and the upper box corresponding to the upper polynomial

$$P_u(z) = a_2 z^2 + a_3 z + a_4 \tag{4.11}$$

has $K = 2^{(2+2+2)} = 64$ corner polynomials. Therefore, the total number of the edges to check for robust stability of $f(z)$ is $H = K * \hbar = 64 * 12 = 768$, where each one can be checked for stability using the method of [12].

This approach requires more computation, but it gives necessary and sufficient condition for the robust stability of interval polynomial $f(z)$ in (1.1) and (1.2).

V. Conclusion

There are several approaches for robust stability analysis of discrete-time uncertain systems with real coefficients [1]–[4]. In this paper, these results are generalized for uncertain systems with complex coefficients. In the case of [1], [2], this generalization is valid, and one needs to check a limited number of *edge* polynomials to obtain necessary and sufficient conditions for robust polynomial stability, while in the case of [3], one needs to check a limited number of *corner* polynomials, which are easier to check at the expense of having only sufficient condition. In the case of [4], a frequency-domain interpretation of the extension of the *corner* approach is presented. In all three approaches, the number of polynomials whose stability has to be checked grows with the polynomial degree, in contrast to the continuous-time case.

References

[1] C. V. Hollot and A. C. Bartlett, "Some discrete-time counterparts to Kharitonov's stability criteria for uncertain systems," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 355–356, 1986.

[2] F. Kraus, M. Mansour, and E. I. Jury, "Robust Schur stability of interval polynomials," in *Proc. 28th IEEE CDC*, Tampa, FL, pp. 1908–1910, Dec. 1989.

[3] F. Kraus, B. D. O. Anderson, and M. Mansour, "Robust Schur polynomial stability and Kharitonov's theorem," *Int. J. Cont.*, vol. 47, pp. 1213–1225, 1988.

[4] M. Mansour, F. J. Kraus, and B. D. O. Anderson, "Strong Kharitonov theorem for discrete systems," in *Proc. 27th IEEE CDC*, Austin, TX, pp. 106–111, Dec. 1988.

[5] N. K. Bose and E. Zeheb, "Kharitonov's theorem and stability test of multidimensional digital filters," *IEE Proceedings*, vol. 133, no. 4, pp. 187–190, Aug. 1986.

[6] P. P. Vaidyanathan, "Derivation of new and existing discrete-time Kharitonov theorems based on discrete-time reactances," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 277–285, Feb. 1990.

[7] A. Katbab and E. I. Jury, "On Kharitonov-type results for complex coefficient interval Schur polynomials," to be published.

[8] V. L. Kharitonov, "On a generalization of a stability criterion," *Izv. Akad. Nauk. Kazakh. SSR Ser. Fiz. Mat.*, vol. 1, pp. 53–57, 1978 (in Russian).

[9] N. K. Bose and Y. Q. Shi, "A simple general proof of Kharitonov's generalized stability criterion," *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 1233–1237, Oct. 1987.

[10] Y. Bistritz, "Stability criterion for continuous-time systems polynomials with uncertain complex coefficients," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 442–448, Apr. 1988.

[11] R. J. Minnichelli, J. J. Anagnost, and C. A. Desoer, "An elementary proof of Kharitonov stability theorem with extensions," *IEEE Trans. Automat. Contr.*, vol. 34, pp. 995–998, Sept. 1989.

[12] N. K. Bose, "Tests for Hurwitz and Schur properties of convex combinations of complex polynomials," *IEEE Trans. Circuits Syst.*, vol. 36, Aug. 1989.

[13] A. C. Bartlett, C. V. Hollot, and H. Lin, "Root locations for a polytope of polynomials: It suffices to check the edges," *Math. Contr., Signals & Systems*, no. 1, pp. 61–71, 1988.

A Pyramidal Delayed Perceptron

G. MARTINELLI, L. PRINA RICOTTI, S. RAGAZZINI, AND F. M. MASCIOLI

***Abstract*—The choice of the structure of a multilayer perceptron and the learning algorithm are two critical issues of this type of neural network. We propose to follow a new approach based on linear programming. The resulting learning algorithm automatically yields the neural network under a pyramidal form and determines the weights of the connections among neurons.**

I. Introduction

A multilayer perceptron can form arbitrarily complex decision regions. As a consequence of Kolmogorov's theorem three layers are sufficient to achieve this goal [1]. The theorem, however, does not suggest any constructive method for shaping the neural network and for determining its topological and constitutive parameters.

