Learning Capability Assessment and Feature Space Optimization for Higher-Order Neural Networks

Leda Villalobos and Francis L. Merat

Abstract—**A technique for evaluating the learning capability and optimizing the feature space of a class of higher-order neural networks is presented. It is shown that supervised learning can be posed as an optimization problem in which inequality constraints are used to code the information contained in the training patterns and to specify the degree of accuracy expected from the neural network. The approach establishes: (a) whether the structure of the network can effectively learn the training patterns and, if it can, a connectivity which corresponds to satisfactorily learning; (b) those features which can be suppressed from the definition of the feature space without deteriorating performance; and (c) if the structure is not appropriate for learning the training patterns, the minimum set of patterns which cannot be learned. The technique is tested with two examples and results are discussed.**

I. Introduction

When an artificial neural net (ANN) is trained with information whose complexity exceeds its learning capabilities, the training session finishes in failure. The reason for this is that no ANN can learn what it is not capable of doing in principle. What an ANN can learn depends upon its structure and the properties of its component elements [17]. Since the process of training is usually very time consuming, it is highly desirable to have a method which establishes whether the ANN is capable of learning the information contained in the training patterns. If learning is infeasible, it is also desirable to have guidance on how to modify either the ANN's structure or the feature space in which the problem is being defined. The use of a procedure with these capabilities would save the time spent in unsuccessful training sessions and expedite the implementation of computationally efficient systems for practical applications.

The usefulness of mechanisms suitable for efficient network complexity reduction and evaluation of learning capacity has been recognized since the early days of the Perceptron [1]. The initial work was primarily concerned with evaluating the linear separability of the training patterns [10] and with finding a connectivity that would produce the correct classification of as many patterns as possible [5], [20]. More recently, Sietsma and Dow [19] have proposed a pruning technique based on the suppression of connectivity which remains unchanged with every epoch. This is done by analyzing every neuron under the presentation of all the training patterns. Karnin [7] has introduced an algorithm which examines the sensitivity of the error function to the inclusion/exclusion of every connection in the ANN. The suppression of the synapses with the lowest sensitivities diminishes the performance of the trained ANN the least. Other approaches for pruning single-layer ANN's recourse to the application of pseudoinverses [21].

This paper introduces another approach to address the problem of efficient ANN structure design and feature space reduction. The approach establishes whether the structure of the ANN can effectively learn the training patterns and, if it can, a connectivity which corresponds to satisfactorily learning. Furthermore, it identifies those features which can be suppressed from the feature space without deteriorating performance. If the structure of the ANN is not appropriate for learning the training patterns, the approach indicates the minimum set of training patterns which cannot be learned.

In Section II, supervised learning is posed as an optimization problem in which inequality constraints capture both the information contained in the training patterns and the degree of accuracy expected from the ANN. In Section III, a linear programming formulation for solving the optimization problem is introduced in the context of higher-order ANN's. Tests and results are presented in Section IV, with conclusions in Section V.

II. Successful Learning as Satisfaction of Constraints

The goal of supervised training is to synthesize a connectivity which enables the ANN to perform satisfactorily when it is consulted with new patterns. This concept discovery process requires sufficient information to be included in the training patterns and that the learning error be small. Ideally, perfect learning would reduce the training error to zero. In practice, training is considered to be successful when the error has been reduced below a small, nonzero value because some degree of discrepancy between the desired and actual outputs is tolerated. This characteristic is exploited in this paper to show that supervised learning can be posed as an optimization problem in which constraints encode both the information contained in the training patterns and the degree of accuracy expected from the ANN.

A. Error Tolerance and Successful Learning

Consider first how learning is assessed in the Delta Rule or the Generalized Delta Rule paradigms. When an ANN is trained, its performance is usually assessed through the computation of an error function given by

$$E = \frac{1}{2PM} \sum_{p=1}^{P} \sum_{m=1}^{M} (T_{pm} - O_{pm})^2 \qquad (1)$$

Manuscript received December 14, 1992; revised April 30, 1993.

The authors are with the Electrical Engineering Department, Center for Automation and Intelligent Systems Research, Case Western Reserve University, Cleveland, OH 44106-7221 USA.

