N96-70675

LEARNING AND DIAGNOSING FAULTS USING NEURAL NETWORKS

Bruce A. Whitehead
Earl L. Kiech
Moonis Ali

Center For Advanced Space Propulsion
The University Of Tennessee Space Institute
Tullahoma, TN 37388

Abstract

Neural networks have been employed for learning fault behavior from rocket engine simulator parameters and for diagnosing faults on the basis of the learned behavior. Two problems in applying neural networks to learning and diagnosing faults are (1) the complexity of the sensor data to fault mapping to be modeled by the neural network, which implies difficult and lengthy training procedures; and (2) the lack of sufficient training data to adequately represent the very large number of different types of faults which might occur. Methods are derived and tested in an architecture which addresses these two problems. First, the sensor data to fault mapping is decomposed into three simpler mappings which perform sensor data compression, hypothesis generation, and sensor fusion. Efficient training is performed for each mapping separately. Secondly, the neural network which performs sensor fusion is structured to detect new unknown faults for which training examples were not presented during training. These methods were tested on a task of fault diagnosis by employing rocket engine simulator data. Results indicate that the decomposed neural network architecture can be trained efficiently, can identify faults for which it has been trained, and can detect the occurrence of faults for which it has not been trained.

Introduction

The objective of our research described in this paper is to employ neural networks for *(i)* learning fault behavior from rocket engine simulator parameters perturbed with noise (termed as sensor data in this paper), and *(ii)* diagnosing faults on the basis of the learned behavior. In a complex system (such as a liquid-fuel rocket engine), there are many possible ways in which components of the system may fail. Only a fraction of these possible failures have been observed and are known to human experts. Human experts have not seen all possible instances of all faults and hence cannot describe the features of the faults sufficiently well to make diagnostic decisions.

This work was supported in part by NASA Grant No. NAGW-1195 and Rocketdyne Contract No. R04QBZ90-032709.

This state of affairs is problematic for training a neural network to recognize component failures based on sensor data. Neural networks learn to recognize faults by being trained with examples of these faults. They are capable of generalizing from some examples of a fault to other examples of the same fault, but they are not capable of recognizing a new fault for which no training examples have been given. If a neural network is trained to recognize a set of faults and then presented with an example of a completely new fault, it will typically either *(i)* find the closest match to the new example among the previously trained faults, or *(ii)* classify the new example as an interpolative "blend" of previously trained faults.

Neither of these classification strategies is appropriate for recognizing a failure which is different from the classes of failures for which the neural network has been trained. What is needed, rather, is an ability to recognize that the new failure is not an example of any previously trained fault. In other words, the neural network must be capable of recognizing the classes for which it has been trained as well as one additional "unknown" class, even though no training examples are available for this unknown class.

If the weights in a neural network are determined solely by the training examples, then the subsequent behavior of the network is also determined by these training examples. Such a network would only be able to classify new examples on the basis of the examples it has seen, and would not be expected to reliably recognize an "unknown" class. We therefore have developed a neural network architecture in which the weights are not determined solely by the training examples. Instead, the weights are determined partly by expert judgment about the type of classification to be performed, and partly by conventional backpropagation training from examples.

This architecture has been tested in the task of sensor fusion of data from the rocket engine simulator. The purpose of the sensor fusion architecture is to classify faults of sensor readings as either *(i)* examples of normal steady-state operation, *(ii)* examples of known classes of component failures, or *(iii)* examples of an unknown class of anomalous behavior.

Sensor data typical of a rocket engine fault (with and without the addition of simulated noise) are depicted in Fig. 1. Four sensors, high pressure fuel turbopump (HPFT) temperature, thrust, chamber coolant valve (CCV) pressure, and main fuel valve (MFV) pressure are monitored during engine operation. The data are normalized with steady-state values for convenience. In the present effort, it is assumed that the engine is operating at some steady-state condition when the fault condition occurs. The fault will be manifested as a deviation of sensor values from the steady-state condition. In the present effort, a time window containing 40 sensor readings spanning four seconds is used.

It is not sufficient for the purposes of diagnosis to simply detect when and whether a deviation from steady-state conditions has occurred; how the deviation is manifested over time is also important. For instance, an observation that a particular sensor parameter is decreasing linearly will likely result in a different diagnosis than that obtained from observing an asymptotic decrease[1]. Therefore, to be effective, a diagnostic system must be responsive to the qualitative (as well as the quantitative) behavior of the engine. The diagnostic process must also exhibit resilience to noise. A noise-corrupted version of the fault is therefore depicted in Fig. 1 for comparison. The 2% noise level means that the standard deviation for a Gaussian distribution of perturbations about the noise-free curves is 0.02. A 2% noise level would be considered excessive for most instrumentation; however, satisfactory operation at this noise level is used as a goal in the present effort.

Neural networks have been employed at UTSI in the past to diagnose the development of fault conditions in jet engines[2-4], using conventional feedforward networks trained with well-known back-propagation algorithms[5]. Although this method was effective in diagnosing faults when given samples of data corresponding to a fault for which the networks were trained, several deficiencies of the conventional feedforward model were noted:

1). Networks can be trained to associate a set of patterns with a set of fault conditions quite readily. However, when presented with an input pattern qualitatively different from those included in the training examples (e.g., a pattern representing a previously unobserved fault condition), the networks can produce spurious categorizations and false positive identifications. There is no method to train a conventional feedforward network to, for instance, activate an output node if the input pattern is not similar to those patterns for which the network is trained.

2). Conventional feedforward networks are limited in the number and type of training examples which can be used. Some of the networks in the jet engine diagnostic system required long training times, which depended not only on the number of training examples required, but also on how similar training examples representing different faults were to each other.

These deficiencies can adversely effect the application of neural networks to the classification of fault patterns. The present work attempts to formulate a neural network architecture and training regimen which will successfully extend the capabilities of feedforward network models.

