Machine Learning Techniques for Fault Isolation and Sensor Placement

James R. Carnes
Advanced Computing Laboratory
Boeing Defense and Space Group
Huntsville, AL 35824
ray@hsvaic.boeing.com

Douglas H. Fisher
Department of Computer Science
Vanderbilt University
Nashville, TN 37235
dfisher@vuse.vanderbilt.edu

November 16, 1992

Abstract

Fault isolation and sensor placement are vital for monitoring and diagnosis. A sensor conveys information about a system's state that guides troubleshooting if problems arise. We are using machine learning methods to uncover behavioral patterns over snapshots of system simulations that will aid fault isolation and sensor placement, with an eye towards minimality, fault coverage, and noise tolerance.

1 Introduction

Accurate and timely fault diagnosis is critical in the life cycle of many physical systems. Seemingly minor faults can, if unremedied, lead to catastrophic faults that disable a system permanently. To identify faults, (human or machine) diagnosticians observe the system's behavior primarily through sensor readings. Sensors should generally be selected to be maximally informative about the state of the system. In the best of all possible worlds, we might expect that sensors should be placed on all measurable quantities of a system; anomalous values on one or more sensors could then readily identify the presence of and help isolate system faults. However, costs are associated with sensors. These costs correspond to actual monetary cost as well as costs due to the physical design constraints of the system such as power, mass, and volume which are at a high premium in systems such as Space Station Freedom. In addition, increased numbers of sensors introduce more information that an operator must attend to; too many sensors can lead to information overload, thus actually contributing to a degradation in (human) diagnostic performance.

In many cases it is neither feasible nor desirable to measure all quantities of a system. Thus, the diagnostician must interact with the system in two other ways: *probing* and *testing*. One can think of probing as sensing a quantity dynamically to determine its value at a particular point in time. In testing we examine component output quantities while systematically varying its inputs.

Probing and testing increase the cost (e.g., time) of diagnosis and may even be impossible on remote systems such as unmanned spacecraft. Moreover, probing and testing are only initiated when there is some indication of a fault. Thus, we would like to judiciously place sensors so that they indicate the existence of faults and focus attention on their plausible causes.

Sensor placement is the task of determining a set of sensors which allows the most accurate determination of the overall state of a monitored system while minimizing costs relating to the number of sensors, power consumption, cost, and weight. Reducing these quantities is particularly important in space platforms due to power and space restrictions. In response, we are using two machine learning methods to identify categories of system behavior that are similar in terms of measurable quantities. In this paper we describe the specific methods used and analyze their results. As we will illustrate, these results can be exploited for purposes of diagnosis and design for diagnosability, notably sensor placement.

We describe a methodology for applying inductive learning systems to the discovery of 'rule bases' for diagnosis. Our primary reason for doing so is to facilitate system design. In particular, rules suggest measurable quantities that are most diagnostic. Given a suitable tradeoff between coverage, accuracy and sensor cost, we envision a tool that aids system designers in sensor selection. We are currently in the process of systematically exploring the interaction between these factors in the context of two learning systems, Quinlan's C4.5 [13] and Fisher's COBWEB [6], with a longer-term goal of developing objective function(s) that reflect such a tradeoff.

2 Supervised Learning Approach

Supervised learning systems discover rules that characterize preclassified observations. For example, supervised machine learning systems are used in medical diagnosis; given patient case histories that record features such as gender, age, aspects of medical history, and a variety of test results, as well as a diagnosis provided by a physician, a supervised system discovers rules that are consistent with the physician-supplied diagnoses. We can also use this technology for purposes of fault diagnosis. In particular, consider the model of a thermal subsystem given in Figure 1.

We have used the following strategy to learn rules that distinguish a variety of conditions that can cause anomalous behavior in this system.

[1] Specify a simulator that represents each major system component as a function that maps component inputs to outputs. Simulation using a model-based methodology similar to Kuipers' [10] begins with an initial state of system parameter settings and propagates parameter changes through component functions until the simulator converges on a steady state.

[2] Associated with each system component are permissible parameter (continuous and discrete) ranges, within which the component is assumed to operate satisfactorily. Initial simulator parameters are systematically perturbed beyond extreme ends of these ranges for each component, thus yielding conditions under which the system is liable to malfunction.

