A New Machine Learning Based Approach to NASA's Propulsion Engine Diagnostic Benchmark Problem

by

Qiyu Wu

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2015 by the
Graduate Supervisory Committee:

Jennie Si, Chair
Teresa Wu
Konstantinos Tsakalis

ARIZONA STATE UNIVERSITY

May 2015

ABSTRACT

Gas turbine engine for aircraft propulsion represents one of the most physics-complex and safety-critical systems in the world. Its failure diagnostic is challenging due to the complexity of the model system, difficulty involved in practical testing and the infeasibility of creating homogeneous diagnostic performance evaluation criteria for the diverse engine makes.

NASA has designed and publicized a standard benchmark problem for propulsion engine gas path diagnostic that enables comparisons among different engine diagnostic approaches. Some traditional model-based approaches and novel purely data-driven approaches such as machine learning, have been applied to this problem.

This study focuses on a different machine learning approach to the diagnostic problem. Some most common machine learning techniques, such as support vector machine, multi-layer perceptron, and self-organizing map are used to help gain insight into the different engine failure modes from the perspective of big data. They are organically integrated to achieve good performance based on a good understanding of the complex dataset.

The study presents a new hierarchical machine learning structure to enhance classification accuracy in NASA's engine diagnostic benchmark problem. The designed hierarchical structure produces an average diagnostic accuracy of 73.6%, which outperforms comparable studies that were most recently published.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF SYMBOLS

Symbol Page

CHAPTER 1

INTRODUCTION

Propulsion gas turbine engine is a large integrated, “multi-physics” system comprising of aerodynamic, chemical, electrical, mechanical, and thermodynamic characteristics [1]. Propulsion gas turbine engine fault/failure diagnostic is one of the most complicated problems in the world.

To encourage creative solutions, NASA created a benchmark propulsion engine diagnostic problem, which is summarized in the software package: Propulsion Diagnostic Method Evaluation Strategy (ProDiMES) [2]. ProDiMES provides high-fidelity engine flight simulations that cover most of the critical engine failure modes. It enables comparison between different diagnostic methods by setting blind test data and failure classification performance metrics.

Some model-based diagnostic approaches have been applied to this benchmark problem. NASA Glenn has developed a weighted least squares (WLS) single fault isolation technique, which consisted of parameter correction, trend monitoring, anomaly detection, event isolation, and the recording of results [3]. University of Liege has developed a 3-module performance analysis tool composed of 1) trend monitoring, 2) fault detection and 3) fault isolation [4]. Wright State University has adopted a Generalized Observer/Estimator [5] for Single Fault Isolation. These three methods have achieved similar diagnostic performance [2].

There are also some data-driven methods applied to this problem, such as machine learning (ML) technique. ML approaches allow a way around the complex physical model of propulsion engine but can be powerful in classification problem by

deciphering data. The machine learning based HSVMkSIR method presented in [1] improves the diagnostic accuracy over the above three model-based methods. It is conceivable that machine learning approaches have even more potentials in this problem.

In the past few decades, apart from collecting tremendous data needed for specific problems, researchers have also made great efforts to improve ML algorithms to deal with those "complicated and difficult" physical problems. There have been some remarkable results. An effective approach to complicated recognition problem is Support Vector Machine (SVM). SVM is a universal function approximator [7]. It maps the input signals into a high-dimensional feature space. A separating hyperplane with high generalization ability is then optimally determined. More than often, there are non-separable training data in complicated recognition problems. In these cases, SVM has proved to maintain its robustness by finding the optimal hyperplane.

Other widely accepted approaches to complicated problems include Multi-layer Perceptron (MLP) [8]. It is also a universal approximator. Based on the back-propagation algorithm [9], MLP has the capability to do multi-class recognition with high fidelity.

These significant results provide us with the basic tools to perform engine diagnostics with ML.

This thesis presents a new hierarchical structure that takes into consideration different failure mode characteristics that are evaluated from various machine learning analysis. Every part in the hierarchical structure is tested and optimized to ensure the best output accuracy. To be comparable with the other diagnostic results, all the ML training and testing are based on the "benchmark problem" presented by NASA. The hierarchical structure proposed in this thesis outperforms other algorithms to our knowledge.

CHAPTER 2

DIAGNOSTIC PROBLEM DESCRIPTION

A. A Challenging Problem

Keeping the gas turbine engine in normal working condition is important yet challenging. Disciplined monitoring is required for the engines in operation. Generally, the monitoring system collects a dataset that includes sufficient engine measurements. Air flow temperature and pressure in different engine positions, actuator positions, rotor speeds, to name a few, are some of the most indicative parameters. During the operation, newly collected data will be sent to processor and compared with what engineers have saved as “normal data”. If the collected data exceeds the normal range engineers have already set, computer would alarm a fault.

Conventional model-based approach has matured diagnostic functions to detect and alarm engine faults. However, due to the complexity of engine characteristics, not all of them are capable of classifying faults with various root causes [1]. The challenging issue is that faults often give rise to a multi-mode symptom. Meanwhile, diagnostic functions are prone to become mathematically under-determinant when only small numbers of measurements are available.

Adding to the complexity, there are hundreds of thousands of engine makes and each of them will operate in various conditions.

In this study, ProDiMES package by NASA is examined in order to propose potential solution to the benchmark problem. The high-fidelity simulator covers most of the critical engine faults, making the ML results in the thesis useful and realistic.

B. Original Dataset Description

ProDiMES package is publicly available since 2013 fall. The Engine Fleet Simulator (EFS) embedded in ProDiMES emulates engine data at either cruise or takeoff condition. Users can specify the type and amount of engines for the simulation. In this study, the data is emulated in cruise condition only. More details about the simulator is available in [2].

The benchmark challenge problem dataset contains 998 simulated engines. Each engine is simulated for 50 flights. For some of these 998 engines, a pre-determined fault mode will be injected at some time during the 50 flights. Once injected, the fault will not be cleared until the last (or 50^{th}) flight. There are totally nine different fault modes. Table 1 shows the fault codes.