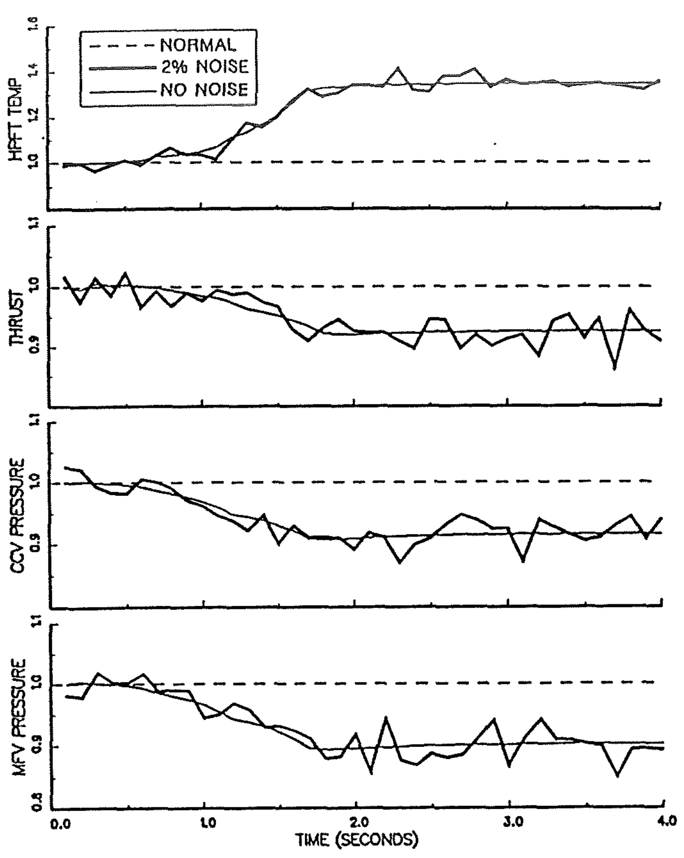

FIGURE 1. Sensor outputs for a 30% blockage of the main fuel valve with 1.5 second onset interval.

Architecture and Method

Consider the problem of inferring the operating conditions of a system from sensor data. Suppose there are M different possible system conditions $C_1 \ldots C_M$ and N different sensors $S_1 \ldots S_N$. In general, each possible condition C_i may be quantitatively characterized by a set of P parameters c_{ip}, $p = 1,2,\ldots,P$. (In our current research, each condition is characterized by two parameters, relating to the severity and onset interval of faults in jet and rocket engines.) The set of all possible system conditions can then be characterized by the set of parameters $\{c_{ip}\}$, where i enumerates the possible conditions and p enumerates the parameters associated with each condition. To infer these conditions, we may sample the value of each sensor S_j over a period of time. If each sensor is sampled at discrete times $t = 1 \ldots T$, then the complete set of sensor data is represented by the set of values $\{s_{jt}\}$, where j enumerates the different sensors and t enumerates the sampling times.

From the point of view of its operation, a system can be represented as a function

$$F : \{c_{ip}\} \rightarrow \{s_{jt}\}$$

which represents the cause-and-effect relationship from system conditions to sensor values. Ideally, if all conditions other than those enumerated could be held constant, F would be a deterministic function. In practice, however, F is usually characterized as a stochastic function in which values of s_{jt} are assumed to be influenced by noise as well as by the conditions $\{c_{ip}\}$. This noise results both from variability in testing conditions and from unreliability of the sensors. Sample data points for this function F can be derived by observing the physical system (under known test conditions) and/or by manipulating a simulation of the physical system.

In either case, the problem of inferring the condition of the system from observed sensor values is the problem of deriving the inverse function

$$F^{-1} : \{s_{jt}\} \rightarrow \{c_{ip}\}$$

which is the mapping from sensor values to hypothesized underlying conditions. For complex real-life systems such as jet and rocket engines, the nature of the function F^{-1} is unknown. However, observations of sensor values caused by known conditions (i.e. observations of F) can be used as example data points for F^{-1}. It might be thought that F^{-1} could readily be learned from these examples by a neural network trained with back-propagation. Each observed set of sensor values $\{s_{jt}\}$ would be input to the network, and the known conditions $\{c_{ip}\}$ would be the desired outputs for back-propagation training. In trials using jet and rocket engine data, however, we observed that two different fault conditions may cause very similar sensor behavior and also that low-severity, slowly developing faults are very difficult to distinguish from normal behavior. Such *differential diagnoses* have proven very difficult for an unstructured back-propagation algorithm to learn.

We propose that learning of the function F^{-1} from sample data points can be more efficient if F^{-1} is decomposed and structured into special-purpose constituent functions, each of which can be learned separately. The proposed decomposition is

$$F^{-1} = R * M * A$$

(where * denotes the composition of functions)

The purpose of the function R is data compression: to reduce the dimensionality of the sensor data $\{s_{jt}\}$ without losing the information necessary to make valid inferences. The purpose of the function M is to map this reduced sensor data $\{s'_{jh}\}$ into hypothesized conditions $\{c'_{jp}\}$. The functions R and M are applied separately to each sensor j. Finally, the purpose of the function A is to perform sensor fusion, i.e., to arbitrate among the hypotheses nominated by different sensors to determine which hypothesis is most likely. Each special-purpose component function R, M, and A is represented independently and trained separately. While the sensor fusion architecture A is the focus of the present study, the preprocessing functions R and M are discussed briefly in the following two subsections. The alternative sensor fusion architectures that were compared are then discussed in detail, and the results of the comparison are given.

Step 1: Data Compression

The data compression function R above is performed by an autoassociative network[6] with at least one layer of semi-linear hidden nodes. There are T input nodes and T output nodes, one for each discrete sampling time $t = 1,\ldots,T$. Data compression is accomplished by connecting the T input nodes to the T output nodes through a much smaller set of

$H << T$ hidden nodes. Node activations are continuous variables. The network is trained to associate input patterns with identical patterns clamped at the output nodes. As a result, training is unsupervised; it is left to the weights between the input, hidden, and output layers of the network to organize in a fashion whereby the mapping is performed correctly.