When facing a particular application, we should determine 1) the topology of the network (number of layers, number of neurons per layer) and 2) the weights of the connections among neurons and the offsets.

The choice of the topology is usually based on qualitative considerations as 1) each neuron of the first layer forms half-plane decision regions; 2) each neuron of the second layer forms a convex decision region having at the most as many plane sides as there are neurons in the first layer; and 3) each output node forms a decision region constituted at the most by as many disconnected regions as there are neurons in the second layer.

The determination of the weights of the connections and of the offsets is carried out by presenting the available examples of the input–output mapping defined by the application at hand. The basic algorithm of learning is that proposed in [2], based on a gradient minimization of the square error between the target and the actual output of the network. The gradient is calculated by a back-propagation technique which involves quite naturally the hidden layers. The gradient descent is plagued by the requirement of determining an appropriate learning rate. Also if there are available several methods for overcoming this inconvenience [3]–[6], they do not yet solve satisfactorily the problem.

We probably need to follow different approaches in order to solve the two previous critical issues in multilayer neural networks. This conclusion is confirmed by the interesting recent results obtained by an algebraic interpretation of the learning algorithm [7]. For this reason we have explored the possibility of using the methodology of linear programming (LP) as a learning tool both for determining the topology and the constitutive parameters of the neural network. Our approach is motivated by the remark that the basic operation of each neuron of the single layer perceptron is similar to LP. As shown in Section II, a modified simplex technique yields very satisfactory results. The repeated application of this technique shapes the neural network under a pyramidal form in the case of a single output. The

Manuscript received June 26, 1989. This letter was recommended by Editor R. Liu.

G. Martinelli and F. M. Mascioli are with the Dipartimento INFOCOM, University of Rome "La Sapienza," Rome, Italy.

L. P. Ricotti and S. Ragazzini are with Fondazione Ugo Bordoni, Rome 00142, Italy.

IEEE Log Number 9037128.

layers of the pyramid operate successively as in the classical multilayer perceptron. In the latter, the signal to be processed is manipulated by the first layer, then the result is passed to the second layer and so on up to the last layer. A similar mechanism characterizes the pyramidal perceptron. However, since the number of layers can be in the latter case much larger than in the former, we denoted our perceptron as delayed. Moreover, owing to the presence of connections between non-contiguous layers it is necessary to explicitly evidentiate the delays in order to take into account the processing time required by each neuron. For this reason we have also shown the delays in the schemes of our perceptrons.

The paper is organized as follows. In Section II the basic algorithm used in the learning algorithm is described. As it will be shown, it derives from a suitable modification of the simplex. In Section III the learning algorithm will be introduced and justified. Finally, in Section IV, the behavior of the proposed neural network will be illustrated by examples.

II. The Modified Simplex Algorithm

The mapping to be realized by the neural network converts a real vector input having M components into a binary output. The case of a multiple binary output will be considered later. The mapping is known through N input–output patterns. The initial scheme of the neural network coincides with a single-neuron single-layer perceptron (SSP).

In Section III we will consider the final scheme obtained after learning. The output of the network is

$$u = f(v)$$
$$v = \sum_{k=1}^{M} a_k I_k + a_0 \tag{1}$$

where $f(v)$ represents the response of a hard-limiter and a_0 is the offset. In the following we will consider the offset as a weight by associating it to a connection between the neuron and an input I_0 always equal to 1.

The values of the weights must satisfy the constraints imposed by the available input–output patterns

$$\sum_{k=0}^{M} a_k I_k^{(i)} \begin{cases} > \Delta, & \text{if } u_i = 1, \\ > -\Delta, & \text{if } u_i = 0, \end{cases} \qquad i = 1,2,N \tag{2}$$

where i denotes the ith pattern and Δ is a positive quantity introduced for a better separation of the two classes of input–output patterns.

