made the LBL algorithm update hidden weights proportional to the derivative of the output error function with respect to the associated hidden weights. Especially, it was proven that the LBL using the new hidden error approximated the standard EBP algorithm with optimal learning rates. The proposed LBL algorithm removed the stalling problem in the conventional LBL algorithms, and thereby greatly speeded up the learning convergence.

REFERENCES

[1] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing.* Cambridge, MA: MIT Press, 1986.

[2] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks,* vol. 1, pp. 295–307, 1988.

[3] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the back-propagation method," *Biol. Cybern.,* vol. 59, pp. 257–263, 1988.

[4] S.-H. Oh, "Improving the error backpropagation algorithm with a modified error function," *IEEE Trans. Neural Networks,* vol. 8, pp. 799–803, 1997.

[5] A. van Ooyen and B. Nienhuis, "Improving the convergence of the back-propagation algorithm," *Neural Networks,* vol. 5, pp. 465–471, 1992.

[6] R. Parisi, E. D. Di Claudio, G. Orlandi, and B. D. Rao, "A generalized learning paradigm exploiting the structure of feedforward neural networks," *IEEE Trans. Neural Networks,* vol. 7, pp. 1450–1459, 1996.

[7] G.-J. Wang and C.-C. Chen, "A fast multilayer neural networks training algorithm based on the layer-by-layer optimizing procedures," *IEEE Trans. Neural Networks,* vol. 7, pp. 768–775, 1996.

[8] F. Biegler-Konig and F. Marmann, "A learning algorithm for multilayered neural networks based on linear least squares problems," *Neural Networks,* vol. 6, pp. 127–131, 1993.

[9] J. Y. F. Yam and W. S. Chow, "Extended least squares based algorithm for training feedforward networks," *IEEE Trans. Neural Networks,* vol. 8, pp. 806–810, 1997.

[10] S. Ergezinger and E. Thomsen, "An accelerated learning algorithm for multilayer pereceptrons: Optimization layer by layer," *IEEE Trans. Neural Networks,* vol. 6, pp. 31–42, 1995.

[11] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 16, pp. 550–554, May 1994.

[12] D.-S. Kim, S.-Y. Lee, and R. M. Kil, "Auditory processing of speech signals for robust speech recognition in real-world noisy environments," *IEEE Trans. Speech Audio Processing,* vol. 7, pp. 55–69, 1999.

[13] Y. Lee, S.-H. Oh, and M. W. Kim, "An analysis of premature saturation in back-propagation learning," *Neural Networks,* vol. 6, pp. 719–728, 1993.

[14] S.-H. Oh and Y. Lee, "Effect of nonlinear functions on correlation between weighted sums in multilayer perceptrons," *IEEE Trans. Neural Networks,* vol. 5, pp. 508–510, 1994.

A Formal Selection and Pruning Algorithm for Feedforward Artificial Neural Network Optimization

P. V. S. Ponnapalli, K. C. Ho, and M. Thomson

***Abstract*— A formal selection and pruning technique based on the concept of local relative sensitivity index is proposed for feedforward artificial neural networks. The mechanism of backpropagation training algorithm is revisited and the theoretical foundation of the improved selection and pruning technique is presented. This technique is based on parallel pruning of weights which are relatively redundant in a subgroup of a feedforward neural network. Comparative studies with a similar technique proposed in the literature show that the improved technique provides better pruning results in terms of reduction of model residues, improvement of generalization capability and reduction of network complexity. The effectiveness of the improved technique is demonstrated in developing neural network (NN) models of a number of nonlinear systems including three bit parity problem, Van der Pol equation, a chemical processes and two nonlinear discrete-time systems using the backpropagation training algorithm with adaptive learning rate.**

***Index Terms*—Backpropagation, feedforward artificial neural network, optimization, pruning.**

I. INTRODUCTION

The development of a feedforward artificial neural network (FANN) model involves the selection of a number of parameters such as number of inputs and hidden units [1]. Once a satisfactory model has been obtained, the problem of optimizing the architecture arises. In this context, optimization is used to refer to reducing structural complexity by minimizing the number of weights in a FANN. Such optimization offers advantages in terms of simpler networks, faster training and better generalization ability to avoid overfitting due to oversized network [2]. Optimization of FANN structure includes the selection of appropriate inputs and outputs [3], [4], the determination of the number and type of nonlinear hidden neurons and the connections between each neuron inside the FANN. Sigmoidal threshold functions are most commonly used for the hidden layer neurons, other alternatives were studied by various authors [5], [6]. Linear threshold functions are used in the output layer commonly [1].