Once training has been completed, classification is accomplished by examining the activations of the hidden nodes. In a sense, the hidden node activations are employed as the "output" of the network. The mapping from input nodes to hidden nodes reduces the dimensionality of the input data from T (the number of time-series samples for a given sensor) to H (the number of hidden node activations). The mapping from hidden nodes to output nodes, on the other hand, has been used to reproduce the original input pattern from the hidden node activations. To the extent that training is successful, therefore, the H hidden node activations will contain sufficient information to reproduce the T time-series sensor values. The H hidden node activations for each sensor S_j thus compress that sensor's time-series values $\{s_{jt}\}$, $t = 1,\ldots,T$ into its hidden node activations $(s'_{j1},\ldots,s'_{jH}) = R(\{s_{jt}\})$, where R is the weighted-summation performed by the connections from the input layer to the hidden layer of the autoassociative network.

This special-purpose network for reducing the dimensionality of the sensor data can be trained very economically. Suppose that it is desired to train the overall architecture $R * M * A$ to classify a large number of temporal patterns (where each temporal pattern consists of a set of data curves, i.e. a set of time-series values for all sensors). While all of these curves will be classified by the mapping M of step 2 below, only a small subset of the training curves is needed to train the data compression network R. For training an autoassociative network to perform R, a small representative subset of the total set of curves is sufficient. Training is done on this subset of curves using an autoassociative back-propagation algorithm with output nodes clamped to the same values as the input nodes. After training is complete, data compression is accomplished by the resulting set of weights from the T input nodes to the H hidden nodes, as explained above.

Step 2: Hypothesis Generation

Once training has been completed for the autoassociative network above, the network is then run (with fixed weights) on the entire set of desired training data. Since no learning is involved at this stage, each sensor data curve $\{s_{jt}\}$, $t = 1,\ldots,T$ can be collapsed into its hidden-node representation $\{s'_{jh}\}$, $h = 1,\ldots,H$ in one iteration of the network. The H hidden node activations for each sensor constitute a compressed representation of the input data for that sensor. The entire set of training data can therefore be converted into a compressed representation with very little computation, i.e., only one iteration of the network per data curve per sensor.

Each data curve will evoke a specific hidden node response which can be represented as a single point $(s'_{j1},\ldots,s'_{jH})$ in the H-dimensional parameter space of hidden node activations. Our studies have shown that data curves generated by smoothly varying the quantitative parameters $\{c_{ip}\}$ of a given fault condition C_i result in hidden node responses which map out a surface. If H hidden nodes are used, the hidden node activations will define a surface in an H-dimensional parameter space. For example, Figure 2 illustrates surfaces generated by training an autoassociative network, with three hidden nodes, on thrust data from the Space Shuttle Main Engine. The larger surface is generated from hidden node activations which correspond to blockage at the main oxidizer valve; the smaller surface corresponds to blockage of the main fuel valve. Both surfaces represent fault conditions over a range of severities and onset intervals. The hidden-node activation resulting from normal (steady-state) operation is represented as a single point in the parameter space.

In the present example, sensor curve characteristics are functions of two parameters: severity and onset interval for each condition C_i. Curves which vary by only one parameter c_{i1} (e.g., main oxidizer valve blockages of the same severity, but of different onset intervals) will evoke corresponding hidden node activations $R(F_j(c_{i1}))$ which define one coordinate direction of the main oxidizer valve surface. That is, $F_j(c_{i1})$ denotes the set of time-series data curves of sensor S_j when parameter c_{i1} is varied. R in turn compresses each curve into a point in the hidden-node space. Curves which vary by another parameter c_{i2} (e.g., main oxidizer valve blockages which vary in severity, but are constant in onset time) will evoke hidden node activations $R(F_j(c_{i2}))$ which define the other coordinate direction of the surface. The entire surface can therefore be mapped out by systematically varying both c_{i1} and c_{i2}. If $c_{i1} \times c_{i2}$ denotes the set of all such combinations of values of the fault parameters c_{i1} and c_{i2}, then $F_j(c_{i1} \times c_{i2})$ denotes the set of time-series curves for sensor j's responses over this range

