MAY 10 1993

U S F I

by Jaques Reifman
and Javier Vitela E.

Argonne National Laboratory, Argonne, Illinois 60439
operated by The University of Chicago
for the United States Department of Energy under Contract W-31-109-Eng-38

~~Results reported in the IFR-TM series of memoranda frequently are preliminary and subject to revision. Consequently they should not be quoted or referenced.~~

This document is not considered OUO-Applied Technology. It was reviewed for Export Controlled Information and found to be suitable for unlimited access and reproduction.

This label reflects Applied Technology instructions issued April 13, 2006, by the the Department of Energy Office of Nuclear Energy. Additional guidance has also been provided by DOE in 2016 and 2018 memos, as well as from NNSA.

Paul Betten. ANL

7/24/2019
Date

~~Released for announcement in NAT.
Distribution limited to participants
in the LMEFB program. Others request
from OSD, DOE.~~

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Argonne National Laboratory, with facilities in the states of Illinois and Idaho, is owned by the United States government, and operated by The University of Chicago under the provisions of a contract with the Department of Energy.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ARTIFICIAL NEURAL NETWORK TRAINING WITH
CONJUGATE GRADIENTS FOR DIAGNOSING TRANSIENTS
IN NUCLEAR POWER PLANTS

by

Jaques Reifman and Javier Vitela E.*

Reactor Analysis Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439

IFR TECHNICAL MEMORANDUM NO. 189

~~Results reported in the IFR TM series of memoranda frequently are preliminary and subject to revision. Consequently they should not be quoted or referenced.~~

~~APPLIED TECHNOLOGY~~

~~Any further distribution by any holder of this document or data therein to third parties representing foreign interests, foreign governments, foreign companies, and foreign subsidiaries or foreign divisions of U.S. companies shall be approved by the Associate Deputy Assistant Secretary for Reactor Systems, Development, and Technology, U.S. Department of Energy. Further, foreign party release may require DOE approval pursuant to Federal Regulation 10 CFR Part 810, and/or may be subject to Section 127 of the Atomic Energy Act.~~

MASTER

*Universidad Nacional Autonoma de Mexico, Instituto de Ciencias Nucleares, Mexico D.F.

~~Released for announcement in NAY.
Distribution limited to participants
in the LMFR program. Others request
from RSDT, DOE.~~ *zb*

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	v
I. INTRODUCTION	1
II. ARTIFICIAL NEURAL NETWORKS AND THE BACKPROPAGATION ALGORITHM	3
A. Artificial Neural Networks	3
B. The Backpropagation Algorithm	6
III. TRAINING NEURAL NETWORKS WITH THE METHOD OF CONJUGATE GRADIENTS	12
A. The Conjugate Gradient Algorithm	12
B. The One-dimensional Search	14
C. Comparison of Conjugate Gradient and Backpropagation	15
IV. NUCLEAR POWER PLANT TRANSIENT CLASSIFICATION	16
A. Transient Database	17
B. Neural Network Architecture	19
V. NEURAL NETWORK TRAINING AND DIAGNOSIS CAPABILITY	21
A. Neural Network Training	21
B. Diagnosis Capability	32
VI. SUMMARY AND CONCLUSIONS	37
ACKNOWLEDGMENTS	38
REFERENCES	38

LIST OF FIGURES

	<u>Page</u>
1. A Typical Feedforward Three-layer Artificial Neural Network	5
2. Representative Learning Curves Obtained by Training the Network with the Method of Conjugate Gradients and with Both Batch and On-line Backpropagation Algorithm	23
3. Effects of the Premature Saturation of the Network Nodes in the Learning Curve Experienced by Training the Network with the Batch Backpropagation Algorithm	25
4. Activation Level of the Three Output Nodes for a Feedwater Pump Trip Event Obtained by Training the Network with the Batch Backpropagation Algorithm . . .	26
5. Learning Curve Obtained by Training the Network with the Method of Conjugate Gradients	29
6. Activation Level of the Three Output Nodes for a Feedwater Pump Trip Event Obtained by Training the Network with the Method of Conjugate Gradients	30
7. The Learning Parameter η_k and Momentum Parameter α_k Estimated by the Method of Conjugate Gradients	31
8. The Effective Step Size or Learning Parameter η_{eff} Used by the Method of Conjugate Gradients	33
9. Activation Level of the Three Output Nodes for a Feedwater Pump Trip Event Not Used in Training	34
10. Activation Level of the Three Output Nodes for a Feed Flow Transmitter Event Not Used in Training	36

LIST OF TABLES

	<u>Page</u>
I. Three Transient Events that Challenge the Heat Removal Safety Function	17
II. List of Plant Parameters	18
III. Target Values for Three Output Nodes Corresponding to the Three Transient Events	20

ARTIFICIAL NEURAL NETWORK TRAINING WITH CONJUGATE GRADIENTS FOR DIAGNOSING TRANSIENTS IN NUCLEAR POWER PLANTS

by

Jaques Reifman and Javier Vitela E.

ABSTRACT

The method of conjugate gradients is used to expedite the learning process of feedforward multilayer artificial neural networks. The proposed method systematically determines both the learning and momentum parameters which are dynamically updated at each learning cycle. The learning parameter is obtained through a one-dimensional search that combines the iterative golden-section algorithm with an analytical cubic interpolation. The momentum parameter is obtained as conjugated directions are calculated at each learning cycle. By dynamically updating the learning and momentum parameters the method avoids the phenomenon of premature saturation of the network nodes and becomes less sensitive to the initial set of weights. The proposed method is compared with the standard backpropagation algorithm in the training of neural networks to classify transient events in nuclear power plants simulated by the Midland Nuclear Power Plant Unit 2 simulator. The comparison results indicate that the rate of convergence of the proposed method is much greater than the standard backpropagation reducing the number of learning cycles and the CPU time. The advantages of the method are more noticeable and important for problems where the network architecture consists of a large number of nodes, the training database is large, and a tight convergence criterion is desired. In addition, the reasons for the premature saturation of the network nodes observed with the backpropagation algorithm are described and suggestions are made to eliminate this undesirable phenomenon.

I. INTRODUCTION

Artificial intelligence techniques in the form of artificial neural networks (ANNs) have recently been proposed¹⁻⁵ to provide support for nuclear power plant operations. The applications of neural networks in nuclear engineering are spread over a wide range of support areas including plant parameter estimation,^{1,2} nuclear power plant transient event classification,³ dynamic system identification,⁴ and thermal performance evaluation of power plants,⁵ among many other applications. The reasons for the sudden large interest in the application of ANNs are due to the recent popularization of the backpropagation (BP) neural network learning algorithm⁶ and its versatility in modeling the input-output relationships of complex nonlinear systems.

Neural networks are composed of many nonlinear computational elements or nodes arranged in patterns reminiscent of biological neural nets. The computational nodes are connected via weights that are updated through training or learning rules to improve performance. Given a set of input-output patterns, an ANN is trained to model its relationships by updating the weights such that the prediction error between the network estimated output values and the known outputs is minimized. From the many different types of ANNs⁷ the one most frequently used, in general as well as in nuclear power plant applications, is the feedforward multilayer perceptron with the BP algorithm as its training mechanism for supervised learning. The standard BP algorithm is basically a gradient descent method and in spite of the fact that global optimization cannot be guaranteed it has proven to be a robust method for training ANNs. However, as with many gradient descent optimization methods, neural network training with the BP algorithm suffers from the following three convergence-related drawbacks:

1. The convergence is slow and becomes slower the closer the algorithm gets to the optimum solution,
2. The algorithm requires judicious selection of parameters, i.e., the learning and momentum parameters, to achieve reasonable convergence and avoid oscillations about the optimum solution, and

3. The rate of convergence is very sensitive to the initial set of weights.

These problems increase the time-consuming task of training ANNs by increasing the number of learning cycles and CPU time necessary for convergence and requiring additional trial-and-error runs to determine good optimization parameters for the particular problem at hand.

