Automated Downlink Analysis for the Deep Space Network

D. Watola
Communications Systems and Research Section

J. B. Hampshire II
Carnegie–Mellon University, Pittsburgh, Pennsylvania

The downlink analyzer (DLA) is a hybrid learning and monitoring system combining classical signal-processing and connectionist (i.e., neural network) pattern classification. It learns to detect and diagnose anomalous operations in the downlink portion of the Deep Space Network, NASA's communications link to all unmanned spacecraft. To meet its learning and monitoring objectives, the DLA must process data sequences gathered from throughout the downlink. Many sequences have low signal-to-noise ratios, and the sample rates for different information sources vary widely. This article describes the technologies employed by the DLA, focusing on aspects that are novel. In the process of learning to date, the DLA has discovered a previously unknown downlink failure mode, providing the first direct evidence that it can teach human engineers, scientists, and operators subtle but important factors that lead to the loss of scientific data.

I. Introduction

The Jet Propulsion Laboratory operates the Deep Space Network (DSN), NASA's communication link to all unmanned spacecraft operating in and beyond the known planetary limits of the solar system. A typical DSN deep-space downlink is expected to recover telemetry from a spacecraft that is billions of kilometers from Earth and emitting fewer than 20 W—less than 1/25,000 of the power output from a typical commercial radio station. Consequently, the burden on the terrestrial antenna, radio-frequency (RF) amplifiers, demodulators, decoders, etc., is such that they must operate near physical and theoretical limits; the incoming signal is faint enough that even small deviations from nominal specifications can result in a complete loss of telemetry. Loss of telemetry means loss of scientific data, with a commensurate reduction in the total information yielded by a mission. Consequently, there is a strong incentive for JPL/NASA to develop a fast and effective means of diagnosing and correcting DSN anomalies.

The complexity of the task and volume of data are both so large that humans frequently cannot diagnose downlink faults in real time. As NASA missions become more demanding, the inadequacy of human abilities to maintain reliable communications is increasingly apparent. This realization has spawned the Downlink Analyzer (DLA) Task, a research project with the following three fundamental objectives:

(1) Provide automated real-time detection and diagnosis of anomalous downlink operations.

(2) Learn heretofore unknown downlink failure modes without human intervention.

(3) Provide statistical analysis tools to downlink operators, engineers, and scientists for further exploration of any autonomously discovered relationships.

Meeting these objectives entails monitoring signals that originate throughout the downlink. Some such signals possess very low signal-to-noise ratios. Sampling rates also vary widely among the myriad data sources. The situation can be further complicated by data subjected to operations invalidating the standard assumptions associated with traditional signal processing; examples include nonuniform sampling, undersampling with aliasing, and nonlinear (e.g., median) filtering. Thus, the problem may not be amenable to purely classical methods.

The downlink analyzer (DLA) is a tool for performing model-based fault diagnosis using a nonparametric empirical model. It achieves its goals using a combination of classical signal processing with a robust connectionist (i.e., neural network) pattern classifier. Though it is targeted to fault identification applications in Deep Space Network downlink subsystems, its principles are applicable to a broad class of statistical pattern recognition and anomaly isolation problems.

This article describes the first complete implementation of the DLA and some of the techniques it employs in order to learn and subsequently diagnose downlink failure modes. In particular, it focuses on technical aspects that are relatively novel. The exposition is divided into seven major parts. The first, Section I, is this introductory segment. Section II describes the overall operation of the downlink analyzer at a high level. Sections III, IV, and V respectively detail the three major subsystems comprising the implementation: the front end or feature extractor that performs fault detection, the neural network responsible for system and fault modeling, and the diagnostic procedure that isolates sources of anomalies. Section VI supplies a practical example of the entire system at work with experimental data. Finally, Section VII concludes the article with a brief summary.

II. System Overview

A block diagram depicting the high-level components and subsystems of the DLA is given in Fig. 1. Raw time series data, usually comprising several sequences of monitor data from a DSN monitor and control subsystem, enter the system at the left of the figure. A very small proportion of these time series represent observable health metrics, or statistics that are strongly correlated with the "health state" of the downlink. The remainder are causal factors characterizing the operation of a downlink subsystem or component thereof. Only one restriction is placed on the content of this data: for each system subject to fault diagnosis, the data must be partitionable into a single principal health metric, h, and a set of relevant causal factors, $\{c_i\}$. There are no other constraints on the nature or content of the inputs, which need not even be synchronously or uniformly sampled.

The principal health metric is selected such that a simple thresholding function, $\Omega_h(h)$, can be formulated to yield the empirical health state of the system, Ω_h. Any number of health classifications can be accommodated, but the DLA posits only two and adopts the convention that $\Omega_h = \Omega_1$ denotes the GOOD or healthy state while $\Omega_h = \Omega_0$ indicates a BAD or anomalous condition. Generally, the empirically derived Ω_h is taken to be equal to the actual system health state Ω even though h is invariably a sample from a stochastic process subject to underlying random process noise, measurement noise, and time delays. Hence, the true mapping from h to Ω is characterized by a set of a posteriori probabilities $\{P(\Omega_i|h)\}$, and Ω_h is only an estimate of Ω. For a principal health metric that is frequently updated and estimated with high statistical confidence, the simplification proves viable.

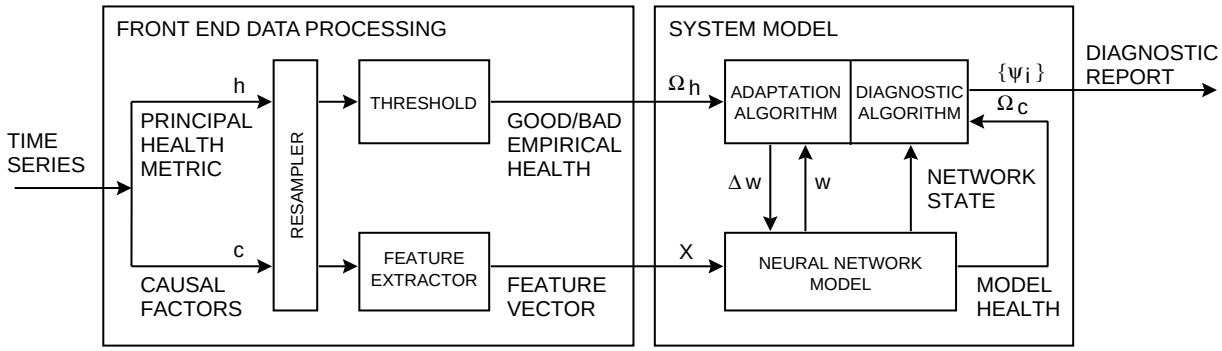

Fig. 1. The downlink analyzer.

Elements of $\mathbf{c} = \{c_i\}$ are chosen such that an arbitrarily complex function $\Omega_c(\mathbf{X})$ can be constructed that also produces the correct value $\Omega_c = \Omega_h = \Omega$ without examining h. Here, $\mathbf{X}$ is a feature vector derived from raw $\mathbf{c}$ samples to facilitate this mapping from $\mathbf{c}$ to Ω_c; its components are arbitrary functions of the causal factors and typically include short-term statistics computed over the c_i data. Thus, $\Omega_c(\mathbf{X})$ constitutes a statistical pattern classifier that distinguishes among health states based on its internal model of the mapping from transformations of observable time series to the empirical health state.

In an operational environment, the DLA applies an analysis technique called discriminative diagnosis to dissect this more complicated model, $\Omega_c(\mathbf{c})$, in order to determine the cause of a transition from healthy state Ω_1 to anomalous state Ω_0 whenever a fault is indicated by the simple model, $\Omega_h(h)$. Results of this mathematical postmortem are a ranked set of saliencies $\{\Psi_i\}$ providing a quantitative measure of the contribution from each changing component in $\mathbf{X}$ to the degraded performance. The final ranked list of causal factors forms the output of the complete downlink analyzer system.

III. Signal Processing for Fault Detection and Feature Computation

A. Downlink Analyzer Signal Processing

The default signal processing applied to incoming DLA data is deliberately kept simple because the nature of the input streams frequently confounds traditional signal-processing strategies. Among the challenges accompanying DSN monitor data are unknown or time-varying latency, nonuniform sampling, asynchronous sampling, widely varying average sample rates, missing samples, and samples that have already been processed with unknown methods. This section describes some of the solutions that the DLA applies to these problems.

