1992

# The importance of input variables to a neural network fault-diagnostic system for nuclear power plants

Terry L. Lanc
*Iowa State University*

DOE/ER/75700--T1



# The importance of input variables to a neural network fault-diagnostic system for nuclear power plants

DOE/ER/75700--T1

DE93 010302

by

Terry L. Lanc

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Mechanical Engineering
Major: Nuclear Engineering

Signatures redacted for privacy

FG02-

Iowa State University
Ames, Iowa
1992

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

## ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to the Duane Arnold Energy Center, especially to Don Vest, for donating many hours of personal and computer time at their nuclear training facility and for data acquisition from their operator simulator. Without their cooperation, this thesis could not have occurred. I want to thank the Electric Power Research Center (EPRC) and Power Affiliates Research Program, directed by John Lamont, for financial assistance through the years of graduate school. I wish to thank the Program of Nuclear Engineering for endowing me the Institute of Nuclear Power Operations (INPO) Fellowship my first year at Iowa State University. Thanks to the excellent faculty and students of the Nuclear Engineering Program for making my study quite enjoyable. Particular thanks go to Rodney Schuler for his eagerness to help me become acquainted with computer systems. Also, I wish to thank John Adams for assisting me to familiarize myself with nuclear power plant systems. In addition, I thank Dr. Danofsky and Dr. Vittal for being on my program committee. Most importantly, I wish to thank my mentor and friend, Dr. Eric Bartlett, for his patience and cheery enthusiasm that always inspired me to perform well. Finally, I want to acknowledge my friends and family for their support.

# CHAPTER 1. INTRODUCTION

## Nuclear Power Plant Safety

This thesis explores safety enhancement for nuclear power plants. Emergency response systems currently in use depend mainly on automatic systems engaging when certain parameters go beyond a pre-specified safety limit. Often times the operator has little or no opportunity to react since a fast scram signal shuts down the reactor smoothly and efficiently. These accidents are of interest to technical support personnel since examining the conditions that gave rise to these situations help determine causality. In many other cases an automated fault-diagnostic advisor would be a valuable tool in assisting the technicians and operators to determine what just happened and why.

In slower operational transients the operator has more time to react to plant conditions. Operators are trained intensively to react a certain way to such conditions, as prescribed in procedural instructions. This is augmented by a basic knowledge of system dynamics that allows them to perform corrective actions that are necessary and proper for mitigation. For such operational transients a fault-diagnostic advisor would be a valuable tool in assisting operators' accident recognition more quickly. Furthermore, the advisor would provide time for the operator to make the required verification of the diagnosis [34].

The results of this thesis are part of an ongoing project at Iowa State University to develop an artificial-intelligence (AI), fault-diagnostic system capable of detecting and classifying operational transients at nuclear power plants. The ultimate goal of the project is to develop and deploy a prototype diagnostic advisor for testing and use in the control room or technical support center at Iowa Electric Light and Power Company's Duane Arnold nuclear power station at Palo, Iowa. The AI fault-diagnostic system designed and analyzed here may offer extra operational safety by early warning and diagnoses of abnormal plant conditions. A fast accident diagnostic system could provide additional time for the operators to confirm its diagnoses and take appropriate actions toward correcting the potential problem.

## Neural Networks

The artificial intelligence technique called artificial neural networks (ANNs, neural networks) are used in this thesis to make a fault-diagnostic system. Perhaps the best definition of neural networks may be provided by Robert Hecht-Nielsen [26]. He describes a neural network as a "parallel distributed information processing structure" comprised of many process elements, commonly called nodes. Each node is connected to at least one other node by signal channels. The node provides an output signal which is fanned out to several nodes as inputs. Neural networks offer a very significant advantage over current expert systems. Whereas an expert system requires that one understand all input variables and how they interact with one another to determine a particular outcome, the ANN does not require such explicit knowledge to be known beforehand and inserted into the system. While an expert system demands that every possible condition and outcome be explored, the ANN merely

learns inductively (without knowledge a priori) outcome classifications as a function of input variables based on some training set. The ANN learns the correct response through training from presented examples and is able to generalize this knowledge. Generalization is the ability to quantitatively estimate certain characteristics or features of a phenomenon never before encountered based on similarities with things previously known [39]. Accurate diagnosis becomes difficult when the monitoring systems are not providing clear and accurate information. Neural networks have the ability to provide correct classification even under such conditions as noisy inputs and intermittent or degraded monitors. ANNs' generalization capabilities are especially useful for determining a solution for accident recognition. It is not always possible to predict all plant system reactions to a particular accident condition. Furthermore, there are a great many of such accidents that have never before occurred [5]. Thus, they need to be modeled by a computer simulation, such as at the facility at the Duane Arnold Energy Center (DAEC). All possible scenarios cannot be anticipated, nor does time and resources allow for their exploration. For the purposes of this research and the illustration of the methods presented, three plant failures and one normal operational mode are investigated.

As previously indicated, ANNs do not require knowledge to be presented and incorporated into them explicitly. A rule-based diagnostic system, on the other hand, is more clumsy since it demands insertion of information directly. Also, too much computer time and programming resources may be required for an expert system to progress through the many possible decision paths. A rule-based system must answer many questions about the condition of the plant, and on the basis of each individual answer (usually only a yes-no statement), it branches closer to the final

desired classification [34]. Designs for elaborate systems tend to become overly complex. While the problem for the computer is merely run-time, the programmer has the unfortunate problem of inserting the required knowledge correctly and understanding why each variable is important and how it may affect the final outcome. An artificial neural network does not require the user to understand why input variables are important, nor how they interact; the user merely knows that particular inputs ARE important. In the past, we mimicked reality by creating models. Then we simulated our models by computer programs that produced answers we desire. With the new approach offered by neural networks, the network itself is an unknown, black box, model that produces desired responses. Essentially, modeling is accomplished by the ANN through its ability to internalize general common features among data. A further advantage for ANNs is they can process inputs and outputs on a continuous scale rather than a binary yes-no or fuzzy multi-level approach. This allows for input variables to consist of actual control room meter readings. It also allows output decisions to be more quantified. The ANNs respond quickly to inputs once trained. Another result of ANNs is that they can separate out features from noisy, or clouded, data, just as the human brain does. This makes them noise- and fault-tolerant. That is, their classification abilities degrade gracefully even when the inputs are clouded with external noise or when an input (for example, a radiation monitor) is giving an improper or faulty response. [4] [5]

## Statement of Objective

The objective of the present work is to investigate the theory and application of the importances of the input variables for neural networks, using three nuclear

power plant failures and one normal mode of operation as a data base. Specifically, computer points or actual control room meter readings are of interest for use as inputs to the neural networks. The specific goal is to determine a means of identifying the most significant input variables necessary to make correct classification of abnormal operating conditions. Several methods are used to accomplish this. First, an intuitive approach is used, based upon practical and technical knowledge of reactor and plant system dynamics which gives a large number of possible variables. Second, a statistical approach is used to reduce the number of variables, based on each variable's linear correlation to the output classifications. This procedure provides three arbitrary input variable sets that are used as the recall sets for three artificial neural networks of various sizes, with 20 inputs, 33 inputs, and 50 inputs. Third, an artificial neural network learning algorithm using backpropagation is developed and trained to some arbitrary level of error. This involves training a neural network with several exemplars taken from the recall set. Then one determines in which patterns large errors are occurring and retrains the network with additional exemplars. Then, as a check on network importance, another algorithm calculates the derivative importances of the variables relative to the output nodes for the fully-trained neural network. The second and third methods are repeated for various statistical levels of correlation, and comparisons are presented in Chapter 4.

## Results

It is found that for the three distinct accident conditions studied the neural network using fewer linearly correlated input variables trains faster with fewer exemplars than the larger networks. The smaller network seems to perform better than

the larger ones, by generalizing more. Additionally, it is found that the derivative importance of an input node for a layered feed-forward neural network does not have an obvious relationship to the linear statistical correlation. These conclusions are justified by examining graphs of recall performances presented in Chapter 4 and are discussed in Chapter 5. It is suggested that the derivative importances for a neural network are a function of the learning procedure and may be sensitive to initial conditions, such as the initial weight vector.

# CHAPTER 2. ARTIFICIAL NEURAL NETWORKS

## Introduction

An Artificial Neural Network (ANN) may be visualized as a model of a living brain. In fact, the Japanese first coined the phrase "natural intelligence" in their Sixth Generation project conference on neurocomputing systems. They described computers based on models of the living physiological brain rather than on models of the traditional serial Von Neumann type [8]. While the concept of neurocomputing is well over 30 years old, it has not come into wide-spread attention until the last decade. The history of neural networks may be traced as far back as 1890, when the American psychologist William James had great insight about the interconnectiveness found within the neurons of the living brain [31]. He describes much that remains relevant even a century later. For instance, in Chapter 9 of his book *Psychology (Briefer Course)* he describes a model of association that is nearly the same as used in modern associative neural networks. He explains a general rule of association, which is a correlation learning rule, and formulates a summing rule for brain activity, which is nearly identical to the sum of inputs weighted by connections to an artificial node used in neural networks. The next most important historical contribution toward neural networks may be the "McCulloch-Pitts" neuron, formulated by Warren S. McCulloch and Walter Pitts in 1943 [36]. They published a paper

on the physiology that governs neuron activity. In it, they develop a mathematical model of a binary neuron that receives inputs from other exited neurons, with identical weights, and compares the sum to a threshold to determine the state of the neuron. These examples demonstrate that neural networks have a colorful history and long association with the neuro-physiologists. The most famous neural network might be Widrow's Adaptive Linear Element (ADALINE) developed in 1963 as a simple bin sorter [45]. However, until the 1980s artificial neural networks had been considered interesting but impractical. In 1982, J.J. Hopfield sparked a resurgence of interest in neural networks when he discussed how such interconnected neurons can have collective computational properties, with a distributed memory [29].

A neural network is made of simple computational elements (nodes) linked to one another by variable weights, much as neurons in a living brain [35]. The layered feed-forward ANN consists of a several layers of nodes, with weights interconnecting the nodes between successive layers. (See Figure 2.1.) The nodes are connected to all nodes in the layer above and below but not to its neighbors in the same layer. For a three-layer network, the first layer is input nodes, which are inactive, the second layer consists of "hidden" nodes, because they do not have direct contact with physical input/output environment, and the third layer is output nodes. The design of network architecture is somewhat arbitrary, except that the number of inputs and outputs is fixed. The user may specify internal parameters such as the transfer function, the learning rate, or the number of hidden nodes or hidden layers. These features will be discussed in more detail in Section 3.2 below. Nodes are typically analog. The simplest nodes take a sum of their inputs multiplied by the associated interconnective weight. A node's output is the result of a non-linear transfer operation on the

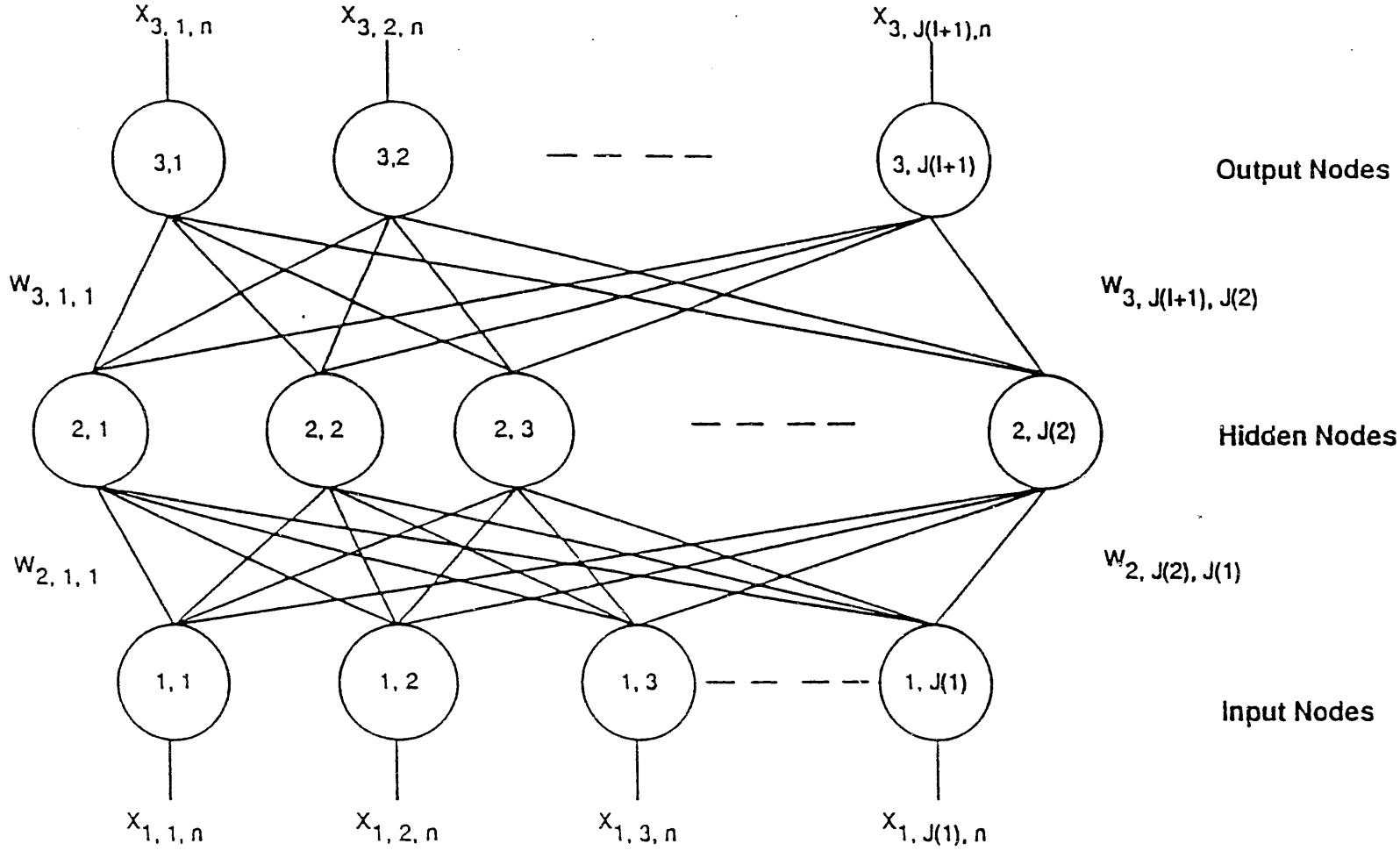


Figure 2.1: An example 3-layer neural network

weighted sum of the inputs to the node [40]. (See Figures 2.2 and 2.3.) Thus, the input is the dot product of the input and weight vectors. The transfer function used is normally fixed as some sigmoidal function defined so that the output of a node is normalized between 0 and 1. A transfer function is essentially a mathematical operation that determines if the process element (node) will output a signal [8]. One typical transfer function used by neural network researchers is the arctangent. Since each node is capable of interdependent and simultaneous calculation, the network, when trained, becomes computationally very fast [35].

