

To the Graduate Council:

I am submitting herewith a thesis written by James Brown Golden III entitled 'A Neural Network for the Analysis of Test Data from the Aeropropulsion Systems Test Facility.' I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Computer Science.

Bruce Whitehead
Bruce Whitehead, Major Professor

We have read this thesis
and recommend its acceptance:

KR Kunkle

Thomas Di

Accepted for the Council:

Cowminkel
Vice Provost
and Dean of the Graduate School

A NEURAL NETWORK FOR THE ANALYSIS OF TEST DATA
FROM THE AEROPROPULSION SYSTEMS TEST FACILITY

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

James Brown Golden III

August 1990

Copyright ©James B. Golden III, 1990

All rights reserved

ACKNOWLEDGEMENTS

This research was performed in part under a contract from Sverdrup Technology, contract number A9B-015. I am indebted to Dr. Virgil Smith, Art Mantone and Lynn Moran of Sverdrup Technology for all their assistance in contract award, data collection and a free flow of ideas. I owe a debt of gratitude to Dr. Bruce Whitehead who exhibited both patience and enthusiasm throughout the course of this thesis and oral exams. It was a learning experience for both of us. I would like to thank the other members of my committee, Dr. Moonis Ali and Dr. Ken Kimble who agreed to review my work and help ensure that it met their high academic standards. I would like to thank my friends, colleagues and coworkers who were forced to put up with me throughout, especially Lisa Pereira who endured it from the beginning. This thesis is dedicated to my mother who taught me to dream, my father who taught me to work, and to Laura who understands the value of both.

ABSTRACT

With the advent of the USAF's Advanced Tactical Fighter and NASA's National Aerospace Plane, demands for concise test data reduction and interpretation will increase beyond the capabilities of current methodologies. As mission complexity increases it becomes apparent that real time data analysis for flight safety, mission control and test conduct becomes a necessary tool.

A neural network is a biologically inspired mathematical model, which can be represented by a directed graph, that has the ability to learn through training. Neural networks have many advantages over current aviation computing systems including the ability to learn and generalize from their environment. Neural networks are excellent for parameter estimation and recognizing patterns in signal data. This thesis discusses a prototype system designed and implemented at the University of Tennessee Space Institute to discover patterns in test data from an engine test cell in order to determine if any part of the system is in failure. The results of this research show that a neural network can be used for fault diagnosis in an engine test cell when the problem of fault monitoring and diagnosis is seen as one of pattern recognition. A two layer semilinear feed-forward neural net is able to separate simulated sensor data into normal and abnormal classes and the addition of a hidden layer makes the network more resistant to noise and improves the ability of the network to classify the type of fault that is occurring.

Neural networks offer test personnel a new tool for the analysis of critical test data. A general working network model can be applied to any system where sensor data is evaluated and fault diagnosis is critical. Significant benefit to the aerospace community is gained by improved safety of test engineers by rapid detection of

faulty conditions. This will be of extreme importance as testing begins on more advanced engine systems such as the National Aerospace Plane and the Advanced Tactical Fighter.

Table of Contents

Chapter	Page
I Introduction	1
1.1 Problem Statement	1
1.2 Approach	1
II Problem Analysis	3
2.1 The ASTF Domain	3
2.2 Fault Monitoring & Diagnosis	6
III Theoretical Development	7
3.1 Neural Networks	7
3.2 Learning Algorithms	11
3.3 The Networks Ability to Generalize from Example	14
3.4 Adaptive Pattern Recognition	15
IV Program Development	17
V Results	24
5.1 A Two Layer Semilinear Feed-Forward Neural Network:	
A Generalized Perceptron Approach	24
5.2 A Three Layer Semilinear Feed-Forward Neural Network:	
A Multilayer Perceptron Approach	27

VI Meaningfulness in the Individual and System Error	33
VII The Networks Ability to Handle Noise	40
VIII The Networks Ability to Classify Types of Faults	44
IX Conclusions	48
Bibliography	50
Appendixes	53
Appendix A: A Mathematical Derivation of a Neural Network	54
Appendix B: A Mathematical Derivation of the Backpropagation Learning Algorithm	57
Vita	61

List of Figures

Figure	Page
1. ASTF Instrumentation and Control Interface Diagram	5
2. A Feed-Forward Neural Network Architecture	8
3. A Single Neural Network Processing Element	9
4. Hyperplanes Formed by Different Training Set Combinations of Normal and Faulty Data in Two Dimensions (Sensor PO and Sensor PL3H) .	22
5. A System Architecture for Analyzing ASTF Test Data	23
6. Hyperplanes Formed by a Two Layer Neural Network in Three Dimensions (Sensor PO, Sensor PL3H and Sensor LVC1B)	25
7. Final Connection Weights for a Two Layer Neural Network	25
8. Hyperplanes Formed by Three Hidden Nodes Graphed in Three Dimensions (Sensor PO, Sensor PL3H and Sensor TL1H)	28
9. Final Connection Weights From Input Nodes to Nodes of Hidden Layer Based on Number of Hidden Nodes	30
10. Final Output Values for Test Data Based on the Number of Nodes in the Hidden Layer	31
11. Total System Error for a Two Layer Neural Network with $\eta = 0.9$ Over 100 Iterations	34
12. Total System Error for a Two Layer Neural Network with $\eta = 0.7$ Over 100 Iterations	34

13. Total System Error for a Two Layer Neural Network with $\eta = 0.5$ Over 100 Iterations	35
14. Total System Error for a Two Layer Neural Network with $\eta = 0.2$ Over 100 Iterations	35
15. Individual Error for a Two Layer Neural Network with $\eta = 0.7$ Over 100 Iterations for the Node Representing Sensor PL3H	36
16. Individual Error for a Two Layer Neural Network with $\eta = 0.5$ Over 100 Iterations for the Node Representing Sensor PL3H	36
17. Individual Error for a Two Layer Neural Network with $\eta = 0.2$ Over 100 Iterations for the Node Representing Sensor PL3H	37
18. Individual Error for a Two Layer Neural Network with $\eta = 0.7$ Over 100 Iterations for the Node Representing Sensor LVC1B	37
19. Individual Error for a Two Layer Neural Network with $\eta = 0.2$ Over 100 Iterations for the Node Representing Sensor LVC1B	38
20. A Two Layer Neural Network Trained and Tested with Noisy Data Over 100 Iterations	42
21. A Three Layer Neural Network with Three Hidden Nodes Trained and Tested with Noisy Data Over 100 Iterations	43
22. A Two Layer Neural Network Classification of Types of Faults with 10% Noise	46
23. A Three Layer Neural Network with Three Hidden Nodes Classification of Types of Faults with 10% Noise	47

SYMBOL TABLE

n	number of elements in layer x
m	number of elements in layer y
x_k	k th component of the input vector x
y_j	j th node in the processing layer y
z_i	i th node in the processing layer z
w_i	weight vector into node z_i
v_j	weight vector into node y_j
y_j^{in}	total amount of input activation to node y_j
y_j^{out}	output activation computed by node y_j
z_i^{in}	total amount of input activation to node z_i
z_i^{out}	output activation computed by node z_i
z_i^{desired}	desired output from node z_i
$\mathbf{x}(1), \mathbf{x}(2), \dots, \mathbf{x}(q)$	series of input vectors $\mathbf{x}(q)$, $q < Q$ used to train the network
Q	a set of training patterns
q	a single training pattern in the set Q
$ Q $	number of vectors in the training pattern
z_i^{error}	error from node z_i (differences between actual output and desired output)
z_i^{MSE}	mean squared error for node z_i over the training set
z^{TSE}	total system error for all nodes in layer z
w_{ij}	connection weight to node z_i from node y_j
Δw_{ij}	change the connection weight specified by the learning rule
α	constant in gradient descent algorithm from which learning rate is derived

v_{jk}	connection weight to node y_j from node x_k
Δv_{jk}	change in connection weight between node x_k and node y_j
δ_{ij}	error between layer y and layer z
δ_{jk}	error between layer x and layer y
η	learning rate used in the back propagation algorithm
f	sigmoid function used to compute output activation of a node from the input activation
g	function to be approximated by a neural network
h	function computed by a neural network

CHAPTER I

INTRODUCTION

(1.1) Problem Statement

Simulated flight tests conducted in the Aeropropulsion Systems Test Facility (ASTF) produce large quantities of sensor data at rates that overwhelm flight test personnel. The task of monitoring signals from test cell sensors and watching for the onset of possible fault situations requires significant experience and a number of test cell personnel. Because of the variety and quantity of test cell data, extensive knowledge and experience is required for accurate differentiation of data types. A neural network architecture that could accurately diagnose the onset of fault situations and alert test cell engineers would result in a significant savings in personnel, downtime because of damaged equipment, and improve test cell safety.

(1.2) Approach

The main approach of this research is to design and analyze a network architecture to monitor sensor data from the ASTF to detect data that falls outside learned parameters, thus signifying a fault. In this research, diagnostic problems in general are conceptualized as a mapping of an input pattern (representing sensor data) to an output pattern which is interpretable as a diagnosis.

A neural network is a biologically inspired mathematical model, which can be represented by a directed graph, that has the ability to learn through training. The action of a neural net may be viewed principally as a mapping through which points in the input space are transformed into corresponding points in the output

space on the basis of designated attribute values (of which class membership might be one) (Pao,1989). It is the goal of this research to design a neural network architecture that can classify ASTF test data as part of a nominal engine test or as data from a system in failure. The words neural network, network, net, and network model will be used interchangeably throughout this thesis.

Three major subsystems that make up the ASTF instrumentation and control systems are the Plant Instrumentation and Control System (PICS), Test Instrumentation System (TIS), and the Automatic Test Control System (ATCS). The ATCS provides facility and engine status monitoring for the detection of unsafe operating conditions having a potential for causing damage to the engine or the test facility (Polce, 1982). Currently, many engineers are needed to monitor data output from the ATCS to check for abnormal operating conditions during a test.

The ATCS is used to control test conditions inside the test cell, control of engine operating conditions, and the detection of abnormal plant and engine operating conditions. Because these tend to be problems in parameter estimation and the detection of events outside expected parameters, the ATCS was chosen as the initial test bed for the network.

CHAPTER II

PROBLEM ANALYSIS

(2.1) The ASTF Domain

The Aeropropulsion Systems Test Facility is designed for testing airbreathing engine propulsion systems over a wide range of altitudes and Mach numbers. Simulated flight tests conducted in the facility will determine the operational and performance characteristics of aeropropulsion systems; thus, the time required for flight tests is shortened, and the risks and expense of flight tests are minimized (Polce, 1982).

The ASTF was designed to permit development and surveillance tests of large airbreathing, jet propulsion engines throughout their operational envelopes by providing the capability for true simulation of flight altitude and flight Mach number conditions. The facility is capable of performing direct-connect testing of turbojet and high-bypass (8:1) turbofan engines up to 75,000 pound-rated, sea level, static thrust throughout the appropriate engine operating envelopes (Polce, 1982). The facility is designed to allow testing in both direct-connect and freejet modes.

In Polce's (1982) thesis he describes different types of tests that will be conducted in the ASTF. These test types require the capability of transient testing, e.g., engine power transients, flight profile transients, or combinations of the two. The facility operating conditions and the corresponding engine operating conditions for testing in the ASTF are:

- (1) Steady Environment — Steady Engine Power Setting.
- (2) Steady Environment — Transient Engine Power Setting.

- (3) Transient Environment — Steady Engine Power Setting.
- (4) Transient Environment — Transient Engine Power Setting (Mission Profile Simulation) (Polce, 1982).

Looking at the testing requirements that the ASTF must perform, it becomes obvious that systems within the test cell must be able to alter both pressures and temperatures, along with a number of valve openings, to regulate test cell conditions. As the system increases in complexity, it becomes an overwhelming task to supervise all the sensors that monitor the facility. Instrumentation and data handling equipment must be provided to measure, transmit, acquire, display, and process those measurements which are required to determine the performance of the test article. The measurement capability for each test cell includes 970 pressure measurements, 820 temperature measurements, and 152 miscellaneous measurements, such as forces, flows, speeds, geometries, vibrations, accelerations, and strains (Polce, 1982).