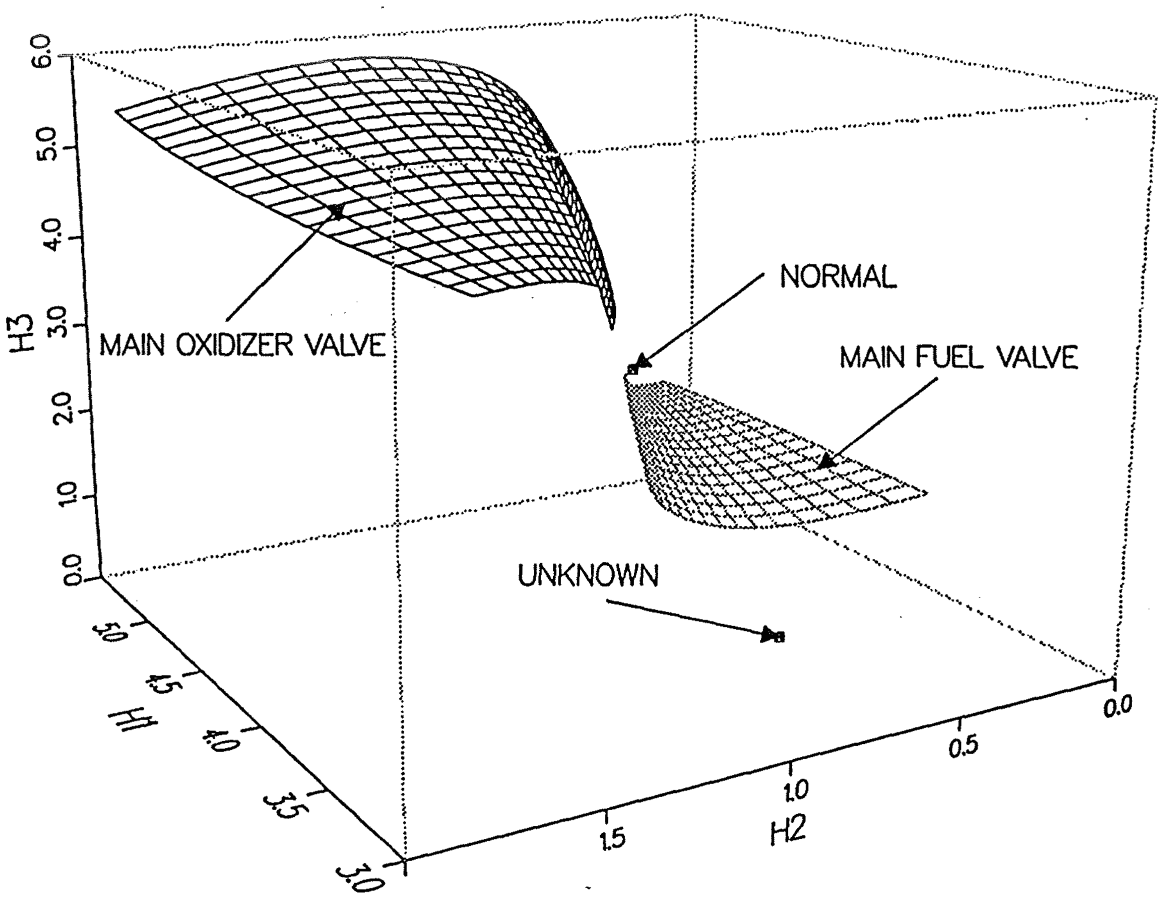

FIGURE 2. Surfaces representing Main Oxidizer Valve and Main Fuel Valve blockages for Thrust sensor. The autoassociative network maps each possible input pattern into one point H1, H2, H3, representing the hidden node activations which result from that input pattern. A class of related input patterns will generate a surface in this space.

of parameters. $R(F_j(c_{i1} \times c_{i2}))$ in turn denotes the surface obtained by compressing these sensor curves into points in the *H*-dimensional hidden-node space. Similarly, a set of main fuel valve blockages covering a range of severities and onset intervals can be used to define another surface.

As explained above, the entire set of training data can be converted into points in the hidden-node parameter space in one iteration of the autoassociative network per data curve per sensor. In the present implementation, these points are simply interpolated to generate a parameter surface for each fault condition. "Training" the map *M* from hidden-node activations to parameters consists merely of (i) generating a point in the hidden-node parameter space for each training example; (ii) labeling each such point with the known parameters (e.g. severity and onset interval) of the example which generated the point; and (iii) interpolating a coordinatized surface from the set of points obtained by varying the parameters of a given condition.

Note that each different sensor S_j will generate its own set of surfaces in *H*-dimensional data compression space, showing the responses of that sensor to the range of conditions in the training data. There will therefore be a separate map M_j for each sensor S_j.

This training process does not in any way require that only three hidden nodes be used, but is fully extensible to higher numbers of hidden nodes. If higher numbers of hidden nodes are used, higher-dimensional surfaces (hypersurfaces) will be generated.

Once a network is trained and the surfaces generated, it may be used for classification of new input patterns. In the example of Figure 2, a new input will be mapped into a new point in multi-dimensional space. If the point lies on or near the surface defined by the training examples, then the resulting hypothesis is that the fault condition represented by that surface is indeed occurring. The closer the new point is to the surface, the stronger is the evidence it provides for that hypothesis. If the new point is close to more than one surface, then more than one hypothesis will be generated, but with different levels of confidence if the distances to the surfaces are different.

After training is complete for steps 1 and 2 above, a new sensor curve $\{s_{jt}\}$, $t = 1,\ldots,T$ can be converted into a new point $(s'_{j1},\ldots,s'_{jH})$ in the hidden-node parameter space, where $(s'_{j1},\ldots,s'_{jH}) = R(\{s_{jt}\})$, R being the mapping performed by the data compression network of step 1. This point in turn can be projected onto each surface $R(F_j(c_{i1}\times c_{i2}))$ yielding values for the parameters c_{i1} and c_{i2} of the hypothesized condition C_i. The degree of evidence for each hypothesized condition is a function of the distance from the new point to the surface representing that condition, as discussed below. If j sensors are operating simultaneously, then this process is carried out separately for each sensor and each surface, projecting to the surfaces that were generated for that particular sensor during training. This will yield a different opinion from each sensor as to the likelihood of various hypothesized conditions. The purpose of the third component of our decomposed architecture is to fuse the information from different sensors into a reliable inference.

Step 3: Sensor Fusion

The key to obtaining a reliable overall inference is the reliability of the differential diagnoses which can be contributed by each sensor. Figure 3 shows a typical problem of differential diagnosis. For this sensor, the main oxidizer valve surface and the main fuel valve surface intersect. The set of hidden-node-activations near this intersection are therefore consistent with blockage of either the main oxidizer valve or the main fuel valve. In this region of the hidden-node space, differential diagnosis by this sensor would be quite unreliable. On the other hand, for points which are near one surface but not near to the other surface, the data is consistent with only one interpretation and therefore the differential diagnosis is more reliable. Finally, points far from both surfaces may indicate either an unknown condition or a faulty sensor. These different possibilities may be represented by the set of distances from a given point $(s'_{j1},\ldots,s'_{jH})$ in the hidden-node space to each of the M surfaces $\{R(F_j(c_{i1}\times c_{i2}))\}$, $i = 1,\ldots,M$ in that space (as well as the distance to the "normal" point $R(F_j(c_0))$ derived from sensor values under normal steady-state conditions).