Once the raw data in Fig. 1 are split into a principal health metric and all other causal factors, the downlink analyzer front end passes each data stream through a user-controlled resampling process. The resampler can leave the data unmodified or has the option to resample the streams on an individual basis. Using this mechanism, time series can be synchronized with each other or to an independent source of reference events (e.g., one sample every 5 s). Provisions exist for inserting (repeating) samples when the underlying process does not emit them often enough and for recognizing the existence of a large gap. In the DLA, the most common application of resampling forces all data streams to be synchronously (but not necessarily uniformly) sampled. This resampling process is not required, but it does make the data more pliable to analysis and presentation outside of the context of the DLA.

After resampling, the principal health metric is transformed into an empirical health state by comparing it to a set of user-supplied thresholds. Threshold selection is usually based on empirical observations and human experience with the target system. Any number of health states may be defined for a principal health metric, but ultimately the empirical health state will be reduced to either GOOD or BAD.

This is the essence of the fault detection process in the DLA: it is accomplished solely by thresholding the principal health metric.

Resampled causal factors usually do not themselves comprise the input to the model. Instead, the DLA front end provides a mechanism for computing features based on resampled data. Support for features derived from raw inputs allows engineers to incorporate some a priori knowledge about the relationships in the system being modeled rather than forcing the neural network to allocate its time and limited functional complexity toward discovering those relationships. In the DLA, features are user-defined, arbitrarily complex functions of the instantaneous (last-known) values of the resampled data. More importantly, features can also be derived from data statistics measured over a feature-specific, user-selected temporal window. The enabling technology is the moving box-plot filter described below.

B. Moving Box-Plot Filters

The moving box-plot filter is a flexible tool for examining time series with samples that arrive at irregular intervals or for combining and analyzing multiple series that are not synchronized. It is based on the box plot, a nonparametric statistical summary first developed by Tukey [7]. Box plots in the DLA follow the methodology described in [1], where the empirical distribution of a hypothetical sample of n points, $\mathbf{x} = \{x_1, \cdots, x_n\}$, is succinctly summarized by nine quantities:

$S_0 =$ the minimum value in the sample set

$S_1 =$ the lower outer fence, defined as the larger of S_0 and $S_3 - 3(S_5 - S_3)$

$S_2 =$ the lower adjacent fence, defined as the larger of S_0 and $S_3 - 1.5(S_5 - S_3)$

$S_3 =$ the boundary between the first and second quartile

$S_4 =$ the boundary between the second and third quartile (i.e., the median value)

$S_5 =$ the boundary between the third and fourth quartile

$S_6 =$ the upper adjacent fence, defined as the lesser of S_8 and $S_5 + 1.5(S_5 - S_3)$

$S_7 =$ the upper outer fence, defined as the lesser of S_8 and $S_5 + 3(S_5 - S_3)$

$S_8 =$ the maximum value in the sample set

All nine statistics are trivial to compute once the samples have been sorted into ascending order; it is only necessary to partition the sorted set into quartiles. Quantities S_0 through S_8 are illustrated for a sample $\mathbf{x}$ ensemble in Fig. 2, which demonstrates how the box plot gives a useful synopsis of the statistical properties of the sample. Specifically, the variance, skewness, and kurtosis of the empirical distribution are clearly indicated by the height of the box, the position of the median within the box, and the relative length of the "fences."

With the addition of a moving window to select the sample set from a time series, the box plot changes from a simple statistical summary to a nonlinear filter with multiple outputs. The resulting filter is a particularly powerful tool for analyzing nonuniformly sampled time series since the observation window is not constrained to be synchronized with any particular event or even to contain a fixed number of samples.

In the DLA, empirical health state estimates and all feature vector components must be issued simultaneously even though the arrival times of samples from the various underlying time series are not

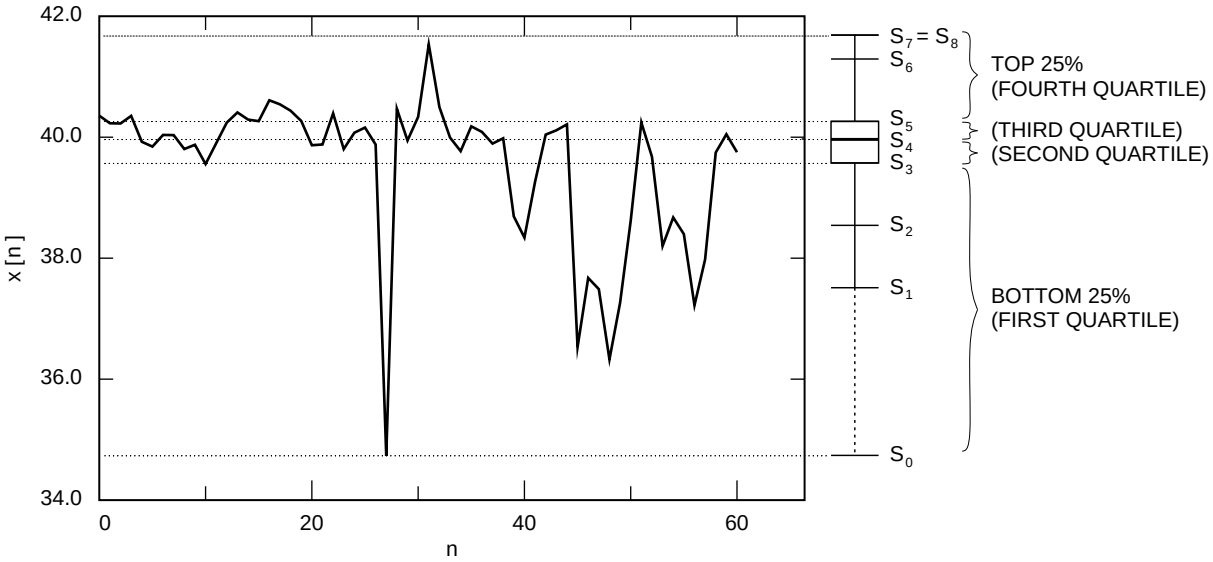

Fig 2. A sample 61-point ensemble and its annotated box plot.

guaranteed. Typically, $(\mathbf{X}, \Omega_h)$ pairs are desired either at regular intervals or synchronously with the arrival of new raw samples of the principal health metric. Moving box-plot filters provide a mechanism for decoupling the health state estimate and feature vector emissions from the input sampling instants without adding much complexity. Therefore, the DLA applies a moving box-plot filter to both principal health metric and causal factor data. Health state estimates are always found by selecting the worst-case value (S_0 or S_8) of the principal health metric from the observation window and comparing that value to the predetermined health class thresholds. Feature vectors are arbitrary functions of the nine filter outputs and the most recent instantaneous sample value. Note that it is not necessary to apply the same moving box-plot filter uniformly everywhere; if observation window durations are tailored to the data, they may vary across the different time series.

Moving box-plot filter technology contributes more to the downlink analyzer than just a convenient solution to the challenges of coordinating asynchronous inputs from multiple sources. It also introduces an element of time dependence into the model, which otherwise is completely feedback-free and lacking in dynamic structure. Without these filters, the simple neural network presented in Section IV below would describe a purely quasistatic mapping from $\mathbf{c}$ to Ω_h and consequently would be incapable of incorporating temporal dependencies among causal factors into its model since feature vectors could then be based only on instantaneous samples. Application of the box-plot filter to extract worst-case values or other sample statistics tends to smear events in the data such that temporal juxtapositions can be detected in addition to simple coincidences. Adding some delay to the system enhances this effect by allowing the filters to appear noncausal, extending the range of the observation window without considering events any more temporally distant from the "current" time instant.