As stated in (1.2), three major subsystems make up the ASTF instrumentation and control system. They are the Plant Instrumentation and Control System (PICS), The Test Instrumentation System (TIS) and the Automatic Test Control System (ATCS) [see Figure 1].

The ATCS performs a number of control functions inside the test cell. The ATCS is used to:

- (1) Control test conditions.
- (2) Control engine operating conditions.
- (3) Coordinate control of test and engine operating conditions with plant configuration changes.

- (4) Detect abnormal plant and engine operating conditions and their response thereto.
- (5) Communicate with facility operators (Polce, 1982).

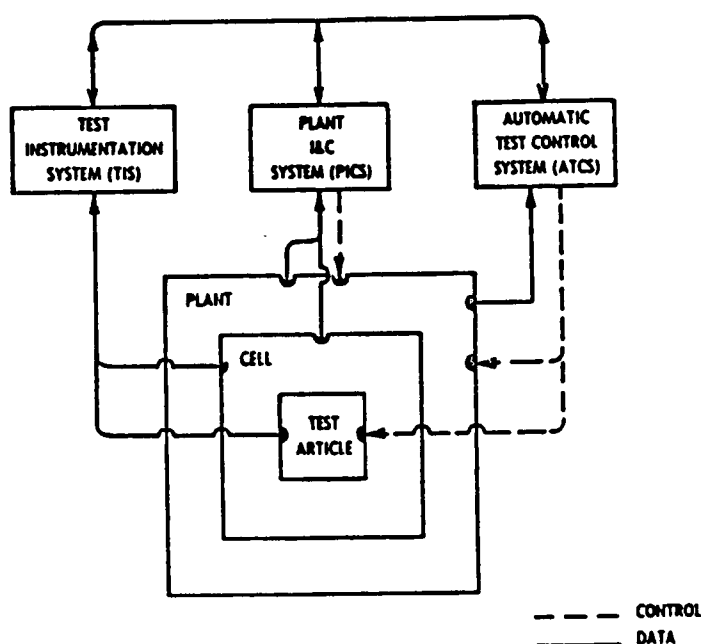


Figure 1. ASTF Instrumentation and Control Interface Diagram.

Because of the importance of the ATCS and because it is used for detection of abnormal conditions inside the test cell, this research concentrates on applying neural network technology to enhancing the ATCS.

The ATCS controls the parameters which establish a desired test condition by positioning forty-six of the facility configuration and control valves. The ATCS is used to control various engine controls and the interface to the test cell engineers is effected at a panel in the test building control room. Pertinent test information is communicated to test cell personnel through four CRT terminals. It is the end goal of this research to establish a better means of monitoring test data from the ATCS.

(2.2) Fault Monitoring and Diagnosis

In order to apply neural networks to fault monitoring and diagnosis, the task of fault detection must be seen as one of pattern recognition. In approaching the ASTF we will say that fault detection is observing data that fall outside expected parameters previously regarded as "normal" operation. Sensor data that fall outside normal boundaries do not necessarily indicate that the system is beginning to fail, but it does indicate that an abnormal occurrence is taking place inside the system that is being studied and warrants further attention. As stated previously, in this research diagnostic problems are viewed as a mapping of an input pattern representing sensor data to an output pattern which is interpretable as a diagnosis.

Analysis of a complex system should use an opportunistic strategy — there is no computationally feasible "legal move generator" that defines the space of solutions in which pruning and steering take place as in classical AI systems (Nii and Feigenbaum, 1978). Bits and pieces of information must be used as the opportunity arises to (relatively) slowly build a coherent picture of the world (system). This is one way in which neural networks are better than current AI methods at analyzing signal data from complex systems.

CHAPTER III

THEORETICAL DEVELOPMENT

(3.1) Neural Networks

As stated previously, we will define a neural network as a biologically inspired mathematical model, which can be represented by a directed graph, that has the ability to learn through training. A neural network model typically has eight major aspects. They are:

- (1) A set of processing units.
- (2) A state of activation.
- (3) An output function for each unit.
- (4) A pattern of connectivity among the units.
- (5) A propagation rule for propagating patterns of activities through the network of connectivities.
- (6) An activation rule for combining inputs to produce a new activation level for a unit.
- (7) A learning rule whereby patterns of connectivity are modified by experience.
- (8) An environment within which the system must operate (Rumelhart and McClelland, 1988).

These will each be described in detail below.

Each processing unit in a neural network model represents some feature-like entity such as an engine sensor. It is the pattern as a whole coming from the system to be studied that is the meaningful level of analysis. The purpose of a unit is to receive input from its neighbors and as a function of the inputs it receives

to output a value to its neighbors. There are three types of processing units in a neural network model: the input units which receive data from sources external to the model, one or more layers of hidden units which are internal to the model only, and a series of one or more output units which sends signals out of the model to effect motoric systems or influence other external systems. Figure 2 shows a three layer feed-forward neural network.

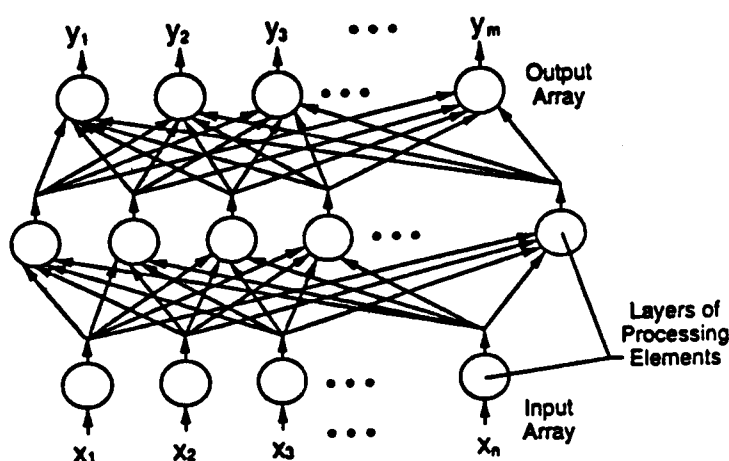
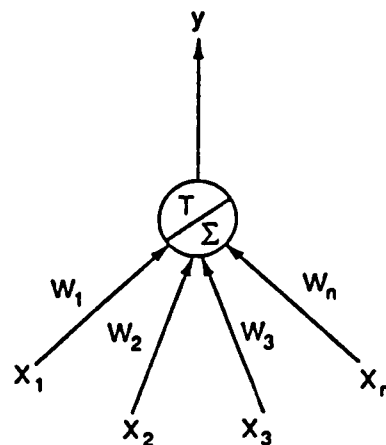


Figure 2. A Feed-Forward Neural Network Architecture

A state of activation is the representation of the state of the system at some time t representing the pattern of activity over the set of processing units. It is the pattern of activation over the set of units that captures what the system is representing at any time.

Associated with each unit y_i there is an output function $y = w \cdot x$ which maps the current state of activation to some output y_i ; $y_i = \sum_j w_{ij}x_j$. There is often some threshold function so that a unit has no effect on its neighboring unit unless the output function exceeds some certain value (Rumelhart and McClelland, 1988).

The pattern of connectivity constitutes what the system knows and determines how it will respond to some arbitrary input. Neural networks are often called connectionist networks because the computations in the net are largely performed by the connection strengths. The signal coming into a processing unit is the inputs from all incoming units multiplied by a weight and summed to get the overall input to that unit, as in Figure 3.



$$\begin{aligned}\bar{x} &= (x_1, x_2, x_3, \dots, x_n) \\ \bar{w} &= (w_1, w_2, w_3, \dots, w_n) \\ y &= \sum_{i=1}^M x_i w_i > T\end{aligned}$$

Figure 3. A Single Neural Network Processing Element

The total pattern of connectivity can be represented by specifying the weights for each of the connections in the system. Positive and negative weights can be seen as excitatory or inhibitory input respectively. The absolute value of the weight gives the strength of the connection. How to determine appropriate weights is dependent on the learning algorithm.

Propagation in the net is the sum of the net excitatory inputs (the weighted sum of the excitatory inputs to the unit) and the net inhibitory inputs (the weighted

sum of the inhibitory inputs to the unit). (Rumelhart and McClelland, 1988). The activity of a unit is regarded as the degree of confidence that a preferred feature is present. To reach a new state of activation a function f (activation rule) will be used to take x and produce some new state of activation. f should be differentiable in order to use certain learning algorithms.

What makes a network useful in the analysis of complex systems is the ability to modify its connectivity through experience. There are three types of modifications to a network: the development of new connections, the loss of existing connections and modifications of the strengths of connections that already exist in the net. The derivation of these modification (learning) rules will be covered under the section on learning algorithms.

The last aspect of the model is the environment in which it will operate. Formally, we will define this as a time varying stochastic function over the space of the input patterns. More simply; at any point in time, there is some probability that any of the possible set of input patterns is impinging on the input units. The environment for this research is the input vectors composed of sensor data coming from the ATCS in the Aeropropulsion Systems Test Facility.

A description of how a neural network processes information follows: (For a more detailed discussion of neural network functions see Appendix A: A Mathematical Derivation of A Neural Network.) An input signal (stimulus) is received by the nodes in the input layer and produces some activation level in a given input unit, which conveys a signal of proportional strength based on the weighting matrix to its [synaptic] connections, or nodes in the next layer. The global effect is that a pattern of activation across the set of input units produces a distinct

pattern of activations across the set of hidden units (Churchland and Churchland, 1990). The process is repeated again as many times as there are hidden layers. As before, an activation pattern across the hidden units produces a distinct activation pattern across the next layer of nodes. In this way the network becomes a device for transforming some input vector representing sensor data into a uniquely corresponding output vector. It is a device for computing a specific function. Exactly what function it computes is fixed by the global configuration of its synaptic weights (Churchland and Churchland, 1990).

There are a number of procedures for training a network and adjusting the weights. Many learning algorithms have been derived over the last two decades. In choosing a learning algorithm for a system to monitor the ASTF we must realize the function that we want to specify is initially unknown to us because of the myriad possible fault situations. We can impose an unspecifiable function on the network as long as we can supply a set of examples of the desired input/output pairs. The network is then trained until it performs the input to output transformation desired.

(3.2) Learning Algorithms

When feasible, learning is a very attractive alternative to explicit programming. This is particularly true when problems do not lend themselves to systematic programming, such as pattern recognition. All neural network learning algorithms have their basis in the neural learning rule discovered by Hebb in 1949. His rule for "Hebbian learning" in a network is as follows:

When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

Neural Networks can be classified on the basis of the patterns stored and their method of recall. Two labels for networks used as associative memories are autoassociative networks and heteroassociative networks. Autoassociative networks associate a data item with itself. Heteroassociative networks associate two different data items — one is used to recall the other. Heteroassociative type memories must be trained with a data pair representing the input and the desired output pattern associated with it. To store the pattern in a training set each input is presented to the system along with the desired output and some learning rule is used to adjust the weights usually based on a difference equation and error between actual and desired outputs. To recall stored information an unfamiliar pattern from a different set of patterns is presented to the network and the system outputs the second member of the training pair, the associated pattern of output.

A heteroassociative network will be used for fault monitoring of the ASTF. Interpretation of the network's outputs and weights should give insight into the meaning of signals in a complex system. If a network is able to respond to a set of "interesting" patterns then this will form the basis for a feature detector which we will use for detecting faults in our complex system.

The learning rule used for this system to discover normal and abnormal patterns of data from the ASTF is backpropagation of error which was discovered

simultaneously by a number of different research groups in the early 1980s. Backpropagation is a heteroassociative, function estimating, neural network that stores arbitrary analog spatial pattern pairs, using multi-layer error-correction encoding between a target output vector and actual output values from the net.

