Real-Time Antenna Fault Diagnosis Experiments at DSS 13

J. Mellstrom and C. Pierson
Ground Antennas and Facilities Engineering Section

P. Smyth
Communications Systems Research Section

This article describes experimental results obtained when a previously described fault diagnosis system was run on-line in real time at the 34-m beam waveguide antenna at DSS 13. Experimental conditions and the quality of results are described. A neural network model and a maximum-likelihood Gaussian classifier are compared with and without a Markov component to model temporal context. At the rate of a state update every 6.4 seconds, over a period of roughly 1 hour, the neural–Markov system had zero errors (incorrect state estimates) while monitoring both faulty and normal operations. The overall results indicate that the neural–Markov combination is the most accurate model and has significant practical potential.

I. Introduction

In previous articles the problem of fault diagnosis of antenna pointing systems has been discussed in great detail [1,2]. Briefly, the problem is that of identifying whether an antenna pointing system is operating normally and, if not, which particular fault has occurred. The information available to the classifier system consists of time-series data from various sensor points in the antenna's pointing servomechanism. Identifying a fault condition is rendered nontrivial by the fact that feedback, redundancy, nonlinearities, and a considerable amount of noise from external disturbances are all present in the servo-control loop. This results in a masking of the underlying cause of a particular problem.

In [2] a hybrid signal-processing/neural-network architecture was applied to the problem. Specifically, various characteristic features were extracted from 4-second blocks of the multichannel data. The most useful features for discriminatory purposes were found to be autoregressive coefficients and standard deviations (estimated for certain channels). In [2] it was further shown that classification accuracies of about 90 percent (on test data sets independent of the training data) were achievable using a neural network model, where there were four classes to be predicted: normal, tachometer failure, noisy tachometer, and compensation loss. By taking into account time correlations in the data, accuracies of about 98 percent were attainable.

The results in [2] were obtained using off-line postprocessing of data collected from the elevation axis drives of the 34-m beam waveguide (BWG) antenna at DSS 13. Specifically, the time series sensor data were recorded directly in digital format by the LabView data acquisition

software running on a Macintosh II computer. At a sampling rate of 50 Hz with 12 channels of interest, this resulted in large volumes of data being recorded; hence, for practical reasons, only 5 minutes worth of data were recorded for each class on any given day of data acquisition, since even a 5-minute batch of such data typically requires about 0.4 Mbyte of storage memory. As described in [2], the 5-minute segments of data were then downloaded to a Sun workstation computer to be used for training and testing various fault diagnosis algorithms. This *off-line* analysis led to the development of specific pattern recognition algorithms described in [2].

In this article, an *on-line* experiment is described where a particular classification model was implemented as part of the data acquisition software, allowing direct on-line estimation and classification at the antenna itself. The primary goal of this experiment was to determine if the pattern recognition system described in [2] could be implemented on-line at low cost and provide reliable estimates of normal and fault conditions in real time. The article describes the experiment setup, the results obtained, and concludes that the system has significant practical potential for antenna monitoring.

II. Implementation of the On-Line Fault Diagnosis Software

In [2] a fault diagnosis system was described which estimates various features from multichannel time-series data over consecutive windows, calculates a posterior probability estimate using these features, and then updates its estimate of the state of the system as a function of the current class estimates based on this window and the previous estimates. In [2] a number of different models were evaluated, having roughly comparable performance in terms of classification accuracy. For this experiment the authors chose to implement one particular model using the LabView software package. The LabView software package is intended for real-time data acquisition and analysis. Its primary advantage is that the system can be easily programmed and modified using an intuitive graphical user interface. Basic data acquisition capabilities, filtering, and signal analysis can be configured directly by editing a block diagram display of the signal flow in the system. For the purposes of this experiment the primary functions to be implemented included analog to digital conversion and buffering of the multichannel sensor data, estimation of the classification features on a block-by-block basis, implementation of the classification equations which estimate class probabilities as a function of the features, and a memory-based algorithm which updates a context-based estimate of the system state given each block.