Various techniques such as optimal brain surgeon (OBS) [7] and optimal brain damage (OBD) [8] have been proposed in literature to prune a fully connected FANN. These techniques, however, require considerable additional computation as they require calculation of the Hessian matrix of the system. In this paper, the pruning method based on the "sensitivity term" proposed by Karnin [9] is reviewed and a more formal procedure for the selection and pruning of weights is proposed. The main attraction of Karnin's technique is that it uses normally available information during a backpropagation (BP) algorithm and hence requires moderate additional computations. The pruning technique is applied to the modeling of a number of nonlinear systems including three bit parity problem, Van der Pol equation and a chemical process (continuous stirred tank reactor) using the BP training algorithm with adaptive learning rate. The results are compared with those obtained using the pruning algorithm proposed by Karnin [9]. An $(l_0 : l_1 : l_2)$ architecture is considered where l_0, l_1 and l_2 correspond to the number of input, hidden, and output

Manuscript received October 30, 1998; revised March 18, 1999.

The authors are with the Department of Electrical and Electronic Engineering, Manchester Metropolitan University, Manchester, M1 5GD, U.K.

Publisher Item Identifier S 1045-9227(99)05483-1.

neurons. It has been shown that a two layered FANN (one hidden layer with nonlinear sigmoidal neurons and one output layer with linear neuron) can uniformly approximate any continuous function to an arbitrary degree of exactness provided the hidden layer contains sufficient number of nodes [10]. Hence only a two-layered network is considered in this work.

II. Motivation

The basis of Karnin's pruning algorithm is to estimate the sensitivity of the global error changes with respect to each individual weight during the training process. The connections which have the smallest sensitivity value (SV) will be pruned and the FANN will be retrained. The SV's are then recalculated and the weight connections corresponding to the lowest SV will be pruned again. The whole process of "training—SV calculation—pruning" will be continued until no further improvement in the FANN model is noticed, i.e., no further decrease of global sum squared error (SSE). However, the two examples presented in [9] required only simple FANN architectures. In addition, *a priori* knowledge was used to prune entire hidden nodes which translates to pruning whole sets of weights connected to those nodes. Pruning based on sensitivity values was only applied after this, and even that stage involved the use of *a priori* knowledge as opposed to following straight forward comparison of SV's as outlined in the paper. Further, if the pruning method were to be applied to a complex NN model having more than 20 or 30 weight connections, it is very likely that several weight connections will have similar low SV's. Without a formal procedure, it will be difficult to decide which or how many weight connections should be pruned at any one time. This formal procedure is crucial especially when no or very little *a priori* knowledge about the system is available. In a large FANN model, it is also possible that weights with low SV's might be dispersed over different layers or concentrated in a small group of neurons. Proper investigation of the local and global effect of pruning is required so as to assure that pruning will not end up with a suboptimal FANN structure. Pruning based solely on global SV's may not yield the most efficient structure as it may result in pruning an entire node whose associated SV's may all be small on global comparison but may actually represent an interaction in a dynamic system that plays an important role.

In this work, a formal selection and pruning technique based on an extension to the SV's is developed to address the above issues. The mechanism of BP training algorithm is investigated to highlight the relative importance of the weight connections. The concepts of hidden neuron to input layer path and output neuron to hidden layer path are introduced to examine the effects of changes in the sets of weights connected to specific nodes in the hidden and output layers. A parameter called local relative sensitivity index (LRSI) is defined and a selection rule is presented to determine which weight connections must be pruned using the LRSI's, and a formal procedure is established for the entire pruning process. The main advantages of this method are that: 1) it offers a systematic approach to pruning FANNs; 2) allows the designer to set threshold values for pruning at a local level; and more importantly 3) prevents pruning of important connections which may be usually pruned if the global pruning method of Karnin is used.

III. Theoretical Foundation

As mentioned in section one, the present studies are based on a two layer FANN with nonlinear sigmoidal threshold function in the hidden layer neurons and linear threshold function output. The connections between the input and the hidden layer are represented by "hidden layer weights" and the connections between the hidden layer and the output layer are represented by "Output Layer Weights."