0	1	2	3	4	5	6	7	11	12
Norm	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Ps3	T24

TABLE I. FAULTS CODES IN THE SIMULATION

Mode 0 in the list is for normal flight condition. Fault modes 1~5 represent gas path component faults, modes 6~7 represent actuator faults and modes 11~12 represent sensor faults.

The total number of flights is 998×50=49,900. For each flight, there are 8 output variables. Thus, the dataset is a three-dimensional 998×50×8 structure. From the 49,900 flights, 13,880 flights are chosen to form the *training* subset, where 7,940 flights are in normal condition and 5,940 are injected with one of the nine fault modes.

Each flight in the training subset is labeled with its fault mode (mode 0 means normal condition). These 13,880 samples with label can be used for training and cross-validation. After that, the whole dataset of 49,900 flights will be used for blind testing.

C. Data Preprocessing for Machine Learning

Due to the unavailability of instant feedback on 49,900-flight blind testing results, we picked out some of the samples from the labeled 13,880 data to constitute a dataset for "independent testing". These picked out data will not be involved in training and cross-validation process. They help us efficiently examine the generalization of the trained networks. "Independent testing" accuracy would serve as indication in the algorithms' optimization process.

We have in total 13,880 available training samples with labels. Every engine mode takes an unequal share of samples, i.e., 7940 samples for mode 0, 680 for mode 1, 680 for mode 2, 670 for mode 3, 660 for mode 4, 640 for mode 5, 620 for mode 6, 670 for mode 7, 710 for mode 11, and 610 for mode 12. For each mode, 10% samples are randomly chosen. Table II shows the actual number of samples involved in training, and the number of samples for the independent testing:

	Training	Independent Testing
Mode 0	7146	794
Mode 1	612	68
Mode 2	612	68
Mode 3	603	67
Mode 4	594	66
Mode 5	576	64
Mode 6	558	62
Mode 7	603	67
Mode 11	639	71
Mode 12	549	61

TABLE II. NUMBER OF SAMPLES IN TRAINING AND INDEPENDENT TEST

D. Brute Force Machine Learning Result

After the data has been preprocessed, we'd like to see the brute force ML performance before any performance optimization is made. Firstly, the supervised ML is implemented with Multi-layer Perceptron network for multi-pattern recognition. Because

of the dominant numbers of normal mode samples, which cover 57.2% of all training samples, the brute force method is implemented in two different ways: one is implemented including normal mode samples and the other not including normal mode samples.

The brute force MLP is tested ten times respectively for both recognitions with and without mode 0, with several different network conditions. Tests show that the adopted brute force MLP is not reliable fault classifier. For each engine mode, the classification result can be different from time to the next, even under the same training network conditions (same number of layers and nodes).

Nonetheless, the best output result from using a MLP is picked out and shown below in Table III and IV. Confusion matrix in Table III was achieved with 1 hidden layer including 15 hidden nodes. Confusion matrix in Table IV was with 1 hidden layer with 30 hidden nodes.

In this benchmark problem, the accuracy of ML outputs is measured by the average value of the diagonal elements in the confusion matrix as specified by the benchmark problem [1]. The best average accuracy of brute force MLP algorithm is 0.629 for dataset without normal mode, and 0.354 for dataset including normal mode. As stated, the other brute force MLP test result accuracy are lower than these two outcomes just summarized.

		Predicted Modes								
		1	2	3	4	5	6	7	11	12
Target Modes	1	0.88	0.04	0	0	0.013	0	0	0.067	0
	2	0.031	0.585	0.046	0	0.015	0	0.046	0.262	0.015
	3	0	0.066	0.79	0	0.016	0	0.016	0.115	0
	4	0.014	0	0	0.972	0	0	0	0.014	0
	5	0.018	0.127	0.091	0	0.727	0	0	0.036	0
	6	0	0.032	0.016	0.048	0	0.484	0.371	0.032	0.016
	7	0.105	0.25	0.079	0	0.013	0	0.316	0.197	0.039
	11	0.107	0.107	0.173	0	0.04	0	0.08	0.493	0
	12	0.057	0.075	0.057	0	0.019	0	0.321	0.057	0.415

TABLE III. CONFUSION MATRIX FROM BRUTE FORCE MLP W/O MODE 0

		Predicted Modes									
		1	2	3	4	5	6	7	11	12	0
Target Modes	1	0.613	0	0	0	0	0	0	0	0	0.387
	2	0.067	0	0	0	0	0	0	0	0	0.933
	3	0	0	0	0	0	0	0	0	0	1
	4	0	0	0	0.946	0	0	0	0	0	0.054
	5	0	0	0	0	0	0	0	0.579	0	0.42
	6	0	0	0	0.043	0	0.565	0.043	0.13	0	0.217
	7	0.03	0	0	0	0	0	0.121	0	0	0.848
	11	0	0	0	0	0	0	0	0	0	1
	12	0.05	0	0	0	0	0	0	0.25	0.3	0.4
	0	0.003	0	0	0	0	0	0	0	0	0.997

TABLE IV. CONFUSION MATRIX FROM BRUTE FORCE MLP WITH MODE 0

CHAPTER 3

ARCHITECTURE DESIGN TO IMPROVE ACCURACY

A. Pair-wise Recognition Method

Table IV shows that almost every fault mode suffers high false positive rate from the normal mode. So we first put aside the normal mode, and look for potential accuracy improvement by attempting for good classification of the nine fault modes only.

From Table III we can see there are four modes with accuracy below 50%: 0.484 for mode 6; 0.316 for mode 7; 0.493 for mode 11; 0.415 for mode 12. Looking into details we notice that, mode 6 suffers from interference from mode 7, mode 7 suffers from mode 11, and mode 12 suffers from mode 7. We then carry out some pairwise MLP test to see whether these misclassifications can be mitigated.

Modes	6-7	6-11	6-12	7-11	7-12	11-12
Accuracy	0.99	0.99	0.99	0.74	0.84	0.95

TABLE V. PAIR-WISE MLP AMONG MODE6, 7, 11, 12

Table V shows good classification accuracy of using pair-wise models. Based on this result, we derive the following classification strategy for modes 6, 7, 11, 12.