A. Data Acquisition and Timing

The LabView software was set up to acquire k consecutive samples on each channel of interest where the data are sampled at a rate of f_s. For the experiments described in this article $f_s = 54$ Hz, which is sufficient given that all signals of direct interest are below 10 Hz. A block size of $k = 200$ was chosen providing a reasonable tradeoff between being large enough to get reliable parameter estimates but not having too large of a delay between classification decisions. Hence although the time series data sampling rate is 54 Hz, the classifier is operating at a much slower rate since a decision every few seconds is deemed quite adequate for this type of application. For simplicity of implementation the data were not pipelined in the LabView software, which meant that data were sampled and collected for the period of τ_1 seconds (where $\tau_1 = k/f_s$) and then processed (parameter estimation, classification, and state update) for another τ_2 seconds. After this total interval of $\tau_t = \tau_1 + \tau_2$ seconds, the next block of sample data was acquired. Hence, the state estimates were updated every τ_t seconds. In the actual experiments to be described later, τ_2 was on the order of 2.7 seconds leading to a state update every 6.4 seconds. The "gaps" in the data of 2.7 seconds (between the 3.7-second blocks) were deemed inconsequential to the overall operation of the system, since no portion of the model relies on an assumption that the window blocks are contiguous in time. However, it should be noted that if this system were to be implemented in an operational mode, a pipelined scheme with no data gaps would be a more elegant solution.

B. Feature Estimation

The particular classification model chosen for the experiment uses 12 features. The first eight features are autoregressive-exogenous (ARX) model coefficients estimated from a model which has the antenna servo-controller elevation rate command as the exogenous (forcing) model input and the motor current of one of the elevation drive motors as the model output (see [2] for more details). These coefficients were estimated from the 200-sample blocks using a standard least-squares estimation algorithm. The other four features used as input to the classifier are: estimated standard deviation of both elevation drive motor currents, estimated standard deviation of the average tachometer sensor, and the estimated standard deviation of the difference of the two tachometer sensors.

C. Classification Using a Neural Network Model

The particular classification model implemented was a multilayer neural network model: a feed-forward network

with a single hidden layer of nonlinear "hidden" units. The model had 12 inputs (one for each of the features) plus a constant bias input set to 1. The particular model used in the experiments had 20 hidden units. The output layer consisted of four units, one for each class: normal, tachometer failure, tachometer noise, and compensation loss. An output *activation* vector $\mathbf{y}(t)$ at time t is calculated as a function of the input feature vector $\mathbf{x}(t)$ as described in detail in Appendix A of [2]. In particular this model can be implemented via two matrix multiplications and a componentwise nonlinear vector transformation, i.e.,

$$\mathbf{y}(t) = \gamma\Big(\mathbf{W_2} \cdot \gamma(\mathbf{W_1} \cdot \mathbf{x}(t))\Big)$$

where $\mathbf{W_1}$ and $\mathbf{W_2}$ are the weight matrices between the input and hidden layers and hidden to output layers, and $\gamma(\mathbf{u}) = (\gamma(u_1), \cdots, \gamma(u_n))$ where $\gamma(x) = 1/(1+e^{-x})$. The ith component of the output vector $\mathbf{y}(t)$ is interpreted as a rough estimate of the posterior probability of class i given the input feature vector $\mathbf{x}(t)$.

D. State Estimation With a Hidden Markov Model

The class probability estimates as produced by the classifier at each time t do not take into account the fact that faults are typically correlated over time. In [2] a heuristic scheme was described to reflect this prior information, where the current estimate of the system state was a function of both the present probability estimates and past probability estimates over a specified number m of past windows. In Appendix A a formal model of this time dependence is introduced using a Hidden Markov Model (HMM). The HMM approach has been successfully used in applications such as speech recognition to model temporal context. The n states of the system (in this case the normal and fault classes, $n = 4$) comprise the Markov model. For this application components of the Markov transition matrix $\mathbf{A}$ (of dimension $n \times n$) are estimated *subjectively* rather than estimated from the data, since there is no reliable database of fault-transition information available at the component level. The *hidden* component of the model arises from the fact that one cannot observe the states directly, but only indirectly via a stochastic mapping from states to symptoms (the features). In speech recognition it is often desired to calculate the most likely sequence of states (perhaps phonemes) given some acoustic evidence. For a fault diagnosis application such as this it is sufficient to calculate at time t the probability that the system is in any particular state at that time. The state vector is denoted by $\mathbf{s}(t)$. As described in Appendix A the probability estimate of state i at time t can be calculated recursively via

$$\hat{\mathbf{u}}(t) = \mathbf{A} \cdot p(s(t-1))$$

and

$$p(s_i(t)) = \frac{\hat{u}_i(t) y_i(t)}{\sum_{j=1}^{4} \hat{u}_j(t) y_j(t)}$$

where the estimates are initialized by a prior probability vector $p(\mathbf{s}(0))$, the $s_i(t)$ are the components of $\mathbf{s}(t)$, $1 \leq i \leq n$, and the $y_i(t)$ are the outputs of the classifier as described in Section II.C.

E. Software Implementation

Implementation of the ARX parameter estimation, classifier equations, and HMM model estimation was straightforward using predefined functions in LabView. All of the above components of the algorithm were implemented, integrated with data acquisition software, and tested in the laboratory in about 3 working days.