Constraints (2) take on the typical form of the linear constraints of an LP problem. We note that some of them could not be satisfied independently from the choice of the objective function. Let us rewrite (2) under the simplex formalism we intend to use:

$$\sum_{k=0}^{M} \left(a_k^{(+)} - a_k^{(-)}\right) I_k^{(i)} = \begin{cases} \Delta + x_i^{(+)} - x_i^{(-)}, & \text{if } u_i = 1 \\ -\Delta - \left(x_i^{(+)} - x_i^{(-)}\right), & \text{if } u_i = 0, \quad i = 1,2,N \end{cases} \tag{3}$$

where the quantities $a_k^{(+)}, a_k^{(-)}, x_i^{(+)}, x_i^{(-)}$ are nonnegative variables and the $I_k^{(i)}$ are constant coefficients. (Note that when the slack variable $x_i^{(-)}$ is equal to zero, the ith constraint is satisfied.)

What is important in the present case is to minimize the number of constraints not satisfied. We cannot express this objective function as a linear function of the previous variables. Since the magnitude of the error when a constraint is violated is not relevant, the minimization of the number of violated constraints cannot be pursued by the choice of a suitable linear objective function like, for instance, the sum of all the slack variables $x_i^{(-)}$. We have found that a result similar to the minimization of our objective function can be obtained by modifying the simplex exchange strategy, i.e., by guiding the exchange mechanism of the base variables. We note that system (3) is initially under a canonic form where all the error variables are base variables. The corresponding solution is obviously not valid. Therefore, we must undertake suitable exchanges in order to eliminate from the base the maximum number of error variables $x_i^{(-)}$. The proposed algorithm for obtaining this result is as follows:

1) Rewrite system (3) in such a manner that all constraint equations are not admissible, i.e., their right-hand terms are negative constant terms. The system is as follows:

$$-\sum_{k=0}^{M} \left(a_k^{(+)} - a_k^{(-)}\right) I_k^{(i)} + \left(x_i^{(+)} - x_i^{(-)}\right) = -\Delta, \qquad \text{if } u_i = 1$$
$$\sum_{k=0}^{M} \left(a_k^{(+)} - a_k^{(-)}\right) I_k^{(i)} + \left(x_i^{(+)} - x_i^{(-)}\right) = -\Delta, \qquad \text{if } u_i = 0,$$
$$i = 1,2,N. \tag{4}$$

In the examples shown in the following, Δ has been arbitrarily chosen to be equal to 1. We have noted that the ordering of the variables in (4) is very important in order to obtain the minimum possible number of violated constraints.

2) Choose the nonadmissible constraint equation with the constant term having maximum magnitude. Determine in this equation the new variable to be included in the base. The search consists in examining the coefficients of the variables following the order established in (4). Choose the first variable with negative coefficient occurring in the search. Let y be this variable.

3) Consider the admissible constraints where the coefficient of y is positive. Calculate for them the ratio between the constant term and the coefficient of y. Carry out the exchange of y using the constraint equation where the said ratio is minimum. If there are not admissible constraints of the above type, carry out the exchange of y using the nonadmissible equation considered in step 2).

4) Apply again steps 2) and 3).

The algorithm ends when there are not further nonadmissible equations. It is evident that the number of steps is finite; it is usually much less than N.

III. Learning Algorithm

The modified simplex algorithm (MSA) is initially applied to the solution of the given problem by means of the SSP.

The SSP scheme is too simple and is usually insufficient to solve the mapping of interest represented by the N given input–output patterns.

On the basis of the actual output of the network obtained by the MSA with respect to the target, we can divide the N patterns in the four groups of Table I.

The behavior of the single SSP taken into consideration is consequently not correct in correspondence to the $(N3 + N4)$ patterns of the groups $C3$ and $C4$. We can remediate this inconvenience by making its offset dependent on the pattern. Namely, we increase the offset (a_0) in the case of the patterns of

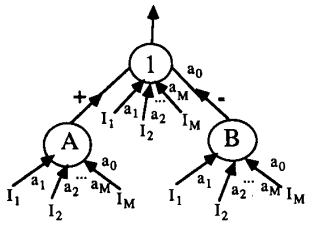

Fig. 1. The neural network considered in the second step of the learning algorithm. The circles denote SSP. The labels + and − indicate excitatory and inhibitory connections.

TABLE I

group	actual output	target	Number of patterns
C1	0	0	N1
C2	1	1	N2
C3	0	1	N3
C4	1	0	N4

TABLE II

group	output of A	output of B
C1	0	don't care
C2	don't care	0
C3	1	0
C4	0	1

group $C3$ and decrease it in the case of group $C4$. This variation of offset is obtained by connecting to the initial neuron two new SSP's as shown in Fig. 1. The neuron A increases the offset in the case of $C3$ and the neuron B decreases the offset in the case of $C4$. It is easy to derive the desired behaviors of the two new neurons in correspondence to the N patterns. Assuming an excitatory connection between the neurons A and 1 and an inhibitory connection between B and 1, their outputs must behave as summarized in Table II.