From Bayes's rule [10],

$$P(A|O)P(O)=P(O|A)P(A)$$

We denote A as the truth mode of an input sample, and O as the output sequence when the sample is put into the six pairwise machines in Table V.

For samples of modes 6, 7, 11 or 12, the probability $P(O)$ is the same. In practice, $P(A)$ – the probability of fault occurrence may differ from mode to mode [7]. In this benchmark problem simulation, different fault modes are equally likely. Thus $P(A)$ is considered the same for each mode.

Thus, these two marginal probabilities in Eqt.1 can be eliminated. The probability of a fault mode prediction given the sample's output sequence from six pairwise machines can be depicted by ***P(O|A)*** – the probability of an output sequence given a specific fault mode. ***P(O|A)*** can be formed with tests. For modes 6, 7, 11 or 12, their labeled samples are put into the six pairwise machines. We can calculate the frequencies of different output sequences. The result is shown in Table VI.

Output sequence \ Truth mode	6	7	11	12
[6 6 6 7 7 11]	0.0806			
[6 6 6 7 7 12]	0.0742			
[6 6 6 7 12 11]	0.0935			
[6 6 6 7 12 12]	0.4419			
[6 6 6 11 7 11]	0.0613			
[6 6 6 11 7 12]	0.0081			
[6 6 6 11 12 11]	0.0758			
[6 6 6 11 12 12]	0.1629			
[6 6 12 7 12 12]				0.1197
[6 6 12 11 12 12]				0.0262
[6 11 6 11 7 11]			0.0183	
[6 11 6 11 12 11]			0.0479	0.0016
[6 11 6 11 12 12]	0.0016			
[6 11 12 7 12 12]				0.0689
[6 11 12 11 7 11]			0.0127	
[6 11 12 11 12 11]			0.0042	
[6 11 12 11 12 12]				0.018
[7 6 6 7 7 11]		0.0164		
[7 6 6 7 7 12]		0.0075		
[7 6 12 7 7 11]		0.0119		
[7 6 12 7 7 12]		0.0239		
[7 6 12 7 12 12]				0.0787
[7 6 12 11 12 12]				0.0311
[7 11 6 7 7 11]		0.0463	0.0014	
[7 11 6 7 7 12]		0.0119		
[7 11 6 11 7 11]		0.0134	0.0634	
[7 11 6 11 12 11]			0.0296	
[7 11 6 11 12 12]				0.0016
[7 11 12 7 7 11]		0.4418	0.0535	0.0049
[7 11 12 7 7 12]		0.294		0.0082
[7 11 12 7 12 11]			0.0042	0.0016
[7 11 12 7 12 12]		0.003		0.3787
[7 11 12 11 7 11]		0.1194	0.5507	0.0082
[7 11 12 11 7 12]		0.009		0.0098
[7 11 12 11 12 11]		0.0015	0.2113	0.0197
[7 11 12 11 12 12]			0.0028	0.223

TABLE VI. FREQUENCIES OF PAIRWISE OUTPUT FOR MODES 6,7,11,12

The strategy to classify modes 6, 7, 11, 12 is that for a given sample, we observe its output and compare the output with those listed in Table VI. The mode that has the

highest frequency will be regarded as the recognition result. The two red highlighted cells in Table VI show where the false classification rate is produced. When the output sequence is [7 11 12 7 7 12], about 5% of mode 11 samples will be recognized as mode 7, and when the output sequence is [7 11 12 11 7 11], about 12% of mode 7 samples will be recognized as mode 11.

Mathematically, there exist 2^6=64 possible output sequences. In Table VI, there are only 36 different sequences. That's the result of the available samples. Very likely, the sequence not listed in Table VI will likely occur when testing with more samples. How do we deal with the other 18 possible sequences?

With our experience in repeated tests, we set the rule that if two or more pair-wise results point to fault mode 6, the recognition output will be mode 6. The same logic is applied to mode 12. If the output is neither mode 6 nor mode 12, the pair-wise between modes 7 and 11 will determine the recognition output. This rule will serve as the complement for this classification strategy.

If this strategy is properly placed in the architecture, the classification accuracy among modes 6, 7, 11, 12 is expected to boost.

B. Self-organizing Map Results

To gain insight on the relationship among different modes intuitively, we turn to the Self-Organizing Map (SOM) algorithm, which can visualize high dimension data.

SOM is an unsupervised machine learning method introduced by Dr. Teuvo Kohonen [11]. Only based on the input data samples, the SOM network can form cluster structure in the data without supervision.

SOM is able to transform high dimensional input data points into a two dimension map. The map contains finite number of neurons which are well aligned. Each input training sample will fall on one of these neurons, called the winner neuron. Generally, samples with similar patterns fall on neurons with close proximity to the winner neuron. Consequently, if we input training samples of different patterns, the SOM output will contain several separable clusters. This visualization provides us with some intuitive view of the relationship among the engine fault modes.

Although SOM is an unsupervised learning process, we can label each neuron by calculating which mode of samples has hit it most. Fig.1 is a labeled SOM result, where four clusters are clearly separated corresponding to mode 1, mode 2, mode 4 and mode 5. That implies those four modes can be well classified.

In complicated recognition problems, SOM results usually contain overlaps, even between just two modes. Fig.2 shows the overlap between mode 0 and mode 2. That means mode 0 and mode 2 cannot be well separated, by using SOM without careful tuning, to say the least.