More specifically, let us define d_{ij} to be the distance from the point $(s'_{j1},\ldots,s'_{jH})$ to the nearest point on the surface $R(F_j(c_{i1}\times c_{i2}))$, or to the normal point $R(F_j(c_0))$ when $i = 0$. Since the point $(s'_{j1},\ldots,s'_{jH})$ was derived from sensor S'_j 's data, and since the surface $R(F_j(c_{i1}\times c_{i2}))$ gives the set of such points predicted by hypothesis C_j, then d_{ij} indicates how far sensor S'_j 's data is from the predictions of hypothesis C_i. This distance can be turned into a consistency measure by defining a tolerance D_i for each surface derived from the variance observed in the set of training data used to determine each surface $R(F_j(c_{i1}\times c_{i2}))$. (A tolerance D_0 can also be defined around the normal point $R(F_j(c_0))$. A new data point closer than this tolerance to the surface should be taken as evidence in favor of the hypothesis, while a data point farther away should be taken as evidence against. We therefore take $D_i - d_{ij}$ as a measure of the *consistency* of the data from sensor S_j with hypothesis C_i.

Suppose, then, that these consistency measures are represented as activations of input nodes in a layered neural network to perform sensor fusion, as in Figure 4. In general, such a network would require $(M+1)\times N$ input nodes X_{ij}, $i = 1,\ldots,M$, $j = 1,\ldots,N$, that is, one node X_{ij} for each pairing between M+1 hypothesized conditions and N sensors (counting the steady-state condition as one hypothesis). Each input node X_{ij} of the sensor fusion network receives a scalar input $D_i - d_{ij}$, where D_i is the tolerance associated with surface $R(F_j(c_{i1}\times c_{i2}))$ and where d_{ij} is the distance between the point $(s'_{j1},\ldots,s'_{jH}) = R(\{s_{jt}\})$ and the surface $R(F_j(c_{i1}\times c_{i2}))$. (Recall that R is the data compression function performed by the autoassociative network trained in Step 1 above, and $F_j(c_{i2})$ gives the series of points generated by parameter c_{i2} of condition C_i during the training of Step 2.) Each such input node activation $D_i - d_{ij}$ therefore represents the *consistency* of data from sensor S_j with hypothesis C_i.

The desired output of the sensor fusion network is the most likely hypothesis based on all sensor data. The network therefore contains M+1 output

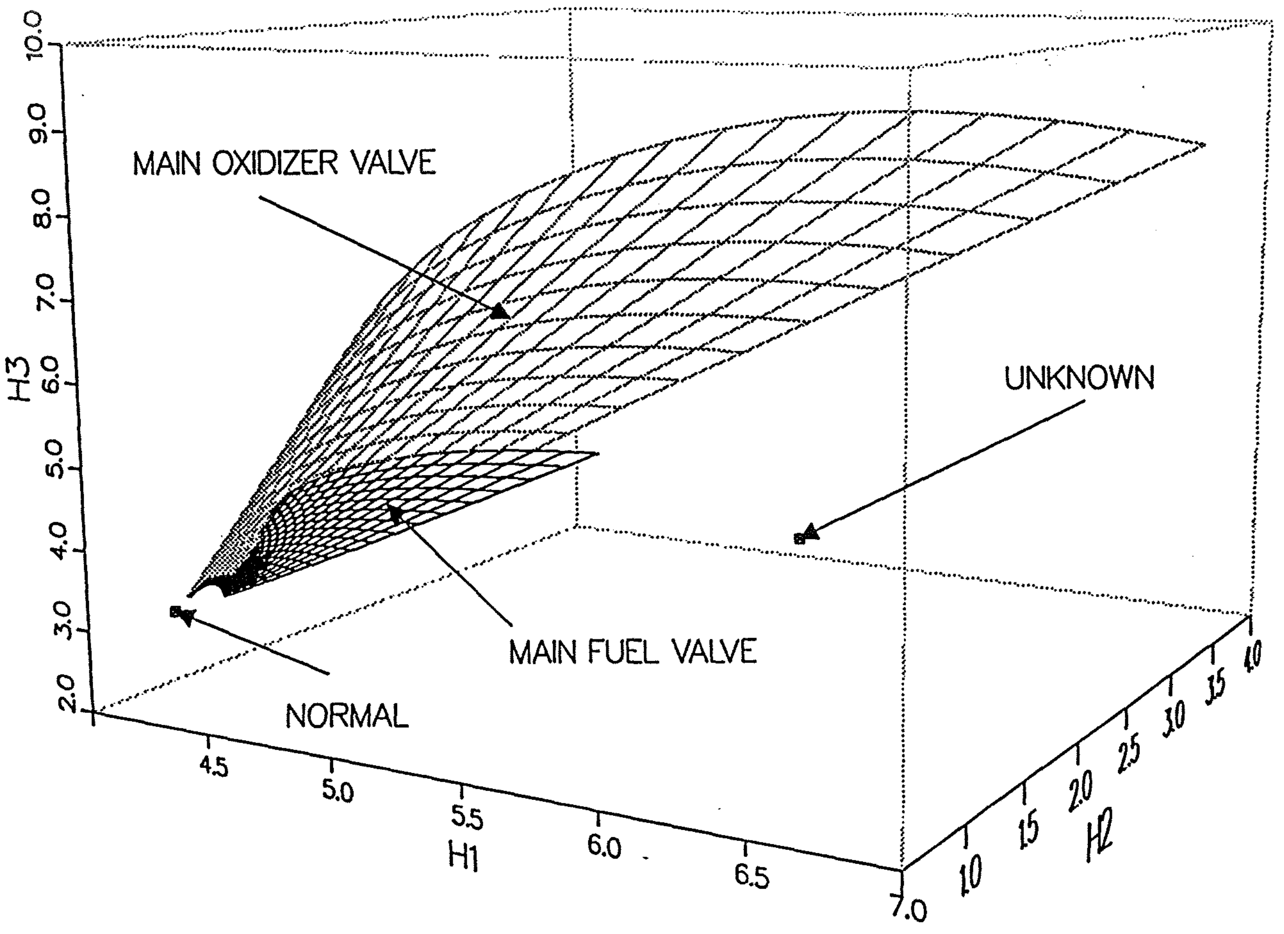

FIGURE 3. Surfaces representing Main Oxidizer Valve and Main Fuel Valve blockages for High Pressure Fuel Turbopump Inlet Temperature sensor.