The number of patterns to be considered in realizing neurons A and B decreases with respect to that of neuron 1, if $N1$ and $N2$ are greater than or equal to 1. This condition can be easily satisfied by the SSP and is actually attained by the MSA.

The repeated application of the MSA requires the introduction of new SSP's in correspondence to every step and the consequent growth of the neural network under a pyramidal form. The growth is limited, since the number of patterns to be considered by the neurons decreases passing from one step to the successive one.

In conclusion the learning algorithm is as follows:

Step 1 (Vertex of Pyramid):

Apply the MSA to the SSP presenting the N input–output patterns representing the mapping of interest. Determine the groups $C1$, $C2$, $C3$, $C4$, of Table I.

Apply again the MSA taking account of the connections to the neurons A and B of the first layer. Determine the weights of these connections and those to the input. If the weight of the connection to A or B is zero, the corresponding neuron is eliminated.

Step 2 (First Layer of the Pyramid):

Apply the MSA to the neurons A and B of the first layer (Fig. 1), if both of them are required. The mappings to be realized are summarized in Table II. In the case A the patterns

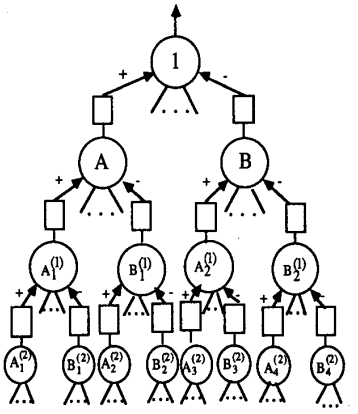

Fig. 2. Pyramidal delayed perceptron. The boxes represent a delay.

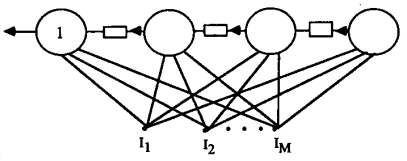

Fig. 3. Modified pyramidal delayed perceptron. The neurons without label may be either of type A or B.

to be considered are $(N-N2)$ and in the case B they are $(N-N1)$. Determine the groups like $C1, C2, C3, C4$. Apply again the MSA to A and B taking account of the connections to neurons $A_1^{(1)}, B_1^{(1)}, A_2^{(1)}, B_2^{(1)}$ of the second layer. Determine the weights of these connections and those to the input. If some of the weights of the connections to the neurons of the second layer are zero the corresponding neuron is eliminated;

Last Step (Last Layer of the Pyramid):

The last step of the learning algorithm regards a layer of neurons which realizes exactly the mappings required by them. This layer is eventually found after a limited number of steps, as guaranteed by the remark on the reduction of the number of patterns to be considered at each step.

The resulting neural network is shown in Fig. 2 in the case of three layers; it is characterized in the general case by a pyramidal shape and by the presence of delays between adjacent layers. It is important to note that each neuron is connected to the input as shown in Fig. 2. The operation of the network starts at the bottom and propagates up to the vertex.

Remark: The shape of the neural network obtained by the above learning algorithm depends on the ordering of the variables $a_k^{(+)}, a_k^{(-)}, x_i^{(+)}, x_i^{(-)}$ in the constraint equations and on the presentation order of the input patterns, when determining the weights of the connections of the neurons belonging to successive layers. In fact, we have noted that if we present the input patterns grouped together on the basis of the same required output, the MSA tends to yield neurons of only one type (A or B). We will present in Sect. IV examples at this regard. As a consequence of this behavior of the MSA we may alter the shape of the resulting network arriving in the limit to the scheme of Fig. 3. It is interesting to further remark that in this last case the number of neurons necessary to meet all the constraints is usually less than that required in the case of a pyramidal shape. The scheme of Fig. 3 looks like a single layer perceptron with delayed connections between adjacent neurons.