## Training a Neural Network

Training a neural network involves presenting the network with a set of patterns, computing the outputs, and comparing the output values to the desired response values. When example patterns are presented, the ANN produces some output signal which is then compared to the desired response signal. The weights are then systematically changed to improve network performance. These weights are normally randomized at the beginning of a training problem. The process of presenting examples and adjusting weights is repeated until the network performs desirably [45]. A common measure of network performance is the root mean square (RMS) error between desired and actual outputs. It is desirable to bring the RMS error, or cost, down as small as possible, a global minimum. However, using the gradient slope descent methods of backpropagation, a neural network tends to become lodged in local minima [5] [28]. If the output for a given node is relatively more significant as an input to a node in the next layer, the nodal output is assigned a greater weight. As the training problem becomes more complex, the training process becomes

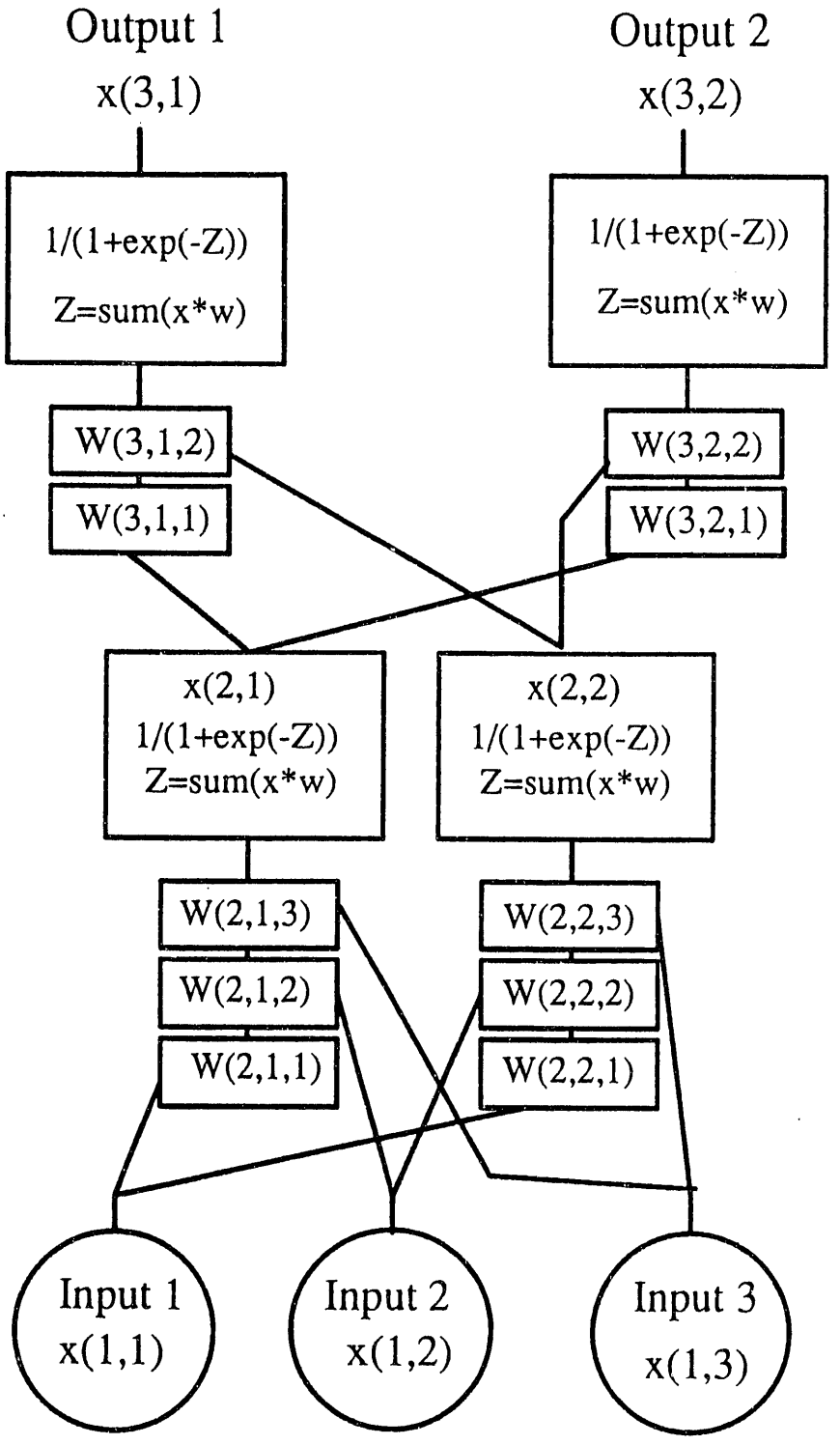


Figure 2.2: Feed-forward activation for a neural network

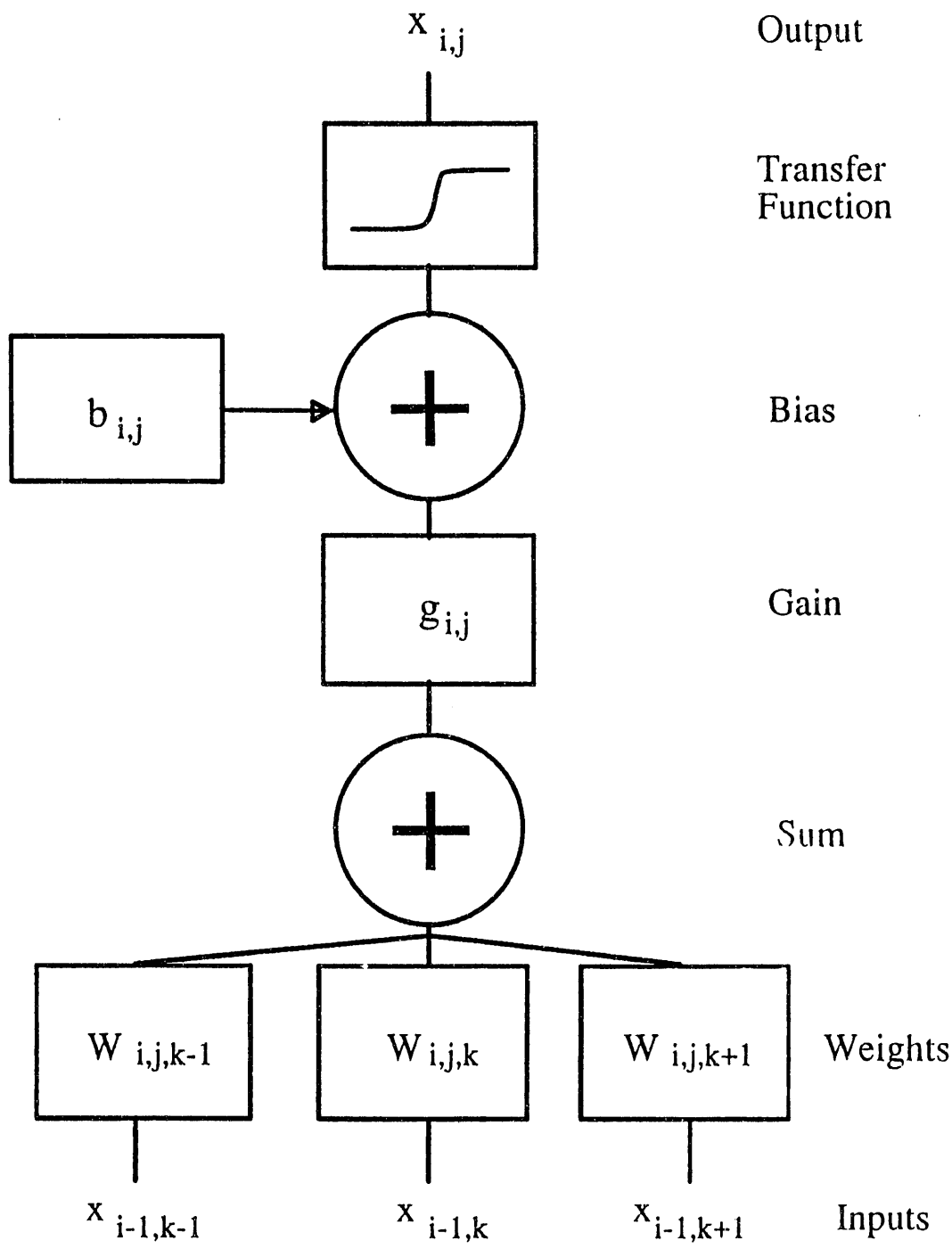


Figure 2.3: An enlarged node

increasingly more complex. Determination of the near optimum weights connecting each node can be achieved by various training schemes. The most common training scheme is employed by propagating the errors backward to the nodes. Such a scheme is described in the next section.

## Theory of Backpropagation: Delta Rule

### Learning in a neural network

Backpropagation is a supervised learning method for determining the nodal interconnection weights for a layered neural network using the gradient slope descent methods mentioned above. Supervised learning involves presenting a pattern to the network and comparing the output values produced to the desired response values. Then, the errors are computed in order to make appropriate changes in certain parameters of the network architecture, particularly the weight vector. The method used to achieve the appropriate changes takes the form of some learning law. The learning law specifies an incremental change to the weight vector that will be used in an iterative fashion that allows the network to converge to a usable weight vector [12]. One of the learning laws used by backpropagation is called the Delta rule, which employs a Least Mean Squared (LMS) error cost function. The Delta learning rule was first developed by Bernard Widrow and Ted Hoff at Stanford University, California, in 1960. Their network was called ADALINE, which stood for ADAptive LINear Element [45]. The Delta rule modifies the processing nodal weights by the following equation, which is called the Delta rule:

$$\mathbf{W}_{new} - \mathbf{W}_{old} = \beta E \frac{\mathbf{X}}{|\mathbf{X}|^2} \tag{2.1}$$

where **X** and **W** are the input and weight vectors, respectively; and $\beta$ and E are the learning rate and error for a node, respectively. (See Figure 2.4.)

This rule attempts to minimize the aggregate statistical least mean square error for the network's output layer. The partial errors are squared so that large errors are weighted more than small errors over the output nodes and over the total number of patterns. A change made to the weight vector using this rule is in accordance with the negative gradient of the weight vector. This will be discussed in more detail in Section 3.2 of this chapter.

The RMS error for a neural network is defined as the root of mean squared differences between the desired output responses and the actual responses of the output nodes to the the input pattern. When the RMS error is minimized, the weight configuration is said to be a near-ideal weight vector. Learning by the method of the Delta rule is often referred to as "downhill" learning since one takes a step in a direction that will reduce the overall error. The Delta rule causes the weight vector to move along the negative gradient, and in this way one tends to travel "downhill." The effectiveness of the Delta rule can be demonstrated by some relatively simple mathematics which follows from Newton's Method. Newton's method is characterized by the use of several derivatives of the function $f(x)$ to obtain an estimate of the zero of $f(x)$, $x^*$ [46]. A Taylor series expansion of $f(x)$ about the value $x = a$ yields [41]

$$f(x+dx) = f(a) + f'(a)(x-a) + f''(a)\frac{(x-a)^2}{2} + ... + f^{(n)}(a)\frac{(x-a)^n}{n!} + ... \tag{2.2}$$

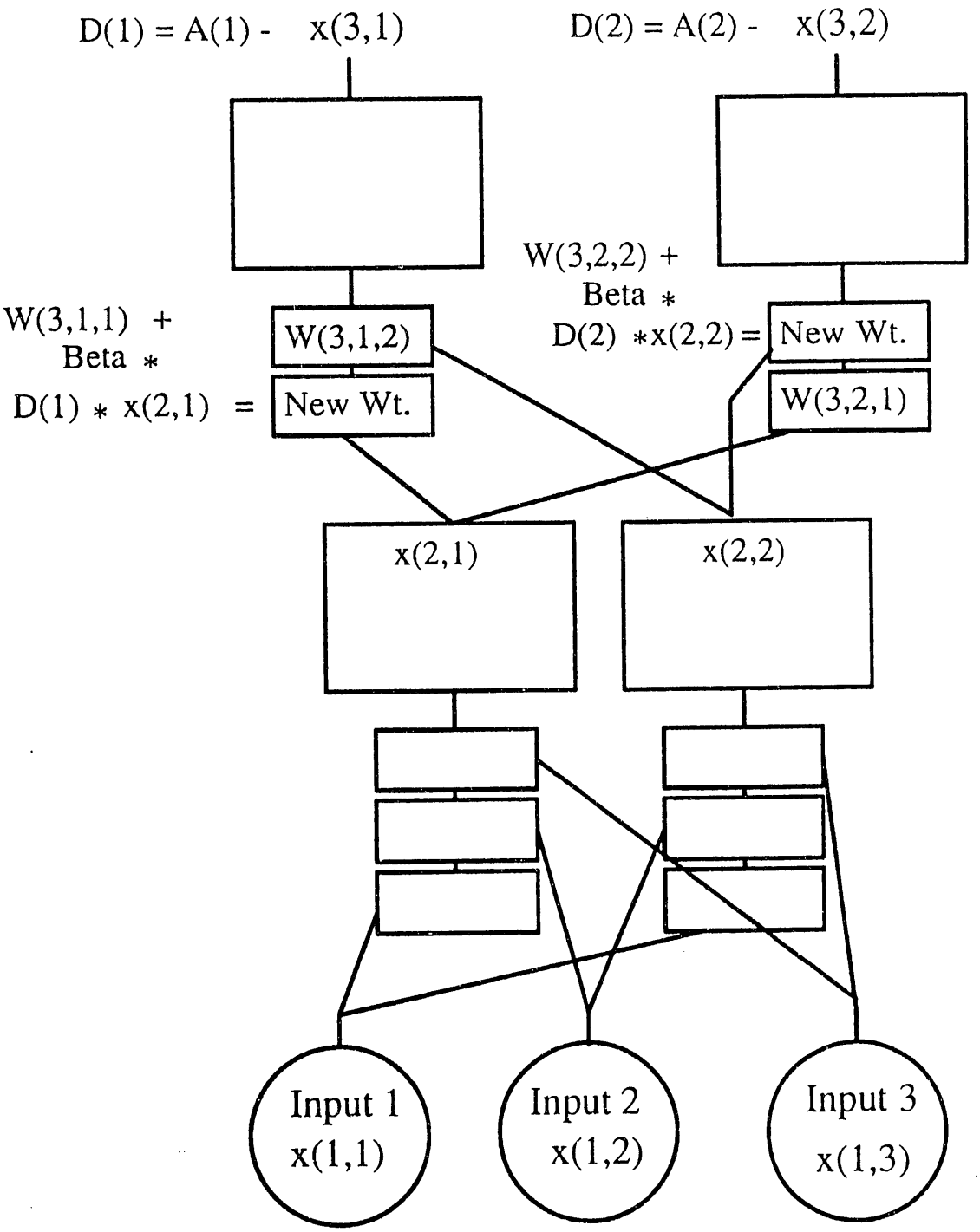


Figure 2.4: The delta rule for backpropagation

Newton's method may be stated: If $f$ is a differentiable function and $x^*$ is a real zero of $f$, and if $x$ is an approximation to $x^*$, then the next approximation is

