Using Decision Trees to Detect and Isolate Simulated Leaks in the J-2X Rocket Engine

Mark Schwabacher
NASA Ames Research Center
Mail Stop 269-3
Moffett Field, CA 94035
650-604-4274
mark.a.schwabacher@nasa.gov

Robert Aguilar
Pratt & Whitney Rocketdyne
RIB-31
Canoga Park, CA 91303
818-586-4986
Robert.Aguilar@pwr.utc.com

Fernando Figueroa
NASA Stennis Space Center
EA41
Stennis Space Center, MS 39529
228-688-2482
Fernando.Figueroa-1@nasa.gov

Abstract— The goal of this work was to use data-driven methods to automatically detect and isolate faults in the J-2X rocket engine. It was decided to use decision trees, since they tend to be easier to interpret than other data-driven methods. The decision tree algorithm automatically "learns" a decision tree by performing a search through the space of possible decision trees to find one that fits the training data. The particular decision tree algorithm used is known as C4.5. Simulated J-2X data from a high-fidelity simulator developed at Pratt & Whitney Rocketdyne and known as the Detailed Real-Time Model (DRTM) was used to "train" and test the decision tree. Fifty-six DRTM simulations were performed for this purpose, with different leak sizes, different leak locations, and different times of leak onset. To make the simulations as realistic as possible, they included simulated sensor noise, and included a gradual degradation in both fuel and oxidizer turbine efficiency. A decision tree was trained using 11 of these simulations, and tested using the remaining 45 simulations. In the training phase, the C4.5 algorithm was provided with labeled examples of data from nominal operation and data including leaks in each leak location. From the data, it "learned" a decision tree that can classify unseen data as having no leak or having a leak in one of the five leak locations. In the test phase, the decision tree produced very low false alarm rates and low missed detection rates on the unseen data. It had very good fault isolation rates for three of the five simulated leak locations, but it tended to confuse the remaining two locations, perhaps because a large leak at one of these two locations can look very similar to a small leak at the other location.[1,2]

TABLE OF CONTENTS

1. INTRODUCTION

The J-2X rocket engine will be tested on Test Stand A-1 at NASA Stennis Space Center (SSC) in Mississippi. A team including people from SSC, NASA Ames Research Center (ARC), and Pratt & Whitney Rocketdyne (PWR) is developing a prototype end-to-end integrated systems health management (ISHM) system that will be used to monitor the test stand and the engine while the engine is on the test stand [1]. The prototype will use several different methods for detecting and diagnosing faults in the test stand and the engine, including rule-based, model-based, and data-driven approaches. SSC is currently using the G2 tool [2] to develop rule-based and model-based fault detection and diagnosis capabilities for the A-1 test stand. This paper describes preliminary results in applying the data-driven approach to detecting and diagnosing faults in the J-2X engine.

[2] IEEEAC paper #1408, Version 1, Updated September 4, 2008

The conventional approach to detecting and diagnosing faults in complex engineered systems such as rocket engines and test stands is to use large numbers of human experts. Test controllers watch the data in near-real time during each engine test. Engineers study the data after each test. These experts are aided by limit checks that signal when a particular variable goes outside of a predetermined range. The conventional approach is very labor intensive. Also, humans may not be able to recognize faults that involve the relationships among large numbers of variables. Further, some potential faults could happen too quickly for humans to detect them and react before they become catastrophic. Automated fault detection and diagnosis is therefore needed.

One approach to automation is to encode human knowledge into rules or models. Another approach is use data-driven methods to automatically learn models from historical data or simulated data. Our prototype will combine the data-driven approach with the model-based and rule-based approaches. This paper focuses on the data-driven approach.

2. THE J-2X ENGINE

Figure 1 - The J-2X rocket engine

The J-2X is a rocket engine currently under development at Pratt & Whitney Rocketdyne [3]. It will be fueled by liquid hydrogen and liquid oxygen. It will be used as the second-stage engine on NASA's Ares I crew launch vehicle [4] and Ares V cargo launch vehicle [5]. It is derived from the J-2 engine, which served as the second- and third-stage engines on the Saturn V launch vehicle. The J-2X engine is shown in Figure 1.