In this report, we present a method for improving learning of neural networks. The proposed method significantly expedites the learning rate of ANNs as compared to the standard BP algorithm, does not require the selection of optimization parameters, and its overall convergence properties are much less sensitive to the initial set of weights chosen to start the learning process. Improved neural network learning is obtained by applying the well-known unconstrained nonlinear optimization method of conjugate gradients⁸ (CGs) to minimize the prediction error. Unlike the BP algorithm which minimizes the prediction error by moving along its gradient direction and is not guaranteed to converge,⁹ the method of CGs moves along the conjugate directions and is guaranteed to obtain the minimum of a quadratic function, apart from round-off errors, in a finite number of steps.⁸ For functions that are not quadratic, which is the case for the prediction error of multilayer neural networks with sigmoid mapping functions, the process is iterative. In the proposed method, the gradient calculations performed by the BP algorithm, as it propagates the prediction error from the output to the input nodes of the network, are used to determine the conjugate directions along which the weights are adjusted. In addition to having better convergence properties and being less sensitive to the initial set of weights than the gradient-descent-based BP algorithm, the method of CGs provides a systematic way to determine and to dynamically update both the learning and momentum parameters at each iteration. By dynamically updating these parameters the method of CGs avoids the phenomenon of premature saturation of network nodes that is often associated with neural network training via the BP algorithm.^{10,11} In this report, we also describe the reasons for the occurrence of the premature saturation of the nodes and provide suggestions for modifying the BP algorithm to avoid the occurrence of this undesirable phenomenon. To demonstrate the advantages of the proposed CG method over the standard BP algorithm we compare the two methods in training a multilayer feedforward perceptron to classify transient

events in nuclear power plants. However, the method is general and can be used to expedite learning for all applications of neural networks. The database of transients used for training the network is generated through the full-scale operator training Midland Nuclear Power Plant Unit 2 (MNP-2) simulator¹² which allows for a realistic representation of all major systems and operating condition of a pressurized water reactor (PWR).

The architecture of ANNs, the standard BP learning algorithm and its major drawbacks are briefly described in Sect. II. In Sect. III, we discuss the method of CGs, show how it is used within a neural network context, describe the one-dimensional search used to dynamically update the learning parameter, and show that the method of CGs provides a systematic approach for dynamically updating the learning and momentum parameters of the BP algorithm. In Sect. IV, we describe the construction of the database of transients, i.e., the set of input-output training data, through the MNP-2 simulator and the architecture of the neural network used in modeling the relationships between plant parameters and transient events. The comparison of training a neural network with the proposed method versus the standard BP algorithm and the diagnosis capability of the trained network is presented in Sect. V, followed by a summary and conclusions in Sect. VI.

II. ARTIFICIAL NEURAL NETWORKS AND THE BACKPROPAGATION ALGORITHM

In this section we briefly summarize the characteristics of feedforward multilayer neural networks and describe the standard BP learning algorithm,⁶ which is generally used to train neural networks.

A. Artificial Neural Networks

Artificial neural networks can be described as nonlinear modeling systems where continuous input vectors are mapped through the network into continuous output vectors. For nuclear power plant transient event classification, each element of the input vector corresponds to a measured plant parameter, while each element of the output vector represents a particular

transient event. Neural networks are composed of many nonlinear computational elements or nodes arranged in patterns reminiscent to biological neural nets. The nodes of feedforward ANNs are generally arranged in L layers, with $L \geq 3$, where information flows from nodes located at the first or input layer to nodes located at the L -th or output layer through the $L-2$ hidden layers. Figure 1 shows a typical architecture of a feedforward three-layer, $L=3$, network with one hidden layer where each layer is composed of J_ℓ ($\ell=1,2,\dots,L$) nodes. In Fig. 1, each one of the J_1 nodes of the input layer receives as input one of the J_1 elements $x_j^{(1)}$ ($j=1,2,\dots,J_1$) of the input vector, which are passed unaltered and serve as the output values of the input layer nodes. The nodes of the hidden and output layers serve as nonlinear transfer functions mapping a multidimensional input received from nodes of the immediately preceding layer to a one-dimensional output $x_j^{(\ell)}$ ($j=1,2,\dots,J_\ell$; $\ell=2,\dots,L$). Each output of the nodes located in the input and hidden layers are distributed to the nodes of the immediately following layer through weighted connections, while each one of the J_L outputs $x_j^{(L)}$ ($j=1,2,\dots,J_L$) of the output layer corresponds to one element of the output vector.

The most widely used transfer function in ANNs is the sigmoid function.⁶ The sigmoid function mapping or activation level of the output $x_j^{(\ell)}$ of the j -th node in the ℓ -th layer, with $\ell > 1$, in Fig. 1 is given by

$$x_j^{(\ell)} = \frac{1}{1 + e^{-u_j^{(\ell)}}} \quad (1)$$

Here $u_j^{(\ell)}$ denotes a linear weighted sum of the outputs $x_i^{(\ell-1)}$ ($i=1,2,\dots,J_{\ell-1}$) of the immediately preceding layer plus a threshold $\Theta_j^{(\ell)}$ of the j -th node in ℓ -th layer

$$u_j^{(\ell)} = \sum_{i=1}^{J_{\ell-1}} w_{ji}^{(\ell)} x_i^{(\ell-1)} + \Theta_j^{(\ell)}, \quad (2)$$

Fig. 1. A Typical Feedforward Three-layer Artificial Neural Network

where $w_{ji}^{(\ell)}$ is the weight connecting the output of the i -th node in the $(\ell-1)$ -th layer to the j -th node in the ℓ -th layer.

B. The Backpropagation Algorithm

Before an ANN can be used to predict the output associated with an input pattern, it needs to learn or be trained from an existing database consisting of pairs of input-output patterns. Given a set of P teaching patterns, where pattern p ($p=1,2,\dots,P$) is composed of inputs $x_{pi}^{(1)}$ ($i=1,2,\dots,J_1$) and target outputs t_{pj} ($j=1,2,\dots,J_L$), neural network learning is obtained by adjusting the connecting weights $w_{ji}^{(\ell)}$ of the network such that the prediction error E_p between the network estimated output $x_{pj}^{(L)}$ ($j=1,2,\dots,J_L$) and the target output t_{pj} , for all P patterns, is minimized. In general, the thresholds $\theta_j^{(\ell)}$ are also adjusted, but since the thresholds can be treated as weights with constant unit inputs we restrict our analysis to the adjustment of the weights $w_{ji}^{(\ell)}$ and assume that the prediction error is a function of the weights only, i.e., $E_p = E_p(\mathbf{w})$. In the standard BP algorithm, which is by far the most widely used method for training feedforward multilayer neural networks, the error E_p for an input-output pattern p is defined as the mean square difference between t_{pj} and $x_{pj}^{(L)}$

$$E_p = \frac{1}{2} \sum_{j=1}^{J_L} (t_{pj} - x_{pj}^{(L)})^2, \quad (3)$$

and the total cumulative prediction error E to be minimized is given by

$$E = \sum_{p=1}^P E_p. \quad (4)$$

The standard BP algorithm popularized by Rumelhart et al.⁶ is basically a gradient descent algorithm for minimizing the prediction error E of Eq. (4) in an iterative fashion. At each iteration, the gradient ∇E is recursively calculated by propagating partial derivatives of E backwards, i.e., from the output to the input nodes, and adjusting the weights proportional to the negative gradient direction until a convergence criterion is satisfied. The BP algorithm is briefly summarized in the following four steps:

1. Randomly select initial values in the range $[-1,1]$ for all weights and set the iteration counter $k=1$.
2. Present P input-output patterns to the network and calculate the differences $t_{pj} - x_{pj}^{(L)}$ ($p=1,2,\dots,P$; $j=1,2,\dots,J_L$). Stop if the convergence criterion $|t_{pj} - x_{pj}^{(L)}| < \epsilon$ is satisfied; otherwise continue to step 3.
3. Compute the components of the gradient $\nabla E(w_k)$ recursively, for iteration k , starting at the nodes in the output layer and working backwards to the nodes in the input layer. To simplify the notation we suppress the iteration subscript k until step 4. A component of $\nabla E(w)$ corresponding to weight $w_{ji}^{(\ell)}$ connecting the i -th node in the $(\ell-1)$ -th layer to the j -th node in the ℓ -th layer is given by,

$$\frac{\partial E}{\partial w_{ji}^{(\ell)}} = - \sum_{p=1}^P \delta_{pj}^{(\ell)} x_{pi}^{(\ell-1)} \quad (5)$$

If the ℓ -th layer is the output layer, i.e., $\ell=L$, then

$$\delta_{pj}^{(\ell)} = (t_{pj} - x_{pj}^{(L)}) x_{pj}^{(L)} (1 - x_{pj}^{(L)}), \quad (6)$$

and for any node in a subsequent hidden layer, i.e., $1 < \ell < L$,

$$\delta_{pj}^{(\ell)} = x_{pj}^{(\ell)} (1 - x_{pj}^{(\ell)}) \sum_{m=1}^{J_{\ell+1}} \delta_{pm}^{(\ell+1)} w_{mj}^{(\ell+1)} \quad (7)$$

4. Update the weights $w_{k+1} = w_k + \Delta w_k$ after iteration k , set the iteration counter $k=k+1$ and return to step 2. In the standard BP algorithm the change in weights Δw_k after the first iteration is given by

$$\Delta w_k = -\eta \nabla E(w_k) + \alpha \Delta w_{k-1}, \quad (8)$$

where η and α are constant parameters known as the learning and momentum parameters, respectively.