IEEE Log Number 9210736.

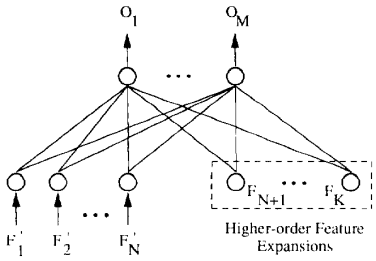

Fig. 1. Higher-order ANN architecture. The most important characteristic of a higher-order ANN is the nonlinear expansion introduced in the feature space. In the figure, the original feature vector has N components; through appropriate nonlinear transformations of these components, higher-order terms are generated and added to the original feature vector effectively expanding it.

where P is the number of training patterns, M is the number of outputs, T_{pm} is the desired mth output associated with the pth training pattern, and O_{pm} is the actual mth output obtained for the pth training pattern.

Training is considered successful when E has been reduced below a small, nonzero value. This implicit discrepancy tolerance acts as an inequality constraint for the ANN. To show the effect it has on the training process, consider the case of a particular training pattern with K features $(F_i; i = 1, 2, \cdots, K)$, and one desired output (T_1). This pattern can be represented by the vector

$$[F_1 \quad F_2 \quad \cdots \quad F_K \quad T_1]. \tag{2}$$

If the discrepancy tolerance for this pattern is given by an upper bound δ_+ and a lower bound δ_-, the ANN will be considered to have learned that pattern if the constraints

$$\text{Output} \geq (T_1 - \delta_-) \tag{3}$$

$$\text{Output} \leq (T_1 + \delta_+) \tag{4}$$

are satisfied. Since it is possible to specify tolerable levels of discrepancy for every pattern, this procedure can be extended to include all the patterns used to train the ANN. Therefore, learning can be posed as an inequality constraints satisfaction problem.

B. Constraints for Higher-Order ANN's

The work presented in this paper concentrates on higher-order ANN's of the type shown in Fig. 1. Multiple versions of these ANN's have been introduced in the literature for solving pattern recognition problems [12], including power system security assessment [14], [21]; adaptive control and plant identification [22]; rotation and translation invariant character recognition [15]; and problems of academic interest [3], [4], [9]. Work showing that these ANN's can be considered a class of universal approximators [6] has also been reported [13]. Although centered on higher-order ANN's, the analysis is extendible to other architectures.

In the most general sense, the mapping from feature space to output space synthesized by an ANN like the one shown in Fig. 1, can be expressed as

$$O_m = g(\text{net}_m) \tag{5a}$$

$$\text{net}_m = \sum_{j=1}^{N} W_{jm} F'_j + \sum_{i=1}^{K-N} W_{im} h^i(\underline{F'}) + \theta_m \tag{5b}$$

where O_m is the mth output, $g(\cdot)$ is the output neuron's activation function, θ_m is the output neuron's threshold, W_{im} is the connection weight from the ith feature to the mth output neuron, F'_j is the jth element of the original N–D feature vector $\underline{F'}$, and $h^i(\cdot)$ represents the ith higher-order feature expansion function. The nonlinear functions $h^i(\cdot)$ could also be replaced, for example, by the output of additional neurons of an extra intermediate layer, the connections from the input layer to this layer being fixed.

To simplify the analysis, the original feature vector $\underline{F'}$ can be concatenated with the nonlinear, higher-order feature expansions, to form the K–D vector F. Hence, the mth output function is given by

$$O_m = g\left(\sum_{j=1}^{K} W_{jm} F_j + \theta_m\right). \tag{5c}$$

Note that this mapping function resembles one learned by a single-layer ANN with a K–D feature vector [23].