3. TEST STAND A-1

SSC operates several rocket engine test stands. Each test stand provides a structure strong enough to hold a rocket engine in place as it is fired, and a fuel feed system to provide fuel to the engine. Test stand A-1 is a large test stand that is currently used to test the space shuttle's main engines, and will be used to test the J-2X [6]. It can withstand a maximum dynamic load of 1.7 million pounds of force. It provides liquid hydrogen and liquid oxygen to the engine being tested, and has numerous sensors on its fuel feed system. Test Stand A-1 is shown in Figure 2.

Figure 2 - Test Stand A-1

4. THE J-2X DETAILED REAL-TIME MODEL

We used data from a high-fidelity physics-based simulator to train and test the data-driven algorithms. The physics-based model chosen for this project is the J-2X Detailed Transient Model or DTM. The J-2X DTM, as the name indicates, is a transient model that accurately models all phases of engine operation including start, mainstage (phase between start and shutdown), and shutdown. The J-2X DTM simulates processes describing rocket engine operation including heat transfer, fluid flow, combustion and valve dynamics. Flowrates, pump speeds, temperatures and pressures are modeled as time dependent differential equations that are updated at a high rate, typically 2000 Hz. Property tables, valve characteristics and turbomachinery efficiency and performance curves are also incorporated in the DTM. DTM's are used to develop safe start and shutdown sequences and for anomaly resolution. The J-2X DTM builds on a long history of DTM's supporting most major Pratt & Whitney Rocketdyne (PWR) rocket engines.

The J-2X DTM underwent modification to enable it to run in "real-time" mode. In real-time mode, the DTM will respond in real world clock time to external stimuli such as

changes in valve position and engine inlet conditions. The latter will comprise the interface to the test stand model. Advances in computer processor technology have made this possible due to the fast update rate required to maintain numeric stability. Faster update rates imply smaller time steps, which result in smaller errors, which result in greater stability. Real-time performance is achieved if a model advances in time (step time) at the same rate as a wall clock. If a processor can perform all calculations in a step time or less, then the model is real-time capable. The step time should also be consistent and set to the longest measured step time corresponding to the longest logical path. Shorter frames are then padded as needed to provide a consistent step time. The J-2X DTM, or any DTM for that matter, was not optimized for real-time operation. Changes that were required include streamlining model code, limiting or eliminating model diagnostic output, and fixing the step time. The J-2X DTM currently uses a variable step time to maintain numeric stability so deterministic timing is not possible. Real-time DTM operation is required when real-time communication with other system components is required such as hardware-in-the-loop testing or for online monitoring of an engine and test stand. Near real-time operation has been demonstrated indicating full real-time operation is feasible in the near future. The modified DTM now has the designation J-2X Detailed Real-Time Model or DRTM.

The DRTM was modified to enable failure mode simulation. Failure modes are modeled as changes to the flowpath of the DRTM (e.g. leaks) or modification of engine parameters (e.g. turbine efficiency) representative of failure signatures. Sensor characteristics, such as lag and bit toggle, and process noise were also modeled to better replicate engine operation. A simulation of cavitation due to low inlet pressure was also added to the DRTM as the primary test stand/engine interface fault mode. As the inlet pressure falls below a certain level, the propellant begins to vaporize and pump performance drops dramatically.

5. DATA-DRIVEN FAULT DETECTION AND DIAGNOSTICS

In our previous work [7, 8], we used unsupervised anomaly detection algorithms to automatically detect faults in Space Shuttle Main Engine data. Unsupervised anomaly detection algorithms are trained using only nominal data. They learn a model of the nominal data, and signal an anomaly when new data fails to match the model. They are useful when few examples of failure data are available. For a rocket such as the Space Shuttle Main Engine, very few examples of failures exist in the historical data. Unsupervised anomaly detection algorithms are therefore useful when using historical data as training data. For the J-2X, no real data is available yet, since the engine has not been built yet. However, we do have a high-fidelity physics-based simulator that can simulate faults. We therefore decided to use supervised learning. When used for fault detection and diagnostics, supervised learning algorithms take as input data from nominal operation and from each failure mode. They learn a model that is able to distinguish between the nominal data and the data for each fault mode. They are able to go beyond the capabilities of unsupervised anomaly detection algorithms by identifying the fault mode, rather than just detecting anomalies.