The learning procedure consists of the net being assigned a random set of weights and receiving a training set pair as input and evaluating the output in a feedforward manner. The initial difference error between target and actual outputs will be quite large. Using the backpropagation procedure, the algorithm calculates a change in weight Δw_{ij} for all w_{ij} in the net for that particular pattern. This procedure is repeated for all patterns in the training set to yield a resulting Δw_{ij} .

$$\Delta w_{ij} = \eta \delta_{ij}(q) y_j(q)$$

$$\text{where } \delta_{ij}(q) = - \sum_q z_i^{\text{error}}(q) \cdot z_i^{\text{out}}(q) [1 - z_i^{\text{out}}(q)]$$

z_i^{desired} is the desired output for node z_i and z_i^{out} is the actual output value of node z_i .

Corrections are made to the weights and the outputs again are evaluated in a feedforward manner based on the number of iterations specified by the system. Discrepancies between actual and target output values again result in evaluation of weight changes (Pao, 1989). If the training is successful, the system error will decrease with the number of iterations the system performs and the procedure will converge to a stable set of connection weights. For a complete derivation of the backpropagation learning rule see Appendix B: Derivation of the Backpropagation Learning Rule.

Backpropagation is not guaranteed to find the global error minimum, only the local error minimum which should be good in enough in determining a fault

situation in the ASTF facility. Backpropagation was chosen for this research because of its ability to store many patterns, its ability to generalize and its ability to perform arbitrary nonlinear mappings (Simpson, 1990).

(3.3) The Network's Ability to Generalize from Example

When considering neural networks for fault monitoring and diagnosis it is important to consider how well the network can generalize from the set of training examples. It is impossible (and impractical) to present the network with all possible examples of a fault. We are expecting the network to learn from a set of examples and be able to generalize for all possible faults.

In diagnosing faults in the ASTF we have an environment, which is a set of sensor values, and call this set X . In this environment we have a concept defined as a function $g : x \rightarrow \{0, 1\}$ which is the presence or absence of a fault. The goal of learning is to produce a hypothesis $h : x \rightarrow \{0, 1\}$ that approximates the concept g (Abu-Mostafa, 1989). In our case h is a pattern recognition system that recognizes the presence of faulty data. To do this we have a number of examples of the concept $(x_1, g(x_1)) \dots (x_n, g(x_n))$, in this case a set of input vectors from the ASTF simulator and the desired outputs. A learning algorithm is one that takes the examples and produces the hypothesis. Performance of the system can be measured by the number of examples needed to produce a good hypothesis.

When backpropagation is used in a feed forward manner, we are guessing that there is a good hypothesis to be gained by setting the weights and thresholds of the network. The set of hypothesis H would then be the set of all functions h that are obtained by setting the weights and thresholds of the net. The learning algorithm picks a hypothesis $h \in H$ that mostly agrees with g on the examples in the

training set. In thinking about the generalizability of the network it is necessary to ask: Does this choice, which is based on the behavior of the examples, hold in general (Abu-Mostafa, 1989).

Generalization can be expected if the number of examples is large enough. Without sufficient number of examples the algorithm cannot produce a good hypothesis. With too many examples the computational task of digesting the examples into a hypothesis proves intractable.

(3.4) Adaptive Pattern Recognition

We will view pattern recognition as the ability to map a series of input patterns into groups of output patterns (such as classes of patterns) or as Pao (1989) defines "a mapping from representation space to interpretation space". The goal of this research with respect to pattern recognition is to determine if a neural network can be used as the operator to carry out this mapping for use in fault diagnosis. In the past, operators were installed in specific pattern recognition systems. We will try to develop a network that can learn the optimal operators through training. The word adaptive means that the system will be able to deal with patterns that are distorted with noise, be able to modify classification rules in accordance with changing circumstances, and be able to self-organize the connectivity of the network in accordance with the structure of pattern data (Pao, 1989).

In order to achieve an optimal operator we must understand the use of discriminants in pattern recognition. A discriminant is a function or an operator that, when applied to a pattern, yields an output that is either an estimate of the class membership of that pattern or an estimate of the values of one or more of the attributes of the pattern (Pao, 1989). Simply put, a discriminant is a type of

operator that produces a mapping from pattern space into attribute space. The discriminant we are after is one that processes a complete pattern at one time rather than one feature at a time. The output of the net can be plotted as a map of decision regions created in the multidimensional space spanned by the input variables. The outputs from a single layer network form half-plane decision regions. A two layer net can form many unbounded regions in the space spanned by the input. The output of multi-layer networks form complex convex regions based upon the intersection of the half-planes formed by each node in the first layer of the network. (Lippman, 1987). These regions, bounded by hypersurfaces, become our discriminants and patterns are classified in accordance whether they are on one side or another of a hypersurface or a set of hyperplanes (Pao, 1989). Using these hyperplanes divides the pattern space into volumes of unique class membership. In this case the classes are simple; normal or faulty operation of the system.

CHAPTER IV

PROGRAM DEVELOPMENT

The system was written in VAX/VMS C on the University's VAX 11/785 using backpropagation code developed by Pao as a foundation (Pao, 1989). The system was written so that different net architectures could be specified including number of layers and number of processing elements. The learning algorithm was the generalized delta rule (backpropagation) in order to learn weights and thresholds with use of training set patterns. Training sets were created from simulated data from an engine test cell simulator based on a mathematical model of the ASTF. The average sampling rate for the data was five times per second and the simulated tests ran between 10 and 20 minutes. Data points were from the 33 ATCS data sensors.

Values of the data points ranged between 0.0 and approximately 600 depending on the sensor and the data being sampled, whether it was a pressure, temperature or a valve opening. Because the data values ranged so widely and the initial connection weights of the network were given a random value between -0.5 and 0.5, the weight vectors would make wide swings around the input vectors as it attempted to converge during training. Therefore the stabilization of the weight vectors would take a long time (large number of iterations). Tests of the network with raw data could not achieve optimal convergence within a limited number of iterations. The network did begin to stabilize, but to achieve the lowest possible system error a large number of iterations were necessary to train the network. This equates to long run times in order for the network to learn from example.

There are two possible solutions to this problem. The first is to randomize

each initial connection weight in a range inversely proportional to the expected magnitude of its input, or to scale the values of each component of the input to be between 0 and 1. For this system the later method was chosen. To achieve an optimal scaling of the input vectors it was important to consider the range of values that a sensor could achieve over a failing system or even over a nominal test with different mission objectives and varying test cell conditions. Data from a temperature sensor might vary 50 degrees even for a test that was running smoothly.

A program was written to take the training set composed of raw data representing both normal and faulty runs of the ASTF, find the maximum and minimum value of each sensor and scale the values using the following method:

$$\text{Scaled Value} = (\text{value} - \text{min}) / (\text{max} - \text{min}) \quad (4.0.0)$$

In this way if the value is equal to the minimum value of the sensor the scaled value will be zero. If the value equals the maximum value of the sensor the scaled value will equal one. The maximum and minimum computed for each sensor was retained in another file so that test vectors that were not part of the training set could be scaled using the same scaling factors as the training set. Although this seems mathematically trivial, it was very necessary in order for the network to converge towards a minimal system error in as few iterations as possible, and to achieve a weight vector that the network could use to produce an accurate diagnosis of the health of the test cell.

Next the desired outputs were specified for each input vector in order to train the network using backpropagation. In order for the network to distinguish between two classes of data, normal and fault, one output node was necessary with

a high value (1.0) if the input vector indicated a fault and a low value (0.0) if the conditions in the test cell were normal. These values were appended to the vectors in the training set depending on whether the vectors represented data known to be coming from a normal ASTF run or from a system that was in failure.

Though variations in the sensor values between a normal test situation and a system in failure might be small, by looking at the complete pattern of input over many sensors a neural network can learn to tell the difference between normal and abnormal data. Initial and subsequent tests of the system confirm this.

Using a training set composed of these vectors the network can be trained with the goal of converging to some near optimum local error minimum that would enable it to differentiate sensor values from a normal test and sensor values from a test where the system is failing. In other words, the network would discover some function that when presented with a new vector (sensor data time slice) would compute some high or low value for the output node. In this way a diagnosis could be achieved as to the status of the ASTF and test cell personnel could be alerted.

There are two things to consider when investigating whether a function can be learned from example:

- (1) Do the examples convey enough information to determine the function?
- (2) Is there a speedy way of constructing the function from the examples? (Abu-Mostafa, 1989).

These questions are crucial to the design of the training set. The training set must contain enough examples to allow the network to determine the function

without trying to represent every known normal or fault situation. In other words, the network must learn to generalize as discussed in section (3.3).

In order to create a training set that is capable of generalization it was necessary to test a number of possible combinations of normal and system-failure vectors. Tests were conducted on the candidate training sets in order to answer the following questions: Does the system trained with the training set converge, and does the network achieve an accurate diagnosis when tested against some arbitrary input? A number of training sets composed of a combination of normal and faulty vectors were used to train a two layer network composed of 22 input nodes and one output node. The combinations ranged from 10 to 40 normal vectors and 10 to 40 faulty vectors. All combinations achieved convergence. However, not all combinations were able to generalize for new input. A few of those combinations are discussed here:

A training set composed of 10 normal vectors followed by 10 fault vectors converged to a normalized system error of 0.000022. A training set of 20 normal vectors followed by 20 fault vectors converged to a normalized system error of 0.000020. Likewise a 30/30 split achieved a system error of 0.000049 and a split of 40/40 converged to a 0.000032 system error. Any of these system errors would be satisfactory for diagnosing the status of the test cell. The crucial test was one of generalizability. Only a network that learned from a training set with at least 40 system-failure vectors was able to accurately classify some arbitrary input as being from a nominal ASTF run or an abnormal one. A system trained on a training set with less than 40 vectors representing faulty sensor data was able to achieve convergence, but all arbitrary input was diagnosed as normal. Likewise, a training set with 40 fault vectors and less than 20 normal vectors was

unable to distinguish between normal and faulty patterns of data. To achieve generalizability, it therefore appears that the training set must contain at least 20 normal vectors and at least 40 vectors from a test cell in failure.

To more clearly demonstrate why one combination of vectors in a training set might classify better than another we can graph the hyperplanes formed by a network trained with a particular training set combination. Figure 4 shows the hyperplanes formed by a two layer network in two dimensions (sensors P0 and PL3H—sensors with the strongest connection strengths). When using different training set configurations, it can be seen that the training set composed of 20 normal vectors and 40 fault vectors separates the classes of data reasonably well. Training sets composed of less than 40 fault vectors were never able to separate the two classes of data. A training set with more than 20 normal vectors tends to push the hyperplane towards the origin. Based on these results this was the training set used by the network for this research.

The training set was presented to the network over some number of iterations (1000) and the weights and thresholds allowed to stabilize. The program was designed to allow the maximum normalized system error to be input and the iterations will continue until the desired system error is reached, or the iterations are completed.

The user interface of the program allows the user to specify whether the system will be used for learning (training) or output generation (testing). Because the code uses dynamic allocation the user can input the number of features in the input pattern (input nodes) and the total number of input samples in the training set. The code also enables the user to specify the number of hidden layers in

the network and the number of nodes in those hidden layers. The code allows variations of the learning and momentum rate for the network based on user input. Since the learning rule uses backpropagation of error it was necessary to be able to vary not only the maximum total error for the system, but the maximum individual error for each node as well. The system architecture is shown in Figure 5.