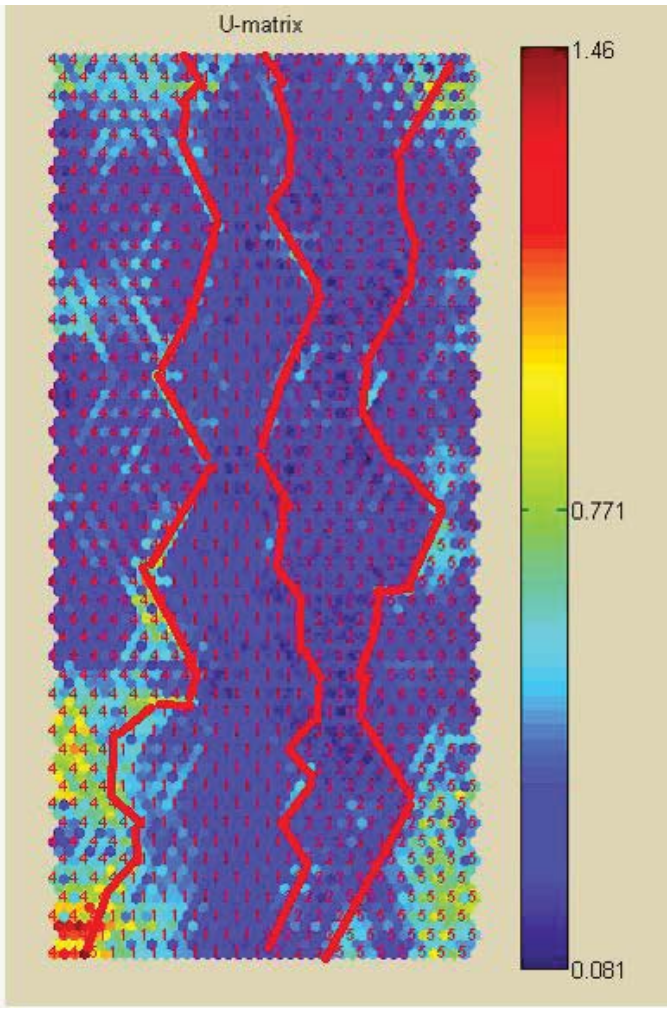

Fig. 1. SOM result for modes 1,2,4,5 samples

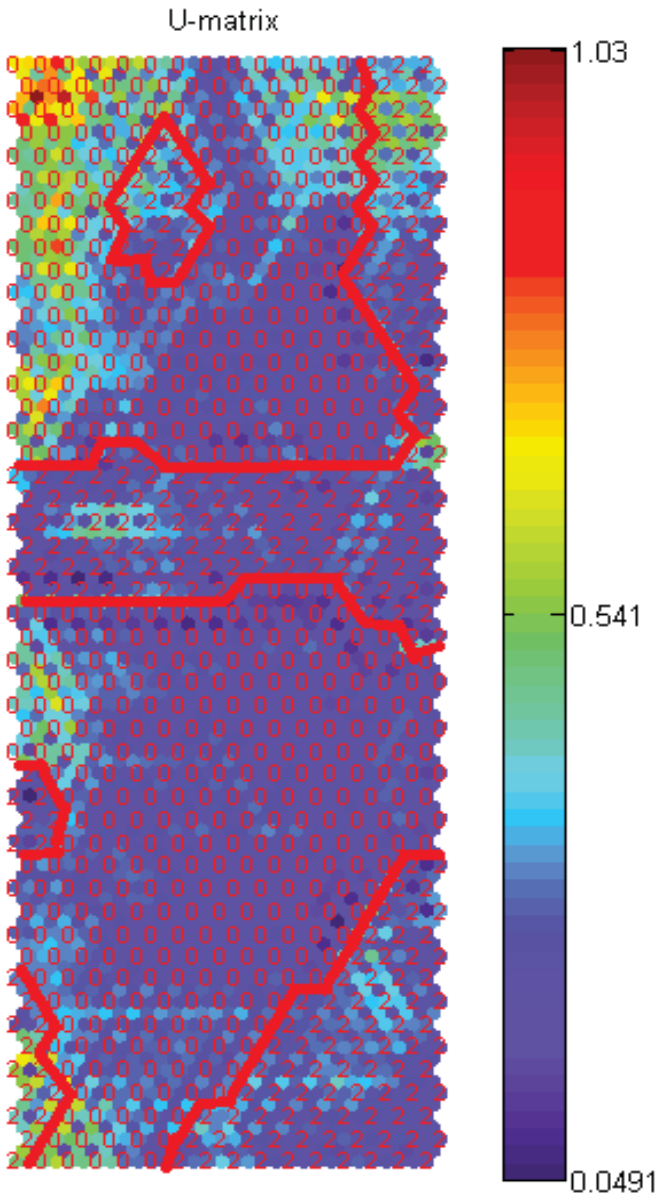

Fig. 2. SOM result for mode 0 and mode 2 dataset

If samples of more modes are put into SOM, the result is even worse. Fig.3 shows the result when all samples of all modes are involved (For clarity, result map uses 8 to mark mode 11, 9 to mark mode 12). These ten modes are totally messed up in the map. That's in line with the poor brute force recognition accuracy by MLP summarized in Table IV.

It would be natural to think of the following: which ones of all the 10 engine modes are likely to overlap with each other? Some modes may only overlap with one of the other nine modes, while some modes can overlap every other mode. The latter ones are more disturbing in ML process, thus denoted as “difficult modes”. To find these “difficult modes”, similar SOM analysis is implemented as those in the Fig.1 and Fig.2.

There exist so many combinations of modes to be tested with SOM. To improve the efficiency, we make use of the insight obtained from Table III and Table IV. For a given target mode, the other modes contribute to the false positive. The modes with high false positive rates are regarded as being more likely to entangle with the target mode.

It has to be noted that even though Table IV shows detrimental effect of mode 0, which contributes high false positive rate with many fault modes, taking mode 0 away does not necessarily enhance the average accuracy greatly. It is easy to understand such an example: a mode 2 sample recognized as mode 0 may be still wrongly recognized as mode 7 when mode 0 is not included in the study.

Given the above analysis and a few SOM tests for verification, it can be concluded: mode 0 is the “most difficult mode”; modes 2, 7, 11 are “difficult modes”. These modes show intensive overlap with other modes in SOM.

On the other hand, there is very little confusion among modes 1, 2, 3; modes 4, 5, 7; and modes 6, 11, 12.