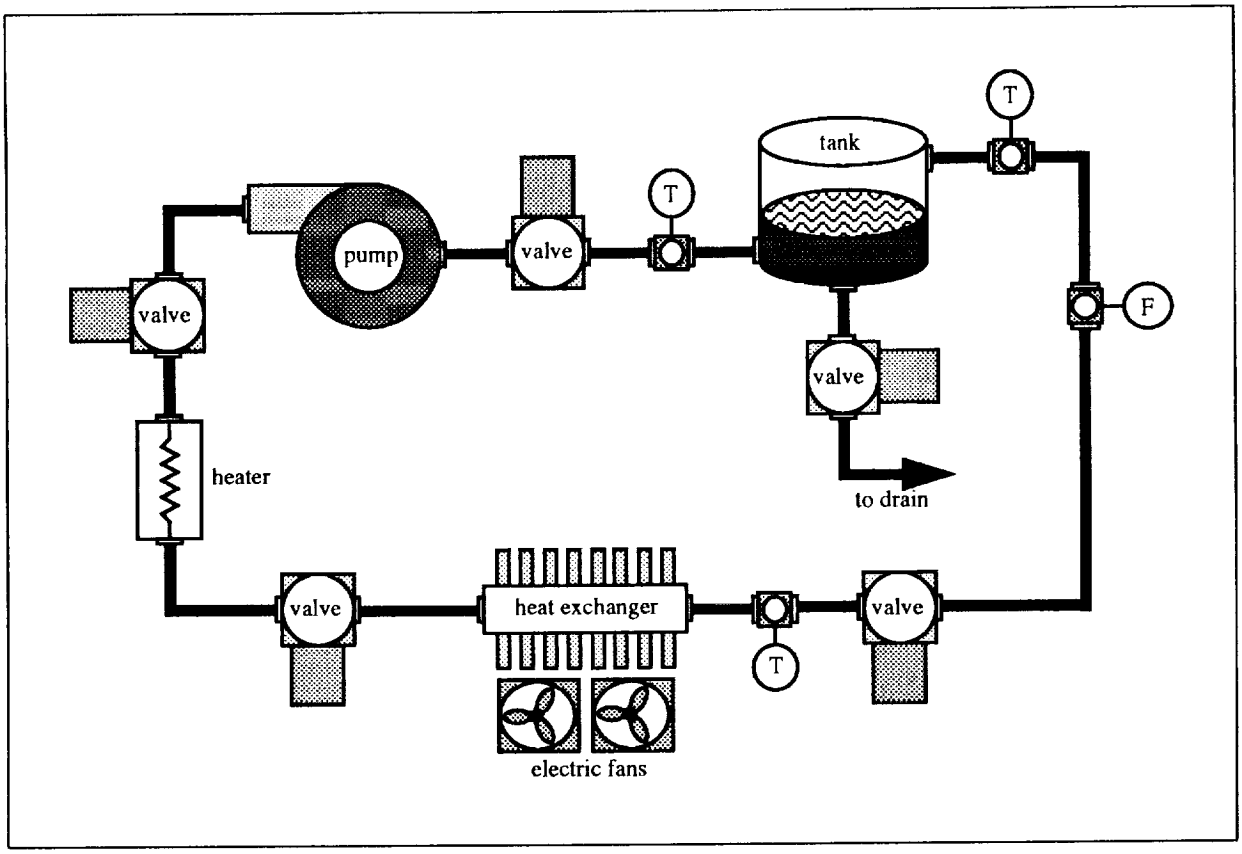

Figure 1: A thermal model.

[3] Each condition set generated in step [2] is propagated through the system until a steady state (or some error condition) is reached. A database record (which consists of measurements from each observable parameter in the system, labeled by the initial perturbed condition) is generated.

[4] The system state descriptions of all simulations are collected together and passed to a supervised learning system.

[5] The learning system forms a decision tree, then extracts rules that distinguish anomalous behaviors that were caused by different parameter perturbations.