As mentioned earlier, the time taken to process a single block of 3.7 seconds worth of data was estimated at 2.7 seconds. Of this time, it was estimated that 2.3 seconds were spent calculating the eight ARX coefficients and the standard deviation terms on the 200 samples, and 0.12 second were spent updating the screen displays; the actual classification equations (the neural network model and Markov equations) took only 0.08 second, and another 0.20 second went to other miscellaneous bookkeeping activities. It is interesting to note that the classification process itself is quite fast taking only 2.9 percent of the processing time, while the feature extraction takes up 85 percent of the time. These numbers are significantly slower than speeds which would be attainable for operational purposes were the system to be implemented using special-purpose signal processing hardware.

III. Experimental Conditions

The experiments were carried out in early November 1991 at the 34-m BWG antenna at DSS 13. Wind conditions were favourable for the duration of the experiments, i.e., no high wind conditions were encountered. Under each desired state (normal and fault classes) the antenna was driven at a typical DSN tracking rate (between 1 and 3 millidegrees per second) along the elevation axis. Normal conditions corresponded to normal antenna operation. Fault conditions were simulated as described previously in [2], i.e., hardware faults were switched into the feedback loop of the elevation servo-control loop in a controlled manner. For safety reasons these faults were introduced when

the antenna was stopped rather than while it was moving (this precluded testing the response of the models in terms of transitions between faults). Each fault condition was then monitored after the antenna had been restarted. Hence, the experimental results in the next section reflect the fact that the data were collected in different batches in this manner. Faults were typically monitored for durations of 10 to 15 minutes and the total duration of data to be reported in the next section amounted to a total of about 1 hour of antenna monitoring. Longer duration monitoring was not practical due to various constraints such as other demands on the antenna and the amount of setup time required to get the monitoring system in place. Due to hardware problems it was not possible to simulate one of the faults described in [2], namely the tachometer noise condition. Hence, the results pertain to normal conditions, the tachometer failure, and compensation loss faults.

IV. Experimental Results

For purposes of comparison, the model implemented at DSS 13 is compared with a simple maximum-likelihood Gaussian classifier whose operation is described in Appendix B. The results with the Gaussian classifier were generated off-line using the recorded feature data. In addition, data were recorded with and without the Markov portion of the model to evaluate the effectiveness of this component. Hence, there are four types of models being compared: Gaussian, Gaussian followed by Markov, neural network, and neural network followed by Markov. Again it is emphasized that the results to be described pertaining to the latter two models were generated on-line in real time.

Table 1 summarizes the overall classification performance of each of the models on five different runs: two for normal conditions, two for the compensation loss fault, and one run for the tachometer failure fault. The units in the table correspond to windows of 3.7 seconds worth of sensor data spaced at 6.4-second intervals. The bottom row of the table tabulates the number of windows per run: hence, for example, 596 windows in total were analyzed, corresponding to about 1 hour and 3.6 minutes total duration. The numbers in the other four rows of the table (one row per model) indicate the number of windows misclassified by each model for each run, except for the final two columns, which give the overall number of misclassifications per model and the percentage misclassified, respectively. Clearly, from the final column, the neural–Markov model (implemented in real time) is the best model in the sense that no windows at all were misclassified. It is significantly better than the Gaussian classifier, which performed particularly poorly under fault conditions. However, under normal conditions it was quite accurate, having only one false alarm during the roughly 30 minutes of time devoted to monitoring normal conditions. The effect of the Markov model is clearly seen to have beneficial effects, in particular reducing the effects of isolated random errors. However, for the compensation loss on day 316, the Markov model actually worsened the already poor Gaussian model results, which is to be expected if the non-Markov component is doing particularly poorly as in this case.

Figures 1 through 5 plot the estimated probability of the true class as a function of time for various models to allow a more detailed interpretation of the results. Note that, given that the true class is labelled i, the estimated probability of class i from the neural network corresponds to the *normalized* output of output unit i of the network at time t, i.e.,

$$\hat{p}_i(t) = \frac{y_i(t)}{\sum_{j=1}^{4} y_j(t)}$$

while the Markov probabilities correspond to the estimates of state i, $p(s_i(t))$, as described earlier. Figure 1 corresponds to normal conditions on day 311 and compares the neural model with and without the Markov processing. Figure 1 demonstrates that the instantaneous probability estimates from the neural model have a large variation over time and are quite noisy. This is essentially due to the variation in the sensor data from one window to the next, since, as might be expected, signals such as motor current are quite noisy. In addition, a large glitch is visible at around 460 seconds. The neural model gives a low probability that the condition is normal for that particular window (in fact a large glitch such as this looks like a tachometer failure problem); however, the Markov model remains relatively unaffected by this single error. Overall, the stability of the Markov model is clearly reflected in this plot and should be advantageous in an operational environment in terms of keeping the false alarm rate to a minimum. Figure 2 is the same as Fig. 1 except that the data were obtained on day 316. Note that in both Figs. 1 and 2, at any particular instant the classifier assigns to the true class a probability of up to only 0.8 or 0.9. In contrast, by modelling the temporal context, the Markov model assigns a much greater degree of certainty to the true class.