IV. System and Fault Modeling

A. Neural Networks as Nonparametric System Models

The mapping from $\mathbf{X}$ to Ω_c is accomplished by a neural network [3,6]. Therefore, instead of resulting from rigorous mathematical analysis of the underlying physical processes that produce the various $\{c_i\}$, this $\Omega_c(\mathbf{X})$ model evolves under the supervision of a robust adaptation algorithm while the network is "trained" to classify empirical data correctly. By providing a general-purpose nonparametric framework for characterizing the desired process, a connectionist model avoids some of the challenges associated

with identifying a suitable parametric model for the system. Of course, literally nonparametric models are an illusion—the connection weights in the neural network, shown as $\mathbf{w}$ in Fig. 1 and fully described below, clearly form a set of model parameters. However, the neural network paradigm represents a nonparametric model in the same sense that classical (i.e., periodogram-based) spectral analysis techniques yield nonparametric frequency-domain process models: the parameters are coefficients of basis functions spanning a particular hypothesis class.

Connection weights in the network are determined by an adaptation algorithm that is essentially an optimization procedure. In the downlink analyzer, the network is exposed to a training set of feature vector/empirical health state pairs carefully selected to be representative of the underlying input space. Network response to these training pairs is monitored as the adaptation procedure iteratively refines the model until acceptable discrimination performance is demonstrated over the training ensemble. Ideally, the training set provides uniform coverage of the multidimensional input space such that, if the network successfully learns those patterns, it will exhibit reasonably good generalization performance when confronted with novel feature vectors during its normal operating regime. Note that training occurs off-line, orthogonal to any real-time operational use.

The remaining subsections in Section IV present the specific neural network used by the DLA, along with the algorithms that mold the network into a robust statistical pattern classifier. Although the DLA uses a fairly conventional neural network paradigm with standard processing units connected in a very common topology, a complete (but terse) description is included here since some of the innovations and variations are both subtle and important. The in-depth explanation also serves to introduce a common notation that is employed for the remainder of this article.

B. Network Architecture and Computation

A neural network can be characterized as a directed graph containing N nodes, representing neurons or processing units, indexed with integers on $[1, N]$. The first N_I units are designated as inputs and may not have edges entering them; these input nodes perform no computation, instead behaving as placeholders for introducing external stimulus to the rest of the network. The remaining nodes are computational units, with the final N_O neurons designated as the observable network outputs. The simple network shown in Fig. 3 illustrates the concepts in this section for $N = 6, N_I = 3$, and $N_O = 1$.

The graph and neural network architecture are described by an $N \times N$ interconnection matrix, $\mathbf{I}$, where the elements $I_{i,j}$ of $\mathbf{I}$ are indicator variables denoting the presence or absence of a connection to neuron i from neuron j:

$$I_{i,j} = \begin{cases} 1 & \text{if a connection exists to neuron } i \text{ from neuron } j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Associated with each edge is a finite connection weight, $w_{i,j}$. Elements of the $N \times N$ weight matrix $\mathbf{W}$ are zero wherever the corresponding element of $\mathbf{I}$ is zero. There are a total of N_w weights in the network, with

$$N_w = \sum_{i=1}^{N} \sum_{j=1}^{N} I_{i,j} \tag{2}$$

Because the weight matrix is sparse, it is often convenient to refer instead to the weight vector, $\mathbf{w}$, containing the subset of elements of $\mathbf{W}$ that represents physical connections.

DLA models are restricted to the class of strictly feedforward multilayer networks. In these architectures, the sets of input and output neurons comprise the input and output layers, respectively. Any

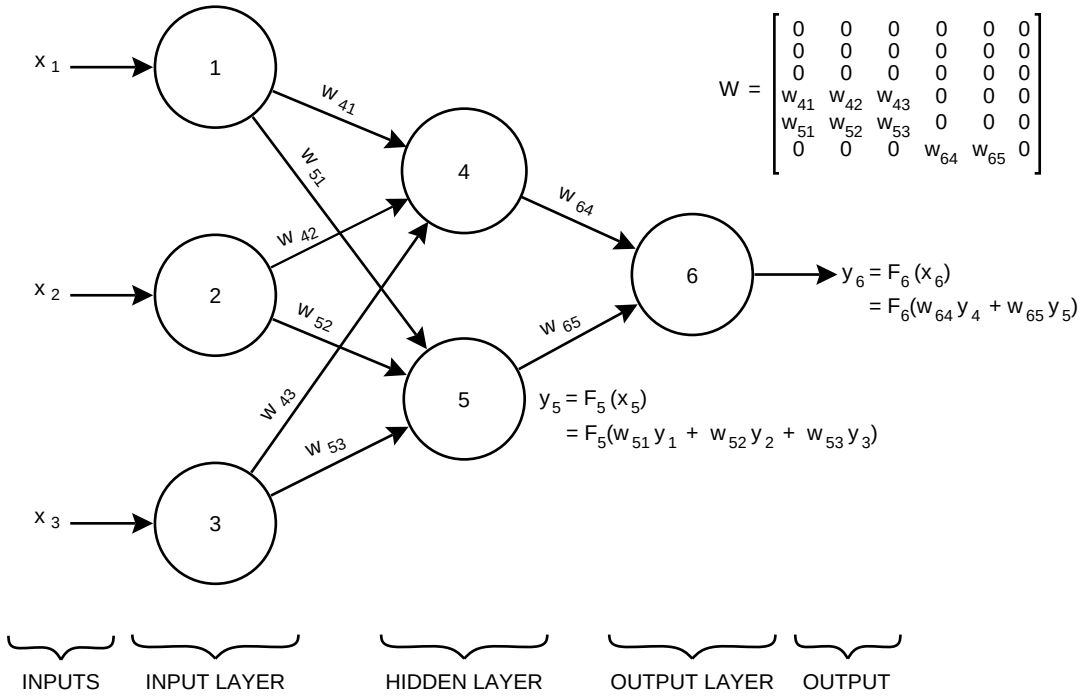

Fig. 3. A sample three-layer, three-input, one-output neural network.

intervening nodes are organized into zero or more discrete "hidden" layers. Connections are permitted only between neurons in adjacent layers and must be oriented in the direction of computation (i.e., toward the output layer). Thus, the graph is acyclic and $\mathbf{I}$ is strictly lower triangular:

$$I_{i,j} = 0, \quad \forall i \geq j \tag{3}$$

Except for input units (which simply pass their inputs to their outputs unchanged), each neuron i computes an output y_i by transforming its net input x_i through a nonlinear activation function $F_i(\cdot)$:

$$y_i = F_i(x_i) \tag{4}$$

where x_i is defined as the sum of neuron inputs weighted by the corresponding connection strengths:

$$x_i = \sum_{j:I_{i,j}=1} w_{i,j} y_j \tag{5}$$

Normally, each (noninput) unit also possesses a bias weight $w_{i,0}$ driven by a virtual input with fixed stimulus $y_0 = 1$; this can be accommodated in the above characterization by augmenting $\mathbf{I}$ and $\mathbf{W}$ with another column.

Two types of activation functions are used by the DLA. All output units use the logistic sigmoid function

$$F(x) = \left(1 + e^{-x}\right)^{-1} \tag{6}$$

in order to constrain the output conveniently between zero and unity. Hidden units, however, have activation functions that are hyperbolic tangents to retain support of bipolar-valued outputs on hidden nodes. Both functions are common choices since they possess several desirable characteristics, viz., boundedness, continuous differentiability, and a "soft saturation" characteristic.

From Fig. 3 and the description above, it is obvious that the overall computation for a feedforward neural network maps length-N_I input vectors $\mathbf{X} = (x_1, \cdots, x_{NI})$ to length-N_O output vectors $\mathbf{Y} = (y_{N-N_O+1}, \cdots, y_N)$, with computation proceeding layerwise from inputs to outputs.

C. Network Training

Because input/output relationships are often complex, the weights necessary to effect a desired mapping from input vectors to output vectors are not usually determined by explicit computation. The DLA employs supervised learning, wherein appropriate weights are found by a feedback-driven iterative optimization procedure. During repeated exposure to input vectors from a carefully selected training set of ordered triplets $\{(\mathbf{X}, \mathbf{Y}, \mathbf{T})_i\}$, the network uses an objective function to assess current network performance as a function of the output vector $\mathbf{Y}$ and the corresponding target vector $\mathbf{T}$. Changes in the objective function drive weight adaptation until $\mathbf{w}$ reaches a local optimum in the weight space.