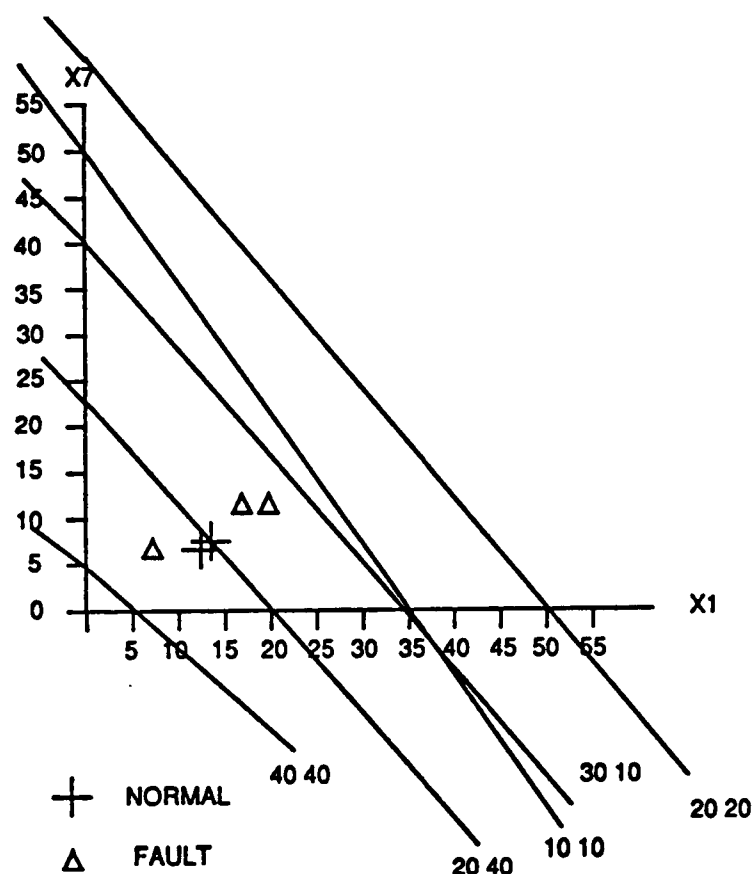


Figure 4. Hyperplanes Formed by Different Training Set Combinations of Normal and Faulty Data in Two Dimensions (Sensor P0 and Sensor PL3H). The first number indicates the number of normal vectors and the second represents the number of faulty vectors; i.e., 20 20 represents 20 normal and 20 fault.

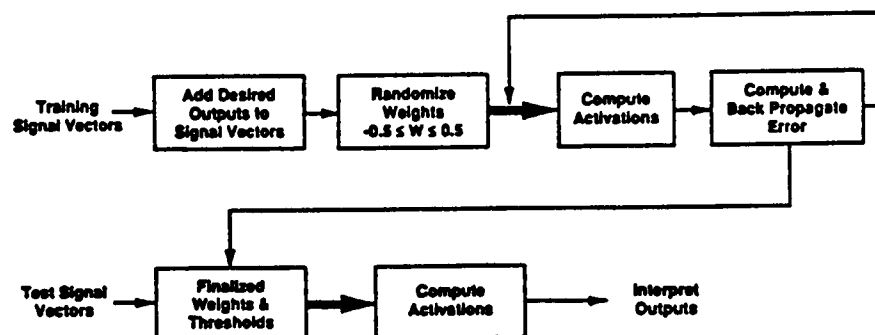


Figure 5. A System Architecture for Analyzing ASTF Test Data

CHAPTER V

RESULTS

(5.1) A Two Layer Semilinear Feed-Forward Neural Network: A Generalized Perceptron Approach

The Automatic Test and Control System (ATCS) in the ASTF uses 33 sensor values for diagnosis of the health of the facility. Eleven of those are commands values from the test cell engineers for changing the test cell environment. The other 22 are sensors that monitor the conditions of the engine and the test cell. In order to catch all possible faults that occur in the ASTF it is necessary to monitor all 22 data sensors. The network was trained using a training set of 60 scaled vectors composed of 20 normal followed by 40 abnormal vectors made up of 22 sensor values and their desired outputs. The network was allowed to iterate 1000 times over the training set to achieve the lowest possible total system error while still achieving 1000 iterations. Finalized connection weights for the input nodes (sensors) to the output node ranged between -1.61 and 3.78. The sensors with the strongest connection weights (highest absolute value) were a pressure sensor (P0) with a weight of 3.78, pressure sensor PL3H with a weight of 3.57 and a valve opening sensor (LVC1B) with a connection weight value of 2.02. The final threshold for the input nodes was -0.46. The hyperplane formed by the two layer network is shown in Figure 6.

In order to interpret exactly what the network found, other than by testing it on new data sets, we can look at how the final weights converged and try to decide if the weight values are telling us anything about the system being analyzed.

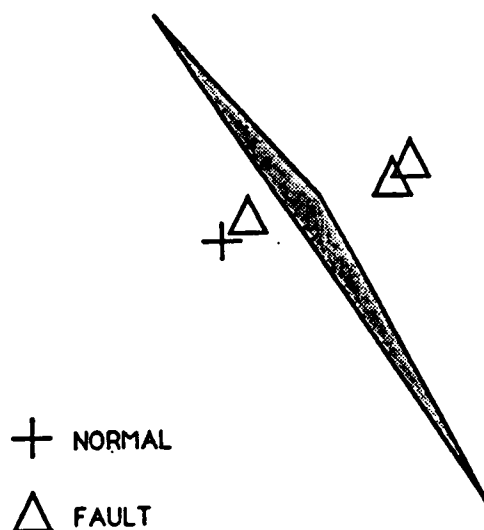


Figure 6. Hyperplane Formed by a Two Layer Neural Network in Three Dimensions (Sensor P0, Sensor PL3H and Sensor LVC1B)

Graphing the final weights in Figure 7 we can see the pattern of connection weights with peaks at sensors P0, PL3H and LVC1B. As far as this two layer network is concerned, the majority of relevant information as to the health of the system can be gained from these sensors. In other words, these sensor values give the best indications as to the status of the system.

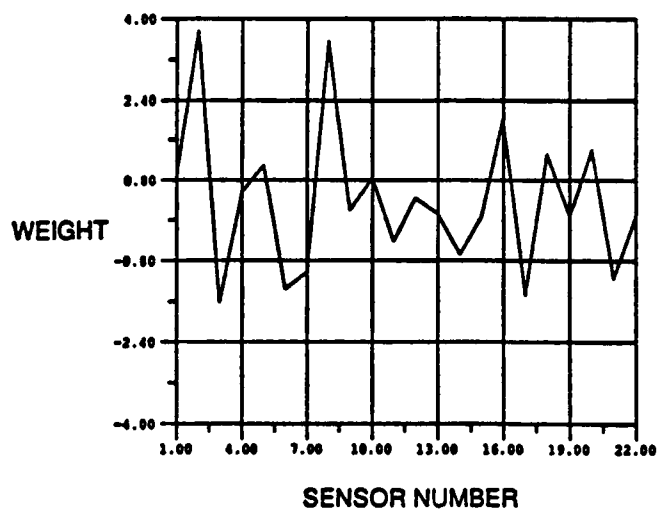


Figure 7. Final Connection Weights for a Two Layer Neural Network

The two layer neural network was tested on seven data sets composed of ten vectors made up of 22 sensor values. Two of those sets were from the training set in order to judge whether or not the system was working correctly. The other five were composed of arbitrary input vectors from the ASTF simulator representing engine test runs new to the network. As expected, the network computed a low output node value for the normal vectors extracted from the training set (0.013) and a high output node value for the vectors representing faulty data taken from the training set (0.999). This is not surprising since we would expect the network to correctly identify input vectors it has seen before. The true test of the system, of course, is its ability to make an accurate diagnosis of the health of the system based on some unknown input. When a set of normal ASTF sensor vectors was presented to the network the value of the output node value was 0.014, a correct diagnosis. Four new data sets composed of faulty sensor vectors was presented to the network and the output node achieved values of 0.990, 0.998, 0.936 and 0.990 respectively. These output values correctly indicate that the system is in failure. This shows that the network architecture is making accurate diagnosis and generalizing over the space spanned by the input variables. A summary of the output values for the network tested on various data sets is given in Table 2.

To confirm that the network was indeed accurately classifying ASTF sensor data from any sensor in the ATCS, the network was trained using a new training set composed of vectors different from those in the original training set. The network was tested against test vectors extracted from the training set. As expected the system achieved high diagnostic accuracy with a mean output value of 0.997 for vectors representing faults and a mean output value of 0.0004 for vectors representing a normal test run. To test the network for generalizability over all sensors

from the ASTF two unknown (not part of the training set) faults were presented to the network from unspecified locations in the test cell. The first vector was an unknown representing a turbine trip at an undisclosed location in the ASTF and the second vector represented a compressor failure in an unspecified leg of the test cell. The network achieved an output value of 0.998 for the first vector and 0.992 for the second vector.

(5.2) A Three Layer Semilinear Feed-Forward Neural Network: A Multilayer Perceptron Approach

A network was constructed using a hidden layer composed of three and five processing elements. Tests were run for each type of network, both on the vectors extracted from the training set and on the same unknown test vectors as for the two layer network. Since a two layer network gave accurate responses, the three layer network should be thoroughly scrutinized to see if it enhances the amount of information gained from the network. Since adding a layer adds computational complexity and adds to the time necessary to train and test the network, a hidden layer should be able to detect hidden features in the patterns of input data not seen by the two layer network, and be able to generalize over many different test sets in order to make it economically worthwhile.

The first three layer neural network used three processing elements in the hidden layer. Connection weights from the input nodes to the first hidden node ranged from -1.60 to 0.76 with highest (absolute value) weights from sensors P0 (-1.55), PL3H (-1.60). Only these two sensors had connection weight strengths greater than 1.0. Connection weights from the input nodes to the second hidden node ranged from -1.59 to 0.71 with greatest weight strengths from sensors P0 (-

1.30) and PL3H (-1.59). Again, only these two sensors had weights whose absolute value was greater than 1.0. Connection weights to the last hidden node ranged from -1.27 to 2.50 with the strongest connections from sensors P0 (2.28), PL3H (2.50) and LVC2A (-1.27). By adding a hidden layer new information was gained on diagnosing the health of the test cell, namely the significance of data from sensor LVC2A. In the two layer neural net the connection strength between LVC2A and the output node was high (1.48), but its relative importance in diagnosing the status of the system was best seen in studying the connection weights between the input layer and the third hidden node. The hyperplanes formed by the three hidden nodes can be seen in Figure 8. The addition of the hidden layer produced enough hyperplanes to form a more complex region to better classify the data.

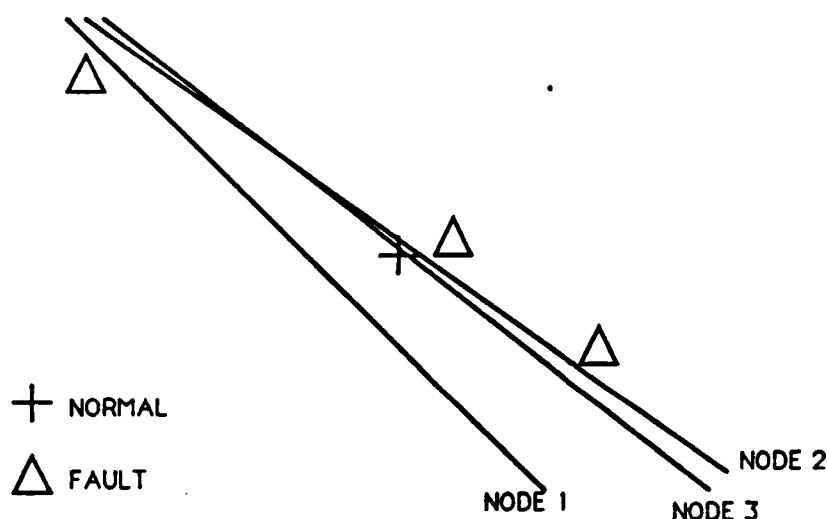


Figure 8. Hyperplanes Formed by Three Hidden Nodes Graphed in Three Dimensions (Sensor P0, Sensor PL3H and Sensor TL1H)

A three layer neural network with five hidden processing elements was also

explored. For all five hidden nodes the strongest connections between the input layer and the hidden layer were from nodes representing sensors P0, PL3H, LVC1B and LVC2A. Based on the finalized weights of the multilayer network, four sensors give the strongest indication as to the health of the ASTF. A summary of connection weights can be seen in Table 1.