$$x_{n+1} = x_n - \beta \frac{f(x_n)}{f'(x_n)}. \tag{2.3}$$

It should be realized that Newton's method fails to guarantee a better approximation to $x^*$ for each succeeding $n$. However, this method has been studied by many mathematicians and is well-known. It works. Attention must be paid in choosing the initial estimate $x_0$ and the step size $\beta$, because a large value of $\beta$ may cause succeeding estimates of $x^*$ to become chaotic. The method may then fail to converge on a solution [46]. When the function $f$ contains low order partial derivatives, which is usually assumed, at any chosen point $x$ the gradient $G(x)$ is defined below. For the case of the neural network, we seek a minimum of $C(\mathbf{W})$. Finding a minimum with Newton's method can be shown like this:

$$G = \frac{\partial C}{\partial W} \tag{2.4}$$

and $H$ is the Hessian matrix defined

$$H_{ij}(W) = \frac{\partial^2 C(\mathbf{W})}{\partial W_i \partial W_j} \quad \text{for} \quad 1 \leq i, j \leq n. \tag{2.5}$$

The Taylor series may be also expanded as

$$C(W + h) = C(W) + \sum_{i=1}^{n} G_i(W) h_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} h_i H_{ij}(W) h_j + \ldots \tag{2.6}$$

or equivalently as

$$C(\mathbf{W}+\mathbf{h}) = C(\mathbf{W}) + \mathbf{G}^T(\mathbf{W})\mathbf{h} + \frac{1}{2}\mathbf{h}^T\mathbf{H}(\mathbf{W})\mathbf{h} + ... \tag{2.7}$$

We want to solve these equations for $dW$ in an iterative fashion so that

$$\mathbf{W}_{t+1} = \mathbf{W}_t + d\mathbf{W}_t \tag{2.8}$$

where $\mathbf{h} = d\mathbf{W}$. If we ignore the second and higher order terms in the Taylor series we get

$$C(\mathbf{W}+d\mathbf{W}) = C(\mathbf{W}) + \mathbf{G}^T(\mathbf{W})d\mathbf{W} \tag{2.9}$$

for a steepest decent [18].

At a local minima $G(\mathbf{W}^*) = 0$ and $H^T(\mathbf{W}^*)$ is positive definite. Then set

$$\mathbf{W}^* = \mathbf{W} + d\mathbf{W} \tag{2.10}$$

and

$$C(\mathbf{W}^*) = C(\mathbf{W}) + \mathbf{G}^T(\mathbf{W})d\mathbf{W}. \tag{2.11}$$

Differentiating this with respect to $\mathbf{W}$ gives

$$0 = \mathbf{G}(\mathbf{W}^*) = \mathbf{G}(\mathbf{W}) + \mathbf{H}(\mathbf{W})d\mathbf{W}. \tag{2.12}$$

So

$$d\mathbf{W} = -\mathbf{H}^{-1}(\mathbf{W})\mathbf{G}(\mathbf{W}). \tag{2.13}$$

For steepest decent, set $\mathbf{H}^{-1} = \mu$. This means that the recursion $\mathbf{W}_{n+1} = \mathbf{W}_n - \mu\mathbf{G}(\mathbf{W}_n)$ will iterate toward a local minimum. Backpropagation minimizes its root-mean-square error cost function by this technique. Using the recursion, the change in weight for a particular connection is directly proportional to the negative gradient evaluated at the most recent weight. So, the Delta rule (and the Generalized Delta rule, discussed later) uses this recursion to iterate toward a weight vector that produces the least mean square error. The learning rate $\beta = \mu$ above "is a measure of the speed of convergence of the weight vector to the" [9] ideal minimum error vector. So, one expects the value of beta to be greater than 0 to insure a negative gradient.

## Backpropagation design

**Backpropagation networks: architecture** Artificial neural networks that use backpropagation are hierarchal and have at least three layers of nodes. Each layer of nodes is connected to the next higher level of nodes, but not normally to nodes within the same layer. The lowest layer consists of the input nodes, and the topmost layer contains nodes that produce output responses to the input pattern. For a three-layer network, the middle layer of nodes is called the "hidden" layer because these nodes have no contact with the physical input/output environment, but merely with the nodes above and below them. The number of input nodes and the number of output nodes are defined by the problem to be solved. That is, the number of input nodes is the number of input variables, and the number of outputs is a function of the pattern classifications desired. The number of hidden nodes is essentially a measure of the number of internalized features as determined by the neural network. Usually, one can make an educated guess about the proper number of hidden nodes. Many

times too few nodes tends to increase training time and reduce the recall abilities. Too many nodes allows the network "memorize" the patterns and not generalize the inputs into general features [10]. Most often, network architectures resemble either a pyramid (with a large number of input nodes connected to a lesser number of hidden nodes connected to even fewer output nodes, eg., 20-17-11-2, where this notation represents 20 inputs, 17 first-hidden layer nodes, 11 second-hidden layer nodes, and 2 output nodes), or else a "Christmas tree" (with a large number of inputs connected to an even larger number of hidden nodes connected to fewer nodes in the next levels, eg., 20-30-11-2).

**Forward activation: the transfer function** When an example pattern is presented to the network input nodes, the pattern is passed along to the nodes in the first hidden (or middle) layer via the interconnecting weights. For one of these hidden nodes the summed input $Z$ is the weighted sum of all the inputs linked to that node, as $Z = \sum(W_i X_i)$ where $i$ is over the total number of input nodes linked to the hidden node. Next, this value is operated on by the node's transfer function. The result of the transfer function $f(Z)$ is called the node's activation. The purpose of the transfer function is to determine the excitation level, or activation, of the node as a response to the input value $Z$. In backpropagation networks this function needs to be differentiable. The sigmoid function was first chosen as the activation function for neural networks using backpropagation by Williams in 1983 [26]. A sigmoid function is a non-linear function characterized by its "S" shape and asymptotically approaches fixed values as the input approaches plus or minus infinity [26]. (See Figure 2.5.) Usually the upper and lower limits are normalized to +1 and 0 or -1, respectively.

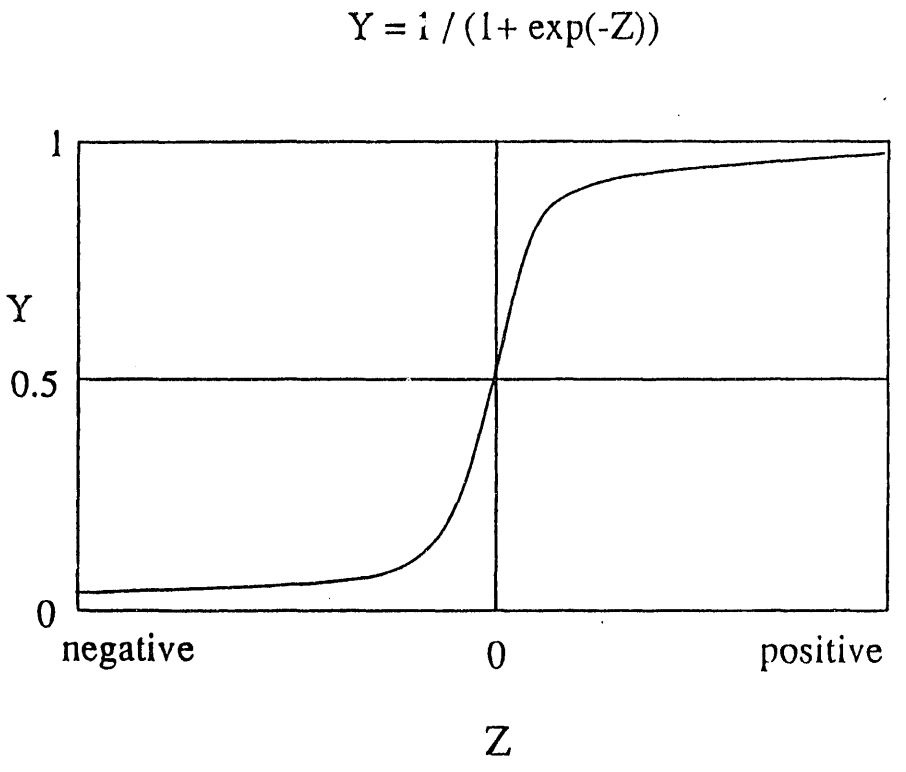


Figure 2.5: The transfer function

The activation function used in most neural network designs using backpropagation is the reciprocal negative exponential as $f(Z) = \frac{1}{1+e^{(-Z)}}$. Sometimes the activation function is modified by introducing a threshold bias to the summation or a gain factor to step up the activation value, $f(Z) = 1/(1 + exp(-g(Z + b))$, where $b$ is the threshold bias term and $g$ is the gain factor. In addition, because an output value of 0 or 1 requires the inputs to be infinitely large in magnitude, people tend to arbitrarily assign values greater that 0.9 as 1 and lower than 0.1 as 0 [10]. Thus, each node in the hidden layer receives its summed input value and produces some output value. These hidden nodes are connected to nodes in the next layer, whereby the outputs of the first hidden nodes are passed via the interconnecting weights as the inputs

to the next higher layer. The activation function is again employed. The process is repeated from layer to layer until the output layer is reached. Here, the outputs of the nodes at the top layer represent the interface to the environment. The whole process of this forward flow of information is an analog parallel process. Usually, however, such neural networks are simulated on serial digital computers. Use of a parallel processing machine can speed up the network learning greatly.

**Backward error flow: generalized delta rule** Once the input pattern set is completely passed forward to the output nodes, the output nodes' values can be compared to the desired values, known beforehand or defined from the training set. Many times the desired responses for the output nodes are binary values. In this way, a well-defined number of classifications can be determined from the data patterns. For example, to classify 8 patterns, one might choose to use 3 binary output switches, since $2^3 = 8$. As stated previously, the actual responses are compared to the desired responses to determine the error in each output node for the pattern. It is desired to propagate these errors backward through the network via the same connecting weights as in the forward pass. These errors are then used to incrementally change the weights of the output nodes according to the Delta rule. (Refer to Section 3.1 of this chapter.) It is a relatively simple matter to send the error signal from the output nodes to the next lower layer with the Delta rule. However, in passing the errors further back to the lower hidden layer nodes the procedure becomes increasingly complex. For this we use the Generalized Delta rule. The Generalized Delta rule involves the derivative of the sigmoidal function to train the hidden layer nodes [10] [27]. The sigmoid activation function $f(Z) = 1/(1+exp(-Z))$ has the graceful feature

of having a simple derivative. This can be demonstrated by taking the derivative $D_x(f(x))$ as $D_x(\frac{1}{1+e^{-x}}) = -1(1+e^{-x})^{-2}(-e^{-x})$ which reduces by some algebra to $f'(x) = \frac{1}{1+e^{-x}}(1 - \frac{1}{1+e^{-x}})$. Or

$$f'(Z) = f(Z)[1 - f(Z)]. \tag{2.14}$$

The Generalized Delta rule is shown below to be

$$W_{new} = W_{old} + E\beta \sum_{patterns} \sum_{nodes} f(Z)f'(Z). \tag{2.15}$$

[9] (See Figure 2.6.)