The optimization procedure used by the DLA is a gradient search [3]. If the objective function is to be minimized, the search is a gradient descent; otherwise, it is a gradient ascent. The general optimization strategy is simple: follow the local gradient, taking small steps in weight space until the gradient vanishes. Thus, at time step (epoch) n, the adaptation proceeds according to

$$\Delta\mathbf{w}^{(n)} = \pm\alpha\frac{\nabla_w\eta(\mathbf{w}^{(n)})}{||\nabla_w\eta(\mathbf{w}^{(n)})||} + \beta\left(\Delta\mathbf{w}^{(n-1)}\right) \tag{7}$$

where $\eta(\cdot)$ is the objective function, $\mathbf{w}^{(n)}$ is the current vector containing all N_w weights in the network, $\Delta\mathbf{w}^{(n)}$ is the latest weight update vector, α is a small positive learning rate constant controlling the length of each step in the direction of the gradient, β is a momentum constant for accelerating the search in regions with negligible higher-order derivatives, and the gradient vector components are given by

$$\left[\nabla_w\eta\left(\mathbf{w}^{(n)}\right)\right]_i = \frac{\partial\eta(\mathbf{w}^{(n)})}{\partial w_i}, \quad i = 1, 2, 3, \cdots, N_w \tag{8}$$

The sign of the first term in Eq. (7) is negative for gradient descent, and positive otherwise. In the DLA, the gradient referenced in Eqs. (7) and (8) above is actually an average gradient accumulated over all patterns (and all objective function scores) in the training set, but adaptation on a pattern-by-pattern basis is also viable. With appropriate choices of α and β, this procedure converges to a locally optimum solution for $\mathbf{w}$.

D. Objective Functions

The most common objective function driving supervised learning is the mean-squared error (MSE) criterion,

$$\eta_{MSE}(\mathbf{Y}, \mathbf{T}) = \frac{1}{N_O}\sum_{i=1}^{N_O}(Y_i - T_i)^2 \tag{9}$$

which is forced to a local minimum over the collection of training triplets $\{(\mathbf{X}, \mathbf{Y}, \mathbf{T})_i\}$ by the adaptation procedure in order to make the ensemble of $\mathbf{Y}$'s closely approximate the ensemble of target $\mathbf{T}$'s. Intuitively, MSE is well suited to neural networks that approximate continuous functions of their input vector components.

The DLA, however, performs a much simpler task: pattern classification. Here, each input vector is associated with one of N_O classes. The target vector components are binary indicator variables with

$$T_i = \begin{cases} 1 & \text{if } \mathbf{X} \text{ belongs to the } i\text{th class} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

and the neuron with the largest output is interpreted as designating the class determined by the network. Mean-squared error is not well suited to this situation, where the target functions are intrinsically discontinuous; in fact, it can be shown that η_{MSE} sometimes increases with improved classification performance [1].

The DLA adopts the classification figure of merit (CFM) objective function,

$$\eta_{CFM}(\mathbf{Y}) = \begin{cases} \sigma_\delta(Y_{(1)} - Y_{(2)}) & \text{if the classification is correct} \\ \sigma_\delta(Y_{(*)} - Y_{(1)}) & \text{otherwise} \end{cases} \tag{11}$$

where $\sigma_\delta(\cdot)$ is the synthetic CFM function with confidence parameter δ [1], $Y_{(1)}$ is the largest network output, $Y_{(2)}$ is the second largest output, and $Y_{(*)}$ denotes the output matching the correct classification. The argument to $\sigma_\delta(\cdot)$ is the discriminant differential, defined as the difference between the output from the correct neuron and the largest other neuron output. It is positive when the classification is correct, negative when incorrect, and generally bounded on $[-1, +1]$ if outputs are limited to [0, 1].

Graphs of $\sigma_\delta(\cdot)$ for various values of δ are available in Fig. 4, which shows the CFM as a continuously differentiable sigmoidal approximation to a counting function. Hence, it has the property that $\sigma_\delta(\cdot) \to 1$ as the discriminant differential approaches δ from below and $\sigma_\delta(\cdot) \to 0$ as the differential becomes increasingly negative. Over the ensemble of training patterns, the average CFM attains its maximum value of unity when all patterns are correctly classified with differentials exceeding the CFM confidence parameter.

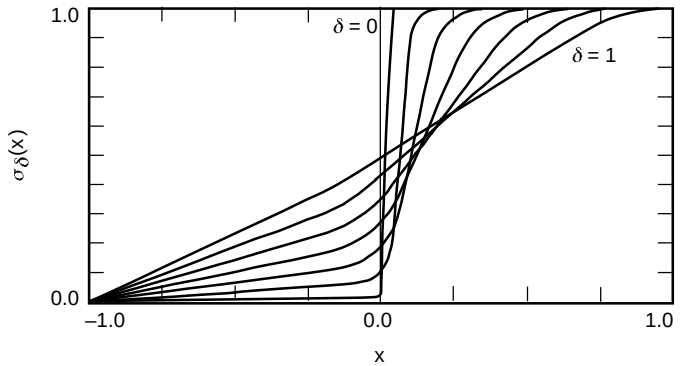

Fig. 4. CFM objective function.

Learning driven by the CFM is differential learning, seeking only to ensure correct classification rather than tackling the more complex task of mimicking a posteriori distribution functions the way MSE-based probabilistic learning does. The less rigorous nature of CFM as a cost/benefit measurement is reflected in Eq. (11), which depends only on two outputs rather than all the components of $\mathbf{Y}$. For further discussion of CFM and the relative merits of differential learning over probabilistic techniques, consult [1].

E. The Back Propagation Algorithm

Back propagation [3,6], or "reverse accumulation," is a well-known procedure for computing the derivatives of η with respect to neuron outputs or weights that are not directly associated with the output layer of a feedforward network. Nearly all developments of the back propagation equations for multilayer networks stress the derivation of an error signal Δ_j for each hidden neuron j, with Δ_j being analogous to the error $Y_i - T_i$ of each output unit. This approach deemphasizes the difference between output and hidden units by synthesizing an equivalent "target value" for each hidden node. Although this unification has some intuitive appeal, it is unnecessarily complicated as well as conceptually inappropriate for pure classification tasks where distance-based metrics have little meaning.

More general back propagation equations that determine the gradient of η over the weight space or node output space for feedforward multilayer networks can be obtained by inspection. The gradient component for a network output unit i can be obtained directly from the objective function $\eta(\mathbf{Y})$:

$$\frac{\partial \eta}{\partial y_i} = \frac{\partial \eta(\mathbf{Y})}{\partial y_i} = \frac{\partial \eta(y_{N-N_O+1}, \cdots, y_N)}{\partial y_i}, \quad i = N - N_O + 1, \cdots, N \tag{12}$$

For hidden- or input-layer nodes, examination of Fig. 3 reveals the general form of the desired derivative:

$$\frac{\partial \eta}{\partial y_i} = \sum_{j:I_{j,i}=1} \frac{\partial \eta}{\partial y_j} \frac{\partial y_j}{\partial x_j} \frac{\partial x_j}{\partial y_i} = \sum_{j:I_{j,i}=1} \frac{\partial \eta}{\partial y_j} \left[F_j'(s_j)\right] w_{j,i}, \quad i = 1, \cdots, N - N_O \tag{13}$$

Thus, the gradient components for all neurons are made available by starting with an explicit computation for the output layer, then propagating these values back through the network toward the input layer and accumulating weighted contributions associated with each forward path. The gradient components with respect to the weights are simply

$$\frac{\partial \eta}{\partial w_{i,j}} = \frac{\partial \eta}{\partial x_i} \frac{\partial x_i}{\partial w_{i,j}} = \frac{\partial \eta}{\partial x_i} y_j \quad \forall i, j : I_{i,j} = 1 \tag{14}$$

F. Differential Back Propagation

Combining CFM-based differential learning with back propagation produces several interesting consequences.