Figures 3 and 4 compare the performance of the Gaussian and neural models on detecting the compensation loss fault. In Fig. 3, the variation in the Gaussian estimates

is quite marked. The Gaussian–Markov model combination, after some initial uncertainty for the first 90 seconds, settles down to yield reasonable estimates. However, the overall superiority of the neural–Markov model is evident. In Fig. 4 (the same fault on a different day), the neural and Gaussian models are compared directly. The variation in the Gaussian estimates is even worse. In fact, the resulting Gaussian–Markov estimates are almost random in nature and have been omitted for clarity.

Figure 5 also compares the Gaussian and neural models but on the tachometer failure fault. Once again the variation over time of the Gaussian estimates is unacceptably large, and again, the resulting Gaussian–Markov estimates have not been plotted due to their almost random nature, i.e., the variation over time on the estimates is so great that the Markov model cannot find any significant correlation.

In Fig. 6, the same information as that for Fig. 3 is plotted but in a different manner. Let $\hat{p}$ be the probability estimate of the true class. Then $1 - \hat{p}$ is the effective error in the estimate; Fig. 6 is a logarithmic plot of this error term. Here, the overall dominance of the neural-Markov model is once again evident, as is the large variation in the Gaussian model.

V. Conclusion

The DSS 13 field results discussed in this article provide convincing evidence for the potential of a pattern recognition system to reliably monitor a DSN antenna drive system in real time. In addition, it should be noted that the monitoring system is unobtrusive and relatively easy to install due its use of inexpensive "off-the-shelf" PC-based hardware and software components.

Table 1. Summary of classification results for various models and runs during days 311 and 316 at DSS 13.

Model	Number of blocks misclassified					Overall misclassification statistics	
	Day 311		Day 316				
	Normal	Compensation loss	Normal	Tachometer failure	Compensation loss	Total no. of misclassifications	Total percent of misclassified
Gaussian	1	15	0	35	50	101	16.94
Gaussian+Markov	1	8	0	3	74	86	14.42
Neural	1	0	4	0	0	5	0.84
Neural+Markov	0	0	0	0	0	0	0.00
Total number of windows	128	60	152	126	130	596	

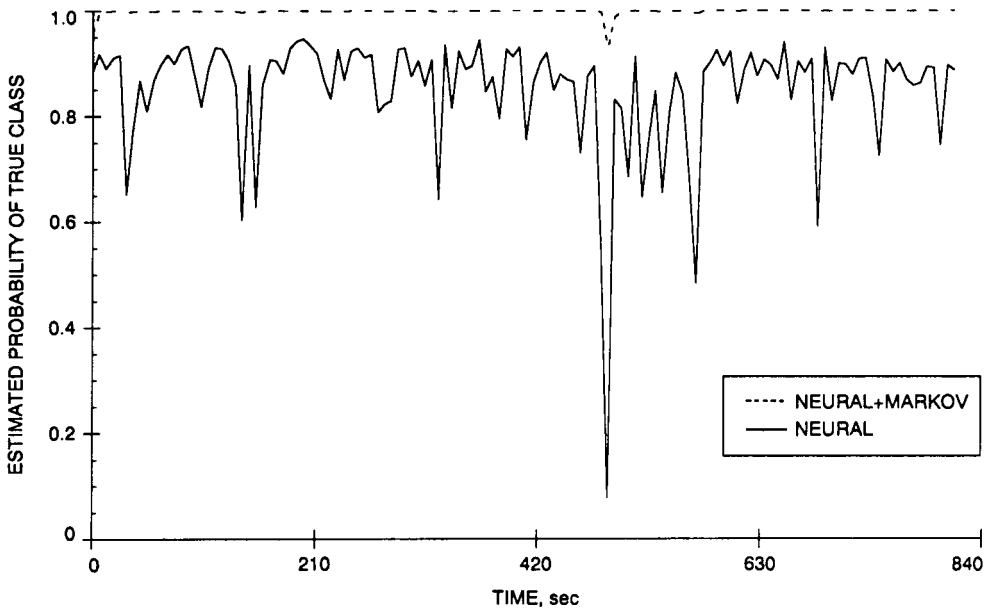

Fig. 1. Comparison of Markov and non-Markov models: estimate of probability of true class (normal conditions) as a function of time for day 311.