The three layer neural network with three and five hidden processing elements (PEs) was tested on the same data sets as the two layer network in order to compare diagnostic accuracy. A summary of the results is given in Table 2. Computing the output for the network with three nodes in the middle layer using data sets composed of normal and faulty vectors extracted from the training set, resulted in an output node value of 0.010 and 0.996 respectively. For the test set composed of new normal sensor data the value on the output node was 0.011. For the four faulty sensor data sets the network achieved the following values: 0.991, 0.994, 0.973 and 0.991. By comparing these values with the outputs from the two layer network it can be seen that diagnostic accuracy was increased somewhat in four out of five test cases by the addition of the hidden layer.

Testing the three layer system with five hidden PEs, the network computed output values of 0.011 for the normal test case extracted from the training set and 0.997 for the abnormal test case similarly derived. For the new unknown test cases the network achieved values on the output node of 0.013 for the normal vectors and 0.992, 0.996, 0.944 and 0.992 for the fault cases respectively. This architecture did better than the two layer system in the same four out of five test cases as the system with three hidden PEs, but achieved a higher diagnostic accuracy than the other three layer network in only three out of five cases.

SENSORS	HIDDEN NODES								
	0	3			5				
	0	1	2	3	1	2	3	4	5
PT2	0.88	-0.79	-0.60	0.70	-0.35	-0.50	-0.40	0.18	0.69
PO	3.78	-1.55	-1.30	2.28	-0.05	-0.97	-1.21	0.98	2.75
TT2	-1.61	0.03	0.16	-0.97	-0.54	0.08	0.10	-0.13	-0.33
PLA	0.56	0.49	0.36	-0.02	0.44	0.38	-0.09	-0.01	0.07
PHC1V	1.10	-0.27	-0.07	0.40	0.00	0.00	-0.46	0.01	0.84
PL1H	-1.35	0.76	0.69	-0.91	-0.02	0.54	0.56	-0.81	-1.01
PL2H	-1.02	0.22	0.16	-0.21	-0.33	0.06	0.72	-0.53	-0.64
PL3H	3.57	-1.60	-1.59	2.50	-0.15	-1.27	-0.87	1.45	1.90
P0EC1	0.20	0.32	-0.20	-0.08	0.3	-0.14	-0.05	-0.03	0.09
TL1H	0.84	-0.87	-0.83	0.92	-0.43	-0.75	-0.04	0.32	0.45
TL2H	-0.42	-0.42	0.29	0.49	-0.42	0.29	0.49	-0.23	0.03
TL3H	0.45	-0.26	-0.01	0.04	-0.08	0.05	-0.40	0.58	-0.14
THC1V	0.13	0.30	-0.10	0.28	0.22	-0.09	0.38	0.21	0.07
TC12	-0.68	-0.44	-0.38	-0.34	-0.40	-0.26	-0.41	-0.40	0.25
LVC1A	0.07	0.34	0.14	0.36	0.37	0.25	0.31	0.32	-0.40
LVC1B	2.02	-0.28	-0.54	0.98	0.46	-0.34	-0.64	0.95	1.13
LVC2A	-1.48	0.73	0.71	-1.27	-0.14	0.52	0.35	-0.29	-1.40
LVC2B	1.33	-0.42	0.00	0.86	0.00	0.07	-0.10	0.21	1.18
LVC3A	0.08	0.24	0.09	-0.41	0.13	0.09	-0.30	-0.14	0.21
LVC3B	1.41	-0.15	-0.83	0.07	0.25	-0.75	-0.81	0.47	0.82
LVTC1	-1.14	0.41	0.61	-0.47	-0.24	0.41	0.74	-0.48	-0.63
LV100	0.09	0.09	0.45	0.05	0.06	0.49	0.04	0.41	0.09

Figure 9. Final Connection Weights from Input Nodes to Nodes of Hidden Layer
Based on Number of Hidden Nodes

Number Of Nodes in Middle Layer	Normal (train)	Fault (train)	Normal (test)	Fault 1 (test)	Fault 2 (test)	Fault 3 (test)	Fault 4 (test)
0	0.013	0.999	0.014	0.990	0.998	0.936	0.990
3	0.010	0.996	0.011	0.991	0.994	0.973	0.991
5	0.011	0.997	0.013	0.992	0.996	0.944	0.992

Figure 10. Final Output Values for Test Data Based on the Number of Nodes in the Hidden Layer

The results of these tests prove that the network is achieving accurate diagnosis of the health of the ASTF over arbitrary inputs and that the network is able to generalize over all possible faults in this instance. Based on the comparisons in Figure 10, it can be seen that a three layer neural network with three hidden nodes achieves a slightly more accurate diagnosis than a two layer neural network. The addition of two more nodes in the hidden layer only results in a small increase in performance, but significantly increases the time needed to train the network. Tests conducted using the two layer network seem to indicate that data coming from the ASTF sensors is linearly separable. Based on the increased run time and greater computational complexity it appears that a two layer neural network

is able to diagnose the status of the engine test cell with high enough diagnostic accuracy not to warrant the addition of a middle layer, specifically the addition of five hidden nodes. However, in this case a multilayer neural net proved useful in discovering sensor information that give important insight into monitoring the health of the test cell. Also Figure 6 clearly shows that a two layer neural network is unable to separate all classes of data, especially in the presence of noise.

The strongest connection weights from the first hidden node to the output node were from sensors PL3H (-1.60), PO (-1.55), and TL1H (-0.87). Using these weights to calculate the intercepts of the hyperplane on a three dimensional graph clearly shows that a multi-layer neural network with three hidden processing elements separates data that was non-linearly separable. The addition of a middle layer adds diagnostic accuracy to the network.

CHAPTER VI

MEANINGFULNESS IN THE INDIVIDUAL AND SYSTEM ERROR

In order to judge the efficiency and accuracy of the network there are other important features to look at other than diagnosis of unknowns. The backpropagation learning algorithm has some drawbacks including the tendency to get stuck in local minimums. Backpropagation is not guaranteed to arrive at the best solution, simply one that is pretty good. It is a gradient descent system and therefore may get stuck in a local minimum that is not necessarily the best solution (best set of connection weights) for the problem. In order to decide how well this system is doing we must look at the way the system arrives at the final weights (what the nodes are doing) and at the gradient curve of the total system error (or how the desired output minus the actual output decreases over time).

In implementing a neural network there are a number of issues that must be kept in mind. One is how a value of η (the learning rate) is to be chosen. A large η corresponds to rapid learning but might also result in large oscillations of the error as the individual processing elements seek a minimum individual error. Figure 9 shows the total system error using over a training set using a learning rate of 0.9 over 100 iterations. The system error makes one wild swing and becomes level at approximately 0.18.

Lowering the learning rate η to 0.7 (as shown in Figure 10) the total system error increases at the first pass through the network and then drops off to a steep decline around the fifth pass through the training set. It appears that a minimum of system error is achieved but we would have stronger confidence in the accuracy

of our system if the gradient descent was smooth.

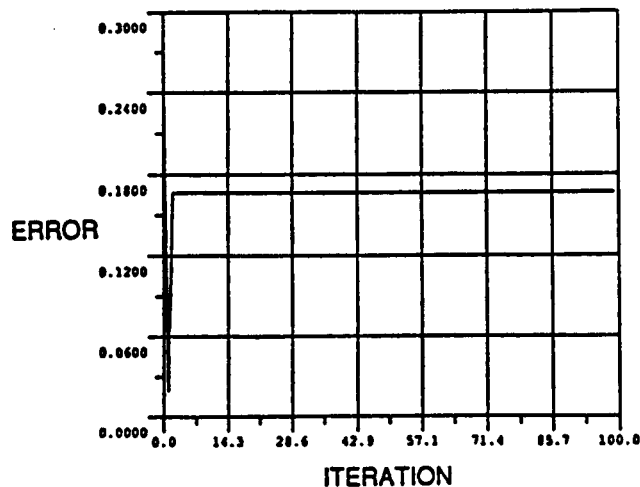


Figure 11. Total System Error for a Two Layer Neural Network with $\eta = 0.9$ Over 100 Iterations

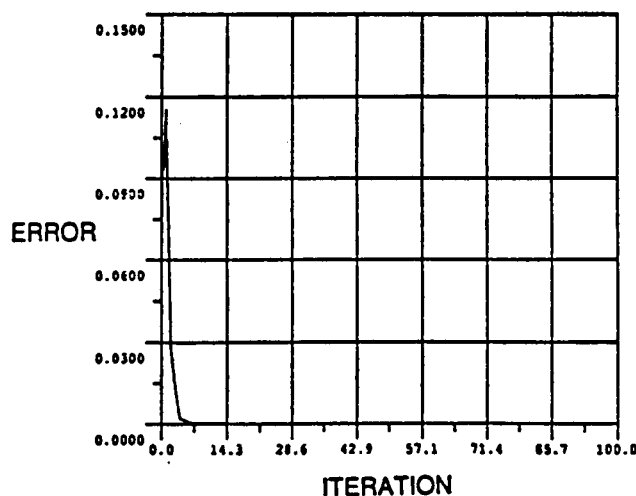


Figure 12. Total System Error for a Two Layer Neural Network with $\eta = 0.7$ Over 100 Iterations

Figure 13 shows the total system error again with a learning rate η of 0.5. This gradient descent makes one climb and then smoothly (rapidly) descends to an error minimum around the 35th pass through the training set. Likewise when $\eta = 0.2$ the total system error smoothly descends from a beginning level to an asymptotic minimum over the full 100 iterations as in Figure 14.

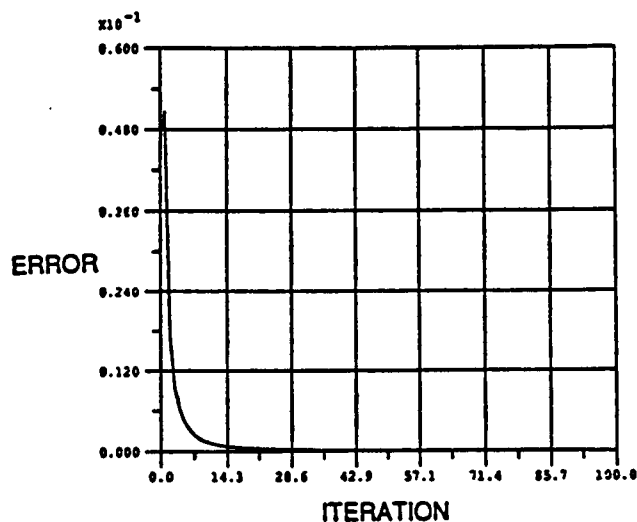


Figure 13. Total System Error for a Two Layer Neural Network with $\eta = 0.5$ Over 100 Iterations

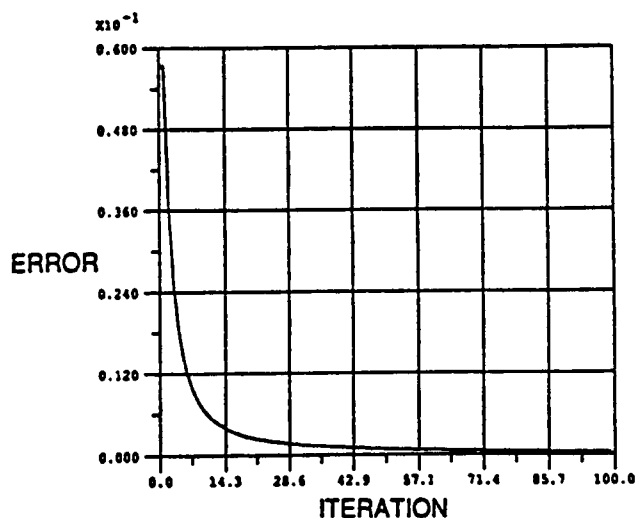


Figure 14. Total System Error for a Two Layer Neural Network with $\eta = 0.2$ Over 100 Iterations

The total system error for the backpropagation network is the sum of the squares of all the individual errors for each node. To understand what is occurring as the curve of the total system error is smoothly decreasing we need to understand what is happening at the level of an individual processing element. Figure 15 shows the individual error for the node representing sensor PL3H, a node that converged

to a high final connection weight, when the system was trained with a learning rate of 0.7. At the second pass through the training set the individual error makes a rapid descent and swings past the optimum error minimum before it swings back to continue its descent towards that minimum. When the system was trained with $\eta = 0.5$ a similar oscillation occurs for the node around the second pass through the training set. The curve is similar to an automobile driver that has missed his turn and doubles back to correct his direction (see Figure 16).