First, the definition of η_{CFM} above in Eq. (11) guarantees that all derivatives of η_{CFM} are zero except those associated with two of the network outputs. Subsequently, only weights contributing to these two top-ranked output units can be modified by the learning rule in Eq. (7). One of these favored outputs always corresponds to the correct class, while the other represents the highest-ranked class other than the correct class. Intuitively, CFM-based back propagation attempts only to increase the output for the correct class while simultaneously reducing the output for the most aggressive competing class without deliberately affecting the discriminant functions for any of the remaining classes. With MSE-driven back propagation, there are no restrictions on which weights may be changed; in general, all network weights

are subject to modification during any given training epoch as the update rule attempts to reduce the error for all neurons. Of course, for the two-class DLA scenario, this distinction is not important.

Derivatives of η_{CFM} also vanish for discriminant differentials that equal or exceed the confidence parameter, δ, associated with $\sigma_\delta(\cdot)$. Accordingly, training samples that produce large enough positive differentials no longer engender weight adaptation. This forms one justification for designating δ as a "confidence"; once the network correctly classifies an exemplar with a differential exceeding δ, it can be said to have learned that pattern with at least confidence δ. The training algorithm no longer needs to allocate portions of network functional complexity toward improving discrimination performance on that example and is instead free to concentrate on more difficult cases. Note, however, that such patterns may still influence learning if the relevant differential is perturbed when weights are modified in response to other training samples.

Also, derivatives of η_{CFM} with respect to a network output y_i depend on another output y_j in addition to y_i. This is in sharp contrast to most probabilistic supervised learning strategies, where each output possesses an error derived from strictly local information. In the two-class problem posed by the downlink analyzer, this results in antisymmetric weight updates—if the initial weights to the output layer are all zero, they will always remain antisymmetric, rendering one output redundant and simplifying the architecture without necessarily penalizing performance. Symmetry breaking, if desired, can be achieved via random initial weights or a few epochs of probabilistic training.

Finally, a subtle aspect of η_{CFM} is found in the fact that the objective function score for a given training example depends only on two outputs. More significantly, which two outputs are involved varies across the training ensemble, i.e., different exemplars effectively have different objective functions. Furthermore, the two specific outputs can change for a given training pattern as the network adapts. This phenomenon is a further reflection of the nature of differential learning.

G. Complexity Reduction

Excessive functional complexity is known to result in overfitting of network weights during training, leading to subsequent poor generalization performance on previously unseen input vectors. This is a penalty for forsaking proper parametric models (specifically those with correct model order) and using an all-encompassing, high-order neural network model in violation of Occam's razor. The problem is exacerbated when available training data are very sparse or otherwise not representative of the true underlying spatial probability distribution of the inputs; in this case, novel feature vector inputs may elicit surprising results if they excite degrees of freedom that were unconstrained during the learning phase. Fortunately, several techniques are available for reducing neural network complexity and ameliorating the effects of a surfeit of free parameters.

Methods of reducing network complexity in the DLA work by suppressing the effects of unimportant parameters. The first technique, weight decay, causes all weights to decay exponentially in magnitude during the training process. Weights to which the classification process is insensitive are permitted to decay to zero, while those that are significant contributors are continuously reinforced by Eq. (7).

A second technique, the optimal brain damage (OBD) algorithm [4], uses the objective function to determine the relative average significance of each weight to the classification of the training ensemble after a locally optimum solution has been reached. Those weights that are deemed relatively unimportant are removed and the network is retrained to reattain the optimum. The pruning procedure may be iteratively applied until performance would degrade beyond an acceptable threshold. Complete details pertaining to the application of the OBD in the context of the DLA are beyond the scope of this article but are available.[1]

[1] D. Watola, "Overview of Downlink Analyzer Neural Network Technology," JPL Interoffice Memorandum 331.1-95-047 (internal document), Jet Propulsion Laboratory, Pasadena, California, November 20, 1995.

V. Fault Diagnosis

The details behind the derivation of the discriminative diagnosis (DD) algorithm and its CFM-engendered differential-learning counterpart, differential discriminative diagnosis (DDD), are sufficiently complex and interesting to deserve a complete treatment elsewhere [2].[2] In this section, enough information is presented to demonstrate how DDD meshes with the rest of the downlink analyzer.

For discriminative diagnosis to be applicable to a neural network, the classifier must be generated by a supervised learning procedure where the model parameters are found by optimizing an empirical cost/benefit function. Models produced for the DLA using back propagation to maximize the CFM statistic clearly meet this condition, which DD and DDD require since they use the same cost function that previously directed learning as the basis for the model analysis at the heart of discriminative fault diagnosis. During training, the cost/benefit metric indicated how well the classifier modeled the relationship between the feature vector and the system health state. Afterward, it serves a complementary purpose, reflecting how much the model health state deviates from some desired nominal state. Of course, this interpretation of η is only meaningful when the classifier forms a robust model of the system; analysis of an inaccurate model yields no useful information.

When the principal health metric announces the presence of an anomaly, i.e., $\Omega_{h1} = \Omega_{c1} = \Omega_0 =$ BAD, diagnosis proceeds by first selecting a previous classification with $\Omega_{h0} = \Omega_{c0} = \Omega_1 =$ GOOD as a reference point. Suitable references are those health metric/feature vector pairs from the recent past exhibiting relatively high values of the discriminant differential; such cases represent classifications where the model has high confidence in its own performance. DDD next constructs an approximate model of the system by computing the Taylor series expansion of the objective function about the reference state:

$$\eta_0(\mathbf{X}) = \eta_0(\mathbf{X}_0) + (\mathbf{X}-\mathbf{X}_0)^T\left(\nabla_X \eta_0(\mathbf{X}_0)\right) + \frac{1}{2}(\mathbf{X}-\mathbf{X}_0)^T\left(\nabla_X^2 \eta_0(\mathbf{X}_0)\right)(\mathbf{X}-\mathbf{X}_0) + O\left(||\mathbf{X}-\mathbf{X}_0||^3\right) \quad (15)$$

where ∇_X denotes the $N_I \times 1$ gradient vector of first derivatives with respect to the $\mathbf{X}$ components, ∇_X^2 refers to the corresponding $N_I \times N_I$ Hessian matrix of second derivatives, and the subscript in η_0 reinforces the notion that all derivatives are evaluated at $\mathbf{X} = \mathbf{X}_0$. To reduce computational complexity, the algorithm truncates the model at the second order by omitting the final term in Eq. (15). Then, by viewing $\mathbf{X}_1$ as resulting from an input perturbation $\Delta\mathbf{X} = \mathbf{X}_1 - \mathbf{X}_0$, a measure of how desirable or undesirable $\mathbf{X}_1$ is with respect to $\mathbf{X}_0$ (in terms of producing the desired health state $\Omega_{h0} =$ GOOD) is obtained from the resulting perturbation in the objective function:

$$\begin{aligned}\Delta\eta_0(\Delta\mathbf{X} &= \eta_0(\mathbf{X}_1) - \eta_0(\mathbf{X}_0) \\ &\approx \left(\Delta\mathbf{X}^T\right)\nabla_X\eta_0(\mathbf{X}_0) + \frac{1}{2}\left(\Delta\mathbf{X}^T\right)\nabla_X^2\eta_0(\mathbf{X}_0)\Delta\mathbf{X} \end{aligned} \quad (16)$$

Here, the loss of equality results from truncation of the third- and higher-order terms in the expression for $\Delta\eta_0$. Contributions of individual input perturbations are isolated from $\Delta\eta_0$ by decomposing Eq. (16) as

$$\Delta\eta_0(\Delta\mathbf{X}) \approx \sum_{i=1}^{N_I}\left(\Delta X_i \frac{\partial\eta_0(\mathbf{X}_0)}{\partial X_i} + \frac{1}{2}\Delta X_i \sum_{j=1}^{N_I}\Delta X_j \frac{\partial^2\eta_0(\mathbf{X}_0)}{\partial X_i \partial X_j}\right) \quad (17)$$

[2] D. Watola and J. B. Hampshire II, "Diagnosing and Correcting DSN Downlink Anomalies With a Robust Classifier," to appear in a future issue of *The Telecommunications and Data Acquisition Progress Report*, Jet Propulsion Laboratory, Pasadena, California.

where the ith summand is defined as the saliency Ψ_i of the ith input. By finding and ranking the negative saliencies produced using this procedure, DDD provides a prioritized checklist of sources of performance degradation to human operators or troubleshooters.

