For a given set of training pattern pairs $\{X_k, T_k\}$ (input patterns and corresponding targets), let $w_{ij}^{[h]}$ (for predetermined i, j, and h) denote the continuous weights obtained by using the standard backpropagation algorithm. We then round these weights to their closest single powers-of-two values. Obviously, in order to quantize $w_{ij}^{[h]}$ to powers of two, normalization is needed. Suppose that the maximum absolute value of weights obtained is $w_{\max}$, then $w_{ij}^{[h]}$ divided by $w_{\max}$ gives the normalized weight $w_{ij}'^{[h]}$ which belongs to the interval $[-1, +1]$.

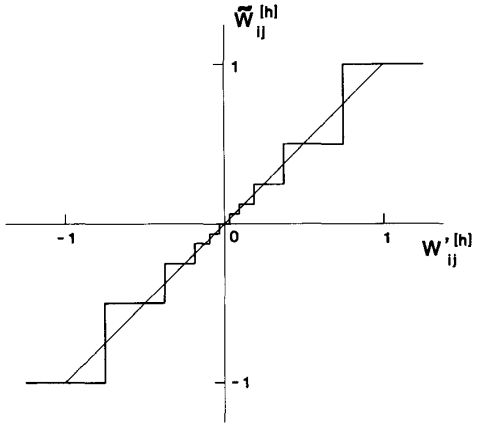

Fig. 3. Quantization curve for $M = 4$.

It can be seen that the input–output mapping relationship will be changed if the normalized weight $w_{ij}'^{[h]}$ is substituted directly into (1). Something has to be done to retain the original input–output relationship. Since the activation function under consideration is sigmoidal as given in (8), this problem can be solved by adjusting the coefficient α properly, i.e., in this case setting $\alpha = w_{\max}\alpha$.

Now, we are in a position to proceed to the second stage, quantizing the normalized weights $w_{ij}'^{[h]}$ to powers-of-two values. The quantization curve for the case of $M = 4$ is given in Fig. 3. Generally, this quantization will result in an increase in SSE. In order to reduce this SSE to a preassigned level with the constraint of powers-of-two weights, our strategy is to adjust adaptively the slope of activation functions employing the method described in Section II-B.

Step-by-step procedures to carry out the algorithm of designing feedforward neural networks with single powers-of-two weights are listed as follows.

1) Starting with small random weights, set $\alpha_j^{[h]} = 1$, where $\alpha_j^{[h]}$ is related with the sigmoid function applied to a neuron j at the layer h. Train the network using the backpropagation algorithm until $\Sigma_{k=1}^{K} e_k < E$.

2) Find the maximum absolute value among all weights $w_{ij}^{[h]}$ obtained in step 1, dentoed by $w_{\max}$, and normalize all weights by $w_{\max}$ as below so that new weights $w_{ij}'^{[h]}$ will fall into the interval $[-1, +1]$.

$$w_{ij}'^{[h]} = \frac{w_{ij}^{[h]}}{w_{\max}}.$$

3) Set $\alpha_j^{[h]} = w_{\max}\alpha_j^{[h]}$.

4) Quantize the normalized weights $w_{ij}'^{[h]}$ to single powers-of-two weights $\tilde{w}_{ij}^{[h]}$ as

$$\tilde{w}_{ij}^{[h]} = Q[w_{ij}'^{[h]}] = \operatorname{sgn}(w_{ij}'^{[h]}) \begin{cases} 0 & |w_{ij}'^{[h]}| < C_M \\ 2^{-m} & C_m < |w_{ij}'^{[h]}| < C_{m-1} \end{cases} \quad (14)$$

where sgn () denotes the sign of () and $C_m = (3/4)2^{-m}$ for $m = 0, 1, \cdots, M$.

5) Calculate the SSE of the network using quantized weights $\tilde{w}_{ij}^{[h]}$ and current values of $\alpha_j^{[h]}$. If SSE $< E$, stop; otherwise, proceed to step 6.

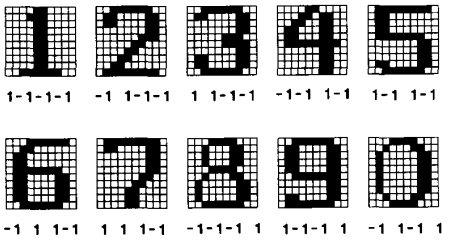

Fig. 4. Training pattern pairs.