The major drawbacks of the BP algorithm are related to the well-known weak convergence characteristics of the gradient descent method in which BP is based. First, the method of gradient descent is affected by the contour of the error-surface in the weight-space. The more skewed the curvatures of the space, the more the search directions oscillate across the ideal path toward the minimum.¹³ In addition, the rate of convergence of the gradient descent method is proportional to the norm of the gradient $|\nabla E|$.⁹ Therefore, gradient descent makes increasingly slow progress the closer it gets to the solution.

The rate of convergence of the BP algorithm depends significantly on the values used for the learning parameter η and the momentum parameter α . For regions in weight-space far away from the optimum and for error surfaces with broad local minima, a large value of η will result in larger changes in the weights yielding a more rapid convergence. However, for regions in weight-space close to the optimum and for problems with steep narrow minima, a small value of η must be chosen to avoid oscillations about the optimum solution. The momentum parameter α , generally chosen between 0 and 1, is the coefficient of Δw_{k-1} which provides a sort of momentum in the weight-space by including the effect of past weight changes on the current

direction of movement.⁶ The momentum term $\alpha \Delta \mathbf{w}_{k-1}$ accelerates convergence in regions of weight-space with relatively constant gradient and oscillatory gradient directions.¹³ The optimum values for η and α depend on the problem being solved and other than a trial-and-error experiments, the standard BP algorithm does not provide a general prescription for selection of optimum values for these two parameters.

Another convergence drawback of the standard BP algorithm is the occurrence of the phenomenon known as premature "saturation" of the network nodes.^{10,11} For certain initial values of the randomly selected weights determined in the first step of the BP algorithm, the output values $x_j^{(l)}$ obtained through Eq. (1), for both hidden and output nodes, may prematurely saturate to 0 or 1 within a few iterations of the algorithm. When this phenomenon occurs, the values of $\delta_{pj}^{(l)}$ in Eqs. (6) and (7) become very small and each iteration of the algorithm produces very small changes in the values of the weights and a negligible move in the direction of the minimum. This behavior persists for a large number of iterations or training cycles until the output values $x_j^{(l)}$ recover from their saturated condition and the weights start moving towards the direction of the minimum. The effect of the premature saturation of the network nodes is an increase in the number of training cycles required for convergence of the BP algorithm. Hence, the sensitivity of the overall convergence of the BP algorithm to the initial random selection of weights will increase the average number of training cycles and may require additional trial-and-error runs to find an appropriate initial set of weights that speeds-up convergence.

Our analysis shows that the process of premature saturation of the network nodes can be divided into three stages: beginning of saturation, saturation plateau, and recovery from saturation. The first stage, beginning of saturation, corresponds to the very first few iterations of the BP algorithm where the learning term $-\eta \nabla E(\mathbf{w}_k)$ "freezes" very quickly at very small absolute values. Saturation occurs when the randomly selected weights places the starting point of the BP algorithm in a region of the weight-space that has skewed error-surface. The skewness of the error-surface causes some components of the gradient $\nabla E(\mathbf{w}_k)$ in Eq. (8) to change signs after a few iterations of the BP algorithm. Since the momentum term $\alpha \Delta \mathbf{w}_{k-1}$ in

Eq. (8) represents the "memory" of previous directions of movement, some of its components will have signs that are opposite to the components of $-\eta \nabla E(\mathbf{w}_k)$. Opposite signs in some components of $-\nabla E(\mathbf{w}_k)$ and $\Delta \mathbf{w}_{k-1}$ when the absolute value of these components of $\alpha \Delta \mathbf{w}_{k-1}$ are larger than the absolute value of the corresponding components of $-\eta \nabla E(\mathbf{w}_k)$ will cause some components of the change in weights $\Delta \mathbf{w}_k$ to be in the "wrong" direction, i.e., opposite to the negative gradient $-\nabla E(\mathbf{w}_k)$. Because at the early stages of the algorithm the components of $\Delta \mathbf{w}_k$ are of the same order of magnitude as the components of $\mathbf{w}_k$, the absolute value of the components of $\mathbf{w}_{k+1} = \mathbf{w}_k + \Delta \mathbf{w}_k$ could rapidly increase in the "wrong" direction. The rapid increase of $\mathbf{w}_{k+1}$ in the "wrong" direction would persist for a few iterations causing a quick saturation of the activation level of the output nodes $x_j^{(l)}$ in Eq. (1) to values close to 0 or 1. As a consequence of the saturation of the output nodes, the values of $\delta_{pj}^{(l)}$ in Eqs. (6) and (7) become very small, since they are proportional to $x_j^{(l)} (1 - x_j^{(l)})$, causing the learning term $-\eta \nabla E(\mathbf{w}_k)$ to "freeze" very quickly at very small absolute values. We should also note that the "wrong" changes in the weights at the first stage of the process of saturation could have the undesirable effect of causing the prediction error E in Eq. (4) to increase.

The second stage of the process of premature saturation of the network nodes, the saturation plateau, corresponds to the iterations of the BP algorithm where the weights $\mathbf{w}_{k+1}$ and the prediction error E remain practically constant. By the end of the first stage, the contributions of $-\eta \nabla E(\mathbf{w}_k)$ to the "correct" changes in $\mathbf{w}_k$ are negligible, while the magnitude of the contributions of $\alpha \Delta \mathbf{w}_{k-1}$ for the "wrong" changes in $\mathbf{w}_k$ have also decreased to a negligible value from its initial large value. Throughout the second stage of the saturation process, the value of $\alpha \Delta \mathbf{w}_{k-1}$ continues to decrease at every iteration of the BP algorithm. Therefore, the net contribution of the two terms, $-\eta \nabla E(\mathbf{w}_k)$ and $\alpha \Delta \mathbf{w}_{k-1}$, is small in comparison with the absolute value of $\mathbf{w}_k$, causing both the weights $\mathbf{w}_{k+1}$ and the prediction error E to remain practically constant. This behavior persists for a large number of iterations until the network starts to recover from saturation initiating the third and final stage of the saturation process. The recovery from saturation corresponds to the end of the process of premature saturation where the weights $\mathbf{w}_{k+1}$ start to change in the "correct" direction allowing the prediction error E to decrease. The recovery from saturation starts when after a number of iterations of the BP

algorithm, the absolute value of the components of $\alpha \Delta \mathbf{w}_{k-1}$ have decreased enough to become smaller than the absolute values of the components of $-\eta \nabla E(\mathbf{w}_k)$. From this point on, the net changes in $\mathbf{w}_{k+1}$ will be in the "correct" direction, i.e., the opposite direction of the local gradient, which allows for the output nodes $x_j^{(l)}$ in Eq. (1) to recover from their saturated activation levels. As a consequence of the recovery from saturation of the output nodes, the values of $\delta_{pj}^{(l)}$ in Eqs. (6) and (7) start to increase, causing the learning term $-\eta \nabla E(\mathbf{w}_k)$ to recover from its "frozen" state acquired during the beginning of saturation. During this stage, the magnitude of the absolute value of the components of $-\eta \nabla E(\mathbf{w}_k)$ increase very slowly at each iteration of the algorithm until the network nodes have completely recovered from saturation causing the weights $\mathbf{w}_{k+1}$ to start approaching their correct values which, in turn, allows for the prediction error E to start decreasing. The larger the difference between the absolute values of the components of $-\eta \nabla E(\mathbf{w}_k)$ and $\alpha \Delta \mathbf{w}_{k-1}$ at the beginning of saturation are, the longer is the recovery process from saturation.