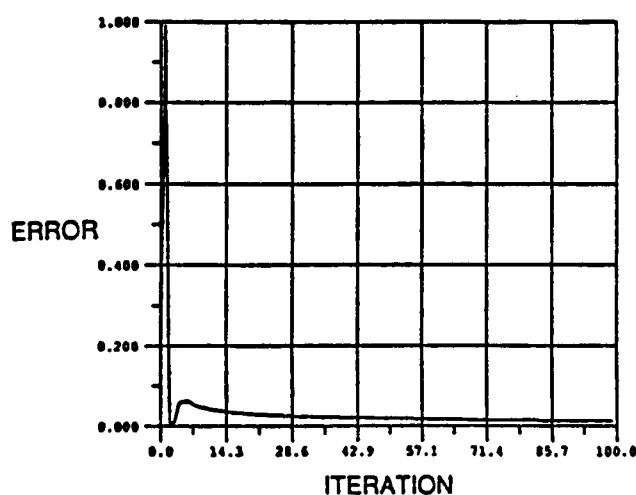


Figure 15. Individual Error for a Two Layer Neural Network with $\eta = 0.7$ Over 100 Iterations for the Node Representing Sensor PL3H

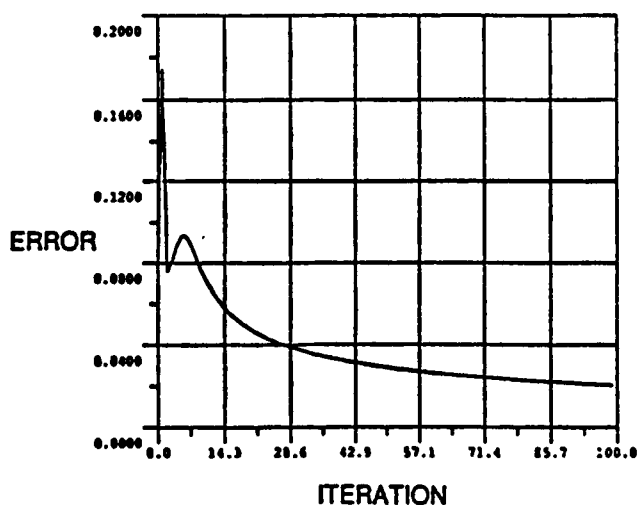


Figure 16. Individual Error for a Two Layer Neural Network with $\eta = 0.5$ Over 100 Iterations for the Node Representing Sensor PL3H

When the system is trained with a learning rate of 0.2 the individual error for sensor PL3H becomes a smoothly descending curve as in Figure 17.

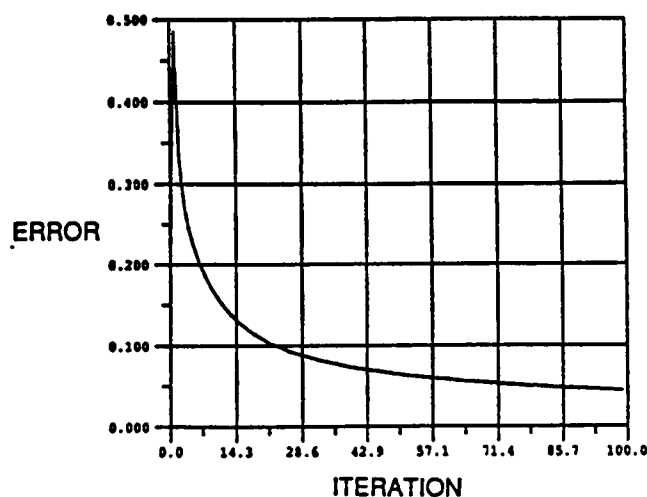


Figure 17. Individual Error for a Two Layer Neural Network with $\eta = 0.2$ Over 100 Iterations for the Node Representing Sensor PL3H

Other individual errors for processing elements representing different sensors show similar results. Figure 18 shows the individual error for the node representing sensor LVC1B when the network was trained using a learning rate of 0.7. When the learning rate was lowered to 0.2 the gradient descent is smooth as in Figure 19.

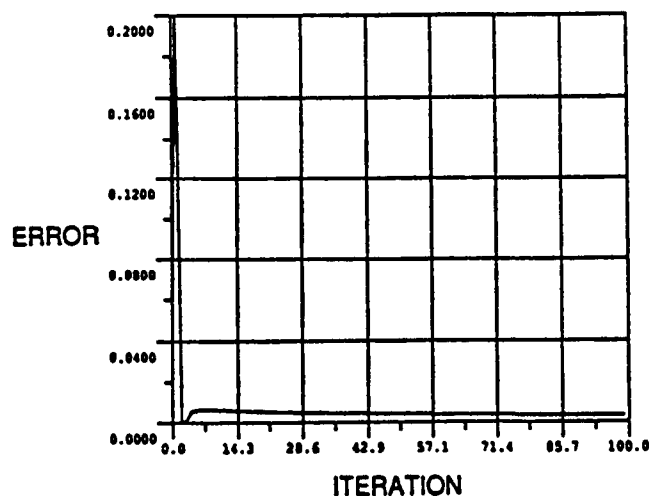


Figure 18. Individual Error for a Two Layer Neural Network with $\eta = 0.7$ Over 100 Iterations for the Node Representing Sensor LVC1B

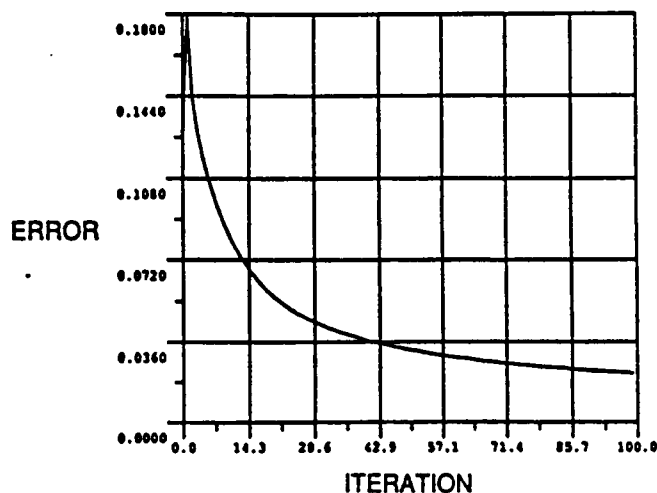


Figure 19. Individual Error for a Two Layer Neural Network with $\eta = 0.2$ Over 100 Iterations for the Node Representing Sensor LVC1B

The network with the reduced learning rate was tested on unknown test vectors to see if there was any change in diagnostic accuracy. The change in the value of the output node over 100 iterations of the training set was approximately 0.05. Where the network trained with a higher learning rate achieved an average output value of 0.994 and 0.980 for two unknown test vector sets, the network trained with a learning rate of 0.2 achieved an average output value for the same unknown test vectors of 0.952 and 0.917

Based on these results it can be seen that a network trained with a high learning rate may have individual errors that oscillate back and forth across the optimum minimum. The learning rate is simply the size of the step the error takes along the gradient descent towards some "good enough" minimum. Reducing the learning rate decreases the size of the step the system takes along the descent parabola in N-dimensional space (in this case 22 dimensional space). In these test cases both a high and a low learning rate achieve a good error minimum. Although

a higher learning rate may achieve faster learning, in designing neural networks to diagnose the health of a complex system over all possible faults it is necessary to have a high confidence that the network is achieving a near global error minimum. To see if the error rate is descending towards an optimal minimum, the individual and system errors can be plotted after each iteration of the training set. Testing the network with a lower learning rate shows little reduction in diagnostic accuracy.

CHAPTER VII

THE NETWORKS ABILITY TO HANDLE NOISE

A great deal of criticism is given to diagnostic systems designed purely around simulated data. Many systems, neural networks among them, do not perform as well on real world data as they did on data from a simulator. One reason for this is the noise that is created when sampling real time sensor data. If a neural network is to be used for diagnosing the status of an engine test cell it must be robust and resistant to noise.

Because of the unavailability of actual engine test data it was necessary to add noise to the data sets created by the ASTF simulator. The original back-propagation program was modified to allow the user to add a certain percentage of noise to both the training set data and to the test set data. Random noise was added to the vectors during every pass through the simulated data in order to better resemble the noise that might be encountered in on-line training and testing of the network. Every time any piece of data was used by the network in either training or testing, the random number generator was re-seeded in order to give completely random noise. Tests of the system were conducted with 5%, 10%, 25% and 35% noisy data. The algorithm used for adding noise to previously used data sets was:

$$\text{Noisy Data Value} = (\text{data value}) + (\text{uniform random number}[-1,1]) * (\text{percent noise})$$

(7.0.0)

A comparison of the noise levels for all seven tests sets for a two layer backpropagation network is given in Figure 20. A two layer backpropagation neural network was trained using a training set modified to include 5% noise. Seven test data sets

composed of ten vectors (two from the training set and five previously unknowns) were used to test the network. For the unknown normal vector set the network computed an average value of 0.051. For the four unknown fault vector sets the average computed values were 0.972, 0.993, 0.874 and 0.967. All these output values diagnose the condition of the noisy simulated data with a high degree of accuracy.

The network was trained again to include 10% noise. The average output value computed by the net for the normal test vector set was 0.051. For the four fault vector sets the respective average output values were 0.973, 0.993, 0.874 and 0.967. The performance of the network degrades slightly as the level of noise increases. For tests conducted with 25% noise the average output value of the network was 0.051 for the normal vectors and 0.973, 0.992, 0.873 and 0.967 for the fault vectors. 35% noise in the training and testing sets resulted in average output values of 0.051 for the normal vector set and 0.973, 0.992, 0.872 and 0.966 for the fault sets respectively.

Because a multilayer neural network spreads its knowledge in the form of connection weights over more nodes, we would expect that it is less affected by noisy data and would diagnose the status of the ASTF with a higher degree of accuracy. This was indeed the case. The tests for noise in a two layer network were repeated with a three layer neural net with three processing elements in the hidden layer, with results as shown in Figure 21.

For a network trained and tested on vectors modified to include 5% noise the average computed output values were 0.044 for the normal test vectors and 0.970, 0.976, 0.927 and 0.968 for the fault vector sets. For a network modified

NOISE	NORMAL TRAINING SET	FAULT TRAINING SET	NORMAL TEST	FAULT TEST 1	FAULT TEST 2	FAULT TEST 3	FAULT TEST 4
5%	0.046	0.998	0.051	0.972	0.993	0.874	0.967
10%	0.047	0.998	0.051	0.973	0.993	0.874	0.967
25%	0.048	0.998	0.051	0.973	0.992	0.873	0.967
35%	0.049	0.998	0.051	0.973	0.992	0.872	0.966

Figure 20. A Two Layer Neural Network Trained and Tested with Noisy Data
Over 100 Iterations

to include 10% noise in training and testing the average output values did not change from the network trained and tested with 5% noise. The outputs also remained unchanged for a network that included 25% noise. Based on this test we can see that a multilayer neural network is robust and resistant to noise in some instances. A slight degradation in diagnostic confidence appears in training and testing a network to include 35% noise. The average output value for the network was 0.045 for the normal vector set and 0.969, 0.976, 0.927 and 0.968 for the fault vector sets.