We have used a supervised learning system known as C4.5 to form a diagnostic rule base. C4.5 has separate programs that (1) construct a decision tree and (2) form a rule base. In particular, C4.5 was used to discriminate the system perturbations ('faults') generated in step [2] of the simulation/learning procedure outlined above. Our thermal model contained a total of 87 fault types. In addition, three versions of each perturbation type were generated, corresponding to cases where the selected parameter value was perturbed just above (or below) acceptable ranges, moderately out of range, and far out of range. Intuitively, these corresponded to conditions of high (low), very high (low), and extremely high (low) values, but each case was labeled by a single fault (e.g., the parameter was 'above acceptable range'). Thus, the decision tree

had to distinguish 87 'faults', derived from over 261 observation sets (snapshots). Each snapshot was represented by 23 system parameter values. Using C4.5, we constructed decision trees much like the one partially shown in Figure 2.

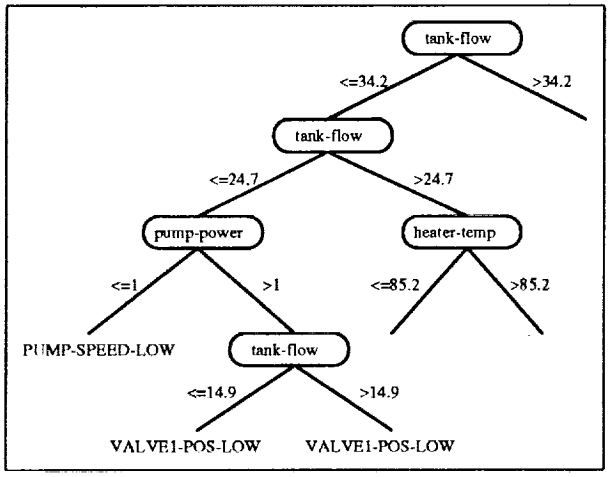

Figure 2: A partial decision tree over anomalous behaviors.

Initially, we are interested in two items: (1) the diagnostic accuracy of this tree, if we insist that faults must be perfectly isolated, and (2) how much the tree 'compresses' the parameters needed to attain a desired accuracy. We call this second factor the *parameter compression ratio.*

In this example, the decision tree correctly and uniquely classified 73% of the snapshots over which it was constructed. Note that the failure to perfectly classify all known behaviors is the result of C4.5's information-theoretic measure which could not reliably distinguish certain behaviors with the existing observable parameter values. These points of ambiguity are precisely where system designers should focus sensor placement efforts in order to better distinguish faults. It required that approximately 18 of the 23 parameters be consulted in order to achieve this accuracy – a parameter compression ratio of $(23 - 18)/23$ or 0.22.

The statistics above reflect a bias that the decision tree (or any rule-based system for that matter) should not attempt to perfectly isolate a fault. However, we can relax the diagnostic task, and allow categorization to identify an observation's fault as one of a small number of possibilities. The tree above will correctly identify each observation as exhibiting 1 of at most 3 fault possibilities (pump-speed-low, valve1-pos-low, valve1-pos-high) in 100% of the cases. Thus, we are are interested in the degree to which the tree isolates a fault. In this case, our minimal *fault compression ratio* is $(87 - 3)/87$ or 0.97.

Three aspects of this inductive analysis are of interest. Each of these speaks to the success of the diagnostic task, and provides guidelines for fault isolation and sensor placement. Our particular concern in this latter regard is with sensor placement.

- The *fault compression ratio* tells us the degree to which a behavior's fault can be isolated using the rule base. Inversely, it is a measure of the extent that we will have to rely on other sources of knowledge and diagnostic procedures, such as an expert or system simulation in conjunction with model-based diagnosis, to discriminate the fault from the reduced set of possibilities.

- The *parameter compression ratio* indicates the proportion of system parameters that need to be accessed for diagnosis over a population of behaviors. This is a guide to the number of sensors that will be required if diagnosis relies simply on sensor values.

- The *diagnostic accuracy* in a system is the percentage of behaviors that are

correctly categorized as one of several possibilities. It measures the *reliability* of diagnosis *within* the rule base, whereas fault compression measures the granularity.

These factors are, of course, interdependent. For example, decreasing allowable fault compression (undesirable) will tend to increase the required parameter compression (desirable), and increase diagnostic accuracy (desirable). In general, we cannot hope to optimize each of these parameters. Rather, design and sensor placement must optimize some tradeoff between them. For example, if accuracy is at a premium, then we may have to accept an decrease in fault compression. This implies a corresponding (but desirable) increase in parameter compression, and an expected decrease in sensor 'cost' as well. However, the undesirable decrease in fault compression implies that diagnostic cost will increase from having to employ secondary diagnostic procedures such as probing, testing, and simulation to a larger extent.