Numerous schemes have been proposed to speed up the convergence of the BP algorithm for training feedforward multilayer neural networks. Basically, the proposed schemes can be grouped into two categories: algorithms based on heuristics^{10,11,14-17} and algorithms stemming from nonlinear optimization techniques.^{13,18-24} The first category of algorithms, based on heuristics, uses empirical adaptation techniques to dynamically update the learning and momentum parameters,^{11,14-17} as well as the slope of the sigmoid function.¹⁰ In spite of the fact that these heuristic schemes have been able to speed up the convergence of the standard BP algorithm for particular problems, their heuristic formulation does not guarantee an increase in the rate of convergence for other problems. The second category of algorithms, based on nonlinear optimization techniques, includes algorithms based on one-dimensional searches for dynamically updating the learning parameter,¹⁸ conjugate gradient methods that use different techniques for determining the conjugate directions and performing the one-dimensional search,^{19,20} Newton and Quasi-Newton methods with different Hessian matrix formulations,^{13,21,22} and extended Kalman Filter techniques.^{23,24} These algorithms have produced great reductions in the number of iterations necessary to train feedforward multilayer ANNs, however, some of these approaches require complex and costly calculations at each iteration which offset their advantage.

In this report we propose to use the unconstrained nonlinear optimization method of conjugate gradients developed by Fletcher and Reeves⁸ to improve neural network learning. Here, the one-dimensional search for the dynamic calculation of the learning parameter is performed by combining the golden section rule²⁵ with a cubic interpolation.²⁶ The algorithm is simple to implement, has reasonable computational costs, and has better convergence properties than gradient descent methods.

III. TRAINING NEURAL NETWORKS WITH THE METHOD OF CONJUGATE GRADIENTS

The method of conjugate gradients is a well-known quadratically convergent gradient method for locating an unconstrained local minimum of a function of n variables.⁸ The method is iterative and for quadratic functions it is guaranteed that the minimum will be located exactly, apart from rounding errors, in a maximum of n iterations. For functions that are not quadratic, which is the case for the prediction error $E(w)$ in Eq. (4), the method is based on the current local quadratic approximation to the function which causes the search process to be iterative rather than a n -step process, and a test of convergence is required. As the function approaches the minimum its approximation of a quadratic function becomes more realistic causing the method to converge faster. This is in contrast with the BP algorithm whose rate of convergence decreases as the minimum is approached.

A. The Conjugate Gradient Algorithm

The method of conjugate gradients obtains the position of the minimum w^* of a convex quadratic function $E(w)$, with $w \in \mathcal{R}^n$, through a sequence of approximations $w_0, w_1, \dots, w_n$ such that $w_n = w^*$. Starting from a randomly selected initial position w_0 , we arrive at the next position $w_1 = w_0 + \eta_0 \mu_0$ by moving along the conjugate direction μ_0 an amount η_0 , where η_0 minimizes $E(w_0 + \eta \mu_0)$ with respect to η . This sequence is repeated until the position of the minimum $w_n = w_0 + \sum_{k=0}^{n-1} \eta_k \mu_k$ is obtained, where μ_k ($k=0,1,\dots,n-1$) are linearly independent conjugate vectors and each scalar η_k ($k=0,1,\dots,n-1$) minimizes $E(w_k + \eta \mu_k)$ with respect to η .

at corresponding positions w_k . In this report, the one-dimensional minimization of $E(w_k + \eta \mu_k)$ is iteratively obtained by combining the golden section rule²⁵ with a cubic interpolation²⁶ and the sequential generation of the conjugate search directions μ_k are obtained through the algorithm developed by Fletcher and Reeves.⁸

The following seven steps summarize the developed conjugate gradient-based algorithm for training feedforward multilayer neural networks. The algorithm is iterative and at each iteration it requires information of the gradient ∇E , which is obtained just as in the BP algorithm described in Sect. II.B, to calculate the conjugate directions μ_k . The seven steps are as follows:

1. Randomly select initial values in the range $[-1, 1]$ for the weights w and set the iteration counter $k=1$.
2. Present P input-output patterns to the network and calculate the differences $t_{pj} - x_{pj}^{(l)}$ ($p=1,2,\dots,P$; $j=1,2,\dots,J_l$). Stop if the convergence criterion $|t_{pj} - x_{pj}^{(l)}| < \epsilon$ is satisfied; otherwise continue to step 3.
3. Compute the gradient $g_k = \nabla E(w_k)$ as in the BP algorithm.
4. Compute the conjugate direction μ_k as follows:

$$\mu_k = \begin{cases} -g_k + \frac{g_k^T \cdot g_k}{g_{k-1}^T \cdot g_{k-1}} \mu_{k-1} & ; \text{ for } k > 1 \\ -g_1 & ; \text{ for } k = 1 \end{cases} \quad (9)$$

5. Obtain η_k which minimizes $E(w_k + \eta \mu_k)$ with respect to η through a one-dimensional search.

6. Update the weights $w_{k+1} = w_k + \Delta w_k$, where the change in weights Δw_k is given by

$$\Delta w_k = \eta_k \mu_k. \quad (10)$$

7. Return to step 2 and update k as follows: if $k < n$ set the iteration counter $k = k + 1$, otherwise if $k = n$ set $k = 1$.

The developed algorithm is similar to the BP algorithm. The first three steps are essentially the same as in the BP algorithm. Steps 4 and 5 corresponding to the calculation of the conjugate directions μ_k and the learning parameter η_k , respectively, have no equivalent steps since in the BP algorithm the gradient directions g_k are used and the learning parameter is fixed. The weight updates in step 6 and the iteration counter update in step 7 are equivalent to step 4 in the BP algorithm. The one-dimensional minimization search performed in step 5 requires attention. If a balance between accuracy and computational speed is not achieved, the time spent evaluating the value of the prediction error function $E(w_k + \eta \mu_k)$ during the iterative process of finding η_k may dominate the overall time of neural network learning. Here, this balance is attained by incorporating the golden section rule with a cubic interpolation.

B. The One-dimensional Search

Analytical and iterative one-dimensional search techniques can be used to find η_k which minimizes $E(w_k + \eta \mu_k)$ with respect to η , with E assumed to be unimodal. Analytical one-dimensional searches assume that E can be approximated by a polynomial of degree M and require at most $M+1$ function evaluations. In feedforward neural networks, a function evaluation corresponds to presenting the P input patterns to the network and evaluating E in Eq. (4). Although fast, the accuracy of analytical techniques depends on the function evaluations being performed in a region close to the actual minimum η_k^* . Iterative one-dimensional search techniques involve sequential function evaluations where at each iteration the search interval in the η -line containing η_k^* is reduced. The main disadvantage of iterative techniques is that to

achieve reasonable accuracy in the location of η_k^* they require a large number of function evaluations.

In order to balance accuracy and computational speed in the one-dimensional search performed at each iteration of the conjugate gradient algorithm, we combine the iterative golden-section algorithm²⁵ with the analytical cubic interpolation technique.²⁶ The golden-section technique is used first to determine lower and upper values of the search interval in which is located, after which, a cubic function of η is fitted to obtain η_k as an estimate to η_k^* . The lower η^l and upper η^u values of the search interval are obtained by sequentially evaluating and comparing the prediction error function $E(\mathbf{w}_k + \eta \mu_k)$ at $\eta = \eta^i$ ($i=1,2,\dots$) defined by the golden-section rule,

$$\eta^i = (1.618)^{i-1} \zeta_k, \quad (11)$$

with

$$\zeta_k = 0.01 \frac{E(\mathbf{w}_k)}{|\nabla E(\mathbf{w}_k)|}, \quad (12)$$

until the search interval $[\eta^{i-2}=\eta^l, \eta^i=\eta^u]$ contains η_k^* . The search interval is partitioned again into a smaller interval using the golden-section rule until a predefined precision is obtained, after which a cubic function of η is fitted in the resulting interval and η_k is obtained as an estimate of η_k^* . Here, the basic step ζ_k in Eq. (12) is determined at each conjugate gradient iteration k , so that in the regions in weight-space where the norm of the gradient $|\nabla E(\mathbf{w}_k)|$ is small the step ζ_k is large and vice versa.