By comparing the results in Figure 20 and Figure 21 it can be seen that the three layer network does better on the unknown normal test vector set and fault vector set four, and much better for the third fault test. A two layer network

trained and tested with 35% noise for fault test set three computes an average output value of 0.872 whereas a three layer network computes an average output value of 0.927. It can be seen from these results that a multilayer neural network is more robust and resistant to noise than a two layer generalized perception. If training is to be conducted on-line in the ASTF a multilayer neural network should be used.

NOISE	NORMAL TRAINING SET	FAULT TRAINING SET	NORMAL TEST	FAULT TEST 1	FAULT TEST 2	FAULT TEST 3	FAULT TEST 4
5%	0.043	0.986	0.044	0.970	0.976	0.927	0.968
10%	0.043	0.986	0.044	0.970	0.976	0.927	0.968
25%	0.044	0.986	0.044	0.970	0.976	0.927	0.968
35%	0.044	0.986	0.045	0.969	0.976	0.927	0.968

Figure 21. A Three Layer Neural Network with Three Hidden Nodes Trained and Tested with Noisy Data Over 100 Iterations

CHAPTER VIII

THE NETWORKS ABILITY TO CLASSIFY TYPES OF FAULTS

In Chapters V and VII we have shown that a neural network trained using the backpropagation learning algorithm can be used to classify data from the Aeropropulsion Systems Test Facility as being normal or indicative of a system in failure. To make this system more useful to test cell engineers it would be helpful for the network to tell us what type of fault is occurring.

The ASTF simulator was used to generate data representative of four types of faults: the ASTF loses one air side compressor during a ramp in test conditions; the ASTF experiences a turbine trip of the only turbine in use; the ASTF loses air side compressors while it is idle; and some unknown fault that has not been diagnosed by the test cell engineers. These faults along with vectors representing normal test data are used to compose a training set in the following way: 30 normal vectors, 20 vectors representing the compressor failure during ramp, 20 vectors representing a turbine trip, 20 vectors representing loss of air side compressors while at idle, and 10 vectors representing an undiagnosed type of fault. This combinations was picked by availability of the data and the results discussed in Chapter IV.

Both a two layer and three layer architecture were tested. In both cases the network was composed of 22 input nodes representing the ASTF sensors and five output nodes representing normal (node 0), compressor failure (node 1), turbine trip (node 2), compressor failure at idle (node 3) and undiagnosed (node 4). Desired outputs were added to the training set with a high value (1) for the node representing the type of fault and a low value (0) for all other output nodes. To

make the data more similar to real time test data 10% noise was added. The networks were trained for 100 iterations with a learning rate of 0.2.

After the network was trained it was tested on two vector sets from the training set and five unknowns. Figure 22 shows the results for the two layer network. For the test set composed of normal vectors the value of node 0 was high (0.371) with all other nodes being relatively low. For the first fault test set the value of node 1 representing a compressor failure was high (0.601) with all other output nodes being relatively low. Likewise for fault test two the value for node 1 was high (0.914) while all others remained low. Fault test three contained data taken from the ASTF during a turbine trip and the value for node 2 was high (0.479). Fault test four (compressor snap at idle) had a high value for node 3 (0.826) and a reasonably high value for node 0 (0.312).

A three layer network with three hidden processing units was tested for its ability to classify types of faults. After the same training configuration as the two layer network the net was tested against the same five test cases. The three layer network did poorly at diagnosing normal test vectors for both the unknown normal and normal data taken from the training set. In both cases the network diagnosed the data as being representative of the undiagnosed fault. However, the multilayer network did very well in diagnosing some types of faults. For fault test one the value of node 1 was high (0.535). For fault test two (another compressor failure) the value of node 1 was 0.876. For the turbine failure fault test three the value of node 2 was 0.848 and for fault test four the value of node three was high (0.864) with all other output nodes being relatively low. These results are summarized in Figure 23.

Based on these results it would seem that the multilayer neural network can diagnose certain types of faults with high degrees of accuracy. However this network architecture is not able to differentiate between normal data and unknowns. While a two layer network does not give output values with as high a confidence as the multilayer network, it is able to diagnose the status of the system with accuracy and with no error during these tests.

OUTPUT NODE	NORMAL TRAINING SET	FAULT TRAINING SET	NORMAL TEST	FAULT TEST 1	FAULT TEST 2	FAULT TEST 3	FAULT TEST 4
0	0.473	0.081	0.371	0.092	0.086	0.072	0.312
1	0.033	0.772	0.081	0.601	0.914	0.002	0.019
2	0.001	0.006	0.001	0.003	0.001	0.479	0.000
3	0.069	0.008	0.019	0.024	0.016	0.135	0.826
4	0.076	0.000	0.079	0.001	0.000	0.000	0.000

Figure 22. A Two Layer Neural Network Classification of Types of Faults with 10% Noise

OUTPUT NODE	NORMAL TRAINING SET	FAULT TRAINING SET	NORMAL TEST	FAULT TEST 1	FAULT TEST 2	FAULT TEST 3	FAULT TEST 4
0	0.279	0.000	0.090	0.001	0.000	0.006	0.097
1	0.012	0.866	0.069	0.535	0.876	0.033	0.003
2	0.000	0.084	0.000	0.043	0.050	0.848	0.048
3	0.057	0.003	0.038	0.015	0.003	0.064	0.864
4	0.514	0.035	0.777	0.062	0.059	0.000	0.067

Figure 23. A Three Layer Neural Network with Three Hidden Nodes Classification of Types of Faults with 10% Noise

CHAPTER IX

CONCLUSIONS

This research has shown that a neural network can be used to differentiate between patterns of data representing a normal test run of the Aeropropulsion Systems Test Facility and sensor data coming from a test cell that is in failure. A network was also built to classify the type of fault that is occurring. For simulated ASTF data a near optimal diagnosis can be achieved using a two layer neural network (generalized perceptron). However, for simulated data with noise such as would occur in actual engine test cell data, a multilayer neural network with three hidden processing elements yields a more accurate diagnosis. The neural network architecture described in this thesis diagnoses the health of the ASTF with a high degree of accuracy after being exposed to an input vector representing sensor data from only one time slice (0.2 sec) and an even higher accuracy after averaging ten time slices, or two seconds, worth of data.

In addition, by monitoring the converging weights of the input processing elements it was seen that the most relevant information used by the 22 input node network comes from four sensors, P0, PL3H, LVC1B and LVC2A based on connection strengths (see figure 9). Graphing the node connection weights and the total system error for various network parameters shows whether the weights oscillate while approaching error minima.

This thesis has demonstrated the usefulness of neural networks for fault monitoring and diagnosis. Further research and implementation of neural network based systems will give test personnel a new tool in the analysis of critical test data. This system will help take neural network technology out of the laboratory

and help create commercial systems for the analysis of complex systems. The most significant potential benefit of the system which has been developed is the improved safety of test cell personnel by rapid detection of faulty conditions. This will be of extreme importance as testing begins on more advanced systems such as the National Aerospace Plane and the Advanced Tactical Fighter.

BIBLIOGRAPHY

BIBLIOGRAPHY

The following is a list of reference articles and texts containing information relevant to the research discussed in this paper:

- Abu-Mostafa, Y.S.: "The Vapnik-Chervonenkis Dimension: Information versus Complexity in Learning". *Neural Computation* (1989).
- Bedworth, M.D. and Bridle, J.S.: "Experiments with the Back-Propagation Algorithm: A Systematic Look at a Small Problem". *Royal Signals and Radar Establishment* (1987).
- Bruck, J.: "A Study on Neural Networks". *International Journal of Intelligent Systems*, Vol. 3, pp. 59-75 (1988).
- Carpenter, G.A. and Grossberg, S.: "ART2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns". *Applied Optics* (1987).
- Casasent, D.P.: "Neural Network Signal Processor (NSP)". *SPIE Vol. 880 High Speed Computing* (1988).
- Chow, M.Y. and Yee, S.: "Application of Neural Networks to Incipient Fault Detection in Induction Motors". Department of Electrical and Computer Engineering, N.C. State. Unpublished (1989).
- Churchland, P.M. and Churchland, P.S.: "Could A Machine Think?". *Scientific American* (January 1990)
- Cox, K.S.: "An Analysis of Noise Reduction Using Back-Propogation Neural Networks". Master's Thesis, AFIT (1988).
- "DARPA Neural Network Study". Final Report. MIT (October 1987-February 1988).
- Dietz, W.E., Kiech, E.L., and Ali, M.: "Neural Network Models Applied To Real-Time Fault Diagnosis". *Journal of Neural Network Computing*, Vol. 1 (1989).
- Hinton, G.E.: "Connectionist Learning Procedures". *Artificial Intelligence* 40 (1989).

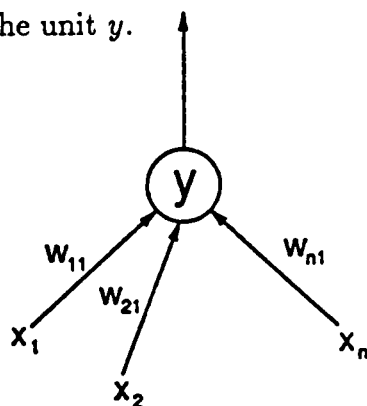
- Johnson, C. And Sehert, P.: "The Automated Flight Test Data System". *AGARD Conf. Porc. on Flight Test Techniques*, Neuilly-Sur-Seine, France (1976).
- Lippman, R.P.: "An Introduction to Computing with Neural Nets". *IEEE ASSP Magazine*, Vol. 4. (1987).
- Mitchell, J.G.: "New Test Capability for Propulsion System Testing". *AIAA Paper 73-1283*, AIAA/SAE 9th Propulsion Conf., Las Vegas, Nevada (November 1973).
- NeuralWorks Professional II Instruction Manual*. NeuralWare Inc. (1988).
- Nii, H.P. and Feigenbaum, E.A.: "Rule Based Understanding of Signals". *Pattern-Directed Inference Systems*. Academic Press (1978).
- Pao, Y.H.: *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley Publishing Co. (1989).
- Polce, R.L.: "A Systems Description of the Aeropropulsion Systems Test Facility (ASTF)". Master's Thesis, UT Knoxville (1982).
- Pomerleau, D.A.: "The Meta-Generalized Delta Rule: A New Learning Algorithm for Learning in Connectionist Networks". *Carnegie-Mellon University Report* (1987).
- Rumelhart, D.E. and McClelland, J.L.: *Parallel Distributed Processing Explorations in the Microstructure of Cognition*, Volume 1: Foundations. The MIT Press (1988).
- Simpson, P.K.: *Artificial Neural Systems: Foundations, Paradigms, Applications and Implementations*, Pergamon Press (1990).
- Touretzky, D.S. and Hinton, G.E.: "A Distributed Connectionist Production System". *Carnegie-Mellon University Report* (1986).
- Werbos, P.J.: "Generalization of Backpropagation with Application to a Recurrent Gas Market Model". *Neural Networks*, Vol. 1, pp. 339-356 (1988).

APPENDIXES

APPENDIX A

A Mathematical Derivation of a Neural Network

Consider some processing unit y . y has a number of inputs $x_1, x_2 \cdots x_n$ either from an input vector or from other processing units. Each input has a connection strength (or weight) to the unit y .