The Generalized Delta rule follows from use of the Chain rule to find the relative change in cost for a change in weight for the given input pattern. This learning law depends on two facts, that (1) the error function (cost) $C(\mathbf{W})$ is differentiable and that (2) $G(W) \approx \frac{1}{N}\Sigma_{n=1}^{N} G_n(W)$, where $N$ is the number of patterns and $G$ is the gradient [18]. The change in cost with respect to a change in weights is expressed as

$$\frac{\partial C(W)}{\partial W_{i,j,k}} = \frac{\partial C(W)}{\partial Z_{i-1,k}} \frac{\partial Z_{i,j}}{\partial W_{i,j,k}} \frac{\partial X_{i-1,j}}{\partial Z_{i,j}} \tag{2.16}$$

where the $X_{i,j}$ is the output signal of the $j$th process element of the $i$th layer for each training pattern presented on a forward pass. Notice that

$$\begin{aligned} \frac{\partial C}{\partial Z_{i,j}} &= \frac{\partial C}{\partial X_{i,j}} \frac{\partial X_{i,j}}{\partial Z_{i,j}} \\ &= \frac{\partial C}{\partial X_{i,j}} f'(Z_{i,j}) \end{aligned} \tag{2.17}$$

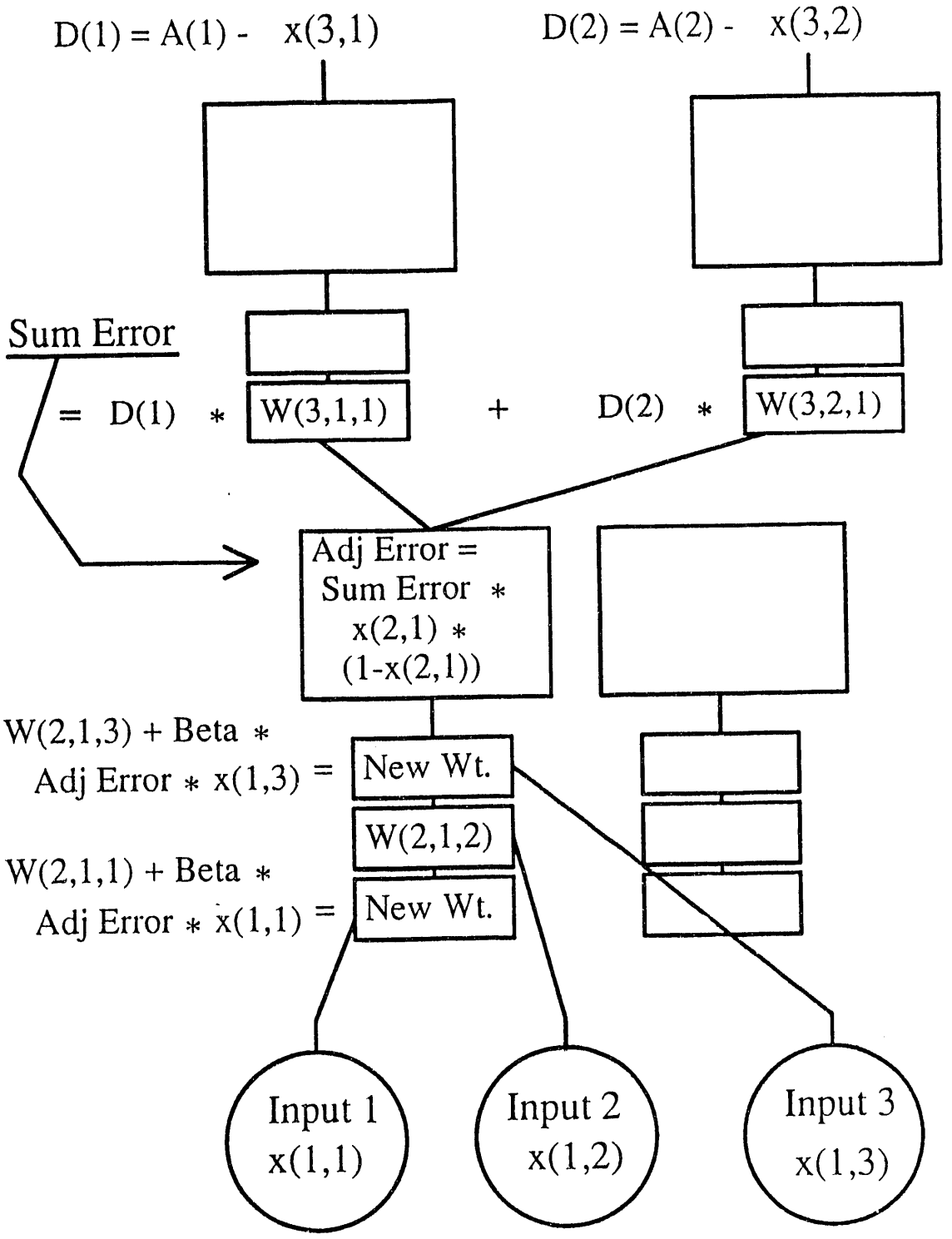


Figure 2.6: Generalized delta rule

where $f'$ is the derivative of the sigmoid discussed earlier in this section. This is the essence of the Generalized Delta rule for backpropagation learning algorithms. To decrease the error function all that is necessary is to adjust the weight vector in the direction $-G(W)$. The rate of the change is specified by the learning constant $\beta$. The amount of weights adjustment should be kept small enough so that the network does not overshoot the minimum along the error surface.

In tracing along a surface of the error function, following the negative gradient, the network occasionally may become lodged in a local minimum. (See Figure 2.7.)

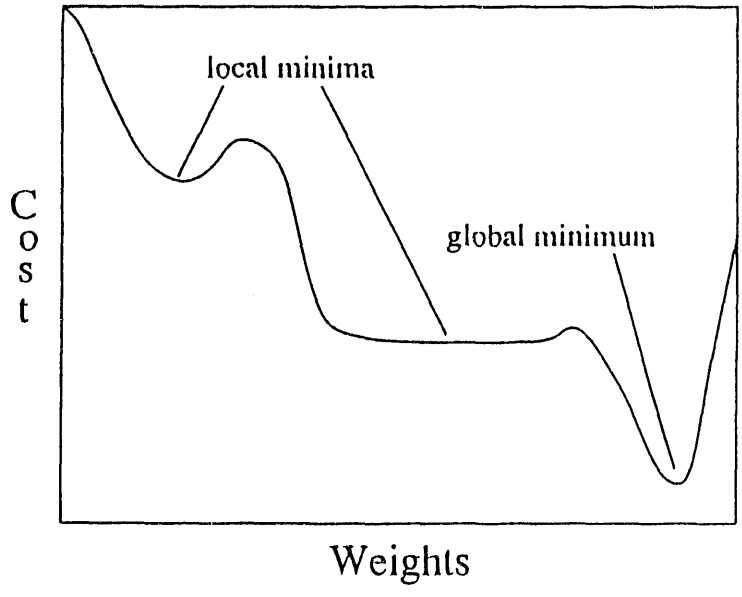


Figure 2.7: Local minima and a global minimum cost for weights

So it is necessary to provide the Delta rule with some extra "momentum" to climb out of these minima. The Delta rule can be revised by the addition of a momentum term,

the constant $\alpha$ multiplied by the change in the weight vector from the previous pattern. Stated mathematically the additional momentum term is $\alpha(W_{new}-W_{old})prev$, where $\alpha$ is the momentum constant. The revised Generalized Delta rule becomes:

$$W_{new} = W_{old} + \beta \sum_{n} \sum_{j} \frac{1}{1+e^{-\sum WX}}(1-X) + \alpha(W_{new} - W_{old})prev. \quad (2.18)$$

Backpropagation is a relatively complex computational paradigm, and is somewhat slow since it is necessary to pass information twice (forward and backward) per pattern per iteration. It requires long training times to learn the set of training examples, often several days of run time for a high-end work station. But it is able to recall almost instantly on the simplest of PCs. For this reason and others, neural networks that use backpropagation are the type of neural network most popular among researchers today.

## Importance of Nodes

The emphasis of some researchers in neural networks is in the area of determining the importance of nodes. Drs. Bartlett and Uhrig from the University of Tennessee at Knoxville Department of Nuclear Engineering explained nodal significance this way: "The importance of a node can be shown to be a function of the outputs of the other nodes in the network. If a node can be shown to have little or no dynamical effect on the output of every node to which its output is an input then it is of little value to the network and has little importance. The total importance of node (i-1,k) is then the sum of the changes of the outputs of the nodes in layer i, with respect to changes in the output of node (i-1,k)," where k is the input weight to the node from

the nodes in the i-1 layer [3]. The importance of a node, in this or in their other work, is essentially the partial derivative of the output classification with respect to the input node. The importance of a node is comprised of a number of factors such as the amount of linear correlation, the amount of information present in the node, and the derivatives of the weights connecting to the node. Each of these factors can be justified by reason, mathematical proof, or by practical experience. The partial importance of node $X_{i-1,k,n}$ to a node in the layer above it $X_{i,j,n}$ is equivalent to the partial derivative of an output node signal relative to the node's output signal which is an input to the output node. This may be expressed as

$$I(X_{i,j,n}, X_{i-1,k,n}) = \frac{\partial X_{i,j,n}}{\partial X_{i-1,k,n}} \tag{2.19}$$

where $I$ is the importance function, $X$ is the output of the particular node, and the subscripts i,j,k, and n are the layer number, the node in in layer i, the node in layer i-1, and the pattern number, respectively. Given this definition of importance, the total importance of a node $X_{i-1,k}$ is the partial importances for that node summed over the total number of output nodes above and over the number of patterns presented. Thus, the importance of a hidden node can be defined as

$$I(X_{i-1,k}) = \sum_{n=1}^{N} \sum_{j=1}^{J(i)} \frac{\partial X_{i,j,n}}{\partial X_{i-1,k,n}}. \tag{2.20}$$

One can determine the importance of any node in the network by applying the Chain rule to this equation. The emphasis of this thesis is in determining the importances of the input nodes. Because networks of four layers have been used, the Chain rule must be applied three times. The importance of any node with respect to nodes below it follows from the derivation provided below:

Using the sigmoidal activation function, the output of any given node $X_{i,j,n}$ is equivalent to $\frac{1}{1+e^{-\sum X_{i-1,k,n} W_{i,j,k}}}$. Let $\eta = -\sum_k X_{i-1,k,n} W_{i,j,k}$. Then, differentiating $X_{i,j,n}$ with respect to $X_{i-1,k,n}$ yields

$$\begin{aligned} \frac{\partial X_{i,j,n}}{\partial X_{i-1,k,n}} &= \frac{\partial X_{i,j,n}}{\partial \eta} \frac{\partial \eta}{\partial X_{i-1,k,n}} \\ &= (1+e^{-\eta})^{-2} e^{-\eta} W_{i,j,k} \\ &= \frac{W_{i,j,k} e^{-\eta}}{(1+e^{-\eta})^2} \end{aligned} \tag{2.21}$$

which, when backsubstituting for $\eta$ reduces to

$$\begin{aligned} I(X_{i-1,k}) &= \frac{1}{N} \sum_{n=1}^{N} \sum_{j=1}^{J(i)} \frac{\partial X_{i,j,n}}{\partial X_{i-1,k,n}} \\ &= \sum_{n}^{N} \sum_{j}^{J(i)} \frac{W_{i,j,k} e^{-\sum X_{i-1,k,n} W_{i,j,k}}}{(1+e^{-\sum X_{i-1,k,n} W_{i,j,k}})^2}. \end{aligned} \tag{2.22}$$

The same notation as before is used. The importance of the input nodes for a 4-layer network can be shown to be

$$\frac{\partial X_{3,j}}{\partial X_{0,k''}} = \frac{\partial X_{3,j}}{\partial X_{2,k}} \frac{\partial X_{2,k}}{\partial X_{1,k'}} \frac{\partial X_{1,k'}}{\partial X_{0,k''}} \tag{2.23}$$

by the Chain rule. The importance of an input node is then

$$\frac{\partial X_{3,j}}{\partial X_{0,k''}} = \frac{1}{N} \sum_{n}^{N} \sum_{k}^{J(2)} \sum_{k'}^{J(1)} \frac{\partial X_{3,j}}{\partial X_{2,k}} \frac{\partial X_{2,k}}{\partial X_{1,k'}} \frac{\partial X_{1,k'}}{\partial X_{0,k''}} \tag{2.24}$$

It is this form that has been applied as the importance function toward determining input variable significances levels. For the purposes of this research, the

importance function, as developed, was applied to the three independently trained neural networks. A recall algorithm calculated the importance values for each of the input nodes for each of the architectures trained. In the code development it was felt that the effects on each output should not nullify one another. That is, if an input node has a positive effect on one output node and a negative effect on another output node, these effects should not cancel. So, the code calculates each partial importance and sums their magnitudes to determine the total importance of an input node. (This makes intuitive sense, since it is difficult to conceive of negative importance. Refer to IMPORT.FOR in Appendix B.) These values are compared with the statistical correlations determined previously for each network architecture. This procedure is discussed in Chapter 3, and the results of that comparison are discussed in Chapter 4.

# CHAPTER 3. PROBLEM AND METHOD OF SOLUTION

## Power Plant Diagnostics

The specific problem to be investigated is to develop an artificial neural network fault-diagnostic advisor for implementation at the Duane Arnold Energy Center using simulated reactor plant data from the DAEC for three abnormal and one normal operating conditions. This network is used to compare the importance function's correlation coefficients with standard statistical linear correlations between input plant variables and output diagnoses. This goal demands a careful examination of nuclear power plant systems. There are innumerous power plant transients that could have been chosen to study. The *Updated Final Safety Analysis Report*, Chapter 15: Accident Analysis [42] and the *Malfunction Cause and Effects Report* [24], both by DAEC, have been consulted. These publications provide most of the significant power plant accidents of interest. Furthermore, several consultations were made with John Adams [1], a former nuclear power plant operator at DAEC, to discuss boiling water reactor design and plant specifics. Adams provided information, including blueprints and technical manuals used by personnel at the DAEC nuclear power station. The analysis done by DAEC and these consultations provided an initial guess at the most significant plant conditions and input variables to be considered for the neural network diagnostic advisor. A list of primary initiating events as well as a list of sec-

ondary events were determined from the DAEC *Cause and Effects Malfunctions List.* Descriptions of the primary initiating events chosen for this analysis are presented in Appendix A. These include a main feedwater-line break, a loss of a feedwater heater, and a design-basis loss-of-coolant-accident. Data was obtained from Duane Arnold Energy Center's nuclear power plant operators training simulator for this thesis work [43] [24] [42] [25].

## Data Collection Process

Data was obtained on 81 nuclear power plant variables in real time as the simulated accidents progressed. A list of all possible variables (actual meter readings) used by the simulating computer at DAEC is also available [32]. Consultations were also made with personnel at DAEC, particularly Mr. Don Vest, who is a simulator instructor [43], to help determine which computer points would prove most useful to identify abnormal plant conditions at their plant. Eighty-one plant variables capable of characterizing most abnormal operating conditions were expected to be used in the training of the diagnostic advisor developed for the present work. A list of 83 such input variables was initially determined [43]. Data acquisition began June 1991. The first condition studied is a design-basis loss-of-coolant-accident. See Appendix A for a full description of this accident. Data for this condition was collected through several minutes after the automatic reactor scram occurred, including about one minute of normal operating conditions prior to the simulated accident, on those 83 plant variables. At a later time another trip was arranged to obtain data for two additional conditions. These events were a loss of feedwater heating to heater A and a main feedwater-line break 100%. These accidents are described in greater detail

in Appendix A. However, by this time, the 83-variable list had been expanded to 96 variables, excluding three redundant variables from prior runs. One or more of these redundant variables was a binary switch to help determine the exact moment at which the condition occurred within the full data sets. For a more complete description of the plant conditions studied at the DAEC, refer to Appendix A [25].