A. Mechanism of Backpropagation (BP) Training

We first present the basic equations of the BP training and then discuss the effect of hidden nodes on weight updates.

Notation:

l_i	Number of neurons in the ith layer ($i = 0, 1, 2$ for the input, hidden, and output layers, respectively).
$x_{(l_0,m)}$	Input to the FANN of dimension (l_0, m) where l_0 is the number of input variables (number of neurons in the input layer) and m is the number of data sets.
o_{ij}	Output of the jth node in the ith layer; and $o_i = [o_{i1} o_{i2} \dots o_{il}]^T$.
net_{ij}	Sum of the inputs to the jth node in the ith layer, and $\mathrm{net}_i = [\mathrm{net}_{i1} \mathrm{net}_{i2} \dots \mathrm{net}_{il}]^T$.
$\mathbf{b}_i$	Bias vector of the ith layer; $\mathbf{b}_i = [b_{i1} b_{i2} \dots b_{il}]^T$ where b_{ij} = bias of node j in the ith layer.
$f_{ij}(\cdot)$	Activation function of the jth node in the ith layer.
$f_i(\cdot)$	Activation function of the ith layer when all the nodes have the same activation function.
t_j	Target output of the jth node in the output layer.
$e_j = t_j - o_{ij}$	Error between the target and actual outputs of the jth node in the ith layer.
w_{ijk}	weight connecting the kth node in the ith layer and the jth node in the $(i-1)$th layer.
w_i	Matrix of the weights connecting all the nodes of the ith and $(i-1)$th layers.

Without loss of generality, we consider a FANN with two layers, one hidden and one output layer. The output of the hidden layer can be written as

$$o_1 = f_1(\mathrm{net}_1 + b_1) = f_1(w_1 x + b_1)$$

and the output of the FANN is given by

$$o_2 = f_2(\mathrm{net}_2 + b_2) = f_2(w_2 o_1 + b_2) = f_2(w_2 f_1(w_1 x + b_1) + b_2).$$

The sum squared error (SSE) in any one iteration is given by $\mathrm{SSE} = \frac{1}{2}\sum_{j=1}^{l_2} e_j^2$ where l_2 is total number of output nodes. The rate of change of the SSE with respect to the weight connecting hidden node k and output node j, is

$$\frac{\partial \mathrm{SSE}}{\partial w_{2jk}} = \frac{\partial \mathrm{SSE}}{\partial e_j}\frac{\partial e_j}{\partial O_{2j}}\frac{\partial O_{2j}}{\partial \mathrm{net}_{2j}}\frac{\partial \mathrm{net}_{2j}}{\partial w_{2jk}}$$
$$= -e_j o_{1k} f'_{2j}(\mathrm{net}_{2j} + b_{2j}), \qquad \text{where} f'_{2j} = \frac{\partial f_{2j}}{\partial \mathrm{net}_{2j}}.$$

Therefore, by gradient descent rule, the update for a weight in the second layer is given by

$$\Delta w_{2jk} = \eta e_j o_{1k} f'_{2j}(\mathrm{net}_{2j} + b_{2j}) = \eta e_j o_{1k} \tag{1}$$

where η is the learning rate and $f'_{2j}(\mathrm{net}_{2j} + b_{2j})$ is always equal to unity as a linear activation function is used for all the output nodes.

Similarly, the rate of change of SSE with respect to the weight connecting input node k and hidden node j is

$$\frac{\partial \mathrm{SSE}}{\partial w_{1jk}} = \frac{\partial \mathrm{SSE}}{\partial e_j}\frac{\partial e_j}{\partial O_{1j}}\frac{\partial O_{1j}}{\partial \mathrm{net}_{1j}}\frac{\partial \mathrm{net}_{1j}}{\partial w_{1jk}} = -x_k f'_{1j}(\mathrm{net}_{1j} + b_{1j})$$
$$\times \left\{\sum_{p=1}^{l_2}[e_p w_{2pj} f'_{2p}(\mathrm{net}_{2p} + b_{2p})]\right\}$$

where p refers to the pth output node and $f'_{1j} = \frac{\partial f_{1j}}{\partial \mathrm{net}_{1j}}$.