The set of n weights can be represented as an n dimensional vector w . The state of activation of the unit y is the inner product of the weight vector with the input vector $x = (x_1, x_2, \cdots x_n)$

$$y = w \cdot x$$

To expand the model; if w_{ij} is the connection strength to the i th output from the j th input then the output of the unit is

$$y_i = \sum_j w_{ij} x_j$$

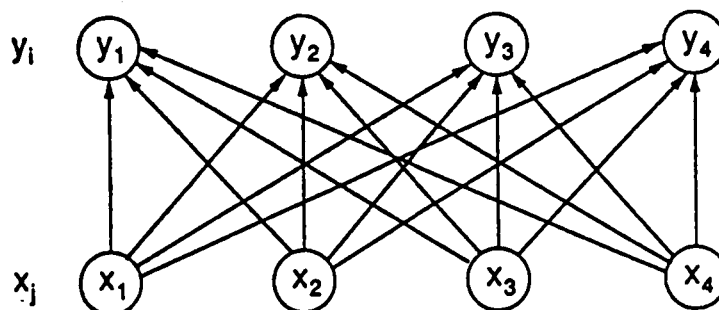
If we normalize the input vectors and designate some threshold function so that the state of activation must exceed that threshold in order to pass on its output to the next layer of processing elements, (the threshold allows the unit to make a

decision), then for a particular unit y_j

$$y_i = 1 \text{ if } \sum_j w_{ij}x_j > T; \text{ else } y_i = 0$$

The properties of the inner product let us picture the system as computing the closeness of the input vectors to the weight vector in N -dimensional space where N is the number of input units.

Represented graphically for a network with 4 input nodes and 4 output nodes the state of situation for the system is:



Adding weights to the diagram

$$\begin{array}{cccc}
 x_1 & x_2 & x_3 & x_4 \\
 \downarrow & \downarrow & \downarrow & \downarrow \\
 w_{11} \rightarrow w_{12} \rightarrow w_{13} \rightarrow w_{14} \rightarrow y_1 \\
 w_{21} \rightarrow w_{22} \rightarrow w_{23} \rightarrow w_{24} \rightarrow y_2 \\
 w_{31} \rightarrow w_{32} \rightarrow w_{33} \rightarrow w_{34} \rightarrow y_3 \\
 w_{41} \rightarrow w_{42} \rightarrow w_{43} \rightarrow w_{44} \rightarrow y_4
 \end{array}$$

Represented algebraically

w_{ij} = weight to y_i from x_j

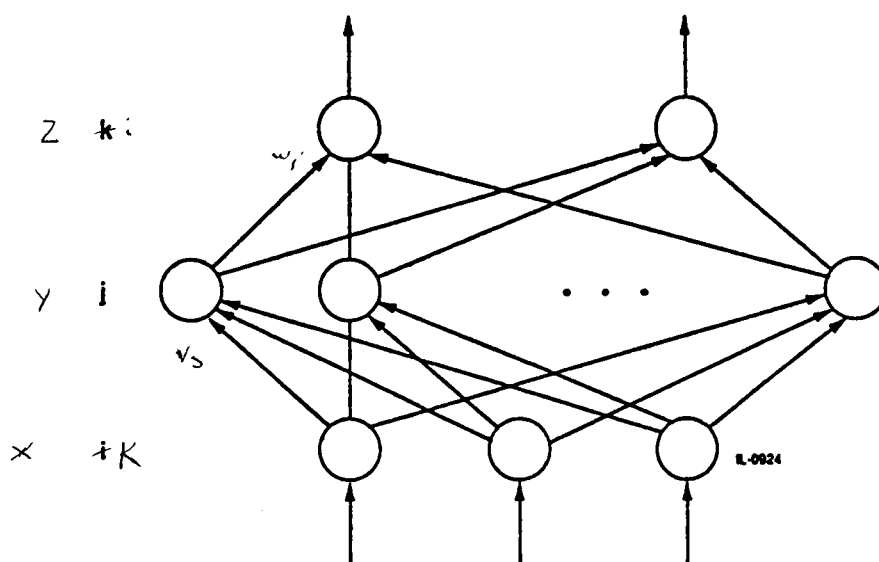
$$y_i = \sum_j w_{ij} x_j$$

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

APPENDIX B

A Mathematical Derivation of the Backpropagation Learning Algorithm*



Consider some three layer neural network with layers x, y, z .

Let v_j be the weight vector into node y_j and w_i be the weight vector into node z_i

Let f be the differentiable Sigmoid function such that

$$f(y) = \frac{1}{1 + e^{-y}} \quad (3.2.1)$$

whose derivative is

$$f'(y) = f(y)[1 - f(y)] \quad (3.2.2)$$

where $f'(y) = df(y)/dy$

* From CS 522 lectures given by Bruce Whitehead, April 16-23, 1990. Lecture notes prepared and edited by Jim Golden, May 1990.

If we say that each node y_j in the hidden layer has a total input activation y_j^{in} and an output y_j^{out} , then for each node in the hidden layer

$$y_j^{\text{in}} = \mathbf{v}_j \cdot \mathbf{x} \quad (3.2.3)$$

and

$$y_j^{\text{out}} = f(y_j^{\text{in}}) = \frac{1}{1 + e^{-y_j^{\text{in}}}} \quad (3.2.4)$$

Also for each node z_i in the output layer

$$z_i^{\text{in}} = \mathbf{w}_i \cdot \mathbf{y}^{\text{out}} \quad (3.2.5)$$

$$z_i^{\text{out}} = f(z_i^{\text{in}}) = \frac{1}{1 + e^{-z_i^{\text{in}}}} \quad (3.2.6)$$

Let Q be the training set of patterns that the network will use to learn. Then

$$\mathbf{x}(1), \mathbf{x}(2), \dots, \mathbf{x}(q), \dots \text{ for all } q \in Q$$

will be a set of training vectors and $|Q|$ will be the number of training vectors in the set. The error on node z_i for each training vector q is

$$z_i^{\text{error}}(q) = z_i^{\text{out}}(q) - z_i^{\text{desired}}(q) \quad (3.2.7)$$

and the mean squared error for node z_i is

$$z_i^{\text{MSE}} = \frac{1}{|Q|} \sum_q [z_i^{\text{error}}(q)]^2 \quad (3.2.8)$$

The total system error for layer z is

$$z^{\text{TSE}} = \sum_i z_i^{\text{MSE}} \quad (3.2.9)$$

$$= \frac{1}{|Q|} \sum_i \sum_q [z_i^{\text{error}}(q)]^2 \quad (3.2.10)$$

Deriving the learning rule for layer z , where α is the learning rate

$$\Delta w_{ij} = -\alpha \frac{\partial z^{\text{TSE}}}{\partial w_{ij}} \quad (3.2.11)$$

$$= -\alpha \partial \left[\frac{1}{|Q|} \sum_{i'} \sum_q [z_{i'}^{\text{error}}(q)]^2 \right] / \partial w_{ij} \quad (3.2.12)$$

$$= \frac{-\alpha}{|Q|} \sum_{i'} \sum_q \frac{\partial [z_{i'}^{\text{error}}(q)]^2}{\partial w_{ij}} \quad (3.2.13)$$

Using the chain rule eqn. (3.2.13) becomes

$$\Delta w_{ij} = \frac{-\alpha}{|Q|} \sum_{i'} \sum_q \frac{\partial [z_{i'}^{\text{error}}(q)]^2}{\partial z_{i'}^{\text{error}}(q)} \frac{\partial z_{i'}^{\text{error}}(q)}{\partial z_{i'}^{\text{out}}(q)} \frac{\partial z_{i'}^{\text{out}}(q)}{\partial z_{i'}^{\text{in}}(q)} \frac{\partial z_{i'}^{\text{in}}(q)}{\partial w_{ij}} \quad (3.2.14)$$

Calculating the partial derivatives, eqn. (3.2.14) becomes

$$\frac{-\alpha}{|Q|} \sum_{i'} \sum_q 2 \cdot z_{i'}^{\text{error}}(q) \cdot 1 \cdot z_{i'}^{\text{out}}(q) [1 - z_{i'}^{\text{out}}(q)] A_{i'j} \quad (3.2.15)$$

where $A_{i'j} = y_j^{\text{out}}(q)$ if $i' = j$ and $A_{i'j} = 0$ otherwise. Therefore

$$\Delta w_{ij} = \frac{-2\alpha}{|Q|} \sum_q z_i^{\text{error}}(q) \cdot z_i^{\text{out}}(q) [1 - z_i^{\text{out}}(q)] y_j^{\text{out}}$$

let

$$\delta_i(q) = - \sum_q z_i^{\text{error}}(q) \cdot z_i^{\text{out}}(q) [1 - z_i^{\text{out}}(q)]$$

then

$$\Delta w_{ij} = \frac{2\alpha}{|Q|} \delta_i(q) y_j^{\text{out}}(q) \quad (3.2.16)$$

Likewise, the learning rule for layer y :

$$\Delta v_{jk} = \frac{-\alpha \partial z^{\text{TSE}}}{\partial v_{jk}} = \frac{-\alpha}{|Q|} \sum_q \sum_i \partial [z_i^{\text{error}}(q)]^2 \quad (3.2.17)$$

and using the chain rule

$$\Delta v_{jk} = \frac{-\alpha}{|Q|} \sum_q \sum_i \frac{\partial (z_i^{\text{error}}(q))^2}{\partial z_i^{\text{error}}(q)} \frac{\partial z_i^{\text{error}}(q)}{\partial z_i^{\text{out}}(q)} \frac{\partial z_i^{\text{out}}(q)}{\partial z_i^{\text{in}}(q)} \frac{\partial z_i^{\text{in}}(q)}{\partial y_j^{\text{out}}(q)} \frac{\partial y_j^{\text{out}}(q)}{\partial y_j^{\text{in}}(q)} \frac{\partial y_j^{\text{in}}(q)}{\partial v_{jk}} \quad (3.2.18)$$

Solving for the derivatives we get

$$\Delta v_{jk} = \frac{-\alpha}{|Q|} \sum_q \sum_i 2z_i^{\text{error}}(q) \cdot 1 \cdot z_i^{\text{out}}(q) [1 - z_i^{\text{out}}(q)] \cdot w_{ij} \cdot y_j^{\text{out}}(q) [1 - y_j^{\text{out}}(q)] x_k(q) \quad (3.2.19)$$

$$= \frac{-2\alpha}{|Q|} \sum_q \left[x_k(q) y_j^{\text{out}}(q) [1 - y_j^{\text{out}}(q)] \sum_i z_i^{\text{error}}(q) z_i^{\text{out}}(q) [1 - z_i^{\text{out}}(q)] w_{ij} \right] \quad (3.2.20)$$

Recall δ_i from eqn. (3.2.16) and let

$$\delta_j = y_j^{\text{out}}(q) [1 - y_j^{\text{out}}(q)] \sum_i \delta_i(q) w_{ij} \quad (3.2.21)$$

then

$$\Delta v_{jk} = \frac{2\alpha}{|Q|} \sum_q \delta_j(q) x_k(q) \quad (3.2.22)$$

let

$$\eta = \frac{2\alpha}{|Q|} \quad (3.2.23)$$

then

$$\Delta v_j = \eta \sum_q \delta_j(q) x_k(q) \quad (3.2.24)$$

In summary

$$\Delta w_i = \eta \delta_i(q) y_j(q) \quad (3.2.25)$$

and

$$\Delta v_{jk} = \eta \sum_q \delta_j(q) x_k(q) \quad (3.2.26)$$

Thus the activation of the network is feed forward but the error is propagated backward because the δ_j 's of the hidden layer are calculated from the δ_i 's of the output layer.

VITA

James Brown Golden III was born June 15, 1963 in Landstuhl, West Germany. He graduated from Meade High School, Fort Meade, Maryland, in 1981. He received his Bachelor of Science degree in Mathematics/Computer Science from Rhodes College in Memphis, Tennessee in 1985. From 1985 to 1988 he served as an officer in the United States Air Force as an aircraft crash investigator at Norton AFB, California. In August of 1988 he entered The University of Tennessee Space Institute as a graduate research assistant. His interests include neural networks, connectionist expert systems, and applying artificial intelligence to problems in aerospace engineering. His permanent home address is: 1036 St. Andrews Place, Nashville, TN 37204.