Consider the particular case of a higher-order ANN trained with P patterns of K features (including the higher-order feature expansions), and one output. Extension to include any number of outputs is immediate. The information contained in the patterns can be captured in a $P \times K$ matrix $(\mathbf{F})$ with the P expanded feature vectors, and a P–D vector $(\underline{T})$ with the corresponding P desired outputs. The ANN is capable of learning this information with the desired accuracy if and only if there exists at least one K–D vector $\underline{W}$, and a scalar θ, such that

$$\begin{bmatrix} \mathbf{F} & \underline{1} \\ \mathbf{F} & \underline{1} \end{bmatrix} \begin{bmatrix} \underline{W} \\ \theta \end{bmatrix} = \begin{bmatrix} \leq g^{-1}(\underline{T} + \underline{\delta}_+) \\ \geq g^{-1}(\underline{T} + \underline{\delta}_-) \end{bmatrix} \tag{6}$$

where $\underline{\delta}_+$ and $\underline{\delta}_-$ are the upper and lower bound tolerance P–D vectors respectively, $g^{-1}(\cdot)$ is the output neuron's activation function inverse, θ is the output neuron's threshold, $\underline{W}$ is the K–D weights vector, and $\underline{1}$ is a P–D vector with all its entries equal to 1.

According to (6), the learning process should lead to the satisfaction of a set of linear inequality constraints. Should the set of solutions be non-empty, there will be at least one connectivity for the ANN which solves the learning task with the desired performance accuracy.

III. Linear Programming Formulation

A system of linear inequality constraints can be solved through the application of linear programming (LP) techniques [2], [11]. Algorithms which solve LP problems operate in two steps. In the first step, it is determined whether the constraints have a non-empty set of feasible solutions; if so, an iterative search for an optimum solution in the set of feasible solutions is carried out in the second step. The search is

guided by an objective function which consists of minimizing or maximizing a linear function of the variables included in the constraints.

A. Establishing Task Learnability

Algorithms for solving LP problems require that all variables be restricted to take only nonnegative values. A variable which has no sign restrictions must be expressed as the difference of two nonnegative variables [11]. Taking this into account, the constraints of (6) can be captured in the following LP formulation

$$\begin{bmatrix} \mathbf{F} & \underline{1} \\ \mathbf{F} & \underline{1} \end{bmatrix} \begin{bmatrix} \underline{W}_A - \underline{W}_B \\ \theta_A - \theta_B \end{bmatrix} = \begin{bmatrix} \leq g^{-1}(\underline{T} + \underline{\delta}_+) \\ \geq g^{-1}(\underline{T} - \underline{\delta}_-) \end{bmatrix} \tag{7}$$

where $\underline{W}_A, \underline{W}_B$ are K–D vectors of nonnegative variables such that $\underline{W}_A - \underline{W}_B = \underline{W}$, and θ_A, θ_B are nonnegative variables such that $\theta_A - \theta_B = \theta$.

The formulation of (7) is appropriate to find out whether the training patterns can be learned with the desired level of accuracy. If they can, the solution will be a connectivity which satisfies all the constraints; otherwise, it will be concluded that no feasible solution exists.

B. Identifying Non-Learnable Patterns

The optimization criterion can be used to identify those patterns the ANN is capable of learning. With the formulation of (7), the existence of any non-learnable pattern will cause the algorithm used to solve the problem to stop before a search for the optimal solution is initiated. Hence, this formulation must be modified to obtain the identity of the non-learnable patterns. The modification should be such that the constraints are always found to have a non-empty set of solutions. This can be done by providing the constraints with a default feasible solution by introducing a pad variable for each training pattern, as shown in (8)

$$\begin{bmatrix} \mathbf{F} & \underline{1} & \mathbf{I} \\ \mathbf{F} & \underline{1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \underline{W}_A & - & \underline{W}_B \\ \theta_A & - & \theta_B \\ \underline{S}_A & - & \underline{S}_B \end{bmatrix} = \begin{bmatrix} \leq g^{-1}(\underline{T} + \underline{\delta}_+) \\ \geq g^{-1}(\underline{T} - \underline{\delta}_-) \end{bmatrix} \tag{8}$$

where I is the $P \times P$ identity matrix, and S_{Ai}, S_{Bi} are nonnegative variables such that $S_i = S_{Ai} - S_{Bi}$, with S_i the ith pad variable.