Using appropriate substitutions, the update for a weight in the hidden layer is given by

$$\Delta w_{1jk} = \eta x_k f'_{1j}(\text{net}_{1j} + b_{1j}) \left\{ \sum_{p=1}^{l_2} (e_p w_{2pj}) \right\} \tag{2}$$

B. Effect of Hidden Nodes on Weight Change

Here we rewrite the weight update equations and show that it is more important to consider local effects while pruning weights in stead of global effects only. Equations (1) and (2) above can be rewritten as follows:

$$\Delta w_{2jk} = C_2 o_{1k} \tag{3}$$

$$\Delta w_{1jk} = C_1 f'_{1j}(\text{net}_{1j} + b_{1j}) \quad \text{where}$$

$$C_1 = \eta x_k \left\{ \sum_{p=1}^{l_2} (e_p w_{2pj}) \right\} \quad \text{and} \quad C_2 = \eta e_j. \tag{4}$$

Considering the weight update for the output layer (3), in any one iteration of the BP phase, C_2 is calculated in the first step at the output layer and this value is used to update all the weights connecting the hidden layer to the output layer. Hence C_2 can be viewed as a constant while investigating the changes of these weights. Similarly for the updating of the weights connecting the input layer to hidden layer (4), C_1 is calculated prior to updating the weights and this value is used in all the calculations. Hence C_1 can also be treated as constant while updating the hidden layer weights.

In each training iteration, for a hidden layer with l_1 neurons, the changes affected in the l_1 sets of weights depend only on the output of the respective hidden neurons to which each set is connected. It is therefore useful to consider the relative importance of a weight in its own set. To facilitate this, each of these sets are described using a hidden neuron to input layer path (HIP) as a subset of the total hidden layer weights. Using similar reasoning, each of the sets of weights in the output layer weights will be described using an output neuron to hidden layer path (OHP) as a subset of the total output layer weights which will consist of l_2 sets where l_2 is the number of output neurons.

Considering the fact that C_1 and C_2 can be viewed as constants, the changes in the connection weights in both the hidden and output layers depend only on the output of one node to which each subset (HIP or OHP) is connected. Karnin's method of pruning [8] does not consider this important aspect and the comparison of sensitivity values is done only on a global basis. Such a global comparison will not consider the location, the relative importance of a weight or the effect of pruning a weight on the training profile of the corresponding HIP or OHP. However, in many applications, interactions may be characterized by a node whose connection weights may be quite small compared to the rest. Hence a more logical pruning would be to compare the weights within each subset so that whole nodes will not disappear unless they are redundant. It is therefore decided to rearrange the set of SV's into groups corresponding to the HIP or OHP they belong to, and comparison of SV's is then performed among the members of each set.

C. Local Relative Sensitivity Index (LRSI)

We introduce an index called local relative sensitivity index (LRSI) which can reflect the relative importance of SV's within the same HIP or OHP. It has the term "local" because the SV's are compared locally in each HIP or OHP. The LRSI of each connection in the specific HIP or OHP is defined as the ratio between the absolute value of each SV to the sum of the absolute values of all SV's in the HIP or OHP. The LRSI of the connection weight between jth and kth node is defined as

$$\text{LRSI}_{jk} = \frac{|\text{SV}_{jk}|}{\sum_{m=1}^{n} |\text{SV}_{jm}|} \tag{5}$$

where n is the number of weight connections in that specific HIP or OHP; SV_{jk} is the sensitivity value of weight connecting the jth and kth node.

By definition an LRSI can only have a value between zero and one. A cut-off value β can now be specified by the designer to reflect the confidence in the model and can be used as a basis for pruning. For example, if an LRSI_{jk} is smaller than certain value say 0.01, then the weight w_{jk} is considered to be less important or redundant and can be pruned. Since the comparison of LRSI's is done in each HIP and OHP, the relative importance of the weight will be considered locally before pruning and at most one connection from each HIP or OHP will be pruned at a time.

D. Pruning and Selection Scheme (PSS)

The pruning and selection scheme based on LRSI's involves eight steps.