## Data Reduction Methods

In order to use the data obtained from DAEC, it was necessary to reformat the raw data files into a form more convenient for use as input to the artificial neural network. This involved several formatting routines, a normalization routine, and a code to add the desired output classifications as data lines. All these codes are written in Fortran [19] and are meant to be run in a specific order. Refer to Table 3.1 and Appendix B for code descriptions and source code listings. The program "SHUFFY" discriminates headings and blank lines and removes them. It also converts the time data from Hour:Min:Sec form to a value in seconds. The program "COLUMNS" takes the data arranged in columns of six variables across by time length and converts it to a form with all variables written across per unit time. The program "SPITTLE" uses a binary importance vector input to discriminate uncommon variables between different data sets. This was necessary because data for the design-basis LOCA was collected on 83 variables, while data for the other two conditions was collected on 96 variables, excluding two and the binary switch. This procedure resulted in three complete data files, each having 81 common variables in the same order. Each of these files contains a short period sample of the normal operating condition followed by the abnormal transient. Next, the three files were concatenated to form one

Table 3.1: Formatting procedure

| |
|---|
| 1) Replace Booleans and/or "Y"-"N" data with "1"-"0." |
| 2) Rename input file as TRIAL.DAT. |
| 3) Run SHUFFY.FOR. |
| 4) Run COLUMNS.FOR (Output of SHUFFY is input as TRIAL2.DAT). |
| 5) Run SPITTLE.FOR (Requires the Binary Importance Vector). |
| 6) Concatenate each formatted accident data set as TRIAL4.DAT. |
| 7) Run NORMAL.FOR. |
| 8) Reduce FINAL.DAT into its component data sets with BREAKUP.FOR. |
| 9) Run FLAG.FOR. |
| 10) Concatenate files to create the test file data set. |

data set. In this data set, the variables were normalized to themselves with the program "NORMAL." The data was normalized from 0.1 to 0.9, rather than typically 0 to 1. This was an attempt to restrict the range of input values to help neural network training. It was then necessary to take the normalized data file and break it up into the three abnormal conditions and three sets of normal data. To accomplish this, a program called "BREAKUP" was used. Each of these six files were more or less assigned an output value with the code "FLAG." These data were placed at the end of each set of variables for each time step. Ultimately, it is desired to normalize all the data to full-scale meter readings of the simulator or actual control room panels. However, this was not performed in this thesis. A list and description of the complete

set of 81 plant variables may be found in Table 3.2.

## Training the Advisor

### Analysis of variance

A statistical analysis was performed on the input variables for the output classifications to help determine statistical relationships between inputs and outputs in the data set. It is felt that a linear correlation between the inputs and the output for the data should be sufficient to determine a means to eliminate unnecessary input variables. Consideration of non-linear relationships among the data would give better results but is not considered in this study. It is necessary to employ the method of least squares for linear regression [38]. Let $y$ be some variable demonstrating some linear dependency on some independent variable $x$. Then the expectation value for $E(y) = A + Bx$. This is the weighted average value for $y$. The method of least squares attempts to make a best-fit line to the data so that the sum of the squared errors for all sample points is kept to a minimum. Mathematically, this is stated as: $min \sum_{i=1}^{n}(y_i - A - Bx_i)^2$. The predictor constants $A$ and $B$ are determined from the equations:

$B = \frac{SS_{xy}}{SS_{xx}}$, and $A = y_{ave} - Bx_{ave}$,

where

$$SS_{xy} = \sum(x_i - \bar{x})^2 = \sum x^2 - \frac{(\sum x)^2}{n} \tag{3.1}$$

and

Table 3.2: Final plant variables used

| VAR | POINT ID | POINT DESCRIPTION |
|---|---|---|
| 1 | A041 | LPRM 16-25 FLUX LEVEL B |
| 2 | A091 | SRM CHANNEL B |
| 3 | B000 | APRM A FLUX LEVEL |
| 4 | B012 | REACTOR TOTAL CORE FLOW |
| 5 | B013 | REACTOR CORE PRESS-DIFF |
| 6 | B014 | CRD SYSTEM FLOW |
| 7 | B015 | RX FW LP A FLOW TEMP-CORRECTED |
| 8 | B016 | RX FW LP B FLOW TEMP-CORRECTED |
| 9 | B017 | CLEAN-UP SYSTEM FLOW |
| 10 | B022 | TOTAL STEAM FLOW |
| 11 | B023 | CLEANUP SYSTEM INLT TEMP |
| 12 | B024 | CLEANUP SYSTEM OULT TEMP |
| 13 | B026 | RECIRC LOOP A1 DRV FLOW |
| 14 | B028 | RECIRC LOOP B1 DRV FLOW |
| 15 | B030 | REACTOR FW CHNL A1 TEMP |
| 16 | B032 | REACTOR FW CHNL B1 TEMP |
| 17 | B034 | RECIRC LOOP A1 INLT TEMP |
| 18 | B036 | RECIRC LOOP B1 INLT TEMP |
| 19 | B038 | RECIRC A WIDE RANGE TEMP |
| 20 | B039 | RECIRC B WIDE RANGE TEMP |
| 21 | B061 | RCT JET PMPS 1-8 FLOW B |
| 22 | B062 | RCT JET PMPS 9-16 FLOW A |
| 23 | B063 | RCT OUTLET STM FLOW |
| 24 | B064 | RCT OUTLET STM FLOW B |
| 25 | B065 | RCT OUTLET STM FLOW C |
| 26 | B066 | RCT OUTLET STM FLOW D |
| 27 | B079 | RRP A MTR VIBRATION |
| 28 | B080 | RRP B MTR VIBRATION |
| 29 | B083 | CRD DRIVE WTR DIFF PRESS |
| 30 | B084 | CRD CLG WTR DIFF PRESS |
| 31 | B085 | TORUSR TEMP #1 |
| 32 | B086 | TORUS AIR TEMP #2 |
| 33 | B087 | TORUS AIR TEMP #3 |
| 34 | B088 | TORUS AIR TEMP #4 |
| 35 | B089 | DRYWELL TEMP AZ EL750 |
| 36 | B090 | DRYWELL TEMP AZ245 EL750 |
| 37 | B091 | DRYWELL TEMP AZ90 EL 765 |
| 38 | B092 | DRYWELL TEMP AZ270 EL 765 |
| 39 | B093 | DRYWELL TEMP AZ270 EL765 |
| 40 | B094 | DRYWELL TEMP AZ180 EL780 |

Table 3.2 continued

| VAR | POINT ID | POINT DESCRIPTION |
|---|---|---|
| 41 | B095 | DRYWELL TEMP AZ270 EL 830 |
| 42 | B096 | DRYWELL TEMP CNTR EL 750 |
| 43 | B098 | TORUS WATER TEMP |
| 44 | B099 | TORUS WATER TEMP |
| 45 | B103 | ILRT DRYWELL PRESSURE |
| 46 | B104 | ILRT TORUS PRESSURE |
| 47 | B105 | TORUS WATER LEVEL |
| 48 | B120 | TORUS RAD MON A |
| 49 | B121 | TORUS RAD MON B |
| 50 | B122 | REACTOR WATER LEVEL |
| 51 | B125 | REACTOR WATER LEVEL |
| 52 | B126 | REACTOR WATER LEVEL |
| 53 | B137 | TORUS WATER LEVEL |
| 54 | B138 | TORUS WATER LEVEL |
| 55 | B150 | CORE SPRAY A FLOW |
| 56 | B151 | CORE SPRAY B FLOW |
| 57 | B160 | RCIC FLOW |
| 58 | B161 | HPCI FLOW |
| 59 | B162 | RHR A FLOW |
| 60 | B163 | RHR B FLOW |
| 61 | B164 | DRYWELL RAD MON (A) |
| 62 | B165 | DRYWELL RAD MON (B) |
| 63 | B166 | POST TREAT ACT |
| 64 | B168 | PRETREAT ACT |
| 65 | B171 | ANALYZER A O2 CONCENTRATION |
| 66 | B172 | ANALYZER A H2 CONCENTRATION |
| 67 | B173 | ANALYZER B O2 CONCENTRATION |
| 68 | B174 | ANALYZER B H2 CONCENTRATION |
| 69 | B180 | CLEAN-UP SYSTEM FLOW |
| 70 | B196 | REACTOR WATER LEVEL-FUEL ZONE A |
| 71 | B197 | REACTOR WATER LEVEL-FUEL ZONE B |
| 72 | B247 | TURB STM BYPASS |
| 73 | B248 | TURB STM BYPASS |
| 74 | E000 | 4160 V SWGR BUS 1A1 A-B |
| 75 | F004 | COND PMP A&B DISCH PRESS |
| 76 | F040 | 1P-1A RFP SUCT PRESS |
| 77 | F041 | 1P-1B RFP SUCT PRESS |
| 78 | F042 | 1P-1A RFP DISCH PRESS |
| 79 | F043 | 1P-1B RFP DISCH PRESS |
| 80 | F094 | FW FINAL PRESSURE |
| 81 | G001 | GEN GROSS WATTS |

$$SS_{xy} = \sum(x_i - \bar{x})(y_i - \bar{y}) = \sum(xy) - \frac{\sum x_i \sum y_i}{n}. \tag{3.2}$$

The standard measure of linearity is the Pearson Correlation Coefficient $R$, defined as

$$R = \frac{SS_{xy}}{\sqrt{SS_{xx} SS_{yy}}}, \tag{3.3}$$

where $SS_{xx}$ is the sum of the squared residuals [38]. Stated in the most convenient computational form, the correlation coefficient is

$$R = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{[n\sum x_i^2 - (\sum x_i)^2][n\sum y_i^2 - (\sum y_i)^2]}} \tag{3.4}$$

[23].

Using this relationship, a Fortran code was created that calculates linear correlation coefficients for each of the input variables. The "ANOVA" code additionally allows the user to specify a threshold value for the correlation coefficient, such that any variable with a correlation coefficient below the threshold magnitude is considered statistically unimportant (at the specified level) and is not used for network training. A reduced data set containing only the statistically important variables is then created. The ANOVA7 routine was used to perform these tasks. (See Appendix B.) A threshold coefficient of 0.3 demonstrated 20 of the original 81 variables were statistically correlated. A list of those 20 variables is described in Table 4.1. At a threshold of 0.28 33 input variables were correlated. A list of those 33 variables is presented in Table 4.2. At a threshold of 0.2 50 input variables were correlated. A list of those 50 variables is presented in Table 4.3. (See Chapter 4.) These reduced

data sets were then used as the recall sets and used to create training sets for the networks' respective architectures. For a list of all 81 variables with their associated statistical correlation level, see Table 3.3. Note that this correlation is for an input variable with respect to some arbitrarily chosen boolean (representing the pattern classification).

### Creating training sets: TFGEN

It was desired to use these reduced data sets to create training sets of example patterns by which to train the artificial neural network. In training a neural network, only a few examples should be used from the full recall data set. One expects the network to make generalizations from the few example patterns toward the greater whole. To accomplish this, another FORTRAN code "TFGEN" was developed to generate the training files. Specifically, TFGEN copies several patterns from the recall data set (consisting of the reduced variable data) into a training file. For a further description, refer to Appendix B. The initial training set for each level of statistical correlation consisted of 12 examples. These 12 patterns were chosen to represent each of the three abnormal sections and each of the three normal conditions preceding the abnormal conditions. Two patterns were taken near the beginning of each normal section, and two patterns were taken from near the end of the abnormal conditions. The patterns for the normal sections were nearly identical so that it did not matter from where the patterns were taken. However, to aid the learning of the network, examples from the accident sections were chosen to reflect plant conditions after reactor scram and near equilibrium conditions.

Table 3.3: Correlation coefficients for all plant variables

| VAR | POINT ID | STATISTICAL CORRELATION | VAR | POINT ID | STATISTICAL CORRELATION |
|---|---|---|---|---|---|
| 1 | A041 | -0.2935 | 41 | B095 | 0.2460 |
| 2 | A091 | -0.2752 | 42 | B096 | 0.3051 |
| 3 | B000 | -0.2938 | 43 | B098 | 0.4187 |
| 4 | B012 | -0.3253 | 44 | B099 | 0.4177 |
| 5 | B013 | -0.3195 | 45 | B103 | 0.2049 |
| 6 | B014 | 0.3274 | 46 | B104 | 0.2047 |
| 7 | B015 | -0.1485 | 47 | B105 | 0.0516 |
| 8 | B016 | -0.1429 | 48 | B120 | -0.0484 |
| 9 | B017 | -0.3087 | 49 | B121 | -0.0484 |
| 10 | B022 | -0.2970 | 50 | B122 | 0.0069 |
| 11 | B023 | -0.2149 | 51 | B125 | -0.1141 |
| 12 | B024 | -0.0828 | 52 | B126 | 0.2332 |
| 13 | B026 | -0.3317 | 53 | B137 | 0.0653 |
| 14 | B028 | -0.3088 | 54 | B138 | 0.0653 |
| 15 | B030 | -0.2178 | 55 | B150 | 0.2813 |
| 16 | B032 | -0.2323 | 56 | B151 | 0.2815 |
| 17 | B034 | -0.2812 | 57 | B160 | 0.0725 |
| 18 | B036 | -0.4333 | 58 | B161 | 0.0456 |
| 19 | B038 | -0.2812 | 59 | B162 | 0.2478 |
| 20 | B039 | -0.4333 | 60 | B163 | 0.2478 |
| 21 | B061 | -0.1766 | 61 | B164 | -0.2374 |
| 22 | B062 | -0.3308 | 62 | B165 | -0.1837 |
| 23 | B063 | -0.2975 | 63 | B166 | -0.2685 |
| 24 | B064 | -0.2973 | 64 | B168 | -0.2576 |
| 25 | B065 | -0.2976 | 65 | B171 | -0.2353 |
| 26 | B066 | -0.2977 | 66 | B172 | 0.0000 |
| 27 | B079 | -0.3344 | 67 | B173 | -0.3051 |
| 28 | B080 | 0.0257 | 68 | B174 | -0.3151 |
| 29 | B083 | -0.1912 | 69 | B180 | -0.3071 |
| 30 | B084 | -0.1918 | 70 | B196 | -0.1141 |
| 31 | B085 | -0.0777 | 71 | B197 | -0.1799 |
| 32 | B086 | -0.0689 | 72 | B247 | -0.0850 |
| 33 | B087 | -0.0764 | 73 | B248 | -0.1028 |
| 34 | B088 | -0.0906 | 74 | E000 | 0.0884 |
| 35 | B089 | 0.2658 | 75 | F004 | 0.1793 |
| 36 | B090 | 0.3018 | 76 | F040 | 0.1733 |
| 37 | B091 | 0.2864 | 77 | F041 | 0.1733 |
| 38 | B092 | 0.2416 | 78 | F042 | -0.3084 |
| 39 | B093 | 0.2864 | 79 | F043 | -0.3088 |
| 40 | B094 | 0.2746 | 80 | F094 | -0.3050 |
| | | | 81 | G001 | -0.1556 |