We are initiating systematic experiments across the range of diagnostic factors, with the eventual goal of defining an objective function that characterizes an appropriate tradeoff between them. Such a function will allow us to bound certain factors (e.g. accuracy, parameter compression or sensor 'cost') and to optimize for the remaining factors (e.g., fault compression). Our current version of C4.5 builds a decision tree based on the diagnosticity of system parameter values. Other variations that take into account the cost of sensing certain values have also been developed by Tan & Schlimmer [15].

A decision tree representation of a rule base is conceptually simple, and it has the desirable aspect of encoding the 'minimal' number of system measurements needed to isolate faults to a certain granularity. However, it also has some well-known disadvantages. Notably, a decision tree is very sensitive to noise in sensed system values (or faulty sensors, which we regard as another type of noise): a single misleading value can lead diagnosis considerably astray. One implication is that the minimality characteristic of decision trees may not be wholly desirable; uncertainty in a domain may insist on some redundancy in the sensed values, in order to better protect against the possibility of noise. Thus, in addition to our studies with C4.5, we are also investigating a second inductive approach known as *clustering*.

3 Cluster-Analytic Approach

A data analyst must often identify similarities and differences between observations. For example, a biologist will categorize a newly discovered organism into a known *genera* based on its similarities with known species of the class and differences with members of competing *genera*. An economist may recognize a trend in the market as having occurred previously, and forecast a particular outcome based on these historical similarities. The need to 'cluster' observations is critical in many fields, including the biological and social sciences, where it has spawned data analysis tools of *numerical taxonomy* or *cluster analysis* (e.g., Jain & Dubes [8]). Clustering methods have also evolved in artificial intelligence (AI) and machine learning (e.g., Michalski & Stepp[11]).

Clustering systems automatically discover categories of observations (events or objects) that are similar along some dimension(s). Once uncovered, these categories may sug-

gest features that characterize the observed data and/or facilitate predictions about the nature of future data. As in scientific endeavors, engineering disciplines can profit from clustering. For example, in diagnosis an observation may be a set of symptoms that collectively indicate a class of events that share a common diagnosis. We believe that discovered clusters can be used dynamically for automated diagnosis, and that like a data analyst, a system designer can use clusters over simulated behavior to facilitate design – in this case sensor placement.

3.1 COBWEB: A sample clustering system

A clustering system constructs a classification scheme over a set of observations. Figure 3 illustrates a classification tree constructed over five observations by a clustering system called COBWEB. Each node (class) in this tree represents a cluster of observations. Each cluster is represented by the distribution of attribute values over members of that node; this illustrative example assumes that observations are represented by attributes of `Size` (`small`, `medium`, `large`), `Shape` (`square`, `sphere`, `pyramid`), and `Color` (`blue`, `green`, `red`). Each leaf of the tree represents a category covering a single observation; the probability of each value in a leaf, $P(A_i = V_{ij}|\texttt{leaf}_k)$, is 1.0 (i.e., present in the corresponding observation) or 0.0 (i.e., absent, in which case it is not explicitly stored at the node). The root of the tree covers all observations, with base rate probabilities $P(A_i = V_{ij}|\texttt{root})$ that reflect global value distributions. In general, each node, C_k, contains probabilities, $P(A_i = V_{ij}|C_k)$, for each attribute value observed in a member of the node. In addition, the proportion of observations stored under each node relative to the node's parent is stored with the node. For example, forty percent of the observations stored under the root are stored under node C_1: $P(C_1|\text{root}) = 0.4$.