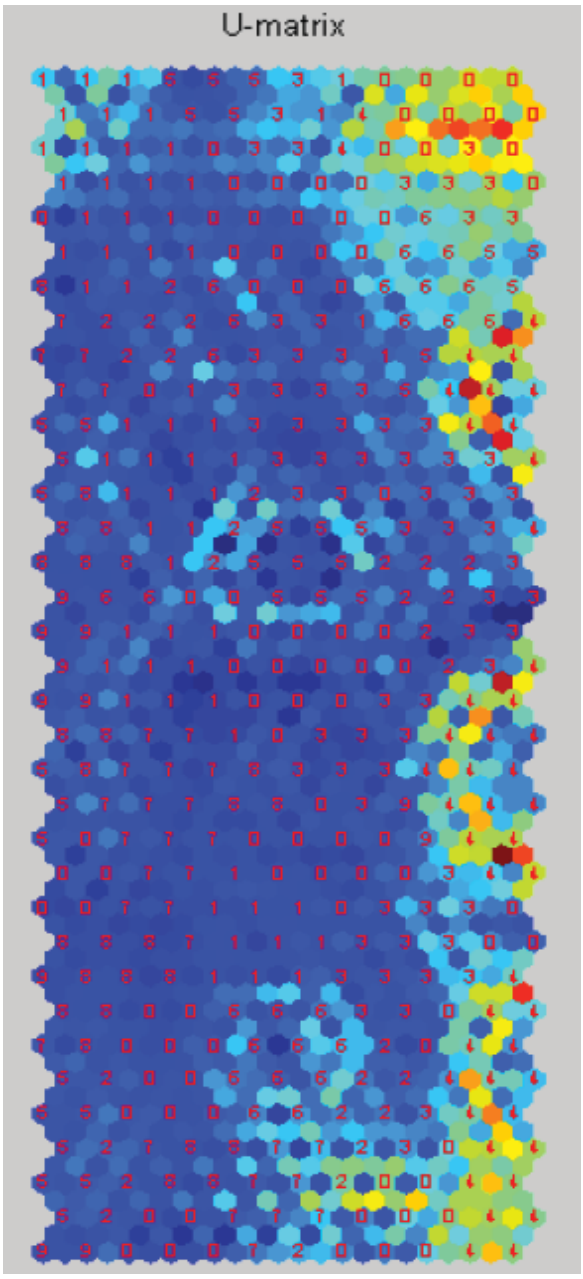

Fig. 3. SOM result for all modes' samples

C. Hierarchical Structure

The SOM result divides the 10 engine modes into 4 groups:

Group	Included mode
I	1, 2, 3
II	4, 5, 7
III	6, 11, 12
IV	0

TABLE VII. FOUR GROUPS OF MODES

The modes in the same group have little overlap with each other (Group IV only contains mode 0). That inspires the idea to reduce the number of modes in ML process. For a given sample, we firstly determine its most likely mode in each group. These likely modes are denoted as Candidate I, Candidate II, and Candidate III (Mode 0 in group IV is by default). Due to the little overlap within each group, the actual mode of the sample would be easy to pick out as one of the candidates. Thus the number of involved modes is cut down to 4 from 10. The possibility of wrong recognition is reduced.

For example, think of a data sample whose mode is to be predicted. Its truth is mode 1, but we do not know it when we do prediction. Denote the pairwise prediction accuracy for mode 1 samples with mode 4 samples, mode 5 samples and mode 7 samples as a_{14}, a_{15}, and a_{17}, respectively. Assume the sample is correctly chosen as mode 1 candidate from the group I, which consists of modes 1, 2 and 3. This assumption is reasonable for we have shown the prediction among modes 1, 2 and 3 is fairly accurate.

If a pure pairwise method is used, which means modes 1, 4, 5, 7 will have pairwise prediction between every two modes and then a majority vote mechanism is used, mode 1 have to win at least 2 out of 3 competitions with modes 4, 5 and 7. Thus, the probability that mode 1 is correctly chosen will be

$$P_1 a_{14} a_{15} + P_2 a_{15} a_{17} + P_3 a_{14} a_{17} + P_4 a_{14} a_{15} a_{17} \quad (1)$$

where P_1, P_2, P_3, P_4 are marginal probability.

There are two important facts about the above probability calculation. This calculated value is an upper bound of the probability that mode 1 can be correctly chosen because when mode 1 only wins 2 rather than 3 times, it is unlikely to win the global pairwise competition. Second fact is that the value of P_1 is dominant over P_2 and P_3

since it is unlikely for mode 1 to confuse with modes 4 or 5 while well separated with mode 7, shown in our tests.

On the other hand, let us calculate the probability of correct prediction when we first pick out candidate from modes 4, 5 and 7 and then do pairwise between mode 1 and the picked candidate. It is easy to calculate that

$$P_{14}a'_{14} + P_{15}a'_{15} + P_{17}a'_{17} \quad (2)$$

where P_{14}, P_{15}, P_{17} are respective frequency of modes 4, 5 or 7 picked out as group 2's candidate. The value of a'_{14}, a'_{15}, and a'_{17} are conditional probabilities of correct prediction conditioned on the candidate mode.

All the parameters in (1) and (2) can be calculated by repeated experiments. The statistics from our experiments have shown (2) produces larger value than (1). Thus we should first select candidates from each group and then do pairwise classification between them.

A hierarchical structure is thus ready to be implemented to realize logical development discussed above. The structure would have two parts: the first part recognizes potential fault modes without involving mode 0 and the second part performs pair-wise recognition between the one of the potential fault modes and mode 0. Fig.4 shows the hierarchical structure.

At the top layer, three networks are trained for Groups I, II, and III. Each network produces a Candidate mode. At the second layer, three Candidate modes will go through another network to determine the most likely mode, denoted as "winner candidate" (mode 0 is not involved in this layer yet). The pair-wise vote mechanism in Section III-A will be used to verify the recognition result. Once the winner candidate is one of modes 6,

7, 11, and 12, the pair-wise mechanism will be triggered. After the verification, the outcome mode will enter the second part to have a pair-wise recognition with mode 0. If the winner candidate is none of modes 6, 7, 11, 12, it will directly go to the pair-wise with mode 0.