The default solution is obtained by making $\underline{W}$ and θ equal to zero, and assigning each pad variable a value which falls in the range corresponding to the satisfactory learning of its associated training pattern. The patterns whose associated pad variables are different from zero in the optimum solution form the smallest set of patterns that preclude learning. Therefore, the objective function should minimize the number of nonzero pad variables. This can be stated as

$$\text{Objective Function} = \min \sum_{i=1}^{P} h(S_i) \tag{9}$$

with

$$h(x) = \begin{cases} 0; & \text{if } |x| = 0 \\ c; & \text{otherwise} \end{cases} \tag{10}$$

where c is any positive real number.

The function $h(x)$ is nonlinear but can be implemented by introducing integer 0/1 variables (variables allowed to take only the values 0 and 1) and including an extra constraint for each pad variable S_i, as indicated in (11)

$$|S_i| - L_i H_i \leq 0 \tag{11}$$

where L_i is a sufficiently large upper bound for $|S_i|$, and H_i is the integer 0/1 variable used to test whether S_i is different from zero.

To satisfy (11), H_i is forced to take the value 1 whenever $|S_i| \neq 0$. If $|S_i| = 0$, H_i can be either 0 or 1. This ambiguity disappears once the optimization criterion is written in terms of the integer variables

$$\text{Objective Function} = \text{Min} \sum_{i=1}^{P} H_i. \tag{12}$$

Since the optimization criterion consists of minimizing the sum of the integer variables, H_i will always take the value 0 whenever $|S_i| = 0$. Thus, no ambiguity exists and H_i behaves according to

$$H_i = \begin{cases} 0; & \text{if } |S_i| = 0 \\ 1; & \text{otherwise} \end{cases} \tag{13}$$

In the solution of an LP problem, whenever a variable without sign restrictions is expressed as the difference of two nonnegative variables, at least one of the nonnegative variables is always zero [11]. Thus, the absolute value of S_i can be expressed as $|S_i| = S_{Ai} + S_{Bi}$. Taking this into account (13) becomes

$$H_i = \begin{cases} 0; & \text{if } S_{Ai} + S_{Bi} = 0 \\ 1; & \text{otherwise} \end{cases} \tag{14}$$

Hence, an LP formulation whose solution indicates which patterns cannot be learned is given by

$$\text{Objective Function} = \text{Min} \sum_{i=1}^{P} H_i$$

subject to

$$\begin{bmatrix} \mathbf{F} & \underline{1} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{F} & \underline{1} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \underline{0} & \mathbf{0} & \mathbf{I} & -\mathbf{L} \end{bmatrix} \times \begin{bmatrix} \underline{W}_A & - & \underline{W}_B \\ \theta_A & - & \theta_B \\ \underline{S}_A & - & \underline{S}_B \\ \underline{S}_A & + & \underline{S}_B \\ & \underline{H} & \end{bmatrix} = \begin{bmatrix} \leq g^{-1} & (\underline{T} + \underline{\delta}_+) \\ \geq g^{-1} & (\underline{T} - \underline{\delta}_-) \\ & \leq \underline{0} \end{bmatrix} . \tag{15}$$

The solution of this LP formulation provides with the following information:

- It indicates whether the ANN is capable of learning the training patterns with the desired level of accuracy. If all the integer variables in the LP solution are equal to zero then from (13) all the pad variables S_i will also be zero. Consequently, (7) will be satisfied and, for every pattern, the actual output will fall in the tolerance range associated with the pattern.

- If the structure is appropriate for learning the training patterns, the solution of the constraint satisfaction problem validates a given connectivity which corresponds to satisfactory learning. Since (7) is satisfied, it is guaranteed the connectivity defined by W and θ is such that the output for every training pattern falls within the range of tolerance.
- If the ANN is not capable of learning all the patterns, one or more of the integer variables will remain nonzero. From (13), the pad variables corresponding to the nonzero integer variables will be also different from zero. Therefore, from (8), the patterns associated with the nonzero pad variables will be incompatible with the ANN structure. They form the smallest set of patterns which cannot be learned with the desired level of accuracy.

C. Optimizing Feature Space

Once it has been established that a particular ANN structure is appropriate for learning the information contained in the training patterns, the next step is to identify those features, if any, which can be eliminated from the feature space without diminishing the ANN's performance.