## Network learning

An artificial intelligent neural network algorithm has been developed that incorporates the supervised learning technique of backpropagation. (See code BP17 in Appendix B.) At first the code used a three-layer network: an input layer, one hidden layer, and an output layer. The networks were trained using data generated from the training file generator code. (No random "noise" was added to the training data.) Initially, 20 input variables were used with 10 hidden nodes and 2 outputs. The network architecture is normally fixed for the number of inputs and the number of outputs. The backpropagation network uses a constant learning rate $\beta$ usually set near 0.2, a gain of 1.0, no bias, and a momentum term near 0.01. The 20-10-2 architecture proved somewhat difficult to train. Variations of the number of hidden nodes did not significantly improve the learning. Finally, a new architecture was developed using two hidden layers. Variations on the number of nodes in each hidden layer were made by trial and error until a relatively optimum architecture of 20-14-8-2 was derived. As stated previously, the level of training for a neural network is measured by a cost function, which is essentially the root mean square error of the outputs [40]. The network achieved a cost less than 0.04 using 12 exemplars, at which point the optimized weight configuration was saved to a file. One such training session typically required running the backpropagation code overnight on a 486 computer. However, this time requirement could be decreased significantly by using a high-end VAX 3100. Next, a neural network having the same architecture and using the derived weights pattern, but only feed-forward activation (no learning), was used to recall the abnormal plant conditions for all the data points. The full data file is 969 patterns long. Thus, the network trained for 12 examples tried to classify conditions for 957 patterns it had

never before seen. The recall algorithm called "REC15" demonstrated in which patterns the network was having difficulties classifying by providing the absolute error (desired value minus output value) for each output node, as well as calculating a RMS error for the recall set. Patterns having large errors were noted, and a new test file was created containing the old examples and several new examples. The new examples were chosen at places where the errors were quite large. Then, using the old weights from the previous training as a starting place, the network trained again. This procedure was repeated several times before the network was able to recall satisfactorily. It was found that 28 patterns were necessary to correctly classify the 969 patterns for the 20-input variable set. The whole procedure was again repeated for a different statistical level that produced 33 variables. A network architecture of 33-20-11-2 was assumed. For this training, 29 example patterns were needed to achieve a RMS error similar to the 20-variable network. The procedure was repeated for a third statistical level that produced 50 significant variables. An architecture of 50-30-17-2 was assumed. For this architecture, 30 patterns were necessary for a satisfactory recall.

## Importance of Input Variables

After training the backpropagation network to a reasonable level of recall error, a feed-forward network was used to calculate the importance values for each of the inputs used. The importance function discussed in Chapter 2 was used to implement this. Recall that the importance function for an input node is suggested as the sum of the products of the partial importances for the network. The values of nodal importance were compared with the standard statistical correlations in the statistical

analysis. The results of that comparison may be found in Tables 4.1, 4.2, 4.3 and are discussed in Chapter 4.

# CHAPTER 4. RESULTS OF THE RESEARCH

## Recall Performance

Three networks were trained using variables chosen according to standard correlations for linearity. Several graphs demonstrate desired outputs compared to the actual trained responses for the recalls on all 969 patterns. These three networks were independently trained with only 28, 29, and 30 example patterns for 20-input, 33-input, and 50-input variables, respectively. The specific variables used were determined from the statistical correlation threshold value specified (described in Chapter 3). Refer to Table 3.3 for the correlation values calculated for each of the 81 plant variables. Generally, training the three networks was repeated until the networks could achieve RMS errors no greater than 0.05 for training sets. The process of training was most easily achieved by the 33-variable network. It trained somewhat faster than the other two for the similar sets of example patterns to the same level of RMS error. Upon presentation of a fourth example training set to the 20-input network, it became difficult to train below an RMS value of 0.06. The results of these three networks follows in graphic form. For each of the three networks (and associated number of input variables, there are three sets of three graphs demonstrating the actual and desired output responses for each accident condition.

## Interpretation

Notice from inspection of these graphs that large errors occur primarily at the onset of each accident. There is little or no error during the normal conditions preceding the accident. Similarly, there is virtually no error near the ends of the transients. The normal condition and the near-end of each of the three sets of data represent stable plant conditions. While the normal data does not vary (it is constant) due to its nature, being full power, middle of fuel cycle, the transients reach an equilibrium state well after the automatic plant scram has occurred.

Additionally, a similar variable selection process and training procedure was repeated for a pseudo-linear correlation coefficient. The ANOVA7 code was modified to provide a sum of magnitudes for individual linear correlation coefficients. Normally, effects of equal but opposite linearity for inputs to outputs tend to cancel one another. However, it was once suggested that both effects would be of concern in determining which variables were correlated to the outputs. Accordingly, the ANOVA7 code was modified and run to provide levels of inputs for three arbitrary levels of correlation. This procedure resulted in three networks comprised of 18, 29, and 53 variables. These networks were then trained on the same data exemplars as their counterpart networks with the same number of hidden layers as before. These networks did not train as quickly, nor as effectively as the others. In fact, the 18-input network never could reach an RMS error below 0.16 for the fourth training set presentation. Obviously, the recall errors were even larger. This suggests that this method may not have been a good method by which to choose input variables. For the data used in these network architectures, the standard linearity coefficient seems to be adequate to reduce the total number of possible input parameters toward accident identification.

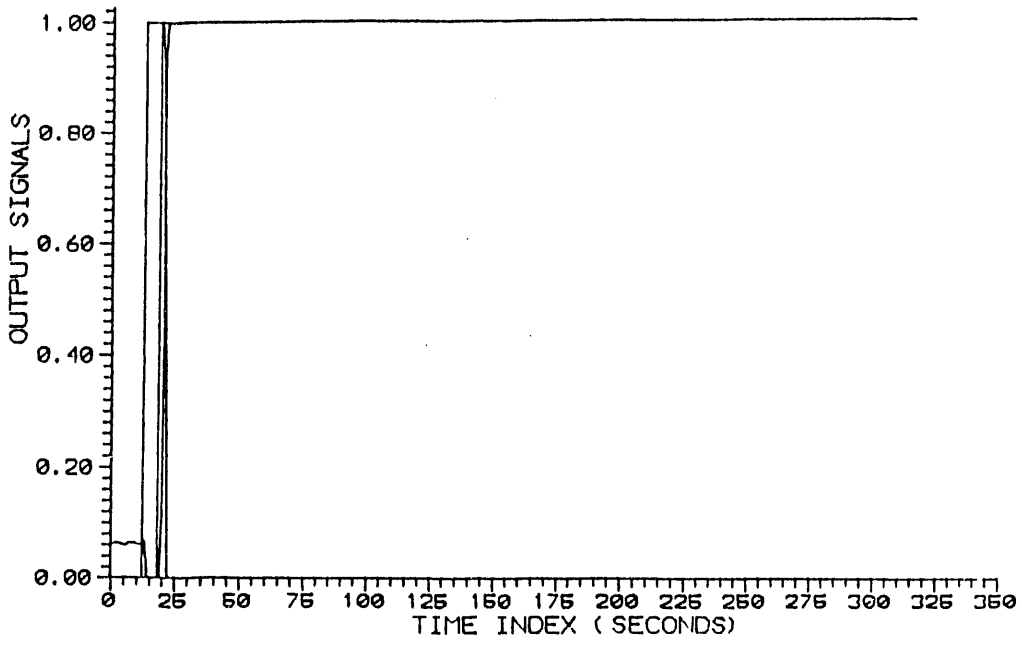


Figure 4.1: Output node activations compared to desired responses for Main Feedwater Line Break for 20 input variables

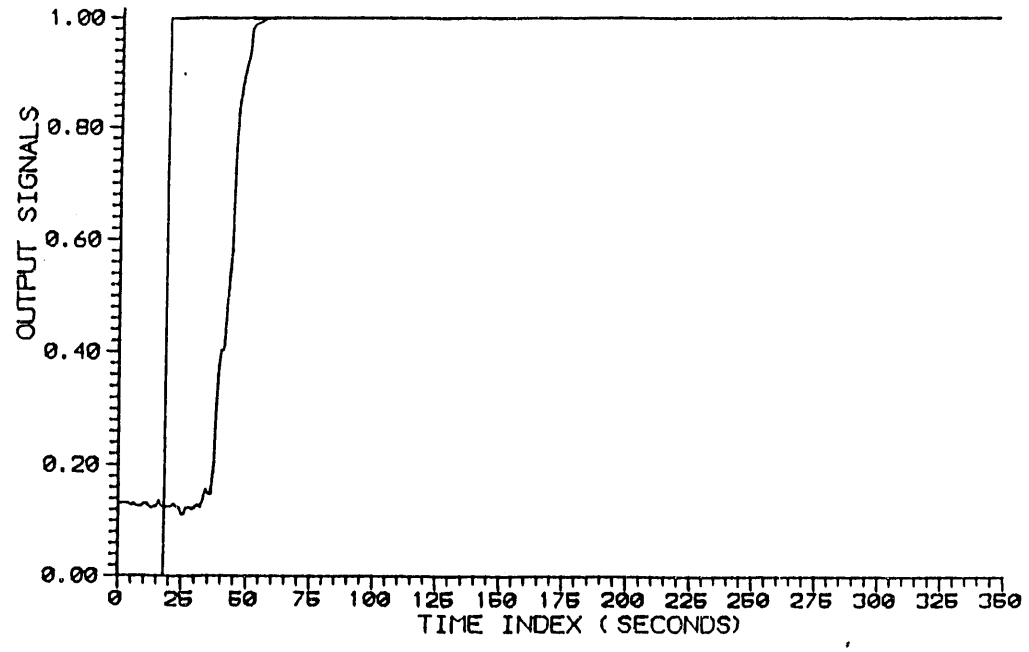


Figure 4.2: Output node activations compared to desired responses for Loss of Feedwater Heating for 20 input variables

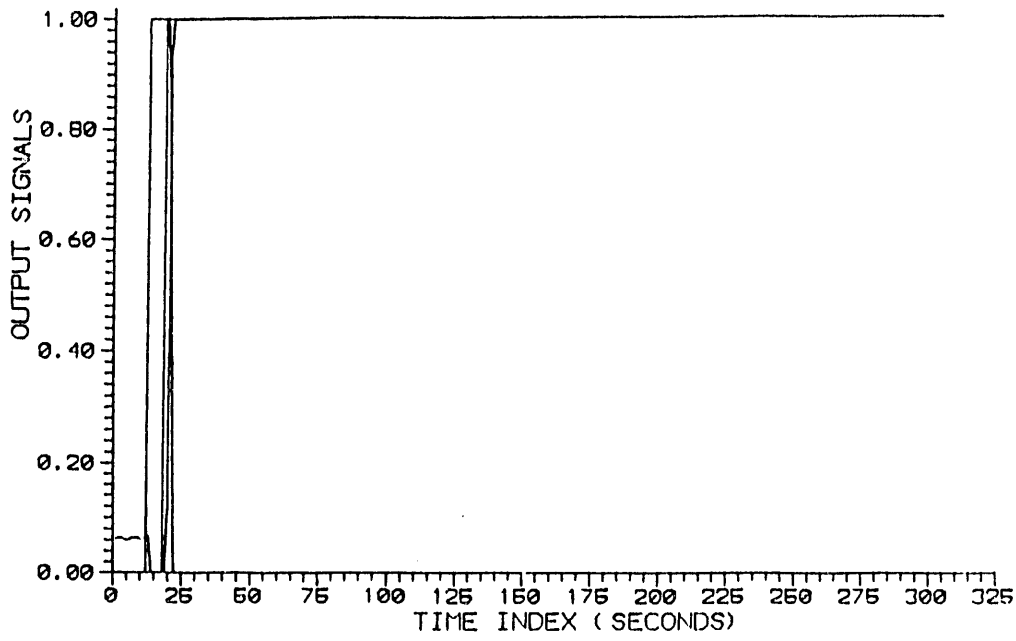


Figure 4.3: Output node activations compared to desired responses for Design basis LOCA for 20 input variables

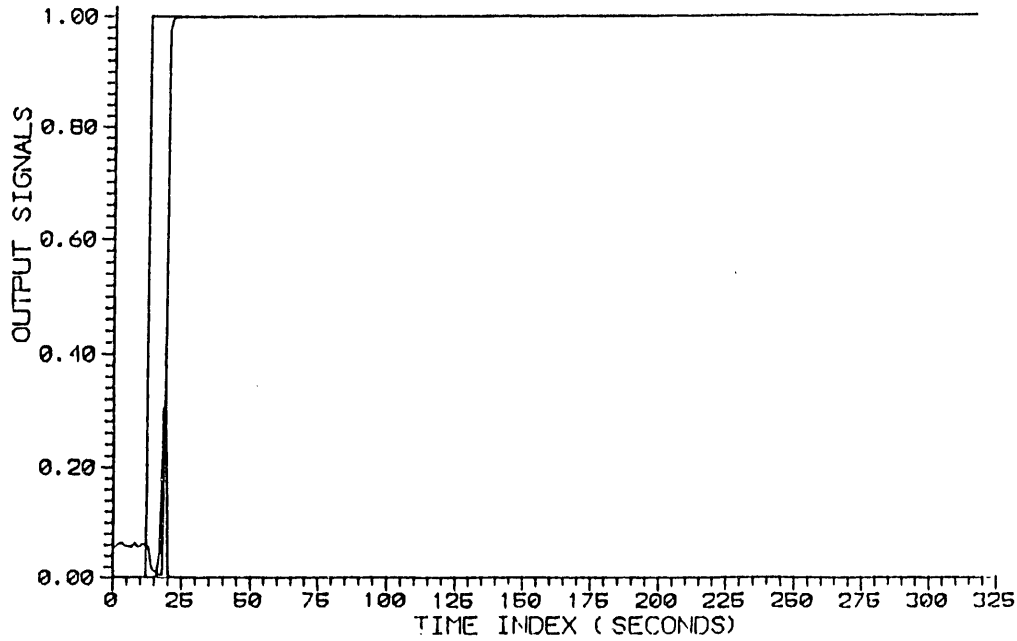


Figure 4.4: Output node activations compared to desired responses for Main Feedwater Line Break for 33 input variables