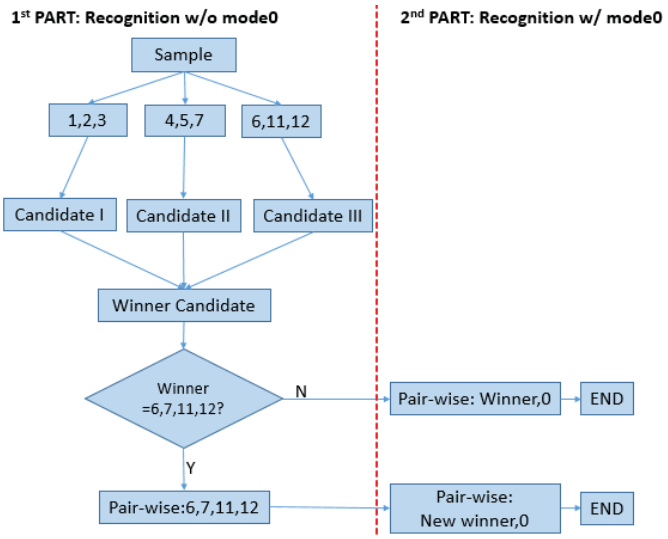

Fig. 4. Hierarchical structure block diagram to recognize sample mode

D. SVM or MLP

After the hierarchical structure is built, the next step is to choose the training algorithms to obtain classification models. The training and independent testing results will help determine whether MLP or SVM is more suitable for this problem.

In general, MLP has a better convergence when the number of nodes in the multi-layer network is increased. MLP will reach its best performance when the number of nodes is large enough [12]. The MLP output accuracy at the top layer of the hierarchy is shown in Table VII. Each MLP has been tuned to its best performance. MLP for Group I (modes 1,2,3) has 1 hidden layer and 15 hidden nodes. MLP for Group II (modes 4,5,7)

has 1 hidden layer and 34 hidden nodes. MLP for Group III (modes 6,11,12) has 1 hidden layer and 31 hidden nodes.

	Top layer output accuracy
Mode1	63/68 (93%)
Mode2	60/68 (88%)
Mode3	58/67 (87%)
Mode4	66/66 (100%)
Mode5	59/64 (92%)
Mode6	61/62 (98%)
Mode7	63/67 (94%)
Mode11	70/71 (99%)
Mode12	49/61 (80%)

TABLE VIII. INDEPENDENT TEST TOP LAYER OUTPUT USING MLP

Then the MLP algorithms are replaced by SVM, and similar output observation is installed at the top layer. In the training process, SVM with linear kernel can barely converge. Then non-linear kernels are tested such as Quadratic kernel, Polynomial kernel, Gaussian Radial Basis Function kernel and Multilayer Perceptron kernel. They converge with the training data in most cases. Among them, the Gaussian Radial Basis Function kernel produces the best output accuracy.

Obviously in this engine fault diagnostic problem, the data points are not linear separable. Thus we should adopt the soft margin SVM [7]. Another reason for using soft margin SVM is to increase model robustness as hard margin SVM would be overly sensitive to data noise. In the case of high-fidelity ProDiMES simulations, the data points are imposed with various noises.

Using the Gaussian Radial Basis Function kernel and tuning the box constraint value *c* for the soft margin until SVM best performance produces the output accuracy at the top layer of the hierarchy.

From Tables VII and VIII we can see that MLP has an average accuracy of 92.3%, and SVM 97.3%. Clearly, SVM outperforms MLP by 5%. Since this comparison has covered all the fault modes, SVM is deemed as a good model selection algorithm for this engine diagnostic problem.

	Top layer output accuracy
Mode1	65/68 (96%)
Mode2	66/68 (97%)
Mode3	66/67 (99%)
Mode4	66/66 (100%)
Mode5	64/64 (100%)
Mode6	62/62 (100%)
Mode7	67/67 (100%)
Mode11	71/71 (100%)
Mode12	51/61 (84%)

TABLE IX. INDEPENDENT TEST TOP LAYER OUTPUT USING SVM

E. Deal with Mode 0, An Indirect Way

So far, we've mainly focused on improving fault mode recognition accuracy. Now we'll try to deal with the "most difficult" normal mode — Mode 0.

We extend our observations to each layer output of our hierarchy to gain a panorama of the SVM accuracy for the independent test dataset.

	Top layer	Candidate I,II,III	Pairwise:6,7,11,12	Pairwise:winner,0
Mode1	63/68 (93%)	61/68 (90%)	/	49/68 (72%)
Mode2	66/68 (97%)	43/68 (63%)	/	20/68 (29%)
Mode3	66/67 (99%)	59/67 (88%)	/	34/67 (51%)
Mode4	66/66 (100%)	66/66 (100%)	/	66/66 (100%)
Mode5	64/64 (100%)	58/64 (91%)	/	56/64 (88%)
Mode6	62/62 (100%)	62/62 (100%)	61/62 (98%)	/
Mode7	67/67 (100%)	/	45/67 (67%)	24/67 (36%)
Mode11	71/71 (100%)	/	57/71 (80%)	4/71 (6%)
Mode12	51/61 (84%)	/	46/61 (75%)	40/61 (66%)

TABLE X. INDEPENDENT TEST OUTPUT AT EACH LAYER WITH SVM

The red highlighted cells in Table X are where prediction accuracy decreased greatly. They all appeared in the last step of the hierarchy, i.e. the pair-wise recognition with mode 0 (normal mode).

In previous analysis, we've stated that mode 0 is entangled with almost every other fault mode. The SVM result has verified this statement. It looks hard to directly determine Mode 0 by ML.