Again, it is critical that the classifier be robust in the neighborhood surrounding $(\mathbf{X}_0, \mathbf{Y}_0, \Omega_{h0})$ for the second-order approximation to be valid. Fortunately, the downlink analyzer provides two powerful means for recognizing and rejecting spurious results. First, meaningless analyses can be avoided in regions where the model is incorrect since the true health state (or at least a robust estimate of the most likely actual health state) is always available in Ω_h for comparison to the model health state. Whenever these two disagree, the model is assumed faulty since the accuracy of the empirical health state is posited. Also, each classification results in an observable discriminant differential; inputs eliciting a relatively low differential (below some specified threshold) can be excluded from use as either $\mathbf{X}_0$ or $\mathbf{X}_1$ since the model does not strongly associate the feature vector with its corresponding output class. Hence, in regions where the classifier fails to discriminate adequately, spurious diagnostic results are avoided or ignored. Inputs mapping to incorrect outputs can be recorded for further analysis or for future off-line training of the network; thus, updating the model is possible when new failure modes manifest.

VI. An Example: KaBLE Link Diagnosis

This section demonstrates the end-to-end performance of the downlink analyzer using monitor data taken from an actual deep-space downlink. The data originate in a series of recordings made in 1993 as part of the Mars Observer Ka-Band Link Experiment (KaBLE) [5] and contain simultaneous measurements from two different downlinks (the dual X-band (8.45-GHz) and Ka-band (32-GHz) carrier-only links employed in the experiment), often with anomalous conditions appearing only in one downlink. Quantities present in the recordings include carrier power-to-noise spectral density ratio (P_c/N_0) estimates made by coherent tone trackers, system noise temperature (SNT) measurements from the total-power radiometer (TPR), pointing-error estimates from the antenna controller, and weather station data. Specifically, the data available are X-band P_c/N_0, X-band SNT, Ka-band P_c/N_0, Ka-band SNT, azimuth, azimuth error, elevation, elevation error, air temperature, water vapor density, wind direction, and wind speed.

In this example, a very simple fault in the DOY-17 X-band downlink is explored. Figure 5 shows a portion of the available time series data for that date. X-band P_c/N_0 is recognizable as the principal health metric by the presence of a small color-coded graph immediately above the time series plot. This extra graphic gives the empirical health state classification for the system based on the health metric; from top to bottom, the four possible health states depicted are VERY GOOD, GOOD, BAD, and VERY BAD. For this data set, the threshold between GOOD and BAD is located (somewhat arbitrarily) at 34 dB-Hz, and there are no VERY GOOD classifications. To the right of each time series is a box plot showing all outputs of the moving box-plot filter for the time instant selected by the vertical time series cursor. Also visible on each box plot is a small square denoting the value of the time series at the cursor location, which coincides with one end of the box-plot observation interval. The box plot for the principal health metric is distinguished by a color-coded background marking the health class boundaries.

At least two qualitatively different types of failures are distinguishable from the principal health metric display in Fig. 5. Between 0030 and 0100, there are many instances where P_c/N_0 makes a brief excursion slightly below acceptable limits, suggesting some intermittent or transient source of performance degradation. From 0245 and 0340, the health state is uniformly VERY BAD with P_c/N_0 measurements consistently low enough to indicate a complete loss of signal. These two distinct behaviors characterize nearly all faults observed in the KaBLE data captured during successful tracking of Mars Observer; only the second type of anomaly is analyzed here because the origin of the fault is known with certainty for DOY 17.

According to the operator's log for DSS 13, the dichroic plate was removed at 0245, resulting in loss of the X-band signal. A dramatic drop in detected P_c/N_0 immediately followed as the tone tracker lost

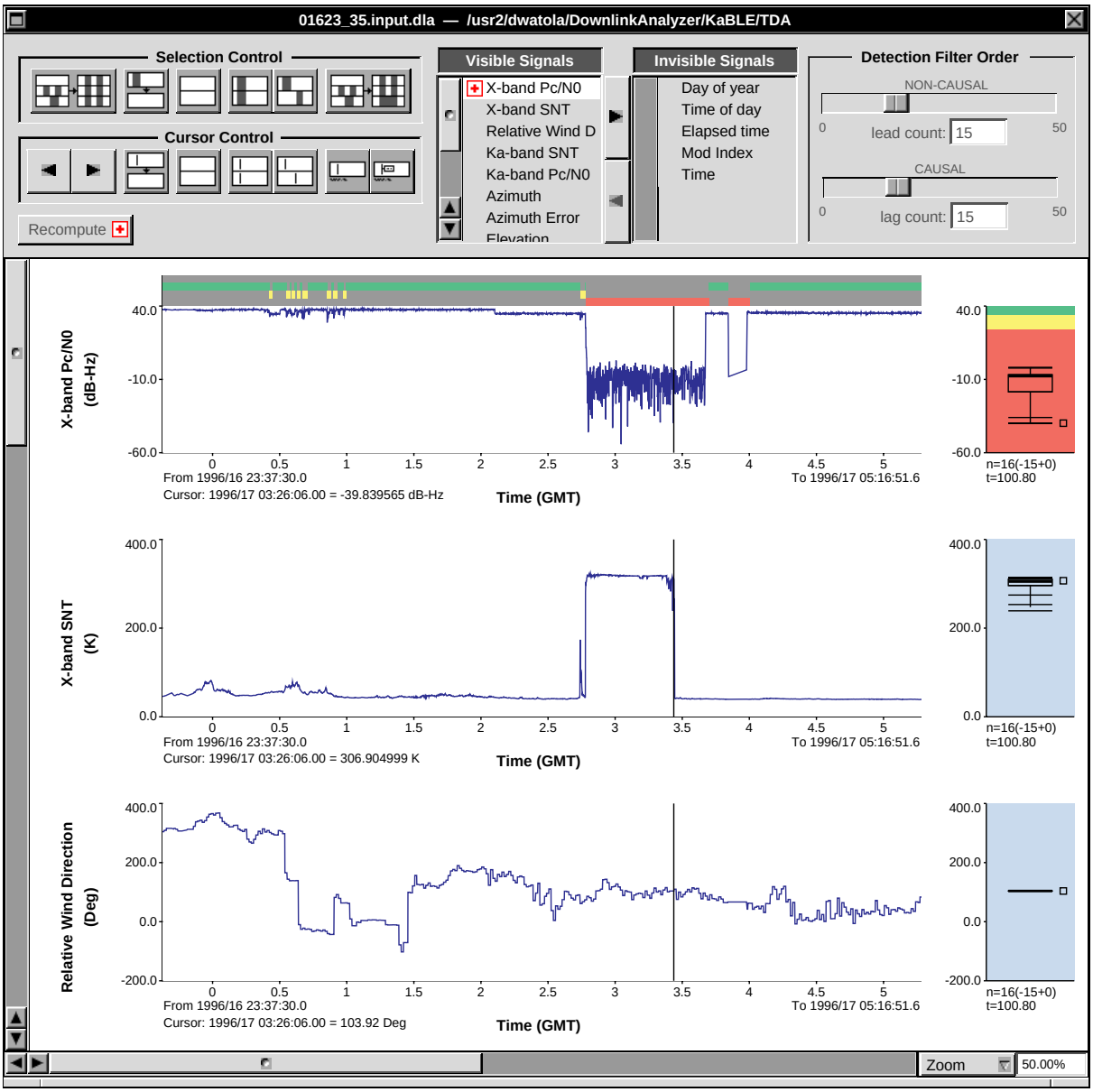

Fig. 5. DLA graphical user interface display for DOYs 16–17 KaBLE data.

its lock on the carrier beacon. The change in principal health metric is visible in Fig. 5, which also shows the X-band SNT registering at approximately room temperature for as long as the dichroic plate was absent. As viewed by the DLA, a transition from the GOOD to BAD and VERY BAD health states occurred almost immediately, because the mapping from h to Ω_h is a simple thresholding operation. Thus, the DLA performs the simplest part of its task, fault detection, with very little computational effort or latency. Unfortunately, the neural network system model does not have access to P_c/N_0 and therefore must perceive the presence of a fault based on the excessive X-band SNT measurement, without explicit knowledge that "high SNT is undesirable." Similarly, the DDD algorithm must recognize an elevated SNT as the proximal cause of the problem solely on the basis of its analysis of the neural network.