C. Comparison of Conjugate Gradient and Backpropagation

Although the weight update defined by Eq. (10) in step 6 of the CG algorithm seems different than the weight update defined by Eq. (8) in step 4 of the BP algorithm, they are completely equivalent. Furthermore, the CG algorithm provides dynamic updates for both

completely equivalent. Furthermore, the CG algorithm provides dynamic updates for both learning parameter η and momentum parameter α used in the BP algorithm. To determine the equivalence between the two equations and identify η and α in the CG framework we substitute μ_k of Eq. (9), for $k > 1$, into Eq. (10) and use the fact that $\mu_{k-1} = \Delta w_{k-1} / \eta_{k-1}$ to obtain

$$\Delta w_k = -\eta_k g_k + \frac{\eta_k}{\eta_{k-1}} \frac{g_k^T \cdot g_k}{g_{k-1}^T \cdot g_{k-1}} \Delta w_{k-1} . \quad (13)$$

If we now compare Eq. (13) with Eq. (8) we may identify the learning parameter η and momentum parameter α of the BP algorithm as the following terms of the CG algorithm

$$\eta = \eta_k , \quad \text{and}$$

$$\alpha = \alpha_k = \frac{\eta_k}{\eta_{k-1}} \frac{g_k^T \cdot g_k}{g_{k-1}^T \cdot g_{k-1}} . \quad (14)$$

For the first iteration of the method of CGs, i.e., $k=1$, and for all iterations that are multiples of $n+1$, where n is the dimension of the weight space, the momentum parameter in the CG algorithm is set to zero. Therefore, except in these cases the expressions used to update the weights at each iteration are equivalent in the two methods. The difference is that the CG method provides a systematic way to determine both the learning and momentum parameters which are dynamically updated at each iteration. This systematic approach eliminates the need for trial-and-error runs used in the BP algorithm to obtain reasonable values of η and α such that the algorithm has a good rate of convergence and avoids oscillations about the optimum.

IV. NUCLEAR POWER PLANT TRANSIENT CLASSIFICATION

To demonstrate the advantages of the method of conjugate gradients described in Sect. III, we train a feedforward multilayer neural network to classify transient events of a nuclear

power plant. The classification problem we select is the same one used by Reifman and Lee²⁷ in which the information-theoretic entropy minimax algorithm²⁸ was applied to generate patterns associated with various plant transients. In that formulation, the patterns were cast as "if... then ..." diagnostic rules that were modularized in a two-level hierarchical knowledge base structure for the anti-core melt safety function class.²⁹ The first level of the structure was represented by a set of rules that characterized challenges to each of the four safety functions, reactivity control, primary inventory control, primary pressure control, and primary heat removal, associated with the anti-core melt class. The second level of the structure was represented by four sets of rules each corresponding to one of the four safety functions of the first level. The second level rules characterized basic transient events whose occurrence would challenge the associated safety function. Here we concentrate on the classification of the three basic transient events whose occurrence would challenge the primary heat removal safety function. The three transients are described in Table I.

TABLE I. Three Transient Events that Challenge the Heat Removal Safety Function

Transient Event	Transient Description
1. Feedwater pump or booster pump trip	Causes an immediate trip of the selected feedwater pump or feedwater booster pump
2. Feed flow transmitter fails high	Causes the feedwater transmitter to transmit a higher flow signal to the control system
3. Turbine control valve fails closed	Causes the turbine control valve to be closed to the selected desired position

A. Transient Database

The inductive learning process of training a neural network requires the existence of a broad and representative database of examples. To construct a realistic transient database for the three transients described in Table I we used a subset of the MNP-2 simulator¹² runs

simulator representing all major systems of a PWR plant and the control room board instrumentation. Each one of the three transient events occurring separately, i.e., assuming single failures, was simulated on the MNP-2 simulator a dozen times. For each one of the 12 simulations of an event, a different combination of failure severity and initial conditions, e.g., different initial power level and fuel burnup, from a steady-state condition was used. Each initial condition of the power plant is associated with a different set of plant parameter values that permit the construction of a broad database. This wide-ranging representation of plant parameters in the database are reflected in classification patterns capable of identifying the transients independently of the state of the plant. Twenty plant parameters, $f_1, f_2, \dots, f_{20}$, were arbitrarily selected from a set of signals throughout the plant that are sent to the control room panel. These 20 plant parameters that were used to characterize the three transients are listed in Table II.

TABLE II. List of Plant Parameters

f_1	=	quench tank pressure (MPa)
f_2	=	quench tank water level (m)
f_3	=	reactor coolant system hot- and cold-leg A temperature difference (K)
f_4	=	reactor coolant system hot- and cold-leg B temperature difference (K)
f_5	=	pressurizer total pressure (MPa)
f_6	=	pressurizer water level (m)
f_7	=	pressurizer total pressure and water level ratio (MPa/m)
f_8	=	reactor power (% full power)
f_9	=	steam generator loop A water level (m)
f_{10}	=	steam generator loop B water level (m)
f_{11}	=	steam generator A exit pressure (MPa)
f_{12}	=	steam generator B exit pressure (MPa)
f_{13}	=	high-pressure turbine throttle pressure (MPa)
f_{14}	=	high-pressure turbine exhaust pressure (MPa)
f_{15}	=	pressure difference across feedwater loop A control valves (MPa)
f_{16}	=	pressure difference across feedwater loop B control valves (MPa)
f_{17}	=	feedwater loop A inlet head pressure (MPa)
f_{18}	=	feedwater loop B inlet head pressure (MPa)
f_{19}	=	letdown water flow (kg/s)
f_{20}	=	makeup water flow (kg/s)

Here we also select the first 40 s after the start of the transient as the time range for diagnostics.²⁷ Similarly, the dynamic behavior of each of the three transients over the 40-s time interval is obtained by converting the 20 plant parameters f_i ($i=1,2,\dots,20$) of Table II into the time rates of change $\Delta f_i/\Delta t$ at three arbitrarily selected discrete times, 10, 25, and 40 s. Thus, each simulation of a transient contributes three data points to the database. The entire transient database used contains 108 data points (three transient types x 12 simulations of each transient type x 3 time rate of change for each simulation) where each of the three transient events is represented by 36 entries.

Before the time rates of changes $\Delta f_i/\Delta t$ are used as input variables to the neural network they need to be scaled to a similar range in order to avoid unnecessary distortions in feature space which strongly affects the rate of convergence of ANNs. If the input variables differ by orders of magnitude, the weights may have to acquire arbitrarily large or small values which will require a large number of iterations before the sigmoid function provides the desired output. Here each input $\Delta f_i/\Delta t$, ($i=1,2,\dots,20$), to the network is normalized to a $[0, 1]$ range with respect to the maximum time rate of change $(\Delta f_i/\Delta t)_{\max}$ of the corresponding variable, where the values of $(\Delta f_i/\Delta t)_{\max}$ are obtained from the 108 data points of the transient database.

B. Neural Network Architecture

The overall definition of the neural network architecture is, in general, tightly coupled with the pattern recognition problem at hand. For instance, for nuclear power plant transient classification, the number of nodes at the first layer or input layer is generally equal to the number of inputs to the network and corresponds to the recorded plant parameters, while the number of nodes at the last layer or output layer generally corresponds to the number of possible transient types. There is not, however, a prescription for determining the number of nodes in the intermediate or hidden layers or even the number of hidden layers of a neural network. Lippman's⁷ interpretation of neural networks as classificatory devices that partition a multidimensional space spanned by the input variables provides some insight to the topology of the hidden layers. One hidden layer is sufficient to define convex regions in hyperspace where the nodes of the first hidden layer correspond to the decision boundaries or hyperplanes that

partition the multidimensional space. A second hidden layer is necessary only to define disconnected and nonconvex regions in the hyperspace where the nodes of the second hidden layer correspond to convex regions.