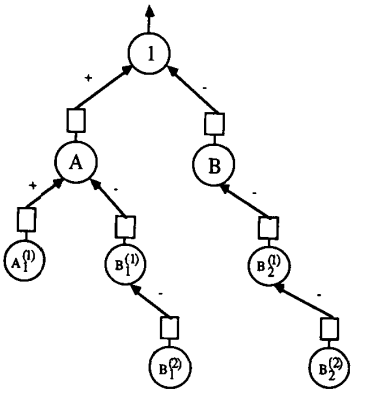

Fig. 4. Neural network obtained in Example 1 on the basis of a random ordering of the outputs of the input patterns.

3.1. Multiple Binary Output

The case of a mapping between a real vector input and a multiple binary output can be simply realized by decomposing it in as many single output mappings as there are outputs. The resulting neural network is characterized by a decoupling among outputs that is desirable in several applications. When this is not the case, we can modify the learning algorithm in order to share the neurons of the layers among the different outputs. For this purpose we apply the algorithm in sequel to the outputs and modify it in correspondence to each step by taking into account the connections between the neuron of interest (layer K) and those of the layer $(K+1)$ already considered in realizing the outputs processed up to that time.

IV. Examples

A remarkable characteristic of the proposed network is the autoconfiguration affected during the training. As a result, only those neurons that are necessary to the correction of the errors are inserted into the network. In fact, in all the examples we have worked out, only few layers of neurons were sufficient to solve the given input–output mapping without errors. Moreover, several excitatory or inhibitory connections were not actually added, so that the configuration of the network is usually much more sparse than the general complete scheme of Fig. 2 tending toward the scheme of Fig. 3. The experience gained in several examples has convinced us that the number of neurons of the pyramidal delayed perceptron is always very close to the minimum number that is required in order to solve without errors a given mapping.

Example 1:

The training sets consist of 103 patterns of 12 binary inputs and one binary output. The set of inputs is obtained affecting a phonological binary coding of the graphemes of words. The output associated with the 12 inputs in a given pattern points out an accent marker that should be used in a text preprocessing for a speech synthesis system, whose details can be found in [6]. The scheme of the pyramidal delayed perceptron obtained in the present case by our algorithm depends on the ordering, based on the required output, of the input patterns presented to the neurons. When the outputs of these patterns are random, we obtain the scheme of Fig. 4. When they are grouped on the basis of a same output we obtain the scheme of the type of Fig. 3. In both cases the number of neurons is less than or equal to that required by the classical perceptron with one hidden layer

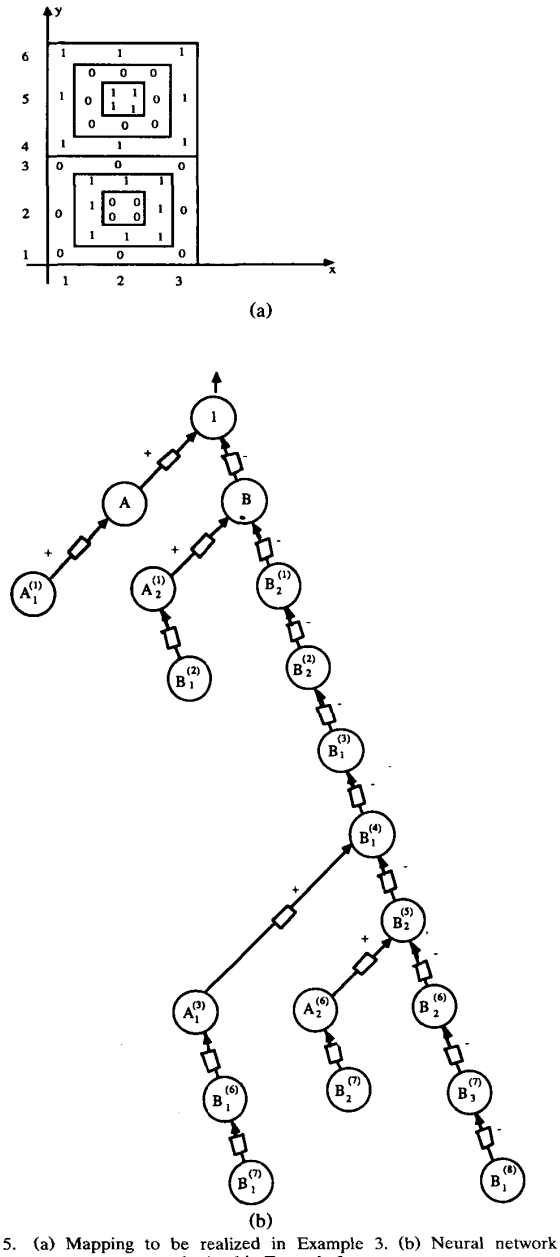

Fig. 5. (a) Mapping to be realized in Example 3. (b) Neural network obtained in Example 3.

trained by back-propagation [2]. Moreover, in this last case several trials, based on different numbers of neutrons, are necessary in order to determine the optimal scheme and each trial requires a large computational cost. By our method the optimal scheme is directly obtained without trials.