nodes, one node Y_i for each possible hypothesis of a fault condition C_i, $i = 1,\ldots,M$, and one node Y_0 for the hypothesis of normal steady-state operation. Given these sets of input nodes and output nodes, and restricting ourselves to one layer of $(M+1)^2$ hidden nodes, we can then define two possible architectures which differ only in the connectivity from the input nodes to the hidden nodes and from the hidden nodes to the output nodes. These two architectures are described below:

Fusion Architecture A is shown in Figure 4. A structured connectivity pattern is defined which contains $(M+1)^2$ hidden nodes H'_{ik}, $i = 0,\ldots,M$, $k = 0,\ldots,M$. First let us consider the case where $i \neq k$. Each hidden node H'_{ik}, $i \neq k$ has random excitatory connections from the set of input nodes $\{X_{ij}\}$, $j = 1,\ldots,N$, and random inhibitory connections from the set of input nodes $\{X_{kj}\}$, $j = 1,\ldots,N$. Hidden node H'_{ik} therefore calculates a weighted sum of evidence from all sensors. In this weighted sum, evidence in favor of hypothesis C_i counts positively, but evidence against hypothesis C_i counts negatively. Conversely, evidence in favor of condition C_k counts negatively in the weighted sum, but evidence against hypothesis C_k counts positively. Hidden node H'_{ik} is thus prewired to receive all data relevant to a differential diagnosis of hypothesis C_i over hypothesis C_k. (Conversely, hidden node H'_{ik} would receive data relevant to a differential diagnosis of hypothesis C_k over hypothesis C_i. For example, if M=2 classes of fault conditions, then there are $M(M+1)$=6 hidden nodes H'_{ik} with $i \neq k$ to perform all pairwise differential diagnoses between hypotheses

C_i and C_k. These are illustrated as the upper 6 hidden nodes in the middle layer of Figure 4.

The M+1 remaining hidden nodes (designated H'_{ii} for convenience, $i = 0,...,M$) are also prewired to perform differential diagnoses, but in this case each differential diagnosis H'_{ii} is between (i) the hypothesis that the current condition of the system is C_i, and (ii) the hypothesis that the current condition of the system belongs to some *unknown* class of anomalies which has not been seen during training. (In the case where M=2, there would be 3 such hidden nodes, illustrated as the lower 3 hidden nodes of Figure 4.) In essence, each of these hidden nodes H'_{ii} is attempting to detect unknown anomalies in general, and more specifically to differentiate the class of such unknown anomalies from a particular known condition C_i. Since by definition no training examples are available for unknown anomalies, the hidden nodes H'_{ii} are prewired with inhibitory connections suitable for detecting unknowns. Recall that the activity D_i–d_{ij} of each input node X_{ij} is negative if the distance d_{ij} from the nearest known condition C_i is greater than the preset tolerance D_i. Therefore, a negative activation in any input node should count as a contribution to the evidence for an unknown anomaly. Moreover, a negative activation in the particular input nodes X_{ij} representing distance from the known condition C_i should especially count as evidence to differentiate an unknown anomaly from the known condition C_i. Each hidden node H'_{ii} for detecting unknown anomalies should therefore have some inhibitory input from all input nodes (as evidence for an unknown anomaly), and stronger inhibitory weights from the particular input nodes $X_{ij}, j = 0,...,M$ (as evidence to differentially diagnose an unknown anomaly from the condition C_i). In figure 4, dotted lines from input nodes to the lower 3 hidden nodes indicate connections which are initially set to strongly inhibitory weights to perform each differential diagnosis. As we have explained, there are also weaker inhibitory connections (not shown in the figure) from all other input nodes to each of the lower 3 hidden nodes.

The weights leading to any given hidden node define a discriminant function which is customarily thought of as a hyperplane in the space of all possible input vectors. All points (input vectors) on one side of this hyperplane result in a positive activation of the given hidden node; all points on the other side result in a negative activation. This discrimination is sharpened by the nonlinear sigmoid function applied to the activation to yield the hidden node's output. The

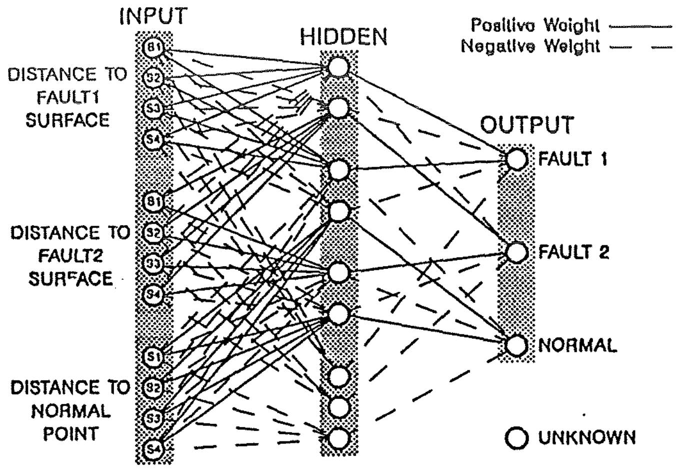

FIGURE 4. Architecture A: Prestructured to perform differential diagnosis (all connections not shown). Solid lines indicate connections initially set to positive wieghts (before training begins); dotted lines indicate connections initially set to negative weights.

weight changes of back propagation training in effect move each hyperplane in a direction which lessens the mean-squared error of the network's output over the training set.