We will not describe the strategy used to build this categorization hierarchy over observations since it is of limited relevance in future discussion, and any of several strategies can be used. However, it is important to note that every clustering system relies on a measure of cluster quality. In COBWEB's case this is a measure of *category utility* derived from Gluck & Corter [3]:

$$CU(C_k) = P(C_k)\times$$
$$[\textstyle\sum_i \sum_j P(A_i = V_{ij}|C_k) \log_2 P(A_i = V_{ij}|C_k)$$
$$- P(A_i = V_{ij}) \log_2 P(A_i = V_{ij})],$$

which rewards clusters that *increase* the certainty inherent in the attribute value distributions. The expression above is appropriate for nominally-valued (i.e., discrete, unordered, finite) attributes, but several variations on this basic scheme (Gennari, Langley, & Fisher [7]; Reich & Fenves[14]) have been adapted to handle observations described over ordinal and continuously-valued attributes as well. The certainty-maximizing measure is used recursively, first to build a partition over the entire population of observations, and then to subpartition each of these initially-constructed clusters, thus yielding a categorization hierarchy. Our particular interest in this process is its ability to discover clusters over snapshots or instantaneous descriptions of system simulations.

3.2 Discovering Fault Modes

We use COBWEB to discover categories of fault conditions over system simulations. This proceeds in much the same way as

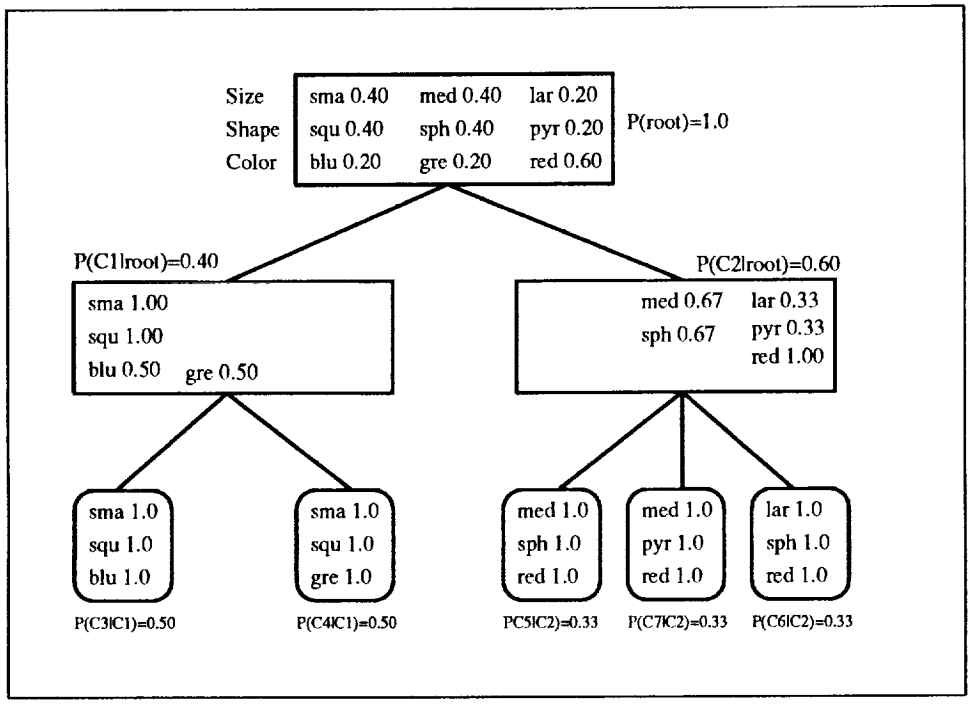

Figure 3: A classification tree constructed by COBWEB.

the simulation/induction procedure of Section 2, except that in Step [4], the snapshots are passed to our clustering system rather than a supervised one. An example of a categorization tree of discovered fault modes for the thermal system is partially shown in Figure 4. Each datum consists of inputs and outputs, for all components, including the single perturbed value (as described in step [2]); that is, each datum is a snapshot of the system. We do not show the probability distributions over all attribute values for clusters, but simply label each low-level node by a descriptor that conveys the fault-mode meaning. Thus, low flow through the radiator and a malfunction to the heater itself both result in high water temperatures (Example 1), despite the fact that this behavior emerges for very different reasons. Similarly, high flow through the pump appears somewhat similar to a second heater malfunction: both result in low water temperatures (Example 2).

As with C4.5, the benefits of clustering are at least two-fold. First, it is difficult for engineers to completely design against system faults in advance. Collectively, simulation and clustering identify fault models that benefit design decision making. For example, a faulty heater may overheat water in the thermal system, but this behavior may appear to be similar to, and thus be clustered with, a radiator (heat exchanger) that does not sufficiently cool water. Second, as with C4.5, these ambiguities can alert analysts to place sensors that better distinguish these conditions.