Example 2 (Parity Problem):

The parity problem requires, when using the perceptron, as many hidden neurons in the first layer as the components of the patterns [2]. Consequently, in the case of four components the network should contain at least five neurons. Applying our method we obtain a scheme requiring only three neurons when

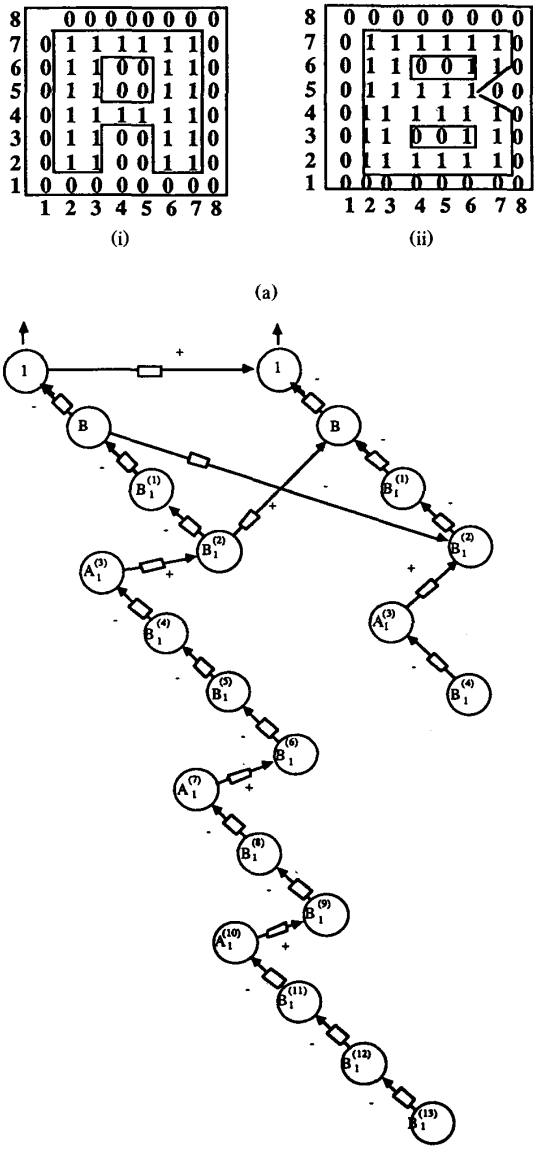

Fig. 6. (a) Mapping to be realized in Example 5. (i) Output 1. (ii) Output 2. (b) Neural network obtained in Example 5.

using the ordering of the input patterns. The values of the offsets and of the weights are as follows:

vertex: $a_0 = -7$, $a_1 = 2$, $a_2 = 2$, $a_3 = 2$, $a_4 = 2$;
neuron A: $a_0 = 5$, $a_1 = -2$, $a_2 = -2$, $a_3 = -2$, $a_4 = -2$;
connection A – vertex: 4;
neuron $B_1^{(1)}$: $a_0 = 3$, $a_1 = -2$, $a_2 = -2$, $a_4 = -2$;
connection $B_1^{(1)} - A = -6$.

Example 3:

The input pattern is a vector having as components the coordinates of a point in a plane. The neural network should distinguish if the point is or not inside the two disconnected regions shown in Fig. 5(a). The input patterns are 40 and represent the coordinates of the dots in Fig. 5(a). The resulting neural network is shown in Fig. 5(b).

Example 4 (Multiple Output, Binary Input):

The input and output are represented by binary bidimensional vectors. The mapping to be realized is shown below.

input	output
0 0	0 1
0 1	1 0
1 0	1 0
1 1	0 1

We apply the procedure described at the end of Section III consisting in sharing the neurons contained in the neural networks already calculated. It is interesting to note that the resulting scheme is obtained with a single neuron of layer 1 that is of type B with respect to the first output and of type A with respect to the other output. The values of the offsets and weights are

first vertex: $a_0 = 3$, $a_1 = -2$, $a_2 = -2$;
second vertex: $a_0 = -3$, $a_1 = 2$, $a_2 = 2$;
neuron B/A: $a_0 = 1$, $a_1 = -2$, $a_2 = -2$;
connection B/A – first vertex: -4;
connection B/A – second vertex: 4.