But that mode 0 is related to the fault modes gives us an idea to "indirectly" recognize it with our hierarchical structure. For the top layer, the output is combination: [Candidate I, Candidate II, Candidate III]. When the 794 Mode 0 samples in the independent test dataset go through the top layer, the output combination has the following distribution:

Combination	Number of samples	Frequency
[2,7,11]	396	0.499
[3,7,11]	191	0.241
[1,7,11]	101	0.127
[3,5,11]	26	0.033
[2,7,12]	23	0.029
[2,5,11]	16	0.020
[1,7,12]	14	0.018
[2,4,11]	9	0.011
[3,7,12]	6	0.008
[2,7,6]	5	0.006
[1,7,6]	3	0.004
[3,7,6]	1	0.001
[3,5,6]	1	0.001

TABLE XI. TOP LAYER OUTPUT DISTRIBUTION FOR MODE 0 SAMPLES

Table X shows that the top 9 most frequent combinations have covered 98.6% of mode 0 top layer outcome. That means the other 18 possible combinations only contribute 1.4% of mode 0 sample outcomes. With this result, to reduce the high false positive rate of mode 0, we set the rule in the architecture that any sample with top layer outcome being in one of those 1.4% combinations can never be mode 0.

This rule effectively tells us which samples are not mode 0, but it does not tell us which samples are actually mode 0. Given the low accuracy of pair-wise recognition with mode 0, we may just remove the second part of the structure in Fig. 3. Then we still input mode 0 samples. These samples will be recognized as one of the nine fault modes. The distribution is shown in Table XI.

Outcome	mode1	mode2	mode3	mode4	mode5	mode6	mode7	mode11	mode12
Frequency	0.083	0.125	0.105	0.018	0.019	0.010	0.166	0.470	0.011

TABLE XII. MODE 0 SAMPLE DISTRIBUTION USING HIERARCHY W/O PAIR-WISE STEP

Tests also show that for a series of flights in normal condition in the benchmark problem, the recognition will follow the outcome distribution in Table XI. So if we adopt a window for a certain length of samples, we can determine whether a fault has occurred by examining failure mode frequency in this window. The judging thresholds are determined according to each mode's respective frequency in Table XI. To prevent false alarms, the thresholds are set a little higher than the frequencies in Table XI. If the frequency of a mode exceeds its threshold, this failure mode is considered onset. Otherwise the engine is viewed as in normal condition. If more than one mode with frequencies exceeding their respective thresholds, the engine is regarded as in normal condition. This seldom happens, though. Table XII lists the chosen thresholds for each fault mode.

Mode	Threshold frequency
1	0.5
2	0.5
3	0.5
4	0.2
5	0.4
6	0.3
7	0.6
11	0.6
12	0.3

TABLE XIII. CHOSEN THRESHOLDS IN BENCHMARK PROBLEM

F. Deal with Mode 0, a Direct Way

In the previous training and testing processes, the number of involved mode 0 samples are approximately 10 times larger than any other individual failure modes. This leads to the imbalanced learning which could be problematic. A simple way to solve this problem is to down sample mode 0 to 1/10 of its original sample size. For simple implementation, a random down sampling is carried out. The down sampled mode 0 has a much better prediction performance when considered together with other failure modes, as shown in Table XIV:

Unbiased	Prediction Accuracy
0-1	0.95 / 0.91
0-2	0.77 / 0.79
0-3	0.97 / 0.94
0-4	1 / 1
0-5	0.98 / 1
0-6	0.95 / 0.98
0-7	0.78 / 0.75
0-11	0.61 / 0.58
0-12	0.97 / 0.87

TABLE XIV. PREDICTION ACCURACY BETWEEN DOWNSAMPLED MODE 0 AND FAILURE MODES

Without down sampling, the binary classification accuracies are as Table XV:

Imbalanced Dataset	Prediction Accuracy Mode 0 / Failure mode
0-1	1 / 0.78
0-2	1 / 0.39
0-3	1 / 0.49
0-4	1 / 1
0-5	0.98 / 1
0-6	1 / 0.87
0-7	1 / 0.3
0-11	1 / 0.08
0-12	1 / 0.64

TABLE XV. PREDICTION ACCURACY BETWEEN UN-DOWNSAMPLED MODE 0 AND FAILURE MODES

The accuracy of the balance case is 87.9%, which is much better than the imbalanced case of 80.7%. Applying this method to the hierarchy produced the following independent test result, with average accuracy = 73%.

		predict									
		1	2	3	4	5	6	7	8	9	0
	1	0.911765	0	0	0	0	0	0	0.014706	0	0.073529
	2	0.014706	0.676471	0.014706	0	0	0	0.044118	0.029412	0	0.220588
	3	0	0.014925	0.850746	0	0.014925	0	0.014925	0.044776	0	0.059701
	4	0	0	0	1	0	0	0	0	0	0
target	5	0	0	0.015625	0	0.984375	0	0	0	0	0
	6	0	0	0.016129	0	0	0.951613	0.016129	0	0	0.016129
	7	0.044776	0.283582	0.134328	0	0.014925	0	0.313433	0.029851	0.044776	0.134328
	8	0.084507	0.126761	0.028169	0	0	0.014085	0.028169	0.380282	0.014085	0.323944
	9	0	0	0.032787	0	0	0	0.098361	0.016393	0.754098	0.098361
	0	0.039043	0.13602	0.027708	0.001259	0.001259	0	0.071788	0.238035	0.008816	0.476071

TABLE XVI. CONFUSION MATRIX FROM HIERARCHY + MODE 0 DOWNSAMPLING

G. Validation Results

For benchmark engine diagnostic problem validation, we solely use the indirect way of dealing with mode 0. The whole 49,900 data points simulated in ProDiMES are used for validation.

The proposed hierarchical prediction algorithm is denoted as the hierarchical radial based kernel SVM (HrbfSVM). HrbfSVM has the following validation confusion matrix.