One possible neural network for use with the KaBLE data is shown in Fig. 6. This architecture, with five hidden neurons, was experimentally determined to be the minimum capable of modeling some of the

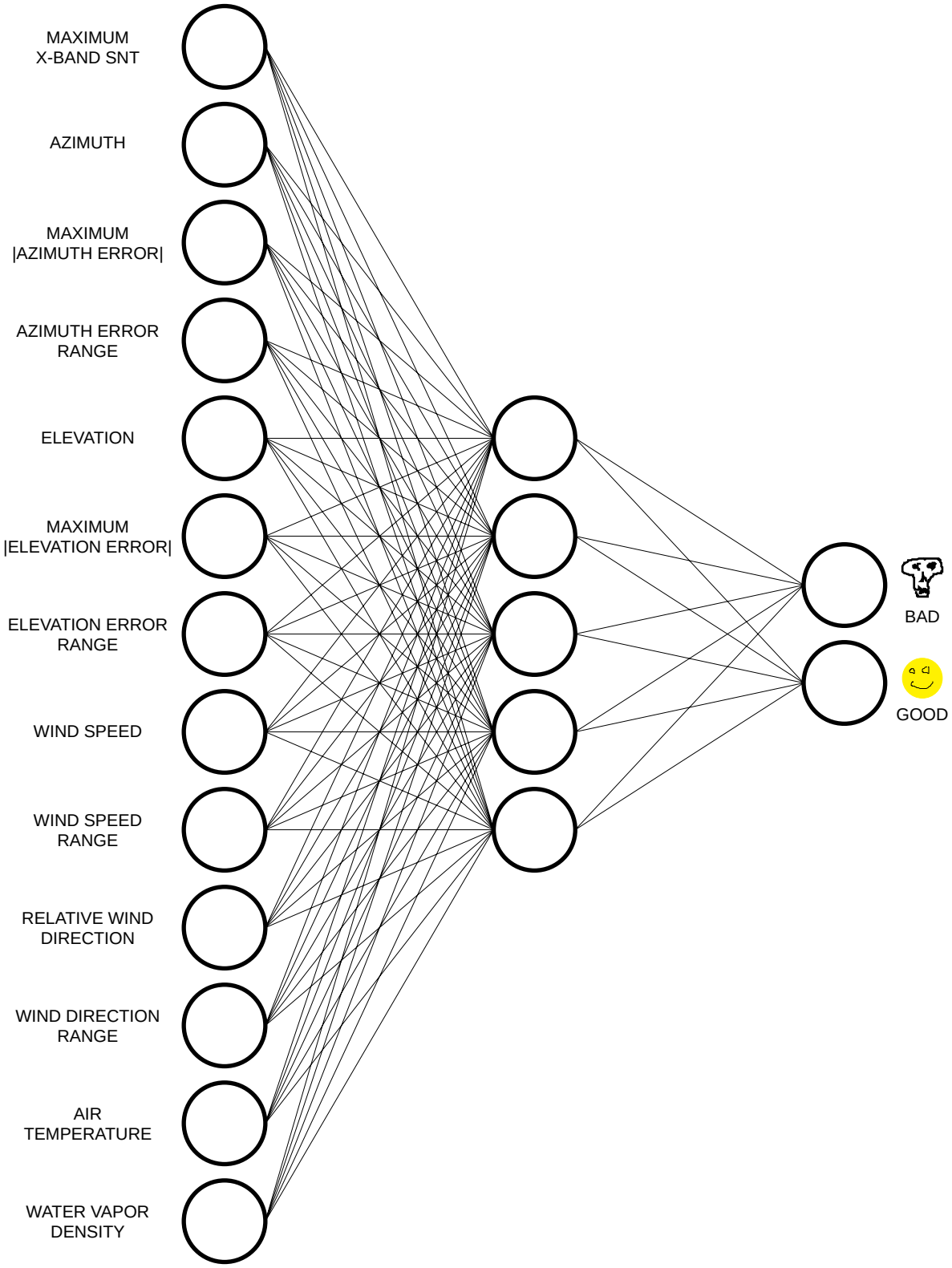

Fig. 6. DLA neural network model for KaBLE.

more subtle wind-related phenomena present in the KaBLE data. The thirteen inputs (features) were also selected after some experimentation. Note that most of the features are not just items from the original time series. Some are transformations of an underlying KaBLE time series, e.g., "maximum |azimuth error|" is the largest absolute value of "azimuth error" found within the observation window. Others are the result of more sophisticated manipulations of the input data; for example, "relative wind direction" is a function of both the "wind direction" input and absolute "azimuth," while "wind direction range" is a short-term statistic computed as the difference between the extreme values of samples of "wind direction" over a 150-s observation window. Although none of these features other than "maximum X-band SNT" are important to the example under consideration, they demonstrate the freedom granted by the DLA software in choosing feature vectors to reduce the burden on the network in its attempt to build a system model.

Figure 7 shows the complete P_c/N_0 history for the DOY-17 track. Highlighted regions mark the 620 data points used to generate a training ensemble for the neural network in Fig. 6. This training set includes samples from both types of faults as well as a few areas where conditions appear nominal. Ordinarily, such a sparse training set would be much too small to engender a useful model of the downlink. However, it is sufficient for this limited demonstration since the resulting neural network is only applied to the remaining DOY-17 data, which do not possess much variety beyond that embodied in the selected samples. The training set deliberately excludes the region of interest near 0245 in the transition region where P_c/N_0 first slips into the BAD class. Instead, it only contains examples of VERY BAD health metric values.

After training this network with the feature vector/health state pairs comprising the DOY-17 training set, the result is an acceptable model for the system (for DOY 17 only) with an 11.1-percent false alarm rate and a 14.4-percent missed detection rate over the 7748 points in the DOY-17 data. Although these performance statistics do not seem very impressive, it should be noted that they are not good indicators of DLA performance in a practical sense. For example, most of the missed faults and many of the false alarms occur while the system is transitioning between states; this is inevitable since the model cannot capture all of the subtleties of the time-domain behavior of the raw data. Of the eight short faults between 0030 and 0100, only two completely escape detection by the neural network and are thus not subject to

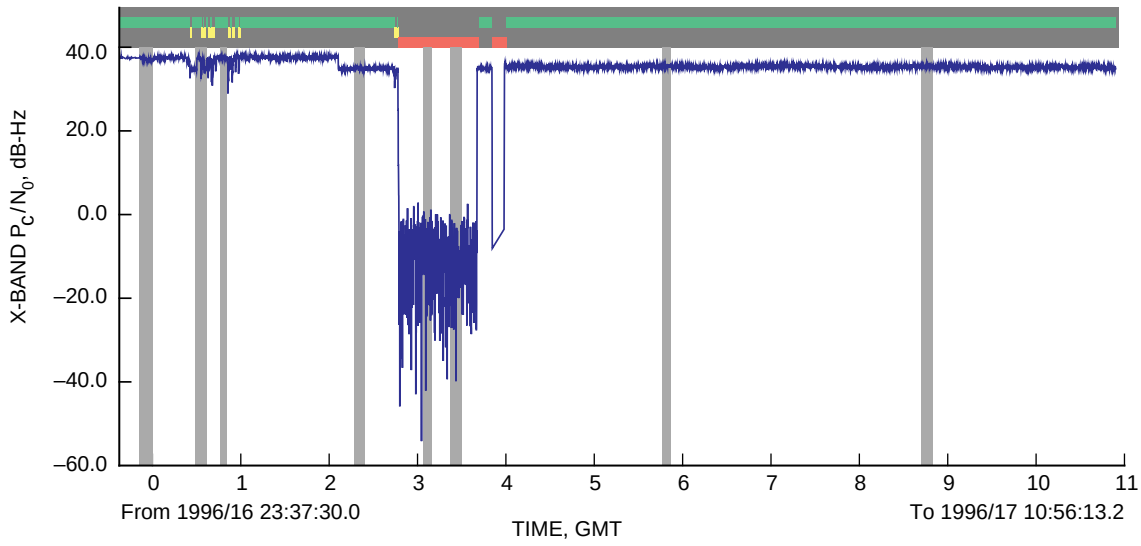

Fig. 7. Complete health metric history for DOYs 16–17 showing training set selection.