Again like C4.5, a COBWEB classification tree can also facilitate fault diagnosis. In particular, categories discovered through clustering associate observable/sensor/test features with component faults that lead to the observed anomalies. We wish to classify an observable set of sensor readings to a

level of the classification tree where a reasonably certain prediction of the underlying fault can be made. However, a categorization and diagnosis procedure is less clear with a COBWEB generated tree, since it does not specify a single value that should be sensed at any particular point as a decision tree does. Rather, we can exploit *characteristic* attribute values of discovered categories to direct sensor testing. There are a number of ways for identifying characteristic (or normative) values, as described in Fisher[6] and Reich & Fenves[14], but suffice it to say that they are values that are typically true of category members, and typically discriminate the category's members from other, contrasting categories. Characteristic values suggest tests that are likely to discriminate the most promising paths of the tree during classification: verification of a characteristic value(s) suggests that the associated path be followed, thus narrowing the plausible faults that are consistent with the known observables; failure to observe the expected value reduces the likelihood that the associated path will lead to a correct diagnosis.

The primary advantage of this strategy over C4.5 is that the categorization tree formed through clustering specifies a number of values at each node of the tree that can be sensed in order to guide further categorization or diagnosis. The decision tree structure is not generally as robust when certain values cannot be reliably sensed because of noise. In contrast, the increased information redundancy of the COBWEB tree is more robust in the face of noise, but redundancy also comes with the corresponding disadvantage that parameter compression is correspondingly lower.

4 Attention Focusing

Consider the space between the decision tree approach and the conceptual clustering approach as a continuum on feature structure. In decision trees the structure is fixed during training so that the order for feature testing during prediction is rigid. There is one feature test at each node with leads to a node at a deeper level (and another test).

In conceptual clustering there is no feature structure. To determine how to branch into the concept hierarchy, one must test every feature in the current node. In some cases this could lead to a significant number of tests (e.g., in our domain example from Section 2).

Optimally, we would like to classify an object or event in as few tests as possible with as few branches as possible. The decision tree approach would seem to have a tremendous advantage in classification of problems with highly independent feature spaces. However, when in a feature space with specific dependencies, it would be nice to cluster tests over these dependencies and branch deeper into the tree with fewer tests. One way in which we accomplish this is to examine the salience of each feature within each node, calculating what amounts to a category utility for each feature within the scope of its parent node.

The order of inspection for features in each node is then relative to its salience. The salience for a feature can be computed in any number of ways. In the equation below we show a general method for calculating salience based on standard deviation.

$$salience_i = \frac{\sum_k P(C_k)\frac{1}{\alpha_{ij}} - \frac{1}{\alpha_i}}{K}$$

where K is the number of classes, $P(C_k)$ is the probability of a particular class, and