Example 5 (Multiple Output):

Two characters are represented on an 8×8 grid. The mapping determines if the dots in the grid belong to one of the two characters ($A = (1,0), B = (0,1)$), to both characters $(1,1)$ or to neither one $(0,0)$. We applied the multiple output procedure with ordering of the patterns and obtained the network of Fig. 6. It is interesting to note that the second output mapping is resolved using a smaller network that takes advantage of the previous network.

V. Conclusions

The choice of the number of layers and of the number of neurons per layer is a critical issue in the actual application of the multilayer perceptron. In the present paper a method is proposed for overcoming this inconvenience. The method requires to modify the structure of the perceptron by introducing suitable delays between adjacent layers and by connecting all the neurons to the input. It is possible by the proposed approach to automatically yield both the configuration of the network and the values of the weights of the connections. The new type of perceptron has a pyramidal shape as the "digital perceptron" introduced in [8]. However, the operation, the learning algorithm, and the type and connection of the input are completely different in the two cases.

References

[1] R. Hecht-Nielsen, "Kolmogorov's mapping neural network existence theorem," in *Proc. IEEE First. Int. Conf. on Neural Networks*, pp. III-11, III-14, 1987.
[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in D. E. Rumelhart, J. L. McClelland (Eds.), *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*. Cambridge, MA: MIT Press, 1986, chap. 3.
[3] D. B. Parker, "Optimal algorithm for adaptive networks: Second order back propagation, second order direct propagation and second order Hebbian learning," in *Proc. IEEE First Int. Conf. on Neural Networks*, pp. II-593, II-600, 1987.
[4] E. D. Dahl, "Accellerated learning using the generalized delta rule," in *Proc. IEEE First Int. Conf. on Neural Networks*, pp. II-523, II-530, 1987.
[5] R. L. Watrous, "Learning gradient methods for connectionist networks: Applied gradient methods of nonlinear optimization," in *Proc. IEEE First Int. Conf. on Neural Networks*, pp. II-619, II-627, 1987.
[6] L. Prina Ricotti, S. Ragazzini, and G. Martinelli, "Learning of word stress in a suboptimal second order back-propagation neural network," in *Proc. IEEE Second Int. Conf. on Neural Networks*, pp. I-355, I-361, 1988.
[7] S. Y. Kung and J. N. Hwang, "An algebric projection analysis for optimal hidden units size and learning rates in back-propagation learn-

ing," in *Proc. IEEE Second Int. Conf. on Neural Networks*, pp. I-363, I-370, 1988.
[8] J. J. Vidal, "Implementing neural nets with programmable logic," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1180–1190, July 1988.
[9] M. Dertouzos, *Threshold Logic*. Cambridge, MA: MIT Press, 1965.

Single Curve for Determining the Order of an Elliptic Filter

P.-M. LIN

***Abstract*—Determination of the order of an elliptic filter is usually done with the aid of filter tables, nomographs, or digital computer programs. In this note we provide a single curve for this purpose.**

I. Introduction

In the literature about analog filter design, the specifications of a low-pass filter are usually given in terms of the attenuation in decibels, denoted by α, as follows (see Fig. 1):

$$\alpha \leqslant \alpha_p, \qquad \text{for } 0 \leqslant \omega \leqslant \omega_p$$
$$\alpha \geqslant \alpha_s, \qquad \text{for } \omega \geqslant \omega_s.$$

For the determination of the orders of filter transfer functions, it is convenient to introduce the following two parameters:

$$\Omega \triangleq \omega_s / \omega_p \tag{1}$$

$$M \triangleq \sqrt{\frac{10^{0.1\alpha_s}-1}{10^{0.1\alpha_p}-1}}. \tag{2}$$

In [1] and several other references, $k = 1/\Omega$ is called the *selectivity* parameter and $k' = 1/M$ is called the *discrimination* parameter. In terms of Ω and M, the required orders of the transfer functions to meet the specifications are given by (see [2] for proofs):