We decided to use a decision tree learning algorithm because the decision trees learned by these algorithms are much easier for human experts to interpret than the models produced by some competing algorithms such as neural networks or support vector machines. Having engineering experts examine the decision trees is very helpful for verifying them before deploying them. The decision tree algorithm automatically "learns" a decision tree by performing a search through the space of possible decision trees to find one that fits the training data. The particular decision tree algorithm used is known as C4.5 [9].

6. RESULTS

In the first experiment, two DRTM simulations were used to train a decision tree. The two simulations each had a leak at the same location, but the leaks were of two different sizes and started at two different times. The simulations included simulated sensor noise, and included a gradual degradation in both fuel and oxidizer turbine efficiency. The simulations also included all four modes, and lasted 500 seconds. The internal timestep was 0.00005 seconds, and the timestep in the recorded data was 0.02 seconds. The decision tree learning algorithm was provided with 31 simulated sensor values for each time step. The resulting tree had 14 nodes. The tree decides whether or not there is a leak at the one location at which the leaks were simulated. Engineering experts on our team examined the tree and concluded that it makes sense. Only five of the 31 sensors appear in the tree, indicating that the learning algorithm determined that is possible to accurately detect the leak using only those five sensors.

A third DRTM simulation was used to test the tree. This simulation had a leak at the same location but again with a different size and at a different time. When applied to this test set, the tree was 99.9957% accurate (meaning that 99.9957% of the time, the tree correctly identified whether there was or was not a leak), which is extremely high accuracy. Only one timestep was classified wrong.

A second set of experiments was performed using 56 DRTM simulations as follows:

- Five leak locations
- Eleven simulations for each leak location, with eleven different leak sizes

Table 1. False Alarm Rates

	leak location					
	3	4	7	8	9	total
false alarm rate	0.0032%	0.0011%	0.0000%	0.0029%	0.0000%	0.0072%

Table 2. Missed detection rates.

		leak location				
Missed detection rate		3	4	7	8	9
leak area (sq in)	0.01	17.92%	18.69%	25.05%	49.71%	30.03%
	0.03	0.02%	0.19%	23.74%	54.73%	29.09%
	0.05	0.02%	0.12%	0.06%	40.90%	18.35%
	0.07	0.00%	0.01%	0.05%	5.77%	6.94%
	0.1	0.00%	0.01%	0.04%	16.91%	0.22%
	0.12	0.01%	0.02%	0.06%	12.06%	0.04%
	0.15	0.01%	0.02%	0.05%	6.57%	0.01%
	0.17	0.00%	0.02%	0.02%	2.96%	0.00%
	0.2	0.00%	0.03%	0.00%	1.92%	0.00%
	0.22	0.00%	0.02%	0.00%	0.05%	0.00%
	0.25	0.01%	0.01%	0.02%	0.62%	0.00%

- One simulation with no leak
- Each simulation was 500 seconds, and the time at which the leak started ranged from 50 to 400 seconds.

A C4.5 decision tree was trained using 11 of these simulations (two for each leak location plus one with no leak), and tested on the remaining 45 simulations. The resulting decision tree has 12,289 nodes, and is thus too big to be easily comprehended by humans. We can offer two possible explanations for the large number of nodes. The first explanation is that C4.5 is overfitting. This hypothesis is supported by the fact that the tree had a lower error rate on the training data than it did on the test data. The second explanation is that the decision function being learned cannot be represented using a decision tree with axis-parallel cuts; C4.5 therefore learns a "stairstep" function consisting of many nodes.

The decision tree decides whether there is no leak or a leak at a particular location (out of the five locations). Table 1 shows the false alarm rates for this decision tree. The false alarm rates in the table answer the following question: Of all the time steps that do not have a leak, for what percentage does the decision tree incorrectly report a leak in each location? The total false alarm rate of 0.0072% is considered to be very good.