From this standpoint, the differential diagnosis performed by each hidden node H'_{ik}, $k \neq i$, can be thought of as a hyperplane intended to separate the data points representing training examples of condition C_i from those representing training examples of condition C_j. While the initial setting of weights biases each hidden node H'_{ik} to perform this differential diagnosis, back propagation will move this hyperplane in whichever direction minimizes the mean-squared error of the network over the set of training examples.

Similarly, the role of each hidden node H'_{ii} can be viewed as a hyperplane intended to distinguish all known conditions C_i on one side, from unknown anomalies on the other side of the hyperplane. M+1 such hyperplanes are created by the hidden nodes H'_{ii}, $i = 0,...,M$. These M+1 hyperplanes are initially placed in different positions due to the stronger inhibitory weights assigned to the inputs $X_{ij}, j = 0,...,M$ than to the other inputs by each particular hidden node H'_{ii}, as explained above. Since no training examples are available for unknown anomalies, back propagation might conceivably reduce the number of known training examples erroneously classified as unknown, but would not be expected to improve the recognition of unknown anomalies as such. We hypothesized, however, that the effect of this inherent limitation of example-based training would be lessened by the prestructuring (initial setting of connections and weights) of architecture A described above.

Finally, each output node Y_i receives initially excitatory connections from the hidden nodes $\{H'_{ik}\}$ $k = 1,\ldots,M$, $(k \neq i)$, initially inhibitory connections from the hidden nodes $\{H'_{ki}\}$ $k = 1,\ldots,M$, $(k \neq i)$, and an initially inhibitory connection from the hidden node H'_{ii}. Each output node Y_i thus receives excitatory input from all differential diagnoses favoring hypothesis C_i, and inhibitory input from all differential diagnoses opposing hypothesis C_i.

Fusion Architecture B is shown in Figure 5. This architecture has the same number of hidden nodes as architecture A, and the same number of input and output connections per hidden node, but with random connectivity from input to hidden and hidden to output layers, and random assignment of initial weights. This architecture is intended to serve as a "control" case against which the effects of the initial structuring of architecture A can be evaluated.

Both architectures for sensor fusion were trained with the same back-propagation algorithm, using the outputs of the training data that were used in step 2 to determine the surfaces $\{R(F_j(c_{i1} \times c_{i2}))\}$, $i = 1,\ldots,M$ for each sensor j. Our hypothesis was that the back-propagation algorithm constrained to the connectivity of architecture A would (i) result in a set of weights from input nodes to hidden nodes which allow the hidden nodes to perform pairwise differential diagnoses, (ii) would reliably differentiate *unknown* engine conditions (not present during training) from the known classes of engine conditions present during training (a capability expected to be lacking in the unstructured back propagation architecture B), and (iii) that this would be accomplished without sacrificing diagnostic performance for known engine conditions, in comparison to the unstructured back propagation architecture.

Testing Procedure

The decomposed architecture (performing data compression, hypothesis generation, and sensor fusion) was trained as described in steps 1-3 above on simulated SSME data for the four sensors illustrated in Figure 1: high pressure fuel turbopump temperature, engine thrust, chamber coolant valve pressure, and main fuel valve pressure. The training set consisted of normal data and two fault conditions, main oxidizer valve (MOV) blockage and main fuel valve (MFV) blockage. Each fault condition was included in the training set at three different levels of severity,

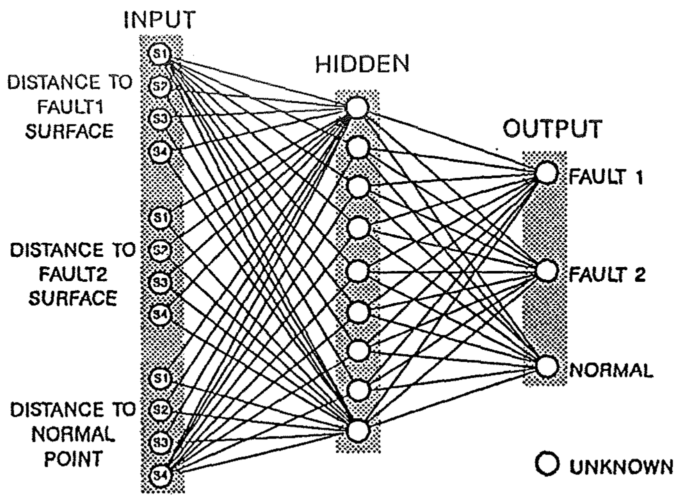

FIGURE 5. Architecture B: Fully connected with random initial weights (all connections not shown). All connections are initially set to random weights drawn from a uniform distribution between -0.1 and 0.1.

and three different onset intervals for each level of severity. In order to compare the performance of sensor fusion architectures A and B, they were each trained on the identical output of Steps 1 and 2 of the training procedure.

After training was completed, the performance of architectures A and B were compared on test data containing the 3 "known" conditions used in training (normal, MOV blockage, and MFV blockage) and 2 additional fault conditions that were not presented during training, Oxidizer Preburner Valve (OPV) blockage and Fuel Preburner Valve (FPV) blockage. These two additional "unknown" conditions were included in order to test each architecture's ability to detect fault conditions which had not been included in the training set. 500 instances of each condition were generated which differed only in the amount of noise added to the simulated data. For each noise level, 100 instances of each condition were generated with the inclusion of random noise at that noise level. During testing each example was classified as one of the three training conditions (normal, MOV blockage, or MFV blockage) based on the maximally active output node, or as "unknown" if none of the output nodes was activated above its threshold value. Each architecture (A and B) was tested using a range of different thresholds, and the threshold yielding the best performance for that network was used.

Table I

Percent correct classification

at various noise levels performed by prestructured architecture A.