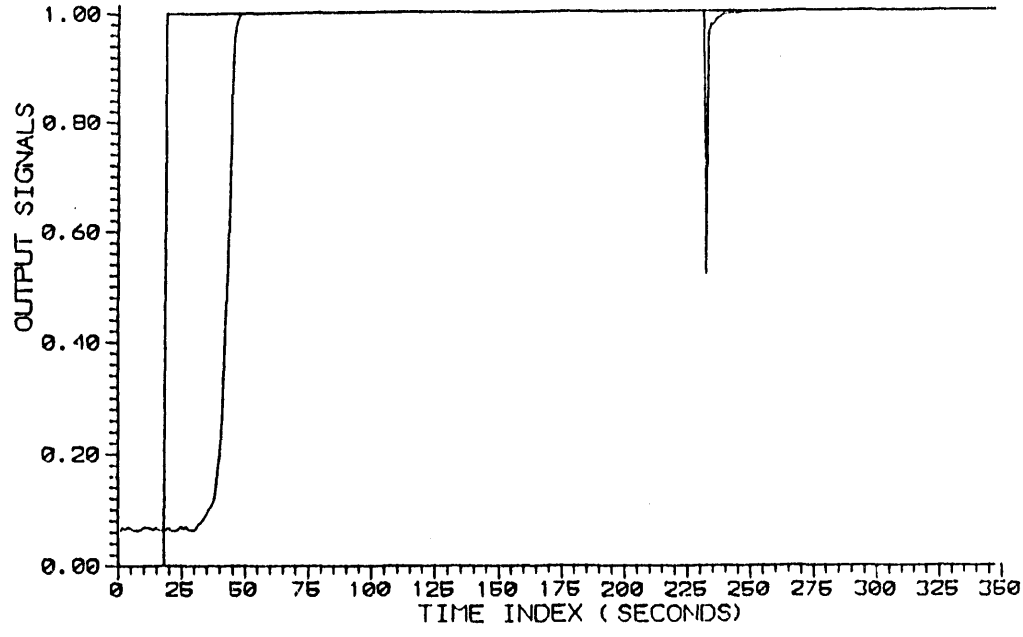


Figure 4.5: Output node activations compared to desired responses for Loss of Feedwater Heater for 33 input variables

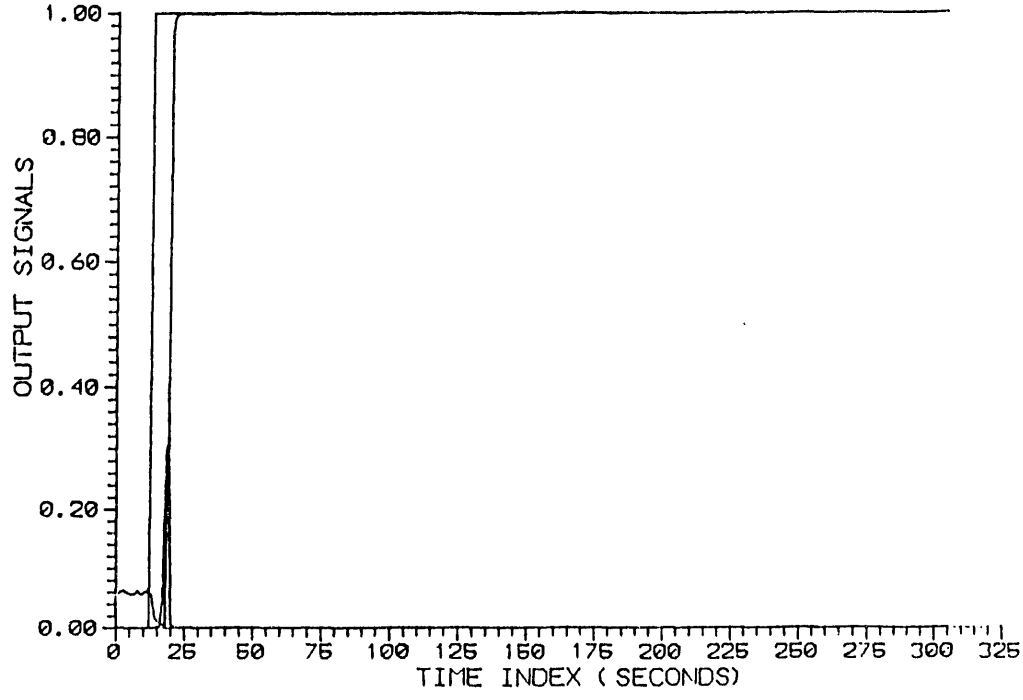


Figure 4.6: Output node activations compared to desired responses for Design basis LOCA for 33 input variables

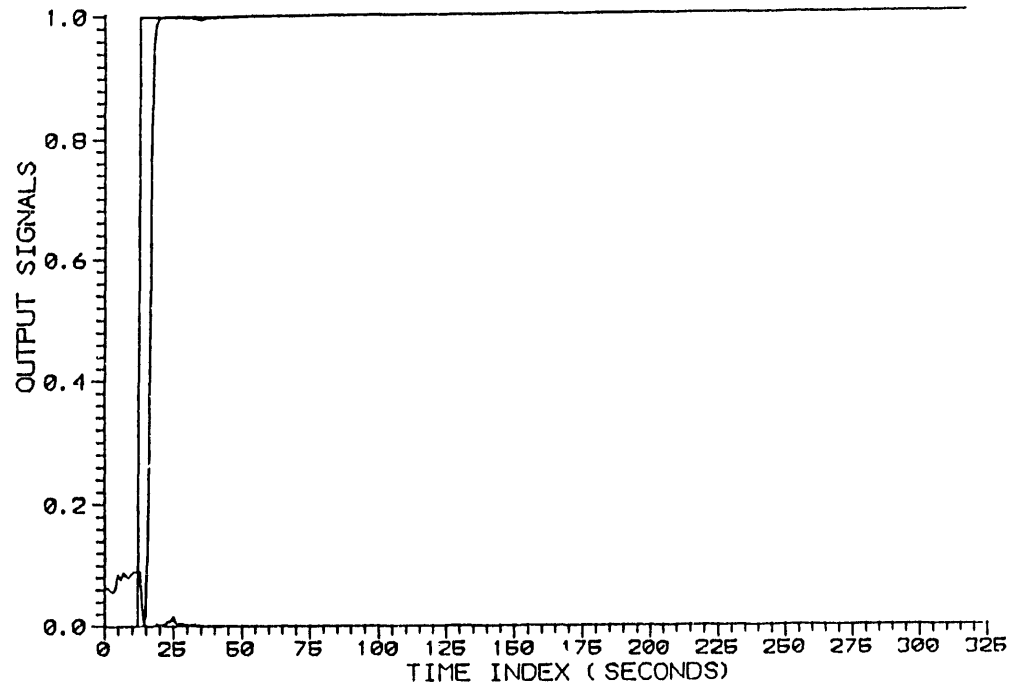


Figure 4.7: Output node activations compared to desired responses for Main Feedwater Line Break for 50 input variables

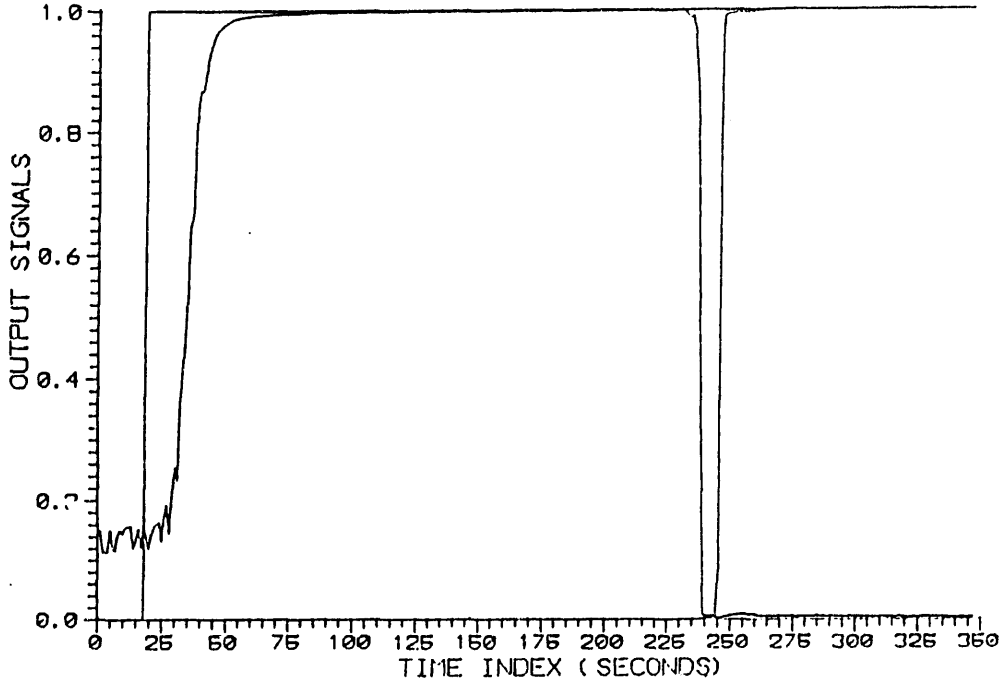


Figure 4.8: Output node activations compared to desired responses for Loss of Feedwater Heating for 50 input variables

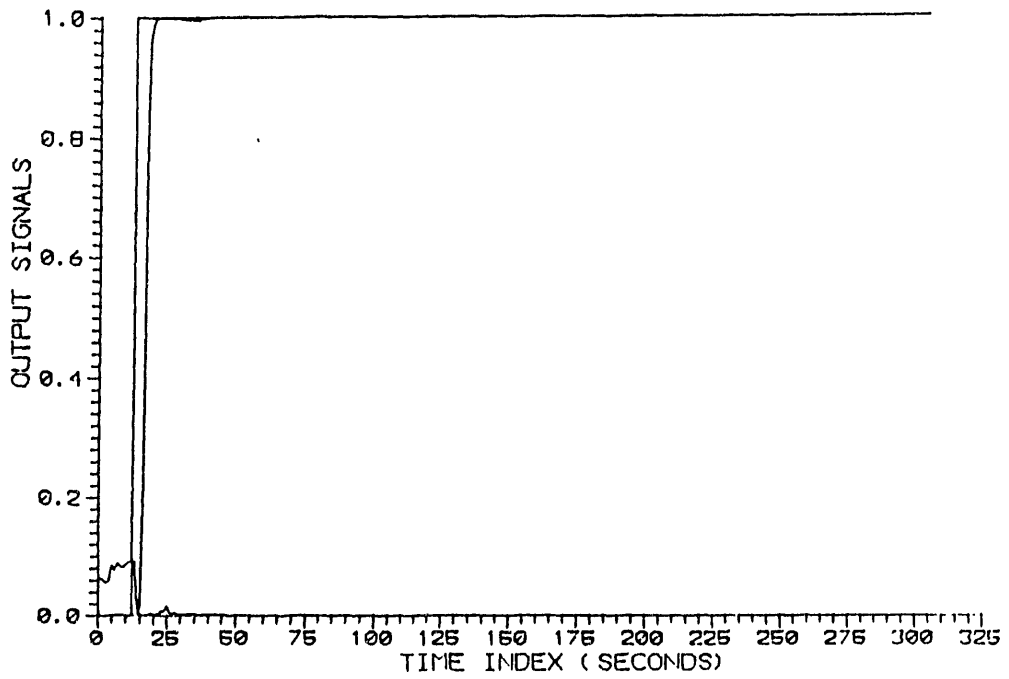


Figure 4.9: Output node activations compared to desired responses for Design basis LOCA for 50 input variables

However, comparisons made between the importances and linear correlation yield no additional information. From inspection, it seems that the derivative importances for the input nodes are independent of their linear correlations.

The statistical correlation values represent a measure of strength of the linear relationship between the input variables and the output variable. In cases where the relationship between two variables in linearly inseparable, such as the exclusive "or," the correlation of linearity will be zero. However, the simplest of neural networks, having 2 inputs, 2 hidden nodes, and 1 output, can easily learn the correct function mapping. Thus, the importance functionality goes beyond the mere linearity imposition of the statistical methods. For a more complete statistical analysis of variances, it is suggested that in the future, one also compare cross-correlations between inputs

as well as fits to higher orders. For further information of the example patterns used, refer to Appendix C.

Table 4.1: Comparison of calculated importances with standard statistical correlations for 20 input variables

| Node | Point ID | Statistical Correlation | Total Importance |
|---|---|---|---|
| 1 | B012 | -0.3253 | 1.2688 |
| 2 | B013 | -0.3195 | 1.1183 |
| 3 | B014 | 0.3274 | 1.0243 |
| 4 | B017 | -0.3087 | 0.1555 |
| 5 | B026 | -0.3317 | 1.3888 |
| 6 | B028 | -0.3308 | 1.3797 |
| 7 | B036 | -0.4333 | 2.9450 |
| 8 | B039 | -0.4333 | 2.8512 |
| 9 | B062 | -0.3308 | 1.3769 |
| 10 | B079 | -0.3344 | 1.3957 |
| 11 | B090 | 0.3018 | 0.2888 |
| 12 | B096 | 0.3051 | 0.3606 |
| 13 | B098 | 0.4187 | 1.2558 |
| 14 | B099 | 0.4177 | 1.2193 |
| 15 | B173 | -0.3051 | 0.8917 |
| 16 | B174 | -0.3151 | 0.7471 |
| 17 | B180 | -0.3071 | 0.1761 |
| 18 | F042 | -0.3084 | 4.8591 |
| 19 | F043 | -0.3088 | 4.7200 |
| 20 | F094 | -0.3050 | 5.5163 |