6) Compute changes in $\tilde{w}_{ij}^{[h]}$ and $\alpha_j^{[h]}$ as in Section II.

7) Update weights $\tilde{w}_{ij}^{[h]}$ with values obtained in step 6 and quantize them to the nearest single powers-of-two as in step 4. If this update can result in a reduced SSE, accept new weights; otherwise, discard them and keep previous values.

8) Update values of $\alpha_j^{[h]}$.

9) Go to step 5.

IV. Simulation Results

Simulations have been conducted in order to test the validity of the above design procedures. The training patterns used in simulations were 10 numerals, each represented by a 10×10 pixel matrix as shown in Fig. 4, and the corresponding targets were 4-b bipolar codes as shown below each pattern. The neural network used was the feedforward model of Fig. 1 with 100 inputs and 4 outputs. The number of hidden layers and the number of neurons at hidden layers were variables. Several combinations of the number of layers, the number of neurons, and the number of quantization levels were used in simulations to test the performance of the algorithm under different conditions.

Two aspects of behavior, i.e., the convergence and generalization properties of the algorithm, have been observed in simulations. For each topology of the network and the number of quantization levels, the network was first trained with the given 10 pattern pairs to obtain both continuous and quantized solutions, i.e., weights for the given problem; then the same set of noisy patterns (original patterns corrupted by noise) was fed separately to both the continuous-weight network and the powers-of-two-weight network to test the generalization ability. Bipolar inputs and outputs with amplitude of ± 0.9 were used instead of ± 1 in order to achieve a faster convergence. Noisy patterns were constructed by inverting randomly a percentage of total elements in training patterns. For the simulations presented below, this percentage was 5.0%. The recall accuracy (the percentage of correct recalls) was obtained by feeding 100 noisy versions of each pattern to the network and taking the average.

Tables I and II show the results for one hidden layer networks under the conditions of different number of hidden neurons and different number of quantization levels. The item with "/" means convergence was never reached. Tables III and IV present the simulation results for networks with different number of hidden layers and hidden neurons when $M = 4$. For the case of two hidden layers, identical size for both layers was assumed, i.e., both hidden layers had the same number of neurons. All the data given in these tables were averages of five runs of the algorithm, starting with different initial weights, which were random numbers uniformly distributed in $[-0.1, +0.1]$. Other parameters used were: $\epsilon = 0.01$; $\epsilon_\alpha = 0.15$; $\mu_\alpha = 0.05$; and $E = 0.1$.

Some conclusions can be drawn from the simulation results. First, convergence was reached in almost all runs. This implies the effectiveness of the algorithm proposed and the possibility of im-

TABLE I
CONVERGENCE SPEEDS (IN NUMBER OF EPOCHES) FOR BP AND POWERS-OF-TWO NETWORKS (100 INPUTS, 4 OUTPUTS, AND 1 HIDDEN LAYER)

No. of Hidden Neurons	BP	Powers-of-Two		
		$M = 2$	$M = 4$	$M = 8$
10	61.6	/	55.2	19.6
20	42.8	1366.6	3.8	3.4
40	32.4	34.0	2.0	2.0
60	28.0	9.4	2.0	2.0
80	25.4	13.0	2.0	2.0
100	24.0	2.2	2.0	2.0

TABLE II
GENERALIZATION CAPABILITIES (IN PERCENTAGE OF CORRECT RECALLS) FOR BP AND POWERS-OF-TWO NETWORKS (100 INPUTS, 4 OUTPUTS, AND 1 HIDDEN LAYER)

No. of Hidden Neurons	BP	Powers-of-Two		
		$M = 2$	$M = 4$	$M = 8$
10	99.58%	/	98.92%	98.98%
20	99.46%	96.14%	99.16%	99.24%
40	99.46%	98.94%	99.28%	99.34%
60	99.48%	99.08%	99.46%	99.50%
80	99.60%	99.00%	99.40%	99.40%
100	99.52%	99.04%	99.42%	99.42%

TABLE III
CONVERGENCE SPEEDS FOR NETWORKS WITH DIFFERENT NUMBER OF HIDDEN LAYERS WHEN $M = 4$ (100 INPUTS AND 4 OUTPUTS)