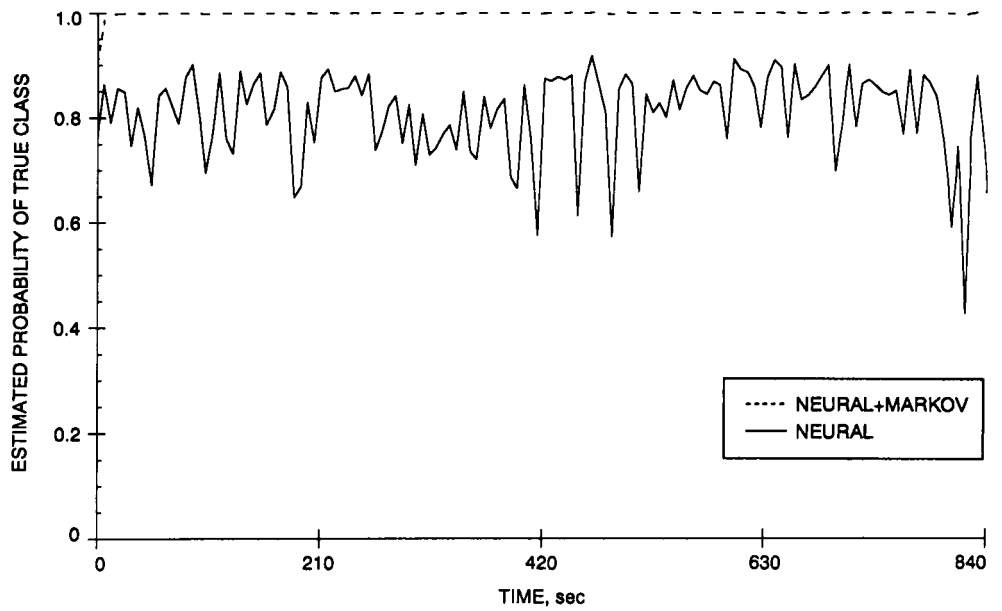

Fig. 2. Comparison of Markov and non-Markov models: estimate of probability of true class (normal conditions) as a function of time for day 316.

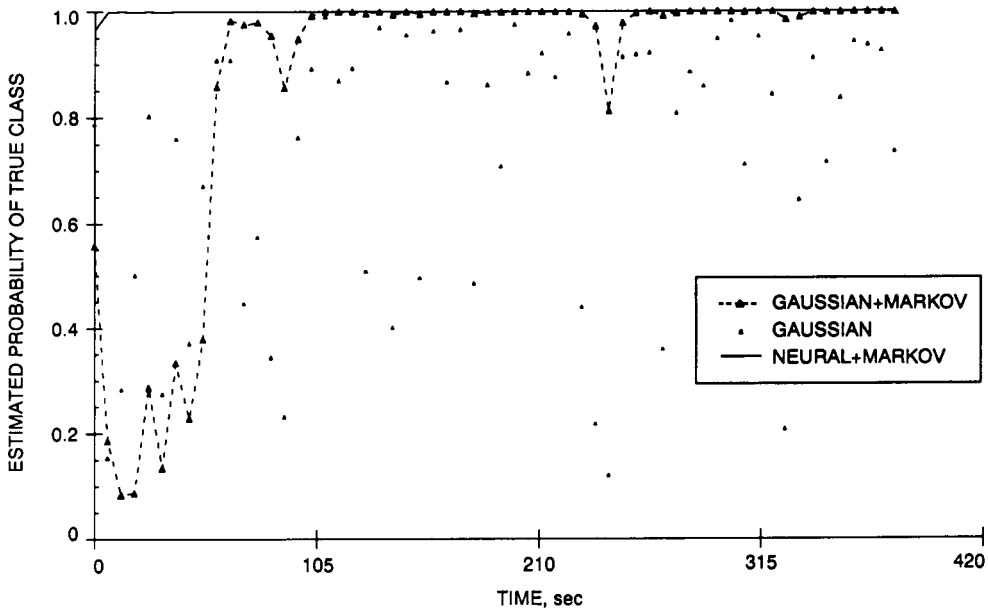

Fig. 3. Comparison of Gaussian–Markov and neural–Markov models: estimate of probability of true class (compensation loss) as a function of time for day 311.