1) Separate the data into a training set and a validation set.
2) Build a fully connected FANN model of the system.
3) Calculate the sensitivity values (SV's) corresponding to each weight connection based on the same equation as Karnin [9]

$$\text{SV}_{jk} = \sum_{n=0}^{N-1} \left[\frac{\partial \text{SSE}}{\partial w_{jk}}(n) \Delta w_{jk}(n) \right] \times \frac{w_{jk}(\text{final})}{(w_{jk}(\text{final}) w_{jk} - (\text{initial}))} \tag{6}$$

where SV_{jk} = Sensitivity value of the weight w_{jk}; $w_{jk}(\text{final})$ = final value of w_{jk}; w_{jk} (initial) = initial value of w_{jk}; $w_{jk}(n)$ = weight change of w_{jk} in the nth iteration; N = number of training iterations and $\frac{\partial \text{SSE}}{\partial w_{jk}}(n)$ = rate of change of the SSE with respect to w_{jk} in the nth training iteration.

4) Calculate the LRSI's of each HIP and OHP using (5).
5) In each layer, prune the weight connections which have LRSI's less than β. In the simulation studies, we have chosen $\beta = 0.01$, i.e., if an LRSI is less than 1% of the total SV in that specific HIP or OHP, it is considered redundant and is pruned.
6) Re-train the FANN based on the pruned network.
7) Compare the model performance of the pruned model with the model prior to pruning.
8) If there is improvement in model performance, repeat steps (3) to (7) until the SSE remains within the specified limit, or does not reduce any further.

The criteria used in the validation of the FANN in the present studies are: 1) Sum squared error in training phase (SSETRAIN); 2) Sum squared error in testing phase (SSETEST); 3) Number of weight connections pruned; 4) Generalization factor ($\text{GF} = \frac{\text{SSETEST}}{\text{SSETRAIN}}$) which reflects the generalization capabilities of the model. The larger the value of GF, the poorer will be the generalization capability of the FANN model.

IV. Simulation Studies

The proposed pruning scheme can be applied to any fully connected neural network model. As this scheme is applicable to a good model developed beforehand, the normal procedure for developing a neural network model is assumed prior to applying this procedure. In other words, network size, training and validation data, issues such as initialization etc. must be chosen using general procedure

TABLE I
COMPARISON OF PERFORMANCE BETWEEN GLOBAL AND LOCAL PRUNING

System	Three bit parity problem		Van der Pol Equation		Chemical Process [16]		non-linear system [17]		non-linear system [18]	
Method	Global pruning	Local pruning	Global pruning	Local pruning	Global pruning	Local pruning	Global pruning	Local pruning	Global pruning	Local pruning
FANN structure	3:4:1 (16 weights)		4:4:1 (20 weights)		10:6:1 (66 weights)		3:36:1 (144 weights)		5:36:1 (216 weights)	
Number of data samples for training	8		200		2000		1000		1000	
Number of data samples for testing	8		200		2000		1000		1000	
weights pruned (%)	6 (37.5%)	6 (37.5%)	0 (0 %)	4 (20%)	5 (7.6%)	17 (25.8%)	3 (2 %)	31 (22%)	18 (8.3%)	29 (13.4%)
Re-training epochs	48	38	40	40	100	120	600	1000	360	100
SSETRAIN	1.4×10^{-5}	1.7×10^{-6}	7.9×10^{-4}	1.8×10^{-4}	1.1×10^{-5}	0.5×10^{-5}	0.0399	0.0269	0.458	0.231
SSETEST	1.4×10^{-5}	1.7×10^{-6}	8.9×10^{-4}	1.8×10^{-4}	1.15×10^{-5}	0.59×10^{-5}	0.0643	0.0364	0.5	0.233
GF	1	1	1	1	1.045	1.18	1.61	1.35	1.09	1.01

normally used for network training [1]. Although various methods of initialization of weights before training the FANN were proposed [11], [12], most pruning techniques including OBS [7] and OBD [8] assumed that the initialization does not affect the results of network pruning provided that a reasonable initialization method was used. In this work, the initialization method proposed in [13] was used which is part of the routine of the neural network toolbox inside MATLAB simulation package [14] which is widely used in the neural networks research community. This method initializes each individual weight such that the output of the hidden neurons connected to these weights will be starting from the active region of the activation function. In order to validate the proposed scheme, a number of test systems were simulated and were pruned using the global pruning procedure of Karnin and also with the proposed local pruning scheme. The bench mark systems used for comparative studies include the three bit parity problem [15], the Van der Pol equation, a chemical process [16], and different nonlinear system models [17], [18]. For each fully developed neural-network model, the global pruning procedure was applied first. As no formal rules were specified for this procedure, the pruning involved deleting one weight with the lowest sensitivity value at a time, and comparing the performance criteria to check if further pruning can be done. Reduced models were also obtained using multiple pruning using Karnin's method. The models were then pruned systematically using the proposed pruning scheme based on local comparison of sensitivity values. The final network sizes, sum-squared errors in the training and testing phase, and the generalization factors were compared (Table I).