		Predicted Modes									
		1	2	3	4	5	6	7	11	12	0
Target Modes	1	0.8	0.02	0.01	0.005	0.002	0.001	0.02	0.05	0.004	0.09
	2	0.02	0.54	0.05	0	0.02	0.003	0.2	0.1	0.003	0.09
	3	0.001	0.03	0.83	0	0.004	0.002	0.01	0.1	0.001	0.02
	4	0	0	0	0.99	0	0	0.001	0.008	0	0
	5	0.004	0.02	0.01	0	0.87	0	0.008	0.04	0	0.04
	6	0	0	0	0	0	1	0	0	0	0
	7	0.02	0.08	0.06	0	0.006	0.002	0.6	0.1	0.009	0.1
	11	0.03	0.04	0.06	0.005	0.01	0.008	0.07	0.6	0.001	0.18
	12	0.02	0.02	0.01	0.003	0.002	0.004	0.07	0.06	0.76	0.06
	0	0.03	0.05	0.08	0.03	0.03	0.04	0.1	0.2	0.02	0.372

TABLE XVII. CONFUSION MATRIX FROM THE DESIGNED ARCHITECTURE

On average, a prediction accuracy of 73.6% has been achieved. It is more than double the brute force accuracy using MLP as shown in Table IV. Some other ML architectures and algorithms are presented in [1] for the same benchmark problem, including Decision Tree (DT), K Nearest Neighbors (KNN), SVM with non-linear kernels (NSVM), hierarchical SVM with kernel sliced inverse regression (HSVMkSIR), and nonlinear-kernel SVM with kernel sliced inverse regression (NSVMkSIR). The comparison between those results and the HrbfSVM are shown in Table XIV.

Used approach	Accuracy	Improvement
HrbfSVM	0.736	/
Brute force MLP	0.354	107.9%
DT	0.3876	89.9%
KNN	0.4495	63.7%
NSVM	0.535	37.6%
HSVMkSIR	0.629	17.1%
NSVMkSIR	0.578	27.3%

TABLE XVIII. COMPARISON WITH OTHER APPROACHES AND METHODS

The comparison result shows that the averaged diagnostic accuracy has improved by 17.1%~107.9% from the HrbfSVM. The most impressive is that HrbfSVM deals with the “difficult modes”—modes 2, 7 and 11, effectively. That is where the HrbfSVM performance gains are over the other methods. We can expect an even better performance if we combine the direct way to deal with mode 0 with the indirect way in the future.

CHAPTER 4

CONCLUSIONS

The study deals with NASA's aircraft engine diagnostic benchmark problem by a hierarchical ML approach. Firstly the brute force MLP generated a result with accuracy of 35.4%. To improve the accuracy, we used SOM to gain intuitive insight on the relationships among various engine modes. The modes which are likely to entangle with each other are separated into different groups. A total of four groups are established. The modes in the same group can be well distinguished. Preliminary analysis has led to the creation of a hierarchical failure recognition structure. SVM, as the workhorse of the proposed hierarchical procedures, with radial based kernel outperformed other SVM and MLP algorithms with its high recognition accuracy. It is used together with additional pair-wise mechanisms to achieve a high performance hierarchical approach to the engine diagnostic problem. Finally, to deal with the most difficult normal mode 0, we made use of sample outputs from the normal mode at different layers of the hierarchy and created an indirect method to recognize mode 0. A direct method is also developed to recognize mode 0 by down sampling, which was shown effective.

In summary, the designed HrbfSVM structure has a better recognition accuracy on this benchmark problem than brute force MLP and other methods. It retains its improved accuracy by enhancing recognition of those "difficult modes", such as modes 2, 7, 11. Nonetheless, we still see room for further improvement. We have not yet combined the direct way and indirect way to recognize mode 0, which has potential to boost the classification accuracy.

If more powerful working algorithms than SVM could be found to be used as the classification engine inside the hierarchical structure, we expect improved accuracy on modes such as mode 2 and mode 7.

In fact, in some complicated image processing problems, “deep learning” network based on back-propagation algorithm has demonstrated high performance in complex classification problems [13]. The specifically designed multi-layer networks have neurons that can optimally weigh the input signal features. Future work may include applying deep structures to the engine diagnostic problem.

The results obtained so far are very encouraging, which can potentially make ML a practical method in aircraft engine diagnostics. Together with the conventional model-based approach, it is likely to improve the safety and security of engine operation. Meanwhile, those ML ideas in dealing with this engine problem will be a good reference in treating other complicated physical problems.

REFERENCES

[1] Link C. Jaw, Yuh-Jye Lee. (June 16-20, 2014) "*Engine diagnostics in the eyes of machine learning*," Proceedings of the 2014 ASME Turbo Expo, Dusseldorf, Germany.

[2] Simon, D. L. (2010). Propulsion Diagnostic Method Evaluation Strategy (ProDiMES) User's Guide.

[3] Volponi, A. J., (2003). "Foundations of Gas Path Analysis (Part I and II)," Gas Turbine Condition Monitoring and Fault Diagnosis (von Karman Institute Lecture Series No. 2003-01), von Karman Institute, Rhode-Saint-Gene`se, Belgium.

[4] Kobayashi, T., Simon, D. L., and Litt, J. S. (2005). "Application of a Constant Gain Extended Kalman Filter for In-Flight Estimation of Aircraft Engine Performance Parameters," ASME Paper No. GT2005-68494.

[5] Ioannou, P. A., and Sun, J., 1996, Robust Adaptive Control, Prentice-Hall, Englewood Cliffs, NJ.

[6] Simon, D. L., Borguet, S., Léonard, O., & Zhang, X. F. (2013). Propulsion Diagnostic Method Evaluation Strategy (ProDiMES): Public Benchmarking Results.

[7] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning,20(3), 273-297.

[8] Ruck, D. W., Rogers, S. K., Kabrisky, M., Oxley, M. E., & Suter, B. W. (1990). The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *Neural Networks, IEEE Transactions on, 1*(4), 296-298.

[9] Hecht-Nielsen, R. (1989, June). Theory of the backpropagation neural network. In Neural Networks, 1989. IJCNN., International Joint Conference on (pp. 593-605). IEEE.

[10] Lin, Y. (2002). Support vector machines and the Bayes rule in classification. Data Mining and Knowledge Discovery, 6(3), 259-275.

[11] Kohonen, T. (1990). The self-organizing map. Proceedings of the IEEE, 78(9), 1464-1480

[12] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.

[13] Deng, L., Hinton, G., & Kingsbury, B. (2013, May). New types of deep neural network learning for speech recognition and related applications: An overview. InAcoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on (pp. 8599-8603). IEEE.