Application of Random Forest to Aircraft Engine Fault Diagnosis

Weizhong Yan
Computing & Decision Sciences
GE Global Research Center
Niskayuna, New York 12309
yan@crd.ge.com

Abstract – Aircraft engine fault diagnosis plays a critical role in modern, cost-effective condition-based maintenance strategy in aircraft industry. Due to several inherent characteristics associated with aircraft engines, accurately diagnosing aircraft engine faults is a challenging classification problem. As a result, aircraft engine fault diagnosis has been an active research topic attracting tremendous research interests in machine learning community. In this paper, random forest classifier, a recently emerged machine learning technique, is applied to aircraft engine fault diagnosis in an attempt to achieve more accurate and reliable classification performance. Our primary objective is to evaluate effectiveness of random forest classifier on aircraft engine fault diagnosis. By designing a real-world aircraft engine fault diagnostic system, this paper investigates design details of random forest classifier and evaluates its performance. In this paper, we also make some efforts on investigating strategies for improving random forest performance specifically for aircraft engine fault diagnosis problem

Keywords: Aircraft engine; classification; diagnosis; performance evaluation; random forests;

I. INTRODUCTION

Aircraft engine fault diagnosis (AEFD) is to detect, isolate, and assess malfunctions/faults and failures of aircraft engine system and its major components. AEFD is the enabler of the modern condition-based maintenance strategy for aircraft engines. The benefits of AEFD may include [YAN, 03]: a) increasing flight safety by early detection of engine malfunctions; b) preventing costly component damage and/or catastrophic failure; c) reducing turnaround time by providing maintenance personnel with information on fault locations (by reducing time for manual fault isolation); d) reducing delays and cancellations by facilitating more on-wing maintenance; and e) increasing engine on-wing time by minimizing scheduled and unscheduled engine removal.

Reliably diagnosing engine faults is a difficult task due to the following intrinsic characteristics of aircraft engines:

1. There exist engine initial quality variations. Engine initial quality variation is the result of the variation of fabrication and assembly. Engine initial quality varies from engine to engine even within the same engine models.
2. Engine quality deteriorates over time. Engine deterioration has many causes, such as, tip clearance changes in the rotating components, seal wear, blade fouling, blade erosion, blade warping, foreign object damage, actuator wear, and blocked fuel nozzles [MKG, 91]. Engine deterioration results in engine performance parameter changes over time (time-varying).
3. Aircraft engines are operated at different points in flight regime. When an aircraft travels from one point to another in flight regime, the engine performance parameters change following the principles of thermodynamics and aerodynamics.

The aforementioned importance and the challenges associated with aircraft engine fault diagnosis have motivated tremendous amount of research efforts on designing a reliable and cost-effective AEFD system.

Over years numerous techniques have been explored for engine fault diagnosis. These techniques range from the traditional gas path analysis (GPA) to modern soft computing technologies, such as, neural networks, fuzzy logic, and evolutional computation. Li's work [LIY, 02] provides a good review of different techniques adopted for engine fault diagnosis. For complex classification problems like AEFD, it is generally accepted that it is difficult to design a single classifier to achieve the desired performance. For such complex problems, both theoretical and empirical studies have proved that using an ensemble of individual classifiers (classifier ensemble) is more effective in achieving improved classification performance. In classifier ensemble, outputs of individual classifiers are combined to arrive at the final classification decision. Such combination strategically utilizes the information provided by the constituent classifiers, thus achieves higher classification performance. Classifier ensemble has been widely used for various applications. For aircraft engine fault diagnosis, classifier ensemble as a classification technique has been explored in several research work, for example, [GKS, 00], [VBT, 03], and [RKS, 01], where potentials of using information fusion to combine information from multiple sources (sensors, models, etc) are investigated. Ogaji and Singh [OSS, 03] also investigated the potential of using a nested neural system to divide a complex problem into a number of

smaller (and hence simpler) problems. More recently, in our previous work [YLG, 06], we proposed fusing three different types of classifiers for improving classification performance and demonstrated the effectiveness of classifier fusion for aircraft engine fault diagnosis. Random forests [BRM, 01] are a recently emerged machine-learning technique. Random forests use an ensemble of trees as their constituent models. While each of the constituent trees is trained on a bootstrap sample of the data, the final outputs are the aggregation of the outputs of the individual classifiers. Random forests are good for both classification and prediction and have shown superior performance over other machine learning techniques for a variety of applications. In this paper, we apply random forest classifier to aircraft engine fault diagnosis. Our primary objective is to evaluate effectiveness of random forest on aircraft engine fault diagnosis. We use a real AEFD as an example to illustrate design details of random forest classifier. We also make some efforts on improving random forest performance specifically for the AEFD problem.

The rest of this paper is organized as follows. Section II describes the aircraft engine fault diagnosis problem. Design details of random forest classifier for aircraft engine fault diagnosis are given in Section III. Results and discussion are given in Section IV. Section V concludes the paper.