The PSS scheme produced pruning in the range of 13.4% to 37.5% out of weight sets ranging between 16 and 216. The maximum number of weights pruned was 31 out of 144 weights. The proposed scheme offers significant advantages over Karnin's global method: 1) The PSS offers a systematic approach and guidelines for pruning at each stage and 2) The PSS leads to better pruning while retaining or improving all the three performance measures compared to Karnin's method. More importantly, the proposed pruning method is more efficient as it provides rules which help pruning a set of weights in a single stage as compared to the two stage process (single weight pruning or multiple weight pruning) of the global SV method.

V. CONCLUSION

In this paper a formal pruning selection scheme based on a sensitivity term was proposed. A mathematical expression of BP was derived to illustrate the importance of local effects in weight changes. a new sensitivity measure, local relative sensitivity index, was defined for each hidden neuron to input path and output neuron to hidden path so as to provide a formal way of interpreting the degree of importance of each weight connection. A procedure for pruning at most one weight connection in each of the weight groups belonging to the HIP or OHP was developed and was applied to the determination of a reduced FANN structure for a number of benchmark systems. Simulation studies showed that, compared to the pruning method based on global SV's, the proposed method produces FANN's with smaller sum-squared error in the testing phase, better generalization capability and reduced complexity of the FANN model in terms of number of weight connections pruned. The amount of retraining effort during the pruning process in the proposed method is either smaller or similar to the one based on global SV's. It was also shown that this formal pruning scheme can prune up to 37.5% of the total number of weight connections of a fully connected model without the need of incurring heavy computational burden of calculation the Hessian matrix of the system while using pruning methods such as OBS [7] and OBD [8] algorithms. Further work is under progress to apply this method to multiple input and output system and classification problems.

REFERENCES

[1] S. Haykin, *Neural Networks—A Comprehensive Foundation*. New York: Macmillan, 1994.
[2] Y. Chauvin, "Generalization performance of overtrained backpropagation networks," in *Proc. Neural Networks EUROSIP Workshop*, L. B. Almeida and C. J. Wellekens, Eds., 1990, pp. 46–55.
[3] D. Soloway and J. Bialasiewicz, "Neural-network modeling of nonlinear systems based on Volterra series extension of a linear model," in *Proc. IEEE Int. Symp. Intell. Contr. Glasgow*, 1992, pp. 7–12.
[4] S. Billings, H. Jamaluddin, and S. Chen, "Properties of neural networks with applications to modeling nonlinear dynamical systems," *Int. J. Contr.*, vol. 55, no. 1, pp. 193–224, 1992.
[5] K. Ho, P. Ponnapalli, and M. Thomson, "A novel MULTI-TYPE architecture for FANN's," in *Proc. Inst. Elect. Eng. 5th Int. Conf.*