No. of Hidden Neurons	One Hidden Layer		Two Hidden Layers	
	BP	Powers-of-Two	BP	Powers-of-Two
10	61.6	55.2	131.2	208.6
20	42.8	3.8	64.6	10.2
40	32.4	2.0	41.0	2.8
60	28.0	2.0	32.0	2.0
80	25.4	2.0	26.4	2.0
100	24.0	2.0	22.6	2.0

TABLE IV
GENERALIZATION CAPABILITIES FOR NETWORKS WITH DIFFERENT NUMBER OF HIDDEN LAYERS WHEN $M = 4$ (100 INPUTS AND 4 OUTPUTS)

No. of Hidden Neurons	One Hidden Layer		Two Hidden Layers	
	BP	Powers-of-Two	BP	Powers-of-Two
10	99.58%	98.92%	99.14%	98.42%
20	99.46%	99.16%	99.52%	99.30%
40	99.46%	99.28%	99.34%	99.04%
60	99.48%	99.46%	99.44%	99.18%
80	99.60%	99.40%	99.60%	99.36%
100	99.52%	99.42%	99.62%	99.42%

plementing feedforward neural networks with single powers-of-two weights. Next, the algorithm proposed can retain similar generalization capability as the original continuous-weight network. This can be seen from the fact that the degrade in performance of powers-of-two-weight networks is only about one percent in almost all the tests. Furthermore, in order to achieve good performance, parameters like the number of hidden neurons and the number of quantization levels should be chosen carefully. Small number of hidden neurons combined with small number of quantization levels should be avoided. Some redundancy in topology of the network is recommended in order to reduce the length of shift registers.

V. CONCLUSIONS

This correspondence proposed a new design algorithm of implementing multilayer feedforward neural networks with single powers-of-two weights. This algorithm starts from the continuous weights obtained by the standard backpropagation and employs adaptive slopes of activation functions to compensate the output error caused by the quantization of weights. The resulted network operates faster and is easier for implementation than the original continuous-weight network while maintaining nearly the same performance. A step-by-step design procedure has been presented. Simulation results have also been provided, which verify the effectiveness of the proposed algorithm.

REFERENCES

[1] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing*, vol. 1. Cambridge, MA: M.I.T. Press, 1986.

[2] H. K. Kwan, "Systolic architectures for Hopfield network, BAM, and multilayer feedforward network," in *Proc. IEEE Int. Symp. Circuits Syst.*, Portland, OR, May 1989, vol. 2, pp. 790–793.

[3] H. K. Kwan and P. C. Tsang, "Systolic implementation of multilayer feedforward neural network with back-propagation learning scheme," in *Proc. Int. Joint Conf. Neural Networks*, Washington, DC, Jan. 1990, vol. II, pp. 155–158.

[4] Y. C. Lim and A. G. Constantinides, "Linear phase FIR digital filter without multipliers," in *Proc. IEEE Symp. Circuits Syst.*, Tokyo, Japan, July 1979, pp. 185–188.

[5] Y. C. Lim, S. R. Parker, and A. G. Constantinides, "Finite wordlength FIR filter design using integer programming over a discrete coefficient space," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 30, pp. 661–664, Aug. 1982.

[6] H. K. Kwan and C. L. Chan, "Circularly symmetric two-dimensional multiplierless FIR digital filter design using an enhanced McClellan transformation," *Proc. Inst. Elec. Eng.* (part G Circuits, Devices, and Systems), vol. 136, no. 3, pp. 129–134, June 1989.

[7] H. K. Kwan and C. L. Chan, "Design of multidimensional spherically symmetric and constant group delay recursive digital filters with sum of powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1027–1035, Aug. 1990, and pp. 1580, Dec. 1990.

[8] M. Marchesi, N. Benvenuto, G. Orlandi, F. Piazza, and A. Uncini, "Design of multilayer neural networks with powers-of-two weights," in *Proc. IEEE Symp. Circuits Syst.*, New Orleans, LA, May 1990, vol. 4, pp. 2951–2954.

[9] M. Marchesi, G. Orlandi, F. Piazza, L. Pollonara, and A. Uncini, "Multilayer perceptrons with discrete weights," in *Proc. Int. Joint Conf. Neural Networks*, San Diego, CA, June 1990, vol. II, pp. 623–630.

[10] A. Rezgui and N. Tepedelenlioglu, "The effect of the slope of the activation function on the back-propagation algorithm," in *Proc. Int. Joint Conf. Neural Networks*, vol. I, Washington, DC, Jan. 1990, pp. 707–710.