Table 4.2: Comparison of calculated importances with standard statistical correlations for 33 input variables

| Node | Point ID | Statistical Correlation | Total Importance |
|---|---|---|---|
| 1 | A041 | -0.2935 | 0.48776 |
| 2 | B000 | -0.2938 | 0.41962 |
| 3 | B012 | -0.3253 | 0.05833 |
| 4 | B013 | -0.3195 | 0.07448 |
| 5 | B014 | 0.3274 | 0.02450 |
| 6 | B017 | -0.3087 | 0.07712 |
| 7 | B022 | -0.2970 | 0.13947 |
| 8 | B026 | -0.3317 | 0.06203 |
| 9 | B028 | -0.3088 | 0.07300 |
| 10 | B034 | -0.2812 | 0.09598 |
| 11 | B036 | -0.4333 | 0.15440 |
| 12 | B038 | -0.2812 | 0.09771 |
| 13 | B039 | -0.4333 | 0.15092 |
| 14 | B062 | -0.3308 | 0.05752 |
| 15 | B063 | -0.2975 | 0.15663 |
| 16 | B064 | -0.2973 | 0.14713 |
| 17 | B065 | -0.2976 | 0.14932 |
| 18 | B066 | -0.2977 | 0.15346 |
| 19 | B079 | -0.3344 | 0.06824 |
| 20 | B090 | 0.3018 | 0.06158 |
| 21 | B091 | 0.2864 | 0.04850 |
| 22 | B093 | 0.2864 | 0.04551 |
| 23 | B096 | 0.3051 | 0.06021 |
| 24 | B098 | 0.4187 | 0.11982 |
| 25 | B099 | 0.4177 | 0.11839 |
| 26 | B150 | 0.2813 | 0.01823 |
| 27 | B151 | 0.2815 | 0.02762 |
| 28 | B173 | -0.3051 | 0.06919 |
| 29 | B174 | -0.3151 | 0.07763 |
| 30 | B180 | -0.3071 | 0.06836 |
| 31 | F042 | -0.3084 | 0.18910 |
| 32 | F043 | -0.3088 | 0.18048 |
| 33 | F094 | -0.3050 | 0.03327 |

Table 4.3: Comparison of calcualted importances with standard statistical correlations for 50 input variables

| Node | Point ID | Stat. Corr. | Total Import. | Node | Point ID | Stat. Corr. | Total Import. |
|---|---|---|---|---|---|---|---|
| 1 | A041 | -0.2935 | 0.060518 | 26 | B091 | 0.2864 | 0.016000 |
| 2 | A091 | -0.2752 | 0.002451 | 27 | B092 | 0.2416 | 0.009549 |
| 3 | B000 | -0.2938 | 0.051671 | 28 | B093 | 0.2864 | 0.023930 |
| 4 | B012 | -0.3253 | 0.026765 | 29 | B094 | 0.2746 | 0.032595 |
| 5 | B013 | -0.3195 | 0.007345 | 30 | B095 | 0.2460 | 0.027561 |
| 6 | B014 | 0.3274 | 0.019327 | 31 | B096 | 0.3051 | 0.041755 |
| 7 | B017 | -0.3087 | 0.005504 | 32 | B098 | 0.4187 | 0.126080 |
| 8 | B022 | -0.2970 | 0.048758 | 33 | B099 | 0.4177 | 0.117940 |
| 9 | B023 | -0.2149 | 0.017363 | 34 | B103 | 0.2049 | 0.019862 |
| 10 | B026 | -0.3317 | 0.024213 | 35 | B104 | 0.2047 | 0.019110 |
| 11 | B028 | -0.3088 | 0.013551 | 36 | B126 | 0.2332 | 0.041600 |
| 12 | B030 | -0.2178 | 0.288540 | 37 | B150 | 0.2813 | 0.027112 |
| 13 | B032 | -0.2323 | 0.254790 | 38 | B151 | 0.2815 | 0.031186 |
| 14 | B034 | -0.2812 | 0.019179 | 39 | B162 | 0.2478 | 0.011887 |
| 15 | B036 | -0.4333 | 0.026897 | 40 | B163 | 0.2478 | 0.008477 |
| 16 | B038 | -0.2812 | 0.016825 | 41 | B164 | -0.2374 | 0.023876 |
| 17 | B039 | -0.4333 | 0.038425 | 42 | B166 | -0.2685 | 0.083582 |
| 18 | B062 | -0.3308 | 0.022804 | 43 | B168 | -0.2576 | 0.027445 |
| 19 | B063 | -0.2975 | 0.066590 | 44 | B171 | -0.2353 | 0.025182 |
| 20 | B064 | -0.2973 | 0.066933 | 45 | B173 | -0.3051 | 0.018833 |
| 21 | B065 | -0.2976 | 0.072434 | 46 | B174 | -0.3151 | 0.005950 |
| 22 | B066 | -0.2977 | 0.075554 | 47 | B180 | -0.3071 | 0.012081 |
| 23 | B079 | -0.3344 | 0.016847 | 48 | F042 | -0.3084 | 0.048562 |
| 24 | B089 | 0.2658 | 0.011456 | 49 | F043 | -0.3088 | 0.048659 |
| 25 | B090 | 0.3018 | 0.042327 | 50 | F094 | -0.3050 | 0.033714 |

# CHAPTER 5. CONCLUSIONS

## Summary

This thesis demonstrates the feasibility of the present research project because it shows that an artificial neural network can be applied toward the classification of operational transients in a nuclear power plant. A neural network can be trained successfully on simulated data so that it correctly identifies the accidents. The goal for the present work is to explore more efficient learning techniques, particularly by the data reduction methods described and by using a large number of plant input variables. Using standard statistical methods and the importance function described in this research, a good determination of which input variables are most important for recognizing the transients of interest in a nuclear power plant can be made.

The process of training three independent networks demonstrates two interesting effects. For fully-connected feed-forward four-layer networks, increasing the number of input variables does not necessarily guarantee better network learning. At least during training, it appears that too few input variables may not provide the network enough information by which to generalize the four conditions. On the other hand, providing a large number of inputs to the network may lead to presenting the network misleading information that may lead to some confusions during training. This was evident in the three networks' respective training times. Moreover, the recalls

demonstrate that the recall performance of a smaller neural network may actually be better than a larger network. The graphs in Chapter 4 of the main feedwater line break accident for each number of input variables clearly illustrates this proposition. Furthermore, the smaller network trained on one fewer example than the next larger one.

It is shown by this work that to make a complete choice of input variables to train an artificial neural network, it is necessary to use the importance function as well as the usual statistical correlations. The statistical methods are not enough to discern between useful and redundant input nodes for a neural network. In terms of the diagnostic advisor project, it is felt that a method has been derived for choosing which nuclear power plant variables are necessary to train for particular anomalies. The significance of this process is the ability to eliminate unnecessary or redundant input variables. For small training problems this is not a large concern, but for a diagnostic system using an initial pool of 300 or more plant variables, it will be necessary to trim this number to a more reasonable amount. Else, the architecture of a network having 300 inputs might require training times that are astronomical.

## Possible Future Work

Further refinement of the importance function would prove useful toward understanding neural network design and in applications toward input data reduction and sophisticated dynamic node architecture learning schemes. For a more complete examination of nodal importances, it is felt a comprehensive statistical evaluation be performed that includes linear and higher-order correlations. Specifically, Information Theory may prove useful toward developing a more comprehensive importance

function. The amount of information (in bits), using an understanding of entropy, being passed through a neural network may demonstrate which input nodes are providing the most useful information.

A neural network fault-diagnostic advisor, once trained and tested, can be adopted within the nuclear power plant simulating facility at DAEC. Having a fast, effective diagnostic system, such as the one proposed, will be of great benefit to the power industry by providing one more automatic safety check on the plant.

Additionally, such analysis, using the derivative importance function described here (or perhaps using a new importance function), of plant variables for a fully-developed diagnostic system may prove useful to control room operators and instructors by suggesting which variables to monitor most closely in order to alert them of a problem.

# BIBLIOGRAPHY

[1] Adams, J. Nuclear Reactor Supervisor, Department of Mechanical Engineering, Iowa State University, personal communication, July/August 1991.

[2] Amit, D. J. "Attractor Neural Networks and Biological Reality: Associative Memory and Learning," *Future Generation Computer Systems*, 6, North-Holland, 1990 (111-119).

[3] Bartlett E. B. and A. Basu. "A Dynamic Node Architecture Scheme for Backpropagation Neural Networks," *Intelligent Engineering Systems Through Artificial Neural Networks,* C. I. Dagli, et al. editors, ASME Press, 1991 (101-106).

[4] Bartlett, E. B. and R. E. Uhrig. "Nuclear Power Plant Status Diagnostics Using Artificial Neural Networks," *Proceedings of the American Nuclear Society Meeting on Frontiers in Innovative Computing for the Nuclear Industry*, September 1991 (644-653).

[5] Bartlett, E. B. "Nuclear Power Plant Status Diagnostics Using Simulated Condensation: An Auto-Adaptive Computer Learning Technique," The Universtiy of Tennessee at Knoxville, University Microfilms, Ann Arbor, Michigan, Ph.D. Dissertation, August 1990.

[6] Bartlett, E. B. and R. E. Uhrig. "A Self-Optimizing Stochastic Dynamic Node Learning Algorithm for Layered Neural Networks," *Proceedings of the IEEE INNS International Joint Conference on Neural Networks*, 2, July 1991 (A-947).

[7] Beyer, W. H. *CRC Standard Mathematical Tables,* 28th edition, CRC Press, Inc., Boca Raton, Florida, 1987.

[8] Caudill, M. "Neural Networks Primer, Part 1," *AI Expert,* December 1987 (46-52).

[9] Caudill, M. "Neural Networks Primer, Part 2," *AI Expert*, February 1988 (55-61).

[10] Caudill, M. "Neural Networks Primer, Part 3," *AI Expert*, June 1988 (53-59).

[11] Caudill, M. "Neural Networks Primer, Part 4," *AI Expert*, August 1988 (61-67).

[12] Caudill, M. "Neural Networks Primer, Part 5," *AI Expert*, November 1988 (57-65).

[13] Caudill, M. "Using Neural Nets: Representing Knowledge, Part 1," *AI Expert*, December 1989 (34-41).

[14] Caudill, M. "Using Neural Nets: Fuzzy Decisions, Part 2," *AI Expert*, April 1990 (59-64).

[15] Caudill, M. "Using Neural Nets: Fuzzy Cognitive Maps, Part 3," *AI Expert*, June 1990 (49-53).

[16] Caudill, M. "Using Neural Nets: Making an Expert Network, Part 4," *AI Expert*, July 1990 (41-45).

[17] Caudill, M. "Using Neural Nets: Diagnostic Expert Nets, Part 5," *AI Expert*, September 1990 (43-46).

[18] Cheney, W. and D. Kincaid. *Numerical Mathematics and Computing*, Second edition. Brooks-Cole Publishing Company, Wadsworth, Inc. Manterey, California, 1985.

[19] Davis, G. B. and T. R. Hoffmann. *FORTRAN 77: A Structured, Disciplined Style*, Third edition, McGraw-Hill Book Company, Inc., New York, 1988.

[20] Deutsch, R. W., R. J. Howarth, and A. M. Mangin. *Introduction to Boiling Water Reactor Nuclear Power Plants*, General Physics Corporation, Columbia Maryland, Second Printing, July 1980.

[21] Engel, C. W. and M. Cran. "Pattern Classification: a Neural Network Competes with Humans," *PC AI*, May/June 1990 (20- 23, 61).

[22] Freeman, W. J. "The Physiology of Perception," *Scientific American*, February 1991 (78-85).

[23] Freund, J. E. *Statistics, a First Course*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1970.

[24] Gould, J. *Malfunction Cause and Effects Report,* Task no. 06000004, Duane Arnold Energy Center, Iowa Electric Power and Light Co., Cedar Rapids, Iowa, March 1991.

[25] Gould, J. *Updated Description of Initiating Events,* Task no. 04000103, Duane Arnold Energy Center, Iowa Electric Power and Light Co., Cedar Rapids, Iowa, June 1991.

[26] Hecht-Nielsen, R. "Theory of the Backpropagation Neural Network," *Proceedings of the International Joint Conference on Neural Networks,* 1, 1989 (593-605).

[27] Hillman, D. V. "Integrating Neural Nets and Expert Systems," *AI Expert,* June 1990 (54-59).

[28] Hirose, Y., K. Yamashita, and S. Hijiya. "Back-Propagation Algorithm Which Varies the Number of Hidden Units," *Neural Networks,* 4, 1991 (61-66).

[29] Hopfield, J.J. "Neural Networks and Physical systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Sciences 79,* 1982 (460-464).

[30] Hopfield, J.J. "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *Proceedings of the National Academy of Sciences 81,* 1984 (579-583).

[31] James, W. *Psychology (Briefer Course),* Holt, New York, 1890.

[32] Kardos, C. *DAEC Process Computer Points List,* Duane Arnold Energy Center, Cedar Rapids, Iowa, February 1991.

[33] Kohonen, T. "The Self-Organizing Map," *Proceedings of the IEEE,* 78, September 1990 (1464-1477).

[34] Lanc, T. L. "Determining the Importance of Input Variables to a Neural Network Fault-Diagnostic Advisor for Nuclear Power Plants," *Twenty-eighth Annual Power Affiliate Report,* Electric Power Research Center and Affiliate Research Program, Iowa State University, Ames, Iowa, May 1991 (293-304).

[35] Lippmann, R. P. "An Introduction to Computing with Neural Nets," *IEEE Acoustics Speech and Signal Processing Magazine,* 4, April 1987 (4-22).

[36] McCulloch W. S. and W. Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics,* 5, 1943 (115-133).

[37] Nolfi, S. and D. Parisi. "Learning to Understand Sentences in a Connectionist Network," *Proceedings of the IEEE International Conference on Neural Networks*, 2, July 1988 (215-219).

[38] Ott, L. *An Introduction to Statistical Methods and Data Analysis*, Third edition, PWS-Kent Publishing Company, Boston, 1984.

[39] Serra, R. and G. Zanarini. *Complex Systems and Cognitive Processes*, Springer-Verlag, Berlin, 1990 (73-193).

[40] Sietsma, J. and R.J.F. Dow. "Creating Artificial Neural Networks That Generalize," *Neural Networks*, 4, 1991 (67-79).

[41] Swokowski, E. W. *Calculus with Analytic Geometry*, Alternate edition, PWS Publishers, Wadsworth, Inc., Boston, 1983.

[42] *Updated Final Safety Analysis Report*, Chapter 15, Duane Arnold Energy Center, Iowa Electric Power and Light Co., Cedar Rapids, Iowa, June 1984.

[43] Vest, D. Control Room Simulator Instructor, Duane Arnold Energy Center, Iowa Electric Power and Light Co., Palo, Iowa, personal communication, 1991.

[44] Werbos, P. J. "Building and Understanding Adaptive Systems: A Statistical/Numerical Approach to Factory Automation and Brain Research," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17, January/February 1987 (7-20).

[45] Widrow, B. and M. A. Lehr "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proceedings of the IEEE*, 78, September 1990 (1415-1441).

[46] Zill, D. G. *A First Course in Differential Equations with Applications*, Third edition, Prindle, Weber, and Schmidt, Boston, Massachusetts, 1986.

# END

DATE
FILMED
5/14/93