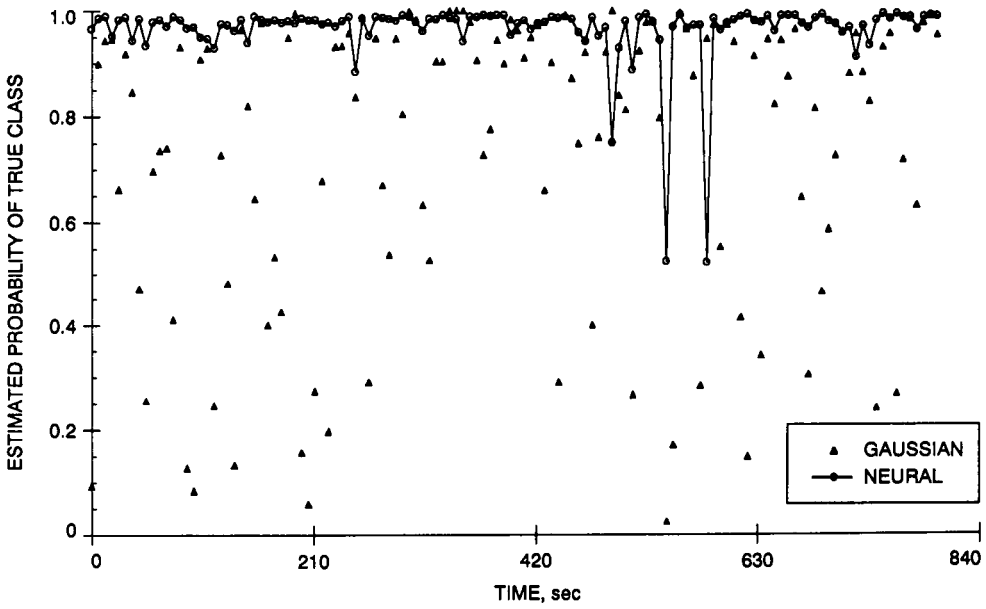

Fig. 4. Comparison of Gaussian and neural models: estimate of probability of true class (compensation loss) as a function of time for day 316.

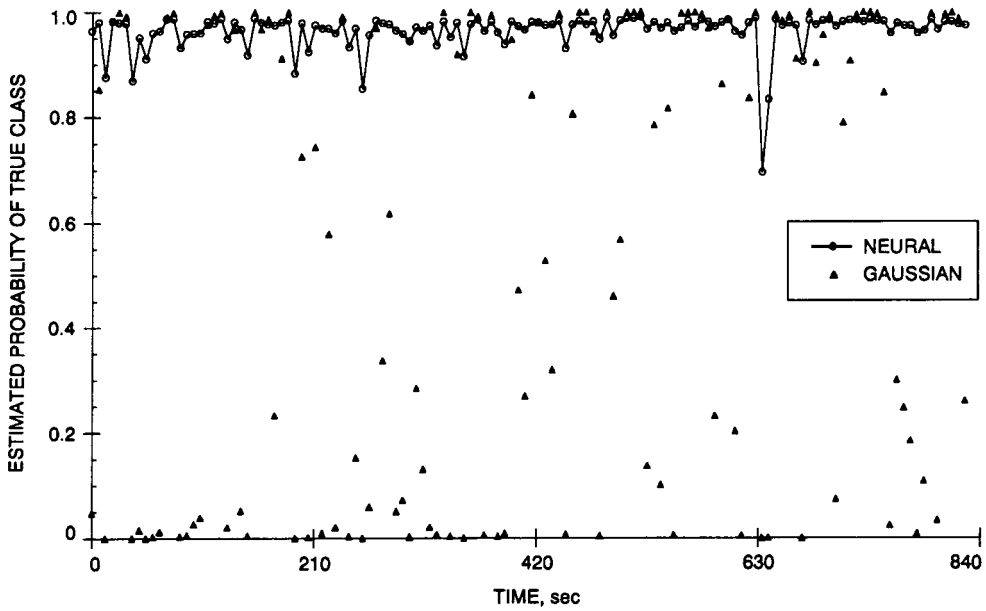

Fig. 5. Comparison of Gaussian and neural models: estimate of probability of true class (tachometer failure) as a function of time for day 316.