Table 2 shows the missed detection rates for the same tree. The missed detection rates in the table answer the following question: Of all the time steps that have a leak of the given size at the given location, for what percentage of the time steps does the decision tree fail to detect the leak? It can be seen that at leak locations 3 and 4, the tree performs very well for leaks of size 0.03 square inches or greater. For leaks at location 7, the tree performs well for leaks of size 0.05 square inches or greater. For leaks at location 9, the tree performs well for leaks of size 0.1 square inches or greater. But for leaks at location 8, the tree does not do a good job of detecting the leak until the size of the leak reaches 0.22 square inches. This reflects the fact that leak location 8 produces the smallest leak rates for a given leak area.

Table 3 shows the misisolation rate for the same tree. The misisolation rates in the table answer the following question: Out of all the time steps in which a leak of a particular size at a particular location occurs, how often is it misidentified as being at a different location? It can be seen that the decision tree does a good job of isolating leaks of size 0.1 square inches or larger for locations 3, 4, and 7, but not for locations 8 or 9. More light is shed on misisolation by the confusion matrix, which is shown in Table 4.

Table 3. Misisolation rates.

Misisolation rate		leak location 3	4	7	8	9
leak area (sq in)	0.01	82.02%	23.15%	9.23%	0.50%	18.89%
	0.03	99.97%	0.90%	2.68%	2.17%	63.99%
	0.05	99.91%	0.55%	0.05%	2.39%	61.17%
	0.07	64.24%	0.35%	0.19%	1.49%	47.77%
	0.1	0.73%	0.43%	0.40%	16.02%	2.57%
	0.12	0.83%	0.54%	0.51%	19.66%	8.74%
	0.15	0.00%	0.33%	0.30%	34.85%	2.48%
	0.17	0.00%	0.00%	0.36%	26.11%	2.64%
	0.2	0.22%	0.19%	0.00%	46.41%	0.67%
	0.22	1.13%	1.11%	0.57%	5.05%	1.13%
	0.25	0.49%	1.64%	0.36%	58.48%	0.38%

Table 4. Confusion Matrix.

no leak	3	4	7	8	9	<-classified as
474581	15	5	0	14	0	no leak
4	51745	67796	413	17	136	3
2243	1800	109719	164	2815	322	4
6210	1995	100	97651	1258	61	7
29748	2788	1444	13	61028	23797	8
14564	1141	1123	127	34542	54756	9

The confusion matrix answers the following question: When there is a leak at a particular location (or no leak), how often does the decision tree say that there is a leak at a particular location (the correct location or an incorrect location)? The first row is false alarms, the first column is missed detections, and the rest of the matrix is misidentifications (except for the diagonal). (Note: The confusion matrix and the false alarm rate table were calculated using only test data, while the missed detection matrix and misisolation matrix were calculated using both training and test data (in order to include every leak size in the set). Because of this difference, the first column of the confusion matrix does not equal the total missed detections in the missed detection matrix.) The matrix shows that locations 8 and 9 are often confused with each other, explaining the high misisolation rates for those two locations. A possible explanation for this confusion is that a small leak at location 8 could look like a large leak at location 9.

7. Related Work

Much of the previous work in fault detection for rocket engines has used unsupervised anomaly detection algorithms. This work has relied on historical data for training purposes, and the historical data has generally not contained enough examples of faults to adequately train a supervised learning algorithm such as C4.5. The work described in this paper is different in that we now have a high-fidelity simulator (the DRTM) that can provide training data for many different fault cases, which has enabled us to use a supervised learning approach.

Martin et al. [10] presents a comparison of six unsupervised anomaly detection algorithms. They ran all six algorithms using Space Shuttle Main Engine data from four space shuttle flights and two test stand firings for training, and eight shuttle flights and four test stand firings for validation. Although they acknowledge that they do not have enough data to make statistically significant comparisons of the relative performance of the six algorithms, they conclude that the algorithm with the best accuracy appeared to be either Orca [11] or the one-class Support Vector Machine, depending on how they classify ground truth in the validation data.

Park et al. [12] applied the BEAM (Beacon-based Exception Analysis for Multi-Missions) system to anomaly detection in SSME data. BEAM has nine components that use nine different approaches to anomaly detection, including both supervised and unsupervised approaches. The work reported in the referenced paper only used one of the nine components: the Dynamical Invariant Anomaly Detector (DIAD), which is an unsupervised anomaly

detection algorithm. Park et al. trained DIAD using data from 16 nominal tests, and tested it using data from seven tests that contained known failures. It detected all of the major failures in these seven tests, although it missed some minor failures and had some false alarms.