Suppose the jth feature can be eliminated; then, the jth connection weight W_j can be made equal to zero, which means that

$$|W_j| = W_{Aj} + W_{Bj} = 0. \tag{16}$$

Integer variables Q_j can be introduced to test whether W_j can be made equal to zero with a procedure similar to the one used to test pad-variables. The objective function in this case is the minimization of the number of connection weights remaining different from zero in the optimum solution. This leads to the formulation

$$\text{Objective Function} = \text{Min} \sum_{j=1}^{K} Q_j$$

subject to:

$$\begin{bmatrix} \mathbf{F} & \underline{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{F} & \underline{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \underline{0} & \mathbf{I} & -\mathbf{L} \end{bmatrix} \begin{bmatrix} \underline{W}_A & - & \underline{W}_B \\ \theta_A & - & \theta_B \\ \underline{W}_A & + & \underline{W}_B \\ & \underline{Q} & \end{bmatrix} = \begin{bmatrix} \leq g^{-1} & (\underline{T} + \underline{\delta}_+) \\ \geq g^{-1} & (\underline{T} - \underline{\delta}_-) \\ & \leq \underline{0} \end{bmatrix} \tag{17}$$

where Q and L are K–D vectors of integer variables and constant values, respectively.

D. Remark

Note that this technique does not work for perceptrons with binary outputs, since there $g^{-1}(\cdot)$ would not be defined. For optimal learning in such systems, methods from quadratic programming have already been reported in the literature [16].

TABLE I
PATTERNS FOR THE PARITY-3 PROBLEM

Pattern Number	Features X1	X2	X3	Desired Output
1	0	0	0	0.1
2	0	0	1	0.9
3	0	1	0	0.9
4	0	1	1	0.1
5	1	0	0	0.9
6	1	0	1	0.1
7	1	1	0	0.1
8	1	1	1	0.9

TABLE II
SUMMARY LP SOLUTION FOR PARITY-3 PROBLEM WITHOUTFEATURE EXPANSION

Variable	Value	Variable	Value	Variable	Value	Variable	Value
H_1	0	H_6	0	S_3	0.0	S_8	0.0
H_2	0	H_7	0	S_4	-2.1972	W_1	4.3944
H_3	0	H_8	0	S_5	-2.1972	W_2	4.3944
H_4	1	S_1	0.0	S_6	0.0	W_3	4.3944
H_5	1	S_2	0.0	S_7	0.0	θ	-2.1974

IV. EXAMPLES

The approach has been tested with the LINDO (Linear Interactive aNd Discrete Optimizer) programming package [18]. This program uses the Simplex Method Algorithm and allows the simultaneous introduction of both real and integer variables into the LP formulation. Two representative examples are presented in this section.

Example 1: This example corresponds to the well-known Parity-3 Problem. Its purpose is twofold: on the one hand, the simplicity of the task facilitates presenting the optimization algorithm in operation; on the other hand, a closed-form solution for the Parity-3 problem with a higher-order ANN has been reported in the literature, which makes comparisons between the algorithm and the results reported by other researchers readily possible.

The three input features are denoted as $X1$, $X2$, and $X3$. The desired outputs were 0.1 for patterns with even parity and 0.9 for patterns with odd parity. Only one constraint was imposed upon each pattern: patterns with desired output 0.9 had no upper bound constraint, while patterns with desired output 0.1 had no lower bound constraint. The upper and lower levels of tolerance were set to 0.0 for all patterns. The activation function was the logistic function

$$g(x) = \frac{1}{1 + \exp(-x)}. \tag{18}$$

The training pattern information is shown in Table I.

The algorithm was first tested with no higher-order expansion. The solution of the LP problem is summarized in Table II. It indicates training would be unsuccessful, a minimum set of incompatible patterns being patterns 4 and 5. The LP solution also indicates the following connectivity produces an optimum solution: $W1 = W2 = W3 = 4.3944$, and $\theta = -2.19\,72$. These results agree with those reported in the literature [12], [21].