Since the results obtained by Reifman and Lee indicate that the transient database for the three transient events of Table I can be successfully partitioned by convex regions in the multidimensional space spanned by the 20 plant parameters of Table II, we chose a network with three layers, one input, one hidden, and one output layer. Our objective here is not to choose the optimal network architecture but to show that the proposed conjugate gradient method improves and accelerates network learning in a reasonably chosen architecture. The input and hidden layers have 20 nodes each, representing the normalized time rate of change for the 20 plant parameters illustrated in Table II, and the output layer has three nodes representing the three transient events illustrated in Table I. The target values t_{pj} , ($j=1,2,3$), of the output nodes for each one of the three transients is illustrated in Table III. For example, when the normalized time rate of change for the 20 plant parameters inputted to the network corresponds to the second transient, feed flow transmitter fails high, the desired output or target values t_{pj} of the network representing the transient are $t_{p1}=0.1$, $t_{p2}=0.9$, and $t_{p3}=0.1$ for nodes 1, 2, and 3, respectively. The values of 0.9 and 0.1 are used instead of the binary values of 1 and 0, respectively, because the weights would have to acquire very large absolute values for the sigmoid function in Eq. (1) to reach the extreme values of 1 and 0.⁶

TABLE III. Target Values for Three Output Nodes Corresponding to the Three Transient Events

Transient Event	Target Values for the Output Nodes		
	1	2	3
1. Feedwater pump or booster pump trip	0.9	0.1	0.1
2. Feed flow transmitter fails high	0.1	0.9	0.1
3. Turbine control valve fails closed	0.1	0.1	0.9

V. NEURAL NETWORK TRAINING AND DIAGNOSIS CAPABILITY

With the conjugate gradient algorithm discussed in Sect. III and the neural network architecture described in Sect. IV.B, we have trained the network to recognize patterns of the three transient events illustrated in Table III. We present the learning curves obtained by training the network with the proposed method and compare it with the standard BP algorithm using both batch and on-line presentation of the transients in the database. We discuss the effort we have made to validate the adequacy of the trained network by using it to diagnose transient events not used in training.

A. Neural Network Training

Given the transient database of 108 data points, the neural network described in Sect. IV.B was trained for numerous training session where at each one of the training sessions the network started from a different set of initial weights, i.e., a distinct position in weight-space. Because a single training session could take more than 6.0 h of CPU time in a SPARC Workstation 2, we limited our experiments to tens instead of hundreds of training sessions. Therefore, we present here representative learning curves instead of average results over the number of training sessions.

We compare the proposed CG method with the standard BP algorithm using both the "batch" version of BP, where the gradient is accumulated over the whole training set, and then weights are updated and "on-line" BP, in which the weights are adjusted after each pattern presentation. The comparisons are performed by constructing "learning curves" where the prediction error E in Eq. (4) is plotted against the number of learning cycles, where each cycle consists of the presentation of the 108 data points. For all three cases, batch-BP, on-line BP, and CG, presented here a convergence criterion $\epsilon = 0.03$ was used. For the two versions, batch and on-line, of the backpropagation algorithm the values of the learning parameter η and momentum parameter α parameters were fixed at 0.1 and 0.9, respectively, for all runs. These are the most frequently used values for η and α at the beginning of trial-and-error runs to determine their optimum values. We did not attempt to determine their optimum values, i.e.,

the one that trains the network in the smallest number of learning cycles, for each set of initial weights due to the large CPU time required for each training session.

Figure 2 shows representative learning curves obtained by training the neural network with the batch BP, the on-line BP, and the CG algorithms. By training the network with the proposed CG algorithm, convergence, i.e., $|t_{pj} - x_{pj}^{(3)}| < 0.03$ for all $j=1,2,3$ and $p=1,2,\dots,108$, is achieved after 804 learning cycles while for the batch BP and the on-line BP algorithms convergence is achieved after 27,160 and 36,275 learning cycles, respectively. Figure 2 makes clear the fact that both batch and on-line versions of the backpropagation algorithm, which are based on the method of gradient descent, make very slow progress as they get closer to the minimum. This is exactly where the quadratically convergent conjugate gradient method excels, since the weight surface is expected to become increasingly quadratic as the minimum is approached and the conjugate directions provide improved orientation to the search directions. Hence, the tighter the error criterion ϵ is, the better is the performance of the proposed conjugate gradient algorithm as compared with the standard backpropagation algorithm. For instance, in a different experiment where the error criterion ϵ was set to 0.01, the CG algorithm converged in 1,152 learning cycles, while neither one of the two versions of the BP algorithm converged within 100,000 cycles.

The CPU time per iteration is, however, larger for the CG method. As described in Sect. III, the conjugate gradient approach requires a one-dimensional search to obtain the optimum learning parameter or step size η_k , i.e., the step size η that minimizes the prediction error E in the conjugate direction μ_k at each iteration k . The one-dimensional search is performed here, as described in Sect. III.B, through the combination of the golden section algorithm with a cubic interpolation which requires additional function evaluations, i.e., forward passes through the network. However, the larger CPU time per iteration does not offset the advantages of the method of conjugate gradients due to the much larger reduction in the number of iterations. In the experiment illustrated in Fig. 2, using a SPARC Workstation 2, the proposed CG method converged in less than 20% of the CPU time used by the batch BP algorithm, and in less than 10% of the CPU time used by the on-line BP algorithm. On-line BP requires more CPU time than batch BP because the weights are updated at each pattern presentation. These results do not imply that we could not obtain optimum values of η and

Fig. 2. Representative Learning Curves Obtained by Training the Network with the Method of Conjugate Gradients and with Both Batch and On-line Backpropagation Algorithms

α that would allow the BP algorithm to converge in less CPU time, although in larger number of learning cycles, than the CG algorithm. But a number of BP trial-and-error runs would be necessary to obtain such optimum values which would result in a much larger cumulative CPU time than the CPU time for one CG run. Similar behavior in CPU time and number of iterations to converge was observed in other comparative experiments with the CG algorithm yielding improved convergence performance in comparison with the standard BP algorithm with tighter convergence criterion.

In Sect. II.B we described the phenomenon of premature saturation of the network nodes that is generally observed in neural network training with the BP algorithm. For a given set of initial weights, the BP algorithm may reach a region in weight-space where after a few first iterations the components of the momentum term $\alpha \Delta \mathbf{w}_{k-1}$ have larger absolute value and opposite sign than the components of the learning term $-\eta \nabla E(\mathbf{w}_k)$ causing the network to saturate. In our experiments we also observed this phenomenon for certain sets of initial weights. By training the network for the same problem, with the same parameters $\eta=0.1$ and $\alpha=0.9$ but a different set of initial weights, the batch BP algorithm experienced premature saturation of its nodes as illustrated by the learning curve in Fig. 3. The premature saturation of the network is indicated by the flat plateaus of the learning curve at relatively high values of the prediction error E . The flat plateaus of the learning curve are characteristic of the first and second stages of the saturation process of the output nodes as described in Sect. II.B. The flat plateaus also characterize the early phase of the third stage, recovery from saturation, when the various output nodes start to recover from their saturated activation levels of 0 or 1. The sharp decrease in E about 350 and after 28,000 learning cycles correspond to later stages of the recovery from saturation process when one of the output nodes has completely recovered from saturation. The cause for the formation of the flat plateaus in the learning curve are tightly dependent on the saturation of the output nodes $x_{pj}^{(l)}$ while the recovery of the network from saturation is influenced by the saturation levels of nodes in both the output and hidden layers of the network.

Figure 4 shows the activation level of the three output nodes $x_{pj}^{(3)}$ ($j=1,2,3$) of the output layer for one of the 108 patterns used to train the network corresponding to the learning curve

Fig. 3. Effects of the Premature Saturation of the Network Nodes in the Learning Curve Experienced by Training the Network with the Batch Backpropagation Algorithm

Fig. 4. Activation Level of the Three Output Nodes for a Feedwater Pump Trip Event Obtained by Training the Network with the Batch Backpropagation Algorithm

presented in Fig. 3. The pattern used represents a feedwater pump trip transient with target values $t_{p1}=0.9$, $t_{p2}=0.1$, and $t_{p3}=0.1$. At the very first learning cycles the activation level of output nodes 1 and 3 diverge rapidly to very small values which characterize the first stage, beginning of saturation, of the process of premature saturation of the network nodes. By the 20-th learning cycle nodes 1 and 3 are completely saturated with $x_{p1}^{(3)} \approx 10^{-8}$ and $x_{p3}^{(3)} \approx 10^{-6}$, which characterize the start of the second stage of the saturation process, while node 2 remains unsaturated. The degree of saturation of the output nodes is a function of the difference between $|\eta \nabla E(\mathbf{w}_k)|$ and $|\alpha \Delta \mathbf{w}_{k-1}|$, determined at the beginning of saturation stage, when some components of the gradient $\nabla E(\mathbf{w}_k)$ change sign. The larger the difference is, the larger is the level of saturation of the node. At the second stage of the saturation process, all the weights $w_{ji}^{(3)}$ connected to the two saturated nodes change very little causing the flat plateaus in Fig. 3. Around the 80-th learning cycle, the absolute value of the components of $\alpha \Delta \mathbf{w}_{k-1}$ associated with node 3 become smaller than the absolute value of the components of $\eta \nabla E(\mathbf{w}_k)$ allowing the activation level of node 3 to start its recovery from saturation, i.e., the start of the third stage, as indicated in Fig. 4. By the 350-th learning cycle, node 3 is completely unsaturated causing a sharp decrease in the value of the prediction error E . Similarly, the recovery for node 1 starts around 6,000 learning cycles and is completed after 28,000 learning cycles where the activation level of all three output nodes are unsaturated and are close to their target values. However, because the convergence rate of the BP algorithm decreases the closer it gets to the minimum, it only converges after 48,392 learning cycles. For the other 107 patterns we observed a similar behavior. Node 2 was never saturated, node 3 was saturated until around 350 learning cycles, and node 1 was saturated until around 28,000 cycles.