Iverson [13, 14] describes an unsupervised anomaly detection algorithm known as the Inductive Monitoring System (IMS) and its use in a Space Shuttle application. After the STS-107 Space Shuttle Columbia disaster, Iverson applied IMS to data from four temperature sensors inside the Shuttle's wings. He trained it using data from five previous Space Shuttle flights, and then tested it using STS-107 data. It detected an anomaly in data from the temperature sensors on the Shuttle's left wing shortly after the foam impact, suggesting in retrospect that with the aid of IMS, flight controllers might have been able to detect the damage to the wing much sooner than they did. More recently, IMS has been deployed to the Mission Control Center at NASA Johnson Space Center, where it is being used to monitor live data from the International Space Station.

Srivastava [15] describes a system that performs unsupervised anomaly detection in sequences of discrete data using envelope detection methods and dynamic Hidden Markov Models. This system has been applied to data from the switches in an aircraft cockpit. In this application, it detects anomalies in the sequence of switch flips made by the pilot, including detecting if the switches are flipped in an unusual order. For example, it can detect if the pilot lowers and raises the landing gear multiple times, instead of just once (as is usually the case in the training data).

Many of the existing approaches to supervised learning for systems health monitoring have used artificial neural networks to model the system. Artificial neural networks are a type of nonlinear model based loosely on the neural structure of the brain, in which the weights of the connections among neurons are automatically adjusted to maximize the fit of the model to the training data [16]. Guo and Musgrave [17] applied neural networks to sensor validation for the SSME. He and Shi [18] found that support vector machines produced better accuracy than artificial neural networks when applied to a pump diagnosis problem. One disadvantage of neural network approaches is that most humans are unable to understand or interpret the neural network models. Support vector machines suffer from similar lack of comprehensibility, but models based on decision trees are generally easier to understand, and therefore more likely to be accepted by human experts.

8. CONCLUSIONS

High-fidelity simulated J-2X data was used to train a decision tree for fault detection and fault isolation. Testing the tree on a separate set of simulated data showed that the tree has very low false alarm rates. It has very low missed detection rates for leaks of size 0.1 square inches or larger at four of the five locations, and adequate missed detection rates for leaks of size 0.2 square inches or larger at the fifth location. The tree almost always correctly isolates leaks of size 0.1 square inches or larger for three of the five locations, but tends to confuse the remaining two locations.

The decision tree described here was delivered to SSC for integration with G2 at Test Stand A-1.

ACKNOWLEDGMENTS

We thank Chuong Luu and Hagop Panossian of Pratt & Whitney Rocketdyne for their help with the DRTM. This work was funded by the NASA Innovative Partnerships Program and by Pratt & Whitney Rocketdyne.

REFERENCES

[1] Figueroa, F., Aguilar, R., Schwabacher, M., Schmalzel, J., and Morris, J., "Integrated System Health Management (ISHM) for Test Stand and J-2X Engine: Core Implementation," Proceedings of the AIAA Joint Propulsion Conference, Reston, VA: American Institute for Aeronautics and Astronautics, Inc, 2008.

[2] Gensym Web site. http://www.gensym.com

[3] Pratt & Whitney Rocketdyne J-2X Web site. http://www.pw.utc.com/vgn-ext-templating/v/index.jsp?vgnextrefresh=1&vgnextoid=8fd0586642738110VgnVCM100000c45a529fRCRD

[4] NASA Ares I Web site. http://www.nasa.gov/mission_pages/constellation/ares/aresl/

[5] NASA Ares V Web Site. http://www.nasa.gov/mission_pages/constellation/ares/aresV/

[6] NASA Test Stand A-1 Web page. http://rockettest.nasa.gov/rptmb/ssc_a1_test_stand.asp

[7] Schwabacher, M., "Machine learning for rocket propulsion health monitoring," Proceedings of the SAE World Aerospace Congress, Vol. 114-1, Society of Automotive Engineers, Warrendale, PA, 2005.