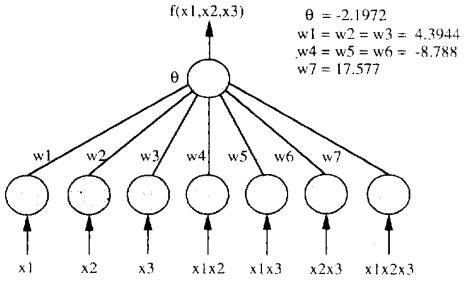

Fig. 2. Neural network structure for solving the Parity-3 problem.

TABLE III
TRAINING PATTERNS FOR FUNCTION $F(x)$

Pattern Number	Feature X	Desired Output F(x)	Pattern Number	Feature X	Desired Output F(x)
1	0.00	0.80	10	0.45	0.32
2	0.05	0.72	11	0.50	0.41
3	0.10	0.62	12	0.55	0.51
4	0.15	0.52	13	0.60	0.58
5	0.20	0.41	14	0.65	0.60
6	0.25	0.32	15	0.70	0.56
7	0.30	0.25	16	0.75	0.51
8	0.35	0.22	17	0.80	0.50
9	0.40	0.25	18	0.85	0.56

The complete third-order outer product expansion having the features $X1, X2, X3, X1X2, X1X3, X2X3$, and $X1X2X3$ was also tested. Sobajic [21] reported this level of expansion is sufficient to obtain appropriate learning. For this expansion, the connectivity synthesized from the solution of the LP problem defines the ANN shown in Fig. 2. This connectivity is exactly as that one given by Sobajic's closed-form solution [21].

Example 2: This example was taken from Klassen *et al.* [8]. Its main purpose is to examine the ability of the optimization technique at identifying unnecessary features in the feature space. The task consists of learning a representation for an unknown, one-variable function with the only available information being the sample patterns listed in Table III.

Klassen *et al.* [8] have reported enhancements of the functional expansion type with the features $\sin(\pi x), \cos(\pi x), \sin(2\pi x), \cos(2\pi x)$, and $\sin(4\pi x)$ are sufficient to train an ANN to solve this task. To test the optimization algorithm, an LP formulation for this enhancement was developed according to (15). Upper and lower discrepancies of 0.015 were allowed for all patterns, which corresponds to specifying an acceptable error E=1.125E-4. The activation function was the logistic function given in (18). The LP solution showed that the expansion is indeed sufficient for satisfactorily training the ANN.

In another test, the formulation was modified according to (17), keeping the same tolerance levels as before. The results showed that the feature $\cos(\pi x)$ is not necessary to achieve the desired performance and, therefore, it can be eliminated from feature space. This later result was confirmed by training an ANN with the reduced set of features. The mapping function obtained with the LP solution appears in Fig. 3, and the corresponding ANN is shown in Fig. 4.

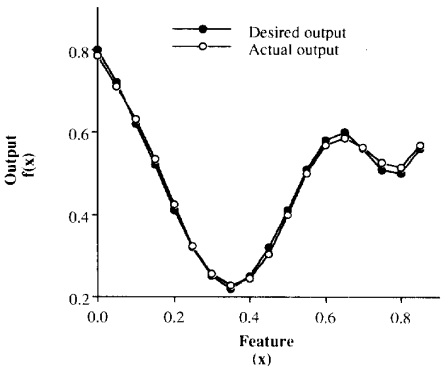

Fig. 3. Training pairs and outputs estimated using the LP solution.

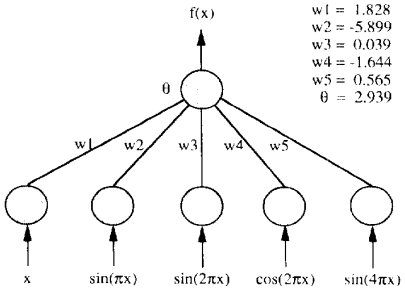

Fig. 4. Reduced ANN architecture obtained from the LP solution.

These results show the specification of an adequate objective function leads to identifying the input vector terms necessary and sufficient to produce an ANN which solves the task with the desired level of accuracy.