Fault	Noise Level (%)				
	0.0	0.5	1.0	1.5	2.0
Normal (Known)	100	100	95	81	70
MOV (Known)	100	100	100	100	100
MFV (Known)	100	97	86	85	80
OPV (Unknown)	100	100	100	100	100
FPV (Unknown)	100	100	100	100	100

Table II

Percent correct classification

at various noise levels performed by fully connected architecture B.

Fault	Noise Level (%)				
	0.0	0.5	1.0	1.5	2.0
Normal (Known)	100	100	94	76	69
MOV (Known)	100	100	100	100	98
MFV (Known)	100	94	86	79	79
OPV (Unknown)	0	0	0	1	5
FPV (Unknown)	0	0	0	0	0

Results and Discussion

Table I shows the results of testing sensor fusion architecture A. As explained above, this neural network architecture was structured prior to training to perform differential diagnoses among the different conditions to be presented during training, and between each of these conditions and the class of unknown faults not presented during training. Table II shows the results of testing architecture B using the same training data and same testing data. Architecture B differed from architecture A in that the connection weights were not structured prior to training, but rather were set to random initial values as typically done in neural networks.

Each column in Tables I and II shows the results obtained for a given noise level in the test data. Within each column, classification performance (percent correct classifications) is shown separately for the normal operation condition and for each fault condition.

The first three rows of each table show the performance on test data representing the three known conditions (normal and two types of faults) that were presented during training. The performance of both architectures declined with the addition of greater levels of noise to the test data, with architecture A performing very slightly better at the higher noise levels, As expected, the two architectures did not differ greatly in their classification performance on these known conditions. This is consistent with our hypothesis that the initial structuring would not detract from the network's ability to correctly classify new examples of the same conditions presented during training.

The last two rows of each table show the results of testing with the two unknown fault conditions which had not been presented during training. Since a conventional back propagation network learns to classify based on its training examples only, we expected architecture B not to be able to recognize new fault conditions as unknown, but rather to classify them into one of the classes that had been presented during training. This is indeed what happened. As shown in the last two rows of Table II, its performance was five percent or less correct classifications at each noise level of these two unknown faults. Recall that results are shown for the best threshold setting for each network. In other words, there was no threshold setting which would allow the output nodes of architecture B to differentiate known examples from unknown examples on the basis of its output node activity.

Architecture A, by contrast, was able to correctly identify new faults as unknown, based on below-threshold activity in all output nodes. This is shown in the last two rows of Table I, in which all examples of new faults at all noise levels are correctly classified on this basis. Since the only difference between architectures A and B is in the initial structuring of connections prior to training, it is reasonable to conclude that the initial structuring of architecture A to perform differential diagnoses allows this network to detect examples of unknown classes that were not presented during training.

The structured back-propagation network of architecture *A* could be viewed as a hybrid between a knowledge-based expert system and an example-trained neural network. In knowledge-based systems, both the general format of the rules and the exact instantiations of the rules are extracted from human experts. In conventional back-propagation networks, hidden nodes serve a function analogous to rules in an expert system. The general format of such a "hidden-node rule" is determined by which input nodes have significant connections to each hidden node, while the exact instantiation of such a rule is given by the exact weights which result from training. In conventional back-propagation, both the format of the hidden-node rules and the exact instantiation of these rules are implicitly determined from training examples by the back-propagation algorithm. The structured back-propagation architecture *A* above represents a hybrid between these two approaches. The general format of its "hidden-node rules" is determined in advance by the connectivity specified for architecture A, and is based on expert judgment about the general utility of rules based on differential diagnosis. Within this general format, however, the specific instantiation of each differential diagnosis rule is determined by the exact weights which are learned from training examples via back propagation. This initial structuring of the back-propagation architecture using expert human judgment allows the neural network to detect the occurrence of faults for which no training examples were presented. This dependence on expert human judgment is much less than in a rule-based expert system, however, since the exact instantiations of the rules are still learned from training examples. Training based on examples should make these "hidden-node" rules easier to maintain than in a conventional rule-based expert system. However, this reduced dependence and implicit learning of "hidden-node-rules" makes it more difficult to provide explanations to the user about the inference process. Nevertheless, forcing the "hidden-node rules" into a predefined format allows the behavior of the network to be more easily compared with the behavior of human experts than in the case of an unstructured back-propagation network. This potentially would allow the network to "explain" the reasons for its diagnosis in terms understandable by a human operator.

Acknowledgement

This work was performed within the Center for Advanced Space Propulsion (CASP) and was supported in part by NASA Grant NAGW-1195 and Rocketdyne Contract No. R04QBZ90-032709. The Center for Advanced Space Propulsion is part of The University of Tennessee-Calspan Center for Aerospace Research, a not-for-profit organization located at UTSI. The authors would like to thank A. M. Norman for his assistance during the course of this effort.

References

1. Cikanek, H.A., "Space Shuttle Main Engine Failure Detection", presented at the 1985 American Control Conference, Boston, MA, June 19-21, 1985.

2. Dietz, W.E., Kiech, E.L., and Ali, M., "Jet and Rocket Engine Fault Diagnosis in Real Time", *Journal of Neural Network Computing*, Vol. 1, No. 1, 1989.

3. Dietz, W.E., Kiech, E.L., and Ali, M., "Pattern-Based Fault Diagnosis Using Neural Networks", The First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Tullahoma, TN., pp 13-23, June 1-3, 1988.

4. Dietz, W.E., Kiech, E.L., and Ali, M. "Classification of Data Patterns Using an Autoassociative Neural Network Topology", The Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, The University of Tennessee Space Institute, Tullahoma, TN, pp. 1028-1036, June 6-9, 1989.

5. Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning Internal Representations by Error Propagation", pp. 318-364, in *Parallel Distributed Processing*, Vol. I, The MIT Press, Cambridge, MA, 1987.

6. Kuczkewski, R.M., Myers, M.H., and Crawford, W.T., "Exploration of Backward Error Propagations as a Self-Organizational Structure", TRW MEAD, One Rancho Carmel, San Diego, CA 92128.