The undesirable effects of the premature saturation of the network nodes could be eliminated from the standard BP algorithm through minor modification in the algorithm. One possible modification involves the selection of a small step size η at the early stages of the algorithm, which could then be increased later in order to speed up convergence. By selecting a small η at the early stages of the algorithm we would force the components of $\Delta \mathbf{w}_k$ to be small in comparison to the components of $\mathbf{w}_k$ which would avoid the premature saturation. Another possible modification involves the suppression of the momentum parameter α , i.e., set $\alpha=0$, at the early stages of the algorithm, which could later be set to 0.9 or other value in order to speed

up convergence. By setting $\alpha=0$ at the early stages of the algorithm we force the components of $\Delta \mathbf{w}_k$ to update $\mathbf{w}_k$ in the negative gradient direction $-\nabla E(\mathbf{w}_k)$, i.e., the "correct" direction, when the components of $\Delta \mathbf{w}_k$ are of the same order of magnitude as the components of $\mathbf{w}_k$, and thus avoiding premature saturation. These two distinct modifications were implemented for the same initial set of weights used in Fig. 3 and succeeded in avoiding the premature saturation of the network nodes. In the first modification, we selected $\eta=0.01$ instead of $\eta=0.1$ for the entire learning cycle. We observed no premature saturation but because of the small step size the algorithm did not converge within 100,000 learning cycles. In the second modification, we set $\alpha=0.0$ for the first 20 learning cycles after which we set $\alpha=0.9$. For this case, the batch BP algorithm converged in 24,921 learning cycles. In spite of the fact that these two possible minor modifications in the standard BP algorithm could be used to avoid premature saturation of the network nodes, they also add more complexity in the heuristic selection of η and α .

The phenomenon of premature saturation of the network nodes is not experienced in neural network training with the method of CGs. By calculating the learning parameter η_k that minimizes $E(\mathbf{w}_k + \eta \mu_k)$ along the conjugate direction μ_k , the method of CGs forces the learning term $-\eta_k \mathbf{g}_k$ to be perpendicular to the momentum term $\alpha_k \Delta \mathbf{w}_{k-1}$ at every iteration. The orthogonality of these two terms causes the change in weights $\Delta \mathbf{w}_k$ in Eq. (13) to always have its projection along the gradient $\mathbf{g}_k$ in the negative direction, i.e., the direction which minimizes the prediction error E ; thus precluding saturation. Hence, the method of CGs not only expedites neural network training but also avoids the premature saturation of the network nodes. Figure 5 shows the learning curve obtained with the method of CGs for the same problem illustrated in Fig. 3. The prediction error E decreases monotonically and convergence is achieved after 725 learning cycles. Figure 6 shows the activation level of the three output nodes $x_{pj}^{(3)}$ ($j=1,2,3$) of the output layer for the same pattern used in Fig. 4. With the CG algorithm the activation levels of the output nodes never become saturated and after 300 cycles the output nodes are already close to their target values.

Figure 7 shows the values of the learning η_k and the momentum α_k parameters in Eq. (14) calculated by the method of CGs for the 725 training cycles required to train the network

Fig. 5. Learning Curve Obtained by Training the Network with the Method of Conjugate Gradients

Fig. 6. Activation Level of the Three Output Nodes for a Feedwater Pump Trip Event Obtained by Training the Network with the Method of Conjugate Gradients

Fig. 7. The Learning Parameter η_k and Momentum Parameter α_k Estimated by the Method of Conjugate Gradients

with the initial set of weights used in Fig. 5. These values are in contrast with the fixed values of $\eta=0.1$ and $\alpha=0.9$ used in the BP algorithm. In this case, the values of the momentum parameter α_k oscillates at each iteration of the CG algorithm with a baseline around 1.0. The values of the learning parameter η_k also vary throughout the training process starting at small values and increasing towards the end of training. Perhaps a more meaningful parameter to present, as illustrated in Fig. 8, is the effective learning parameter or step size η_{eff} defined as the product of the learning parameter η_k and the norm of the conjugate direction $|\mu_k|$. The effective step size is the actual amount that the weight w_k moves in weight-space along the conjugate direction μ_k . The value of the effective step size decreases at the later stages of the training process which indicates the necessity to better account for the quadratic nature of the prediction error function near the minimum. This decrease in η_{eff} also avoids the oscillatory behavior that can be observed in the BP algorithm if the value of the fixed learning parameter η is too large causing overshooting of the minimum.

B. Diagnosis Capability

The validation of the diagnosis capabilities of the trained neural network with the conjugate gradient method is performed through a blind test. Two simulated transient events, one representing the feedwater pump trip and one representing the feed flow transmitter failure, which were not used to train the neural network were used for the blind test. The two events were simulated with randomly selected initial conditions and failure extent which allow for the verification of the degree of context independence or interpolation of the trained neural network.

As the blind test was also expected to demonstrate the time-varying diagnostic ability of the static feedforward network, the test was designed to simulate on-line diagnostics. At every second, the 20 plant parameters of Table II are converted into their corresponding time rate of change which are then normalized before being fed to the 20 nodes of the input layer of the network to obtain the activation levels for the three output nodes. Figure 9 shows the network activation levels for the three output nodes during the first 40 s of the feedwater pump motor trip transient. Within 20 s into the transient the network is capable of correctly identifying the event, which is evidenced by the fact that after 20 s the activation levels for the three output

Fig. 8. The Effective Step Size or Learning Parameter η_{eff} Used by the Method of Conjugate Gradients

Fig. 9. Activation Level of the Three Output Nodes for a Feedwater Pump Trip Event Not Used in Training

nodes are very similar to the target values t_{pj} ($j=1,2,3$) of 0.9, 0.1, and 0.1, respectively, representing a feedwater pump motor trip as indicated in the first row of Table III. Actually, after 5 s into the transient the trained neural network already provides good evidence of the occurrence of the correct transient.

A mild deviation from the expected results is observed between 10 and 20 s into the transient where the activation level of the first output node decreases as the activation level of the second output node increases. The activation level of the first node decreases by the same amount that the second node increases such that the addition of the two activation levels remain constant at 1.0 and the addition of the three activation levels remain constant at 1.1. This behavior of the network allows us to interpret the activation levels of the output nodes as membership functions used in fuzzy logic.^{30,31} Since each one of the three output nodes is associated with one of three possible transient events, the activation level of each node corresponds to the membership function of a given input vector of plant parameters to the corresponding transient event represented by the node. For example, the activation level of the first output node represents the membership function of a given input vector of plant parameters to the feedwater pump trip transient event. The closer the activation level of a node is to 0.9, the larger is the certainty about the occurrence of the corresponding transient event. Hence, the behavior of the output nodes of the diagnosis problem in Fig. 9 can be interpreted as in the fuzzy logic approach for transient identification,^{30,31} where the increase in the membership function for one transient would automatically cause the membership of one or both remaining transients to decrease.

The results of the blind test for the feed flow transmitter transient is illustrated in Fig. 10. Within a couple of seconds into the transient the trained neural network is capable of identifying the transient as evidenced by the close agreement between the target values of the three output nodes presented in the second row of Table III and the results presented in Fig. 10. The activation levels for output nodes 1 and 3 are approximately 0.1 while the activation level for output node 2 is approximately 0.9 as expected. Similar blind test results were obtained when the network was trained by either batch or on-line versions of the backpropagation algorithm allowing us to conclude that the diagnosis capabilities of the network seemed to be independent of the algorithm used for training.