Butterworth Filter

$$n_B \geqslant \frac{\log(M)}{\log(\Omega)} = \frac{\ln(M)}{\ln(\Omega)} \tag{3}$$

Chebyshev and Inverse Chebyshev Filter

$$n_c \geqslant \frac{\cosh^{-1}(M)}{\cosh^{-1}(\Omega)} \tag{4}$$

Elliptic Filter

$$n_E \geqslant \frac{f_E(M)}{f_E(\Omega)} \tag{5}$$

where

$$f_E(x) = \frac{K\left(\sqrt{1-(1/x^2)}\right)}{K(1/x)} \tag{6}$$

and $K(*)$ is the complete elliptic integral of the first kind defined as

$$K(k) \triangleq \int_0^{\pi/2} \frac{d\lambda}{\sqrt{1-k^2\sin^2\lambda}}. \tag{7}$$

Manuscript received September 22, 1989. This paper was recommended by Editor R. Liu.
The author is with the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.
IEEE Log Number 9037130.

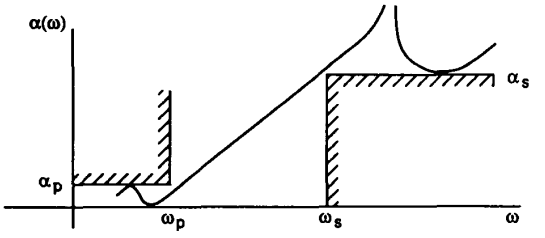

Fig. 1. Attenuation specification of a low-pass filter.

Although the present day scientific calculators can handle the evaluation of (3) and (4) with a few key strokes, none has a built-in function for (7) which is needed in (5) for n_E. One has to resort to a filter table (see, e.g., [3]–[5]), a nomograph (see, e.g., [6], [7]), or a computer program (see, e.g., [8], [9]) to determine n_E. It will be a great convenience if the $f_E(x)$ versus x curve is available, from which we can quickly investigate the various tradeoffs among the parameters $(\alpha_p, \alpha_s, \omega_p, \omega_s, n_E)$. The purpose of the present note is to fill this void by providing a sufficiently accurate plot of the $f_E(x)$ versus x curve, shown in Fig. 2, and illustrate its applications.

II. An Example of Application

The following example illustrates the use of $f_E(x)$ versus x curve.

Example: Given $\alpha_p = 1$ dB, $\alpha_s = 25$ dB, $\omega_p = 1000$ r/s, and $\omega_s = 1100$ r/s, find the lowest orders required of Butterworth, Chebyshev, and elliptic filters.

Solution:

$$\Omega = \omega_s/\omega_p = 1.1$$
$$M = \sqrt{\frac{10^{0.1\times 25}-1}{10^{0.1\times 1}-1}} = 34.89.$$

For n_B and n_C, we use a calculator to evaluate (3) and (4). The results are: $n_B \geqslant 3.552/0.095 = 37.39$ and $n_C \geqslant 4.245/0.444 = 9.56$. For n_E, we read from Fig. 2(a) to get $f_E(35) \cong 3.2$, and from Fig. 2(b) to get $f_E(1.1) \cong 0.71$. From (5), we have $n_E \geqslant 3.2/0.71 \cong 4.50$. Since the orders of transfer functions must be integers, we use $n_B = 38$, $n_C = 10$, and $n_E = 5$. The advantage of the elliptic filter is obvious in this case.

Since we decide to use $n_E = 5$, which is greater than the value 4.5 as calculated from (5), we can exceed the specifications in different ways. For example, we may retain the same values for $(\alpha_s, \omega_p, \omega_s)$ but reduce the passband ripple. The new α_p is then determined as follows:

$$f_E(M) = n_E f_E(\Omega) = 5 \times f_E(1.1) = 5 \times 0.71 = 3.55$$
$$M \cong 65 \text{ (from Fig. 2(a))}$$
$$\alpha_p = 0.32 \text{ dB (from (2))}.$$

In a similar manner of using Fig. 2, we can determine other possibilities of utilizing the margin gained from using $n_E = 5$ instead 4.5. Some of the possible trade-offs are given below:

n_E	α_p	α_s	$\Omega = \omega_s/\omega_p$
5	0.32	25	1.1
5	1	30.4	1.1
5	1	25	1.06

The choice of a particular tradeoff is governed by other factors. For example, it is known that if α_p is reduced, then the phase