V. CONCLUSION

A simple technique for assessing the learning capability of an ANN and optimizing the feature space has been presented. Although the technique is formulated based on the architecture of a higher-order single-layer ANN, it is extendible to include other architectures. It takes advantage of the implicit accuracy tolerance associated with every training pattern in supervised learning. This tolerance is used to show that supervised learning can be posed as an optimization problem in which inequality constraints capture the information contained in the training patterns and the degree of accuracy required from the ANN. The solution of the appropriately defined optimization problem indicates whether the structure of the ANN can effectively learn the patterns, and if so, it gives a connectivity which corresponds to satisfactorily learning; it also indicates which features, if any, can be suppressed from feature space without deteriorating performance. If the structure is not appropriate for learning the task, the solution identifies the minimum set of training patterns precluding learning.

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers who provided with many valuable suggestions.

REFERENCES

[1] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[2] S. I. Gass, *Linear Programming: Methods and Applications*. New York: McGraw-Hill, 1985.

[3] C. L. Giles and T. Maxwell, "Learning, invariance and generalization in higher-order neural networks," *Applied Optics*, vol. 26, pp. 4972–4978, 1987.

[4] C. L. Giles, C. B. Miller, and D. Chen, "Learning and extracting finite state automata with second-order recurrent neural networks," *Neural Computation*, vol. 4, p. 393, 1992.

[5] R. G. Grinold, "Comment on pattern classification design by linear programming," *IEEE Trans. Computers*, vol. C-18, pp. 378–379, 1969.

[6] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[7] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 239–242, 1990.

[8] M. Klassen, Y. H. Pao, and V. Chen, "Characteristics of the functional link net: a higher order delta rule net," in *Proc. IEEE 2nd Annual Int. Conf. Neural Networks*, San Diego, CA, 1988.

[9] Y. C. Lee, G. Doolen, H. H. Chen, G. Z. Sun, T. Maxwell, H. Y. Lee, and C. L. Giles, "Machine learning using a higher order correlation network," *Physics 22D*, pp. 276–306, 1986.

[10] O. L. Mangasarian, "Linear and nonlinear separation of patterns by linear programming," *Op. Res.*, vol. 13, pp. 444–452, 1965.

[11] K. G. Murty, *Linear Programming*. New York: Wiley, 1983.

[12] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley, 1989.

[13] Y. H. Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture and functionalities," *IEEE Computer*, to appear.

[14] Y. H. Pao and D. J. Sobajic, "Combined use of unsupervised and supervised learning for dynamic security assessment," *IEEE Trans. Power Syst.*, vol. 7, p. 878, 1992.

[15] S. J. Perantonis and P. J. G. Lisboa, "Translation, rotation and scale invariant pattern recognition by high-order neural networks and moment classifiers," *IEEE Trans. Neural Networks*, vol. 3, pp. 241–251, 1992.

[16] P. Rujan, "A fast method for calculating the perceptron with maximal stability," *J. de Physique I*, vol. 3, pp. 277–290, 1993.

[17] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, vol. 1, D. E. Rumelhart and J. L. McCleland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[18] L. Schrage, *Linear, Integer and Quadratic Programming with LINDO*. New York: Scientific, 1984.

[19] H. Sietsma and R. J. F. Dow, "Neural net pruning—Why and how?" in *Proc. IEEE Int. Conf. Neural Networks*, vol. 1, San Diego, CA, pp. 325–332, 1988.

[20] F. W. Smith, "Pattern classifier design by linear programming," *IEEE Trans. Computers*, vol. C-17, pp. 367–372, 1968.

[21] D. J. Sobajic, "Artificial neural networks for transient stability assessment of electric power systems," Center for Automation and Intelligent Systems Research, Case Western Reserve University, Tech. Rep. TR-88-127, 1988.

[22] D. J. Sobajic, Y. H. Pao, and D. T. Lee, "Autonomous adaptive synchronous machine control," *International Journal of Electrical Power and Energy*, vol. 14, p. 166, 1992.

[23] J. M. Zurada, *Introduction to Artificial Neural Systems*. St. Paul: West Publishing, 1992.