Fig. 10. Activation Level of the Three Output Nodes for a Feed Flow Transmitter Event Not Used in Training

VI. SUMMARY AND CONCLUSIONS

The developed CG algorithm used for neural network learning of nuclear power plant transient classification eliminates key drawbacks of the standard BP algorithm. First, the rate of convergence of the proposed method is much greater than the standard BP algorithm and it reduces both the number of iterations and the CPU time required to train the network to recognize patterns of the transient events. Unlike the standard BP algorithm, the convergence rate of the proposed method does not necessarily decrease the closer it gets to the optimum solution. Second, the method of CGs provides a systematic mechanism to determine both the learning and the momentum parameters which are dynamically updated at each learning cycle. This eliminates the sequence of trial-and-error runs necessary in neural network training with the BP algorithm to obtain optimum values for these two parameters. Finally, the method of CGs is less sensitive to the initial set of weights than the standard BP algorithm since the phenomenon of premature saturation of the network nodes is not observed in the proposed method. This allows for smaller variations in the number of learning cycles and CPU time required for training a neural network when different initial set of weights are used.

The good convergence characteristics of the conjugate gradient method substantially decrease the effort involved in the time-consuming task of training ANNs. This advantage of the CG method is most noticeable and important for problems where the network architecture consists of a large number of nodes, i.e., when the optimization problem has a large dimension, the number of input-output training patterns is large, and a tight convergence criterion is desired. The proposed method, however, requires a one-dimensional search which can affect the overall performance of the algorithm significantly and should be considered carefully. Considerations should include the trade-off between accuracy of the one-dimensional search, which could influence the convergence of the overall method, and the CPU time involved in the search, which could account for the major portion of the total CPU time. Here, a balance between accuracy and CPU time in the one-dimensional search is achieved through the combined use of the iterative golden-section algorithm with an analytical cubic interpolation. In this report we also describe the reasons for the premature saturation of the network nodes which often occurs with training via the BP algorithm and provide suggestions for modifying the algorithm to avoid this undesirable effect.

Neural network training with the method of conjugate gradients presented here is of course not limited to the classification of nuclear power plant transient events. The method is general and could be used to expedite the training process of other applications of ANNs in nuclear power operations. Future work should attempt to apply the CG method to improve learning of recurrent neural networks⁴ which try to account for time-varying nonlinearities. This is a rather challenging problem since recurrent networks feed the output of each node back to itself either directly or through other nodes which increases the difficulty in training this type of networks.

ACKNOWLEDGMENTS

The first author was supported by the U.S. Department of Energy, Nuclear Energy Program.

REFERENCES

1. M. S. Roh, S. W. Cheon, and S. H. Chang, "Power Prediction in Nuclear Power Plants Using a Back-Propagation Learning Neural Network," Nucl. Technol., **94**:270 (1991).
2. B. R. Upadhyaya and E. Eryurek, "Application of Neural Networks for Sensor Validation and Plant Monitoring," Nucl. Technol., **97**:170 (1992).
3. E. B. Bartlett and R. E. Uhrig, "Nuclear Power Status Diagnostics Using an Artificial Neural Network," Nucl. Technol., **97**:272 (1992).
4. A. G. Parlos, A. F. Atiya, K. T. Chong, and W. K. Tsai, "Nonlinear Identification of Process Dynamics Using Neural Networks," Nucl. Technol., **97**:79 (1992).
5. Z. Guo and R. E. Uhrig, "Use of Artificial Neural Networks to Analyze Nuclear Power Plant Performance," Nucl. Technol., **99**:36 (1992).
6. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," Parallel Distributed Processing: Explorations in the Microstructure of Cognition, I:319, D. E. Rumelhart and J. C. McClelland, Eds., MIT Press (1986).
7. R. P. Lippman, "An Introduction to Computing With Neural Networks," IEEE ASSP, **4**(2):4 (1987).

8. R. Fletcher and C. M. Reeves, "Function Minimization by Conjugate Gradients," The Computer Journal, 7:149 (1964).
9. D. A. Wismer and R. Chattergy, Introduction to Nonlinear Optimization: A Problem Solving Approach, Elsevier North-Holland, Inc. (1978).
10. A. Rezgui and N. Tepedelenlioglu, "The Effect of the Slope of the Activation Function on the Back Propagation Algorithm," Proc. IJCNN Intl. Conf. Neural Networks, I:707, Washington, D. C. (January 15-19, 1990).
11. J. R. Chen and P. Mars, "Stepsize Variation Methods for Accelerating the Back-Propagation Algorithm," Proc. IJCNN Intl. Conf. Neural Networks, I:601, Washington, D. C. (January 15-19, 1990).
12. Midland Plant Simulator Units I and II Manual, Consumers Power Company, Midland, MI (1984).
13. R. L. Watrous, "Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization," Proc. IJCNN First Intl. Conf. Neural Networks, II:619, San Diego, CA (June 21-24, 1987).
14. R. A. Jacobs, "Increased Rate of Convergence Through Learning Rate Adaptation," Neural Networks, 1:295 (1988).
15. A. A. Minai and R. D. Williams "Acceleration of Back-Propagation Through Learning Rate and Momentum Adaptation," Proc. IJCNN Intl. Conf. Neural Networks, I:676, Washington, D. C. (January 15-19, 1990).
16. J. Higashino, B. L. De Greef, and E. H. J. Persoon, "Numerical Analysis and Adaptation Method for Learning Rate of Back Propagation," Proc. IJCNN Intl. Conf. Neural Networks, I:627, Washington, D. C. (January 15-19, 1990).
17. T. Samad, "Backpropagation Improvements Based on Heuristic Arguments," Proc. IJCNN Intl. Conf. Neural Networks, I:565, Washington, D. C. (January 15-19, 1990).
18. E. D. Dahl, "Accelerated Learning Using the Generalized Delta Rule," Proc. IJCNN First Intl. Conf. Neural Networks, II:523, San Diego, CA (June 21-24, 1987).
19. R. Battiti, "Optimization Methods for Back Propagation: Automatic Parameter Tuning and Faster Convergence," Proc. IJCNN Intl. Conf. Neural Networks, I:593, Washington, D. C. (January 15-19, 1990).
20. N. Baba, "A Stochastic Optimization Approach for Training the Parameters in Neural Networks," Proc. of the Intl. Workshop on Computationally Intensive Methods in Simulation and Optimization, 55, Laxenburg, Austria (August 23-25, 1990).

21. D. M. Himmelblau, "Introducing Efficient Second Order Effects into Back Propagation Learning," Proc. IJCNN Intl. Conf. Neural Networks, I:631, Washington, D. C. (January 15-19, 1990).
22. S. Becker and Y. Le Cun, "Improving the Convergence of Back-Propagation Learning with Second Order Methods," Proc. of the 1988 Connectionist Models Summer School, Carnegie-Mellon University, 29, San Mateo, CA (1989).
23. R. S. Scalero, and N. Tepedelenlioglu, "A Fast Training Algorithm for Neural Networks," Proc. IJCNN Intl. Conf. Neural Networks, I:715, Washington, D. C. (January 15-19, 1990).
24. F. Pallmieri and S. A. Shah, "Fast Training of Multilayer Perceptrons Using Multilinear Parametrization," Proc. IJCNN Intl. Conf. Neural Networks, I:696, Washington, D. C. (January 15-19, 1990).
25. E. H. Haug and J. S. Arora, Applied Optimal Design, John Wiley & Sons (1979).
26. D. G. Luenberger, Linear and Nonlinear Programming, Addison-Wesley (1984).
27. J. Reifman and J. C. Lee, "Reactor Diagnostics Rule Generation Through Statistical Pattern Recognition," Nucl. Sci. Eng., 107:291 (1991).
28. R. Christensen, "Entropy Minimax Multivariate Statistical Modeling - I: Theory," Intl. J. General Systems, 11:231 (1985).
29. W. R. Corcoran, N. J. Porter, J. F. Church, M. T. Cross, and W. M. Guinn, "The Critical Safety Functions and Plant Operations," Nucl. Technol., 55:690 (1981).
30. J. C. Lee, P. Rank, J. Reifman, E. Hawkes, and D. K. Wehe, "Detection of Abnormal Transients in Nuclear Power Plants," Proc. of the Brazilian Meeting on Reactor Physics and Thermal Hydraulics, 105, Sao Paulo, Brazil (September 17-28, 1991).
31. P. Rank, J. Reifman, and J. C. Lee, "Fuzzy Diagnosis of Multiple Failures in Nuclear Power Plants," Trans. Am. Nucl. Soc., 64:176, San Francisco, CA (November 10-14, 1991).