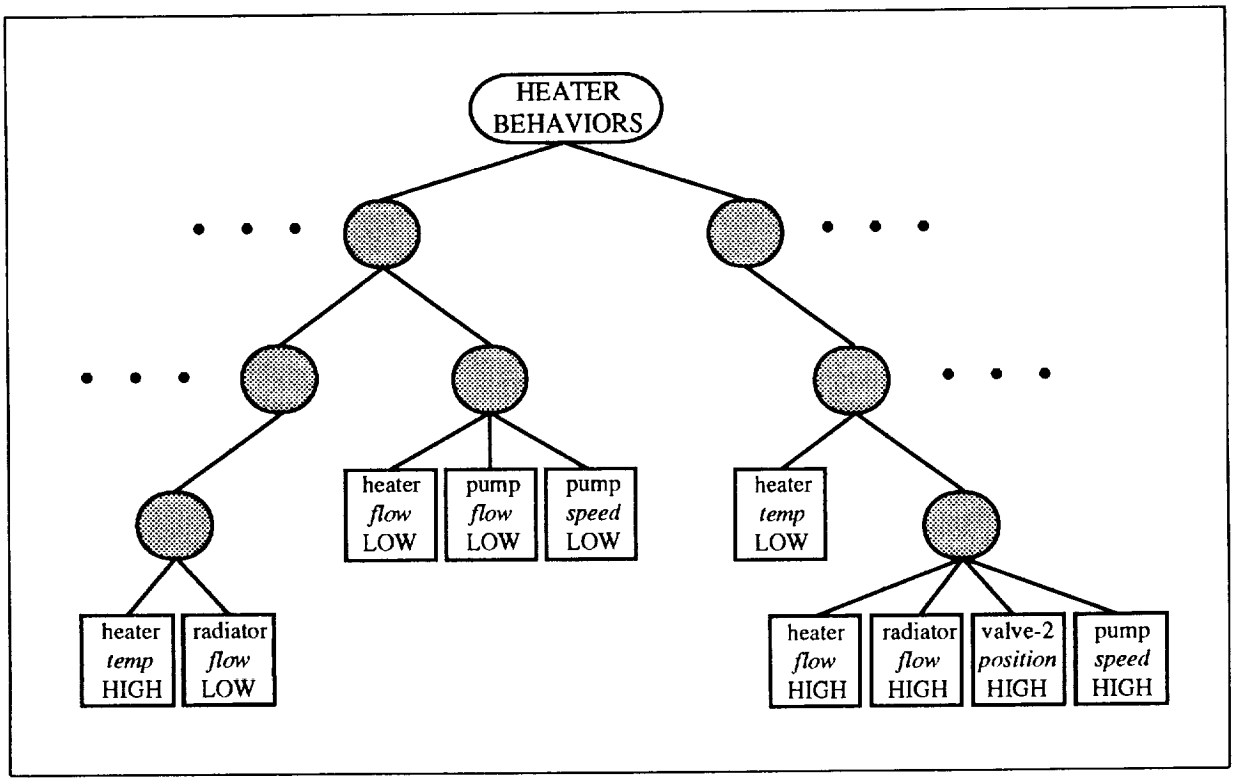

Figure 4: A partial classification tree of fault modes for the thermal model.

α_{ij} is the standard deviation of the feature within class k.

Using the notion of salience, an algorithm can be derived that focuses attention on the most informative features to test before branching into a behavior hierarchy. The following describes our algorithm for attention:

1. Select an unseen feature with probability based on salience scores stored at the parent.

2. Compute the salience of the selected feature; store this new score at the parent.

3. Compare the category utility score for the best classification, x, based *only* on features inspected so far.

4. Consider all remaining unseen features; if these were to match the second best classification, would the score be better than x?

5. If yes, goto step [1], otherwise ignore remaining attributes and branch to new node.

A problem closely associated with the calculation of feature salience is the selection of parametric measurements to ensure *complete* and *cost-effective* diagnosis. In analyzing a design for fault isolation we examine several additional factors, or properties, that belong to the device used for sensing a particular feature. A partial list of factors governing sensor selection follows:

So, when looking at which salient features to actually measure, an objective equation to minimize cost and maximize feature coverage must be designed. Below we offer a

· response time	· maintainability
· launch weight	· I/O performance
· criticality	· power consumption
· reliability	· procurement price
· repeatability	· number of sensors
· accuracy	· operating temperature
· resolution	· operating pressure

Table 1: Factors for sensor selection.

general form for such an objective equation:

$$min \sum w_i f_i'$$

where $\sum_i w_i = 1$, $f_i \in \{f_1, \cdots, f_i, \cdots, f_n\}$ are n sensor factors, and $f_i' = \| f_i \|$ is a normalized value representing the sensor factor within some range.

The following algorithm can be used for selecting which salient features to measure in a system under design.

1. Set threshold for objective equation.
2. Apply objective equation.
3. Collect sensor recommendations.
4. If *parameter compression* and *fault compression* (from decision tree analysis) are exceeded, then adjust threshold; goto root-node and restart. Otherwise branch and goto step [2].

5 Related Work

Work currently underway at JPL complements our research. JPL's AI Group has identified numerous factors that influence optimal sensor placement in Chien, Doyle, & de Mello[1], Chien, Doyle, & Rouqette[2], and Doyle & Fayyad[5]. Among these are factors that relate to the diagnosticity of sensors - i.e., the ability of sensed system quantities to predict the presence and location of faults. Roughly, diagnosticity is measured by simulating a fault on a system model, and then observing the changes to various model quantities. Quantities that differ most relative to their normal state (and possibly their value during other, competing fault conditions), are judged good predictors of that particular fault. In general, the approach makes pairwise comparisons between the same quantities under two different fault modes, and two different quantities under identical fault conditions. The approach appears to be generally helpful, but the utility of pairwise comparisons is limited. In contrast, our two learning approaches seek patterns or rules across multiple dimensions (i.e., multiple fault modes, and multiple sensed quantities) of system behavioral snapshots simultaneously. This approach can provide a more global perspective on system behavior, and makes certain multidimensional patterns explicit to the designer.