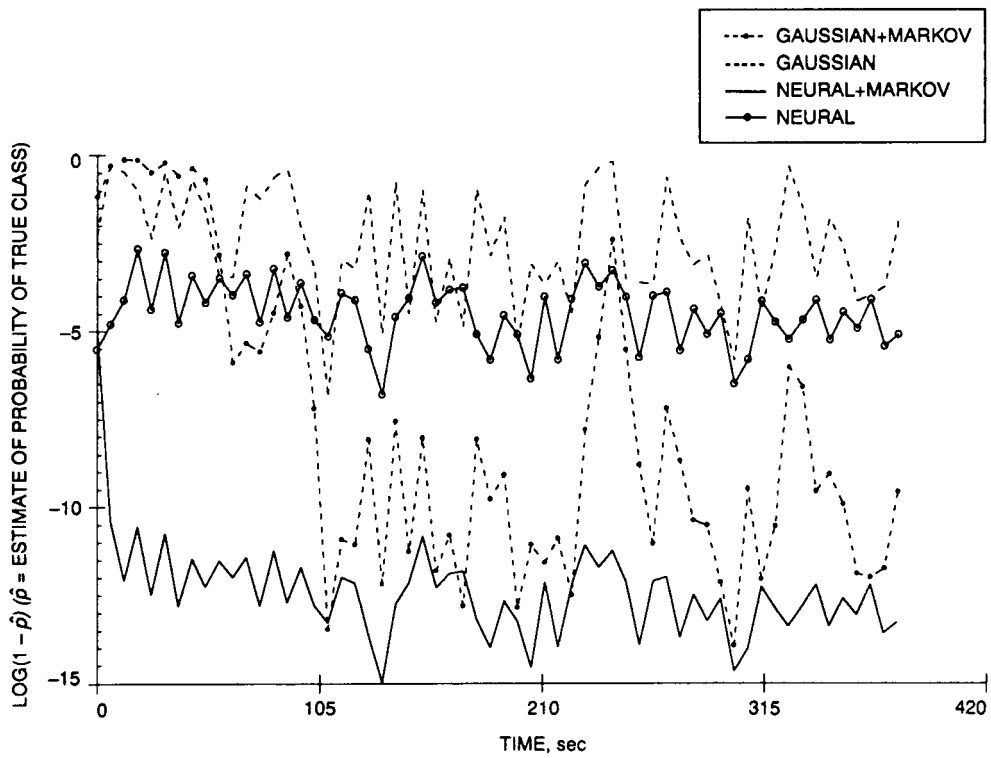

Fig. 6. Comparison of Gaussian–Markov and neural–Markov models: plot of log $(1 - \hat{p})$, where $\hat{p}$ is an estimate of the probability of the true class (compensation loss) as a function of time for day 311.

Appendix A
Hidden Markov Models

Assume that time t is discretized and the system of interest is always in one of n states. Further assume the system of interest is described by a Markov model, i.e., at each time t the probability that the system is in any state i, $p(s_i(t))$, is only a function of the state j that the system was in at time $t-1$. Strictly speaking, this is a *first-order* Markov model. Given that the system was in state j at time $t-1$, the probability $a_{ij} = p(s_i(t)|s_j(t-1))$ is the *state transition probability* and the $n \times n$ matrix $\mathbf{A}$ with components a_{ij} characterizes the Markov model.

In practice, however, the states may not be directly observable, giving rise to the notion of a *hidden* Markov model. Instead, each state produces an observable set of symptoms or features $\mathbf{x}(t)$ at each time t. The features $\mathbf{x}(t)$ are not a deterministic function of the state, rather they are described by a probability density function $p(\mathbf{x}(t)|s_i(t))$. Note that the probability of obtaining a particular $\mathbf{x}(t)$ depends on which state i the system is in at time t. It is this fact that makes it possible to infer the probability that the system is in any state i simply by observing the features $\mathbf{x}(t)$.

This model describes quite closely the fault diagnosis problem, namely the system (the antenna elevation-axis servo-control loop) is in some unknown state (normal and failure modes) at each time instant that the classifier looks at the sensor data. The neural network classifier learns the *instantaneous* relationship between states and features, i.e., it provides an estimate of $p(s_i(t)|\mathbf{x}(t))$. However, the best estimate of the state at time t uses all of the information available up to time t, namely the observed feature vectors $\{\mathbf{x}(t), \cdots, \mathbf{x}(1)\}$. Hence, the best estimate of state i requires the calculation of $p(s_i(t)|\{\mathbf{x}(t), \cdots, \mathbf{x}(1)\})$, which is now derived:

$$p(s_i(t)|\{\mathbf{x}(t), \cdots, \mathbf{x}(1)\}) = \frac{p(s_i(t), \{\mathbf{x}(t), \cdots, \mathbf{x}(1)\})}{p(\{\mathbf{x}(t), \cdots, \mathbf{x}(1)\})}$$

$$= \frac{1}{p(\mathbf{X})}\alpha_i(t)$$

where $\mathbf{X} = \{\mathbf{x}(t), \cdots, \mathbf{x}(1)\}$ and $\alpha_i(t) = p(s_i(t), \{\mathbf{x}(t), \cdots, \mathbf{x}(1)\})$

$$= \frac{1}{p(\mathbf{X})}\sum_{j=1}^{n} p(s_i(t), \mathbf{x}(t), \cdots, \mathbf{x}(1), s_j(t-1))$$