[8] Schwabacher, M., Oza, N., and Matthews, B., "Unsupervised Anomaly Detection for Liquid-Fueled Rocket Propulsion Health Monitoring," Proceedings of the AIAA Infotech@Aerospace Conference, Reston, VA: American Institute for Aeronautics and Astronautics, Inc., 2007.

[9] Quinlan, J.R., C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.

[10] Martin, R. A., Schwabacher, M., Oza, N., and Srivastava, A., "Comparison of Unsupervised Anomaly Detection Methods for Systems Health Management Using Space Shuttle Main Engine Data," Proceedings of the Joint Army Navy NASA Air Force Conference on Propulsion, 2007.

[11] Bay, S. D. Schwabacher, M., "Mining Distance-Based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule," Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York: Association for Computing Machinery, 2003.

[12] Park, H., Mackey, R., James, M., Zak, M., Kynard, M., Sebghati, J., and Greene, W., "Analysis of Space Shuttle Main Engine Data Using Beacon-based Exception Analysis for Multi-Missions," Aerospace Conference Proceedings, Vol. 6, pp 6-2835–6-2844. Washington, DC: Institute of Electrical and Electronics Engineers, 2002.

[13] Iverson, D. L., "Inductive System Health Monitoring," Proceedings of the International Conference on Artificial Intelligence, IC-AI '04, Vol. 2 & Proceedings of the International Conference on Machine Learning; Models, Technologies & Applications, MLMTA '04, 2004.

[14] Iverson, D. L., "System Health Monitoring for Space Mission Operations," Aerospace Conference Proceedings, Washington, DC: Institute of Electrical and Electronics Engineers, 2008.

[15] Srivastava, A. N., "Discovering System Health Anomalies Using Data Mining Techniques," Proceedings of the Joint Army Navy NASA Air Force Conference on Propulsion, 2005.

[16] Bishop, C. M., Neural Networks for Pattern Recognition. Oxford University Press, 1995.

[17] Guo, T.-H. and Musgrave, J., "Neural Network Based Sensor Validation for Reusable Rocket Engines.," Proceedings of the American Control Conference, Vol. 2, pp. 1367–1372, 1995.

[18] He, F., and Shi, W., "WPT-SVMs Based Approach for Fault Detection of Valves in Reciprocating Pumps," Proceedings of the American Control Conference, 2002.

BIOGRAPHY

Mark Schwabacher earned his Ph.D. in computer science in 1996 from Rutgers University. His thesis work applied artificial intelligence to engineering design. He has worked at NASA Ames Research Center for ten years, where he has worked on several ISHM activities. He served as the Software Lead of the NASA X-37 Integrated Vehicle Health Management Experiment, and is currently leading the development of the Ares I-X Ground Diagnostic Prototype in collaboration with NASA Kennedy Space Center, NASA Marshall Space Flight Center, and the Jet Propulsion Laboratory. He has also applied anomaly detection algorithms to Earth science and to aviation security.

Bob Aguilar received a B.S. from the University of Arizona in Aerospace Engineering and an S.M. in Aero-Astro Engineering from M.I.T. in 1987. He has worked at Pratt & Whitney Rocketdyne for 25 years in the areas of simulation, controls, data analysis and health manage-ment. He has worked on many PWR engine programs including Space Shuttle Main Engine, XRS-2200 (X-33), and RS-68. He is currently leading development of the J-2X Failure Detection System.

Fernando Figueroa was born in Peru and received his B.S. degree in Mechanical Engineering from the University of Hartford in 1983. He pursued graduate studies at Penn State University, receiving an M. S. degree in 1984 and a Ph.D. degree in 1988, both in Mechanical Engineering. He was in the faculty of Mechanical Engineering at Tulane University from 1988 to 2000, and an Associate Chair of Advanced Instrumentation and control in Mechanical Engineering at the University of New Brunswick, Canada (1995-1997). Since 2000, he has been at NASA Stennis Space Center, Mississippi. His interests include intelligent sensors and systems, intelligent integrated system health management, advanced sensor technologies, robotics, mobile robots, ultrasonic sensing, and controls. Applications are focused on rocket engine test stands, NASA ground operations systems, spacecraft, and Lunar/Mars Outposts.