Artificial Neural Networks, Cambridge U.K., Publication 440, 1997, pp. 239–244.
[6] M. Fukumi, S. Omatu, and M. Teranishi, "A new neuron model 'Cone' with fast convergence rate and its application to pattern recognition," *Syst. Comput. Japan*, vol. 22, no. 1, pp. 91–98, 1991.
[7] B. Hassibi, D. Stork, and G. Wolff, "Optimal brain surgeon and general network pruning," in *IEEE Int. Conf. Neural Networks*, 1993, vol. 1, pp. 293–299.
[8] Y. Cun, J. Denker, and S. Solla, "Optimal Brain Damage," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, vol. 2, pp. 598–605, 1990.
[9] E. Karnin, "A simple procedure for pruning backpropagation trained neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 239–242, 1990.
[10] G. Cybenco, "Approximations by superpositions of a sigmoidal function," *Math. Cont. Signal Syst.*, vol. 2, pp. 303–314, 1989.
[11] Y. Lee, S. H. Oh, and M. W. Kim, "An analysis of premature saturation in backpropagation learning," *Neural Networks*, vol. 6, no. 5, pp. 719–728, 1993.
[12] N. Weymaere and J. P. Martens, "On the initialization and optimization of multilayer perceptrons," *IEEE Trans. Neural Networks*, vol. 5, pp. 738–751, 1994.
[13] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in *Int. Joint Conf. Neural Networks*, 1990, vol. 3, pp. 21–26.
[14] H. Demuth and M. Beale, *MATLAB Neural Network Toolbox User's Guide*, The MathWorks, 1997.
[15] M. Riedmiller, "Advanced supervised learning in multilayer perceptrons—From backpropagation to adaptive learning algorithms," *Comput. Standards Interfaces*, vol. 16, pp. 265–278, 1994.
[16] S. P. Schooling, M. Thomson, and M. Mackay, "The application of a nonlinear identification and control scheme to a continuous stirred tank reactor," *Advances Process Contr.*, vol. 4, pp. 219–226, 1995.
[17] F. C. Chen and H. K. Khalil, "Adaptive control of a class of nonlinear discrete time systems using neural networks," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 791–801, 1995.
[18] L. Jin, P. N. Nikiforuk, and M. M. Gupta, "Adaptive control of discrete time nonlinear systems using recurrent neural networks," in *Proc. Inst. Elect. Eng. Contr. Theory Applicat.*, 1994, vol. 141, no. 3, pp. 169–176.

A Local Minimum for the 2-3-1 XOR Network

Ida G. Sprinkhuizen-Kuyper and Egbert J. W. Boers

***Abstract*— It was assumed proven that two-layer feedforward neural networks with $t-1$ hidden nodes, when presented with t input patterns, can not have any suboptimal local minima on the error surface. In this paper, however, we shall give a counterexample to this assumption. This counterexample consists of a region of local minima with nonzero error on the error surface of a neural network with *three* hidden nodes when presented with *four* patterns (the XOR problem). We will also show that the original proof is valid only when an unusual definition of local minimum is used.**

***Index Terms*— Error surface, feedforward neural network, gradient-based learning, local minimum, XOR function.**

I. Introduction

Until recently, it was assumed that the error surfaces of two-layered feedforward neural networks using the sigmoid transfer function with $t-1$ hidden nodes do not have suboptimal local minima when presented with t patterns. This was proved in papers by Poston *et al* [5] and Yu [10]. During our investigation of several error surfaces of small neural networks when presented with the task of learning the XOR problem [7]–[9], we found regions of suboptimal local minima for the 2-2-1 XOR network. This network has two instead of three hidden nodes, so the proof of Poston *et al.* and Yu is not applicable. However, we noticed that some of these suboptimal local minima do not disappear by adding an extra hidden node. This contradicts the beforementioned proofs. This is the main result that is presented in this paper: an example of suboptimal local minima with nonzero error for the error surface of a two-layer feedforward network with three hidden nodes.

In the paper by Poston *et al.* [5] it is proved that for "normal" problems (i.e., for problems without contradictory training examples) with t training examples, a neural network with $t-1$ "real" hidden nodes (one extra node with a fixed output of one is used to provide the threshold value of the output nodes, totaling t hidden nodes) is able to represent these training examples exactly, with zero error. Moreover, Poston *et al.* prove that the error surface of such a neural network cannot have local minima with error greater than zero. They do not give an explicit definition of local minimum, but the most commonly used definition of local minimum (which is used in this paper) is: A local minimum is a point with a neighborhood such that in each point of that neighborhood the value is greater or equal than the value of that point. They formulate their result as: *Therefore there cannot exist a $\mathbf{w}$ from which we can reach a good point via continuous change* only *by allowing a temporary increase in error. Local minima with nonzero error are impossible*.[1]

Yu [10] generalizes this result to training sets *with* contradictions. For such training sets it is not possible to find weight combinations resulting in error zero, but such error surfaces will have an absolute minimum value for the error. The proof by Yu has been criticized by Hamey [3], and in a reaction to Hamey's remarks, Yu (see [3])

Manuscript received November 22, 1998; revised March 4, 1999.

The authors are with the Department of Computer Science, Leiden University, The Netherlands.

Publisher Item Identifier S 1045-9227(99)05484-3.

[1] Note that the second statement does not follow from the first statement.