$$= \frac{1}{p(\mathbf{X})}\sum_{j=1}^{n} p(s_i(t), \mathbf{x}(t)|\mathbf{x}(t-1), \cdots, \mathbf{x}(1), s_j(t-1))$$

$$\times\ p(\mathbf{x}(t-1), \cdots, \mathbf{x}(1), s_j(t-1))$$

$$= \frac{1}{p(\mathbf{X})}\sum_{j=1}^{n} p(s_i(t), \mathbf{x}(t)|s_j(t-1))\alpha_j(t-1)$$

by the Markov assumption and the definition of α_j

$$= \frac{1}{p(\mathbf{X})}\sum_{j=1}^{n} p(\mathbf{x}(t)|s_i(t))p(s_i(t)|s_j(t-1))\alpha_j(t-1)$$

since $\mathbf{x}(t)$ is assumed independent of past states

$$= \frac{1}{p(\mathbf{X})} \sum_{j=1}^{n} \frac{p(s_i(t)|\mathbf{x}(t))p(\mathbf{x}(t))}{p(s_i(t))} a_{ij}\alpha_j(t-1)$$

by Bayes' rule and the definition of a_{ij}

$$= \frac{p(\mathbf{x}(t))}{p(\mathbf{X})} \sum_{j=1}^{n} \frac{\hat{p}_i(t)}{p(s_i(t))} a_{ij}\alpha_j(t-1)$$

---

where $\hat{p}_i(t)$ is the instantaneous probability estimate of state i given $\mathbf{x}(t)$ (namely, the normalized output of the neural network in this case), and $p(s_i(t))$ is the prior probability of state i. Because (by definition)

$$\frac{1}{p(\mathbf{X})} \sum_{i=1}^{n} \alpha_i(t) = 1$$

$p(\mathbf{x}(t))/p(\mathbf{X})$ can be treated as a constant and factored out. Hence, by virtue of the recursion relation defined on the α's above and given $p(\mathbf{s}(0))$, the probability of each state at time t can be computed recursively from earlier state information. These basic recursion relations are known as the *forward–backward* relations in the literature. For a more extensive discussion on hidden Markov models and their applications, the reader is referred to [3].

For the experiments described in this article, the initial states $p(s_i(0))$ were set equal to the priors, which in turn were taken to be $1/n = 0.25$. The a_{ij} were set to 0.99 for $i = j$ and $0.01/3$ for $i \neq j$.

Appendix B

A Maximum-Likelihood Gaussian Classifier

Consider that there are n classes ω_i, $1 \leq i \leq n$. In turn, the features are described by a d-component feature vector $\mathbf{x}$. For a Gaussian classifier, one assumes that the probability density $p(\mathbf{x}|\omega_i)$ is multivariate normal for each class, i.e.,

$$p(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} e^{(-1/2)(\mathbf{x}-\mu_i)^t \Sigma_i^{-1}(\mathbf{x}-\mu_i)}$$

where μ_i is a d-component mean vector and Σ_i is a $d \times d$ covariance matrix.

By Bayes' rule,

$$p(\omega_i|\mathbf{x}) \propto p(\mathbf{x}|\omega_i)p(\omega_i)$$

where $p(\omega_i)$ is the prior probability of class i. Given a particular feature vector $\mathbf{x}$, one calculates $\gamma_i = p(\mathbf{x}|\omega_i)p(\omega_i)$, from which one gets

$$\hat{p}(\omega_i|\mathbf{x}) = \frac{\gamma_i}{\sum_{j=1}^{m} \gamma_j}$$

which is the posterior estimate of the probability of class i given the feature data. For the results reported in this article, Σ_i was assumed to be diagonal since there were not sufficient data to obtain reliable estimates of all the elements of the full $d \times d$ ($d = 12$) covariance matrix for each class. The diagonal variance terms σ_i and the means μ_i in each dimension were estimated directly from the data using maximum likelihood estimates. The priors $p(\omega_i)$ were chosen to be $1/n = 0.25$ for each class.

References

[1] P. Smyth and J. Mellstrom, "Initial Results on Fault Diagnosis of DSN Antenna Control Assemblies Using Pattern Recognition Techniques," *TDA Progress Report 42-101*, vol. January–March 1990, Jet Propulsion Laboratory, Pasadena, California, pp. 136–151, May 15, 1990.

[2] J. Mellstrom and P. Smyth, "Pattern Recognition Techniques Applied to Performance Monitoring of the DSS 13 34-Meter Antenna Control Assembly," *TDA Progress Report 42-106*, vol. April–June 1991, Jet Propulsion Laboratory, Pasadena, California, pp. 30–51, August 15, 1991.

[3] A. B. Poritz, "Hidden Markov Models: a Guided Tour," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing 1988*, IEEE Press, New York, pp. 7–13, 1988.