Furthermore, our approach to sensor placement is guided by an explicit model of the diagnostic process. This top-down approach contrasts with JPL's bottom-up approach, which is primarily responsible for enumerating a wider variety of factors that play a role in sensor placement. Our primary focus on a single aspect (i.e., information-content) of system parameter values that might act as good sensors is a disadvantage of our approach relative to JPL's. However, we view the two approaches as complementary, and are pursuing links between them.

6 Concluding Remarks

Our approach to sensor selection is distinguished from others in that it is guided by an explicit model of diagnosis; this top-down methodology promises principled criteria for sensor placement. Although our models of diagnosis are primarily useful for design, the rule bases developed through clustering and supervised methods could be used directly for diagnosis as well – either autonomously or by a human user. In this, we recognize the importance of both rule-based and model-based approaches as contrasted in Keller[9] and Davis[4]. Our bias is that inductive approaches can never replace model-based approaches in any but the most trivial of applications. As Keller points out, 'compiled' knowledge is most helpful in diagnosing relatively routine faults. To attempt a rule-based approach that covers idiosyncratic faults as well (i.e., achieves very high fault compression) invites 'overfitting' (i.e., unacceptably low accuracy and/or unacceptably low parameter compression). The overfitting phenomenon is well-known in machine learning, but inductive approaches to compilation for diagnosis have not traditionally addressed the issue, as shown in Pearce[12]. Rather, an ideal tradeoff between coverage, cost, and accuracy must only assume that a certain diagnostic burden is taken on by the compiled rule base. Our primary goal is to limit, but not eliminate, the space of faults that need be explored by probing, testing, and simulation.

References

[1] S. Chien, R. Doyle, and L. Homem de Mello. Model-based reasoning approach to sensor placement for monitorability. In *Proceedings of the Space Operations, Applications, and Research Symposium*, Houston, TX, 1991.

[2] S. Chien, R. Doyle, and N. Rouquette. Sensor placement for diagnosability in space-bourne systems: A model-based reasoning approach. In *Second International Workshop on the Principles of Diagnosis*, Milano, Italy, October 1991.

[3] J. Corter and M. Gluck. Explaining basic categories: feature predictability and information. *Psychological Bulletin*, 1992.

[4] Randall Davis. Form and content in model-based reasoning. In *Proceedings of the AAAI Workshop on Model-Based Reasoning*, Detroit, MI, August 1989.

[5] Richard J. Doyle and Usama M. Fayyad. Sensor selection techniques in device monitoring. In *Proceedings of the 2nd Conference on AI Simulation and Planning*, Cocoa Beach, CA, April 1991.

[6] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139-172, 1987.

[7] John H. Gennari, Pat Langley, and Douglas H. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40, 1989.

[8] A.K. Jain and R.C. Dubes. *Algorithms for Cluster Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1991.

[9] Richard Keller. In defense of compilation. In *Proceedings of the Second AAAI Workshop on Model-Based Reasoning*, Boston, MA, August 1990.

[10] B. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, December 1986.

[11] R.S. Michalski and R.E. Stepp. Learning from observation: Conceptual clustering. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 331–403, San Mateo, CA, 1983. Morgan-Kaufmann.

[12] D.A. Pearce. The induction of fault diagnosis systems from qualitative models. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 353–357, Saint Paul, MN, 1988. Morgan-Kaufmann.

[13] J.R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[14] Y. Reich and S. Fenves. The formation and use of abstract concepts in design. In D. Fisher, M. Pazzani, and P. Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, San Mateo, CA, 1991. Morgan-Kaufmann.

[15] M. Tan and J. Schlimmer. Two case studies in cost-sensitive concept acquisition. In *Proceedings of the Eigth National Conference on Artificial Intelligence*, pages 854–860, Boston, MA, July 1990.