DLA diagnosis. The remaining six anomalies can still be analyzed using DDD even though the model produces occasional incorrect classifications in their vicinity. Furthermore, those faults not found by the model are still flagged by the DLA, so the problem does not escape detection entirely.

Also, nearly all of the false alarms occur between 0400 and 0530, the windiest 1.5 hours of the pass. Because the only gusty winds in the training set are strongly correlated with anomalies, the model incorrectly generalizes in this region and frequently produces BAD classifications even though the downlink is healthy. Thus, the false alarm performance would be greatly improved by the addition of a few minutes of new training data from this windy region.

To apply the diagnostic procedure to the fault event near time 0245, it is necessary to choose two classifications: one with a BAD health state to be investigated and another having the desired GOOD health state for use as a reference. Figure 8 shows the two cases selected for this example. Here, the feature vector 2090 produces the very first BAD output associated with the removal of the dichroic plate. Classification 2088 is a suitable reference point since it is a recent previous input that gave a correct GOOD classification with a relatively high discriminant differential, meaning that the output for the GOOD class responds much more strongly to the input features than that of the BAD class. On the other hand, 2089 is not a viable reference since it lies in the transition region where the model is not robust.

Differential discriminative diagnosis on these two selections yields the saliencies shown in Fig. 9. Note that the change in "maximum X-band SNT" is flagged as the most significant contributor to the performance degradation, and that the saliencies for most of the other input perturbations are not within an order of magnitude of the saliency for SNT. The selection of "wind speed range" as a close competitor to SNT is due to a coincidental change in wind characteristics; as described above, the training set is too sparse for the neural network to discover that not all wind-related events result in faults. A stronger cue that the diagnosis may be questionable is the presence of a positive saliency (for the "wind speed" feature) of the same order of magnitude as the largest negative saliency—a sign that the second-order

Fig. 8. DLA classifications at anomaly onset.

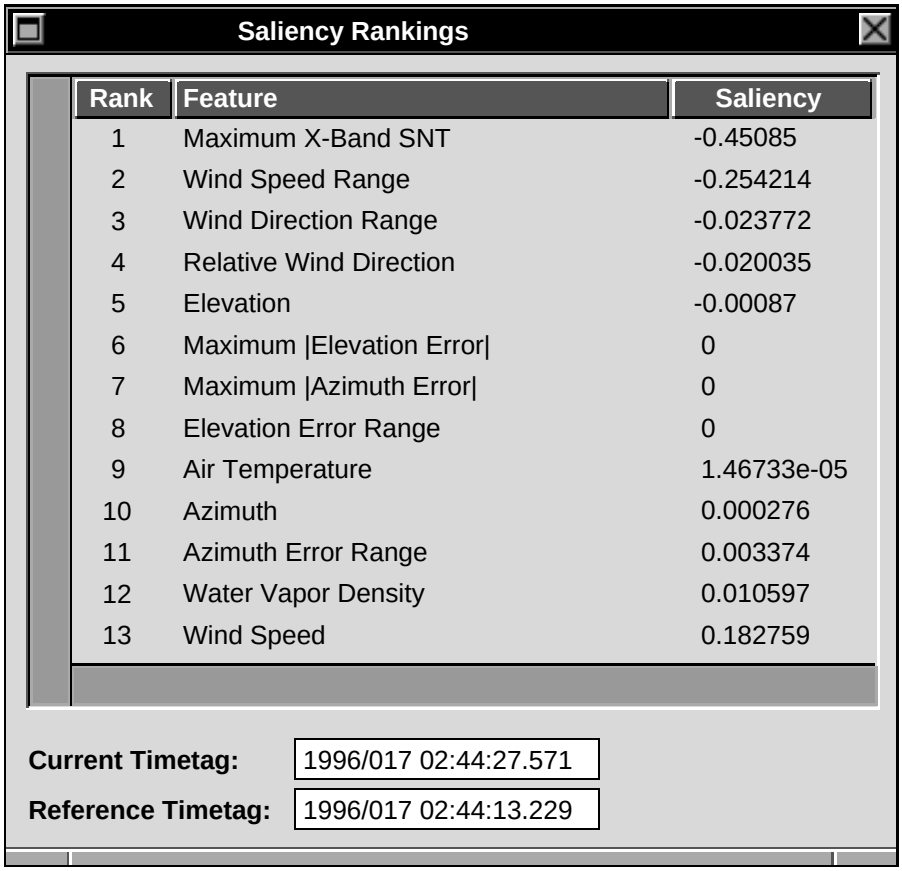

Fig. 9. Saliencies computed using differential discriminative diagnosis for Fig. 8.

approximation is not very good for the selected points. In spite of these problems, DDD successfully isolates the large change in the X-band SNT as the source of the problem. Selecting a faulty point with a higher discriminant differential (e.g., >0.6 instead of the 0.13 in this example) for analysis nearly always gives unambiguously correct results.

This example demonstrates recognition and analysis of a fault with a very obvious cause; even without the station log, it is clear that SNT is the culprit. Armed with the DDD procedure, the downlink analyzer is capable of detecting and diagnosing much more subtle downlink failures. KaBLE data contain numerous events where P_c/N_0 drops below acceptable limits without any clear reason. An early version of the downlink analyzer traced some of these lost signal events to rapid wind shifts at low wind velocity. JPL engineers reviewing 1964 wind-tunnel test results for DSS 13 confirmed that rapidly shifting low-velocity winds can induce underdamped oscillations, placing mechanical loads on the pointing apparatus large enough to cause downlink failure. This represents the first direct evidence that the downlink analyzer can discover heretofore unknown downlink failure modes and, in effect, teach human design engineers, scientists, and operators what it has learned. Full details of this diagnosis will be available in an upcoming article that examines discriminative diagnosis and its application in much more detail.[3]

[3] Ibid.

VII. Summary

The downlink analyzer is a versatile system for performing fault detection and diagnosis in DSN downlink systems or, indeed, any system that fits the principal health metric/causal factor paradigm. One reason it is such a flexible tool is its ability to deal effectively with multiple unsynchronized data sources, including the facility to perform joint computations on samples that arrive at arbitrary intervals. This ability is largely due to the use of the moving box-plot filter in both fault detection and feature vector computation. Other technical innovations powering the downlink analyzer are the use of a nonparametric statistical pattern classifier for learning DSN causal relationships, the application of differential learning to achieve minimum probability of error discrimination with minimum complexity, and the discriminative diagnosis procedure for locating and ranking likely causes of anomalies based on mathematical analysis of the empirically derived neural network model. The current DLA implementation is a significant step toward meeting the primary goals of providing automated detection and diagnosis of faults, learning novel failure modes without human intervention or guidance, and providing useful tools for fault-related data analysis.

References

[1] J. B. Hampshire II, *A Differential Theory of Learning for Efficient Statistical Pattern Recognition*, Ph.D. Thesis, Carnegie–Mellon University, Department of Electrical and Computer Engineering, Pittsburgh, Pennsylvania, September 1993.

[2] J. B. Hampshire II and D. Watola, "Diagnosing and Correcting System Anomalies With a Robust Classifier," *IEEE Proceedings of the 1996 International Conference on Acoustics, Speech, and Signal Processing*, Atlanta, Georgia, pp. 3507–3509, May 1996.

[3] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Redwood City, California: Addison-Wesley, 1991.

[4] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal Brain Damage," *Advances in Neural Information Processing Systems 2*, edited by D. S. Touretzky, San Mateo, California: Morgan Kaufmann, pp. 598–605, 1990.

[5] T. A. Rebold, A. Kwok, G. E. Wood, and S. Butman, "The Mars Observer Ka-Band Link Experiment," *The Telecommunications and Data Acquisition Progress Report 42-117, January–March 1994*, Jet Propulsion Laboratory, Pasadena, California, pp. 250–282, May 15, 1994.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing*, vol. 1, Cambridge, Massachusetts: MIT Press, pp. 318–362, 1986.

[7] J. W. Tukey, *Exploratory Data Analysis*, Reading, Massachusetts: Addison Wesley, 1977.