II. PROBLEM DESCRIPTION – AIRCRAFT ENGINE FAULT DIAGNOSIS

Aircraft engine fault diagnosis (AEFD) is to detect and isolate engine faults/malfunctions based on information such as sensed engine performance parameters and flight regime information. AEFD is a classical classification problem. Let's assume the sensed engine performance parameters and flight regime information at time t are represented by a vector $\mathbf{x}_t = [m_1, m_2, ...m_k]$. Also assume engine has $1+p$ possible conditions/health states (a normal condition plus p different faulty conditions), i.e., $\mathbf{C} = [C_1, C_2, ...C_{p+1}]$. AEFD is then to infer engine instantaneous condition by assigning $\mathbf{x}_t$ to one of the $1+p$ health states, $\mathbf{C}$. Mathematically, AEFD performs a nonlinear mapping from an input vector $\mathbf{x}_t$ to a condition space $\mathbf{C}$, i.e., $F: \mathbf{x}_t \in \Re^k \rightarrow \mathbf{C} \in \Re^{p+1}$.

An aircraft engine is a complex system and there are potentially a number of possible faults occurring in different sub-systems or components. In this paper, we limit our study to engine gas path faults. More specifically, we design an engine diagnostic system to detect and diagnose the following 6 engine gas path faults:

1. Fan fault (FAN) – Fan blade damage, typically occurring due to bird strikes or other foreign object damage (FOD) during takeoff.
2. Compressor fault (CMP) – Compressor blade damage, compressor contamination, or abnormal operation.
3. High Pressure Turbine fault (HPT)– Typically a partial loss of one or more blades, most common during high power conditions.
4. Low Pressure Turbine fault (LPT) – Typically a partial loss of one or more blades, most common during high power conditions.
5. Customer Discharge Pressure fault (CDP) – Leakage in excess of the desired bleed level commanded by the aircraft and communicated to the Full Authority Digital Electronic Control (FADEC).
6. Variable Bleed Valve fault (VBV) – Variable bleed valve doors not closing according to FADEC command.

The 9 sensed engine performance parameters that are used as inputs to the diagnostic system are listed in Table 1. These are the typical engine performance parameters that are popularly used for engine gas path fault diagnosis. In addition, we construct 2 extra parameters based on the domain experts' experience. They are the ratio of compressor exit and inlet pressures and the ratio of fuel flow rate and compressor exit pressure. Including the two ratio parameters, we have a total of 11 variables (see Table 1) that are used as inputs to the diagnostic system.

Table 1: The 11 variables used as inputs for AEFD

No.	Variable Description
1	Fuel flow rate
2	Core speed
3	Compressor inlet pressure
4	Compressor Exit pressure
5	Fan tip exit pressure
6	Compressor inlet temperature
7	Compressor exit temperature
9	HP turbine exit temperature
9	LP turbine exit temperature
10	No.4/No.3
11	No.1/No.4

As described above, AEFD is to infer engine condition based on monitoring the changes of the sensed engine performance parameters. Such changes are usually subtle, which makes fault diagnosis a difficult classification problem. Furthermore, engine performance parameters also change as aircraft operating at different points in flight regime even though the engine condition is unchanged. The flight regime induced changes of engine performance parameters mask the changes induced by engine faults, thus make the engine fault diagnosis even more difficult. Addressing the flight regime issue has always been an important effort in designing engine fault

diagnostic systems. In this paper, the flight regime issue is addressed by flight regime mapping as proposed by the author in his previous work [YAN, 03]. It essentially maps the engine performance parameters from the actual flight regime to sea level static equivalent, which compensates for flight regime induced changes, thus accentuates the engine condition related changes of engine performance parameters. In this study, for designing aircraft engine fault diagnosis, we perform flight regime mapping on engine performance data first and design random forest classifier based on the mapped data.

III. DESIGN OF RANDOM FOREST CLASSIFIER FOR AIRCRAFT ENGINE FAULT DIAGNOSIS

A. Random forests

Random forest (RF) is one of recently emerged ensemble learning methods. Since first introduced by Leo Breiman [BRM, 01] in 2001, it has been popularly used in various applications (both classification and prediction) due to its superior performance and its relative simplicity in design. Random forest is an ensemble of unpruned trees. Final output of random forest is the aggregation of the outputs of the individual trees. To minimize the correlation among the individual trees while maintaining the strength of the individual trees, thus to improve the classification accuracy, each of the individual trees are constructed with two sources of randomness injected. First, each tree is trained on a dataset that is a bootstrap replica (randomly drawn, with replacement) of the training data. Second, at each node of each tree, the algorithm determines the best split based on a small subset of variables that is randomly picked from the complete variable set. Also, each tree is fully grown in order to achieve low bias and high variance of the tree outputs.

Features of random forest include [BRM, 01]: 1) it performs favorably over most of other classifiers; 2) it is computationally efficient on large datasets; 3) it can handle well large number of input variables; 4) it is relatively robust against over-fitting; 5) it can deal with missing data, and 6) it is relatively robust to outliers and noise.

B. Standard design of random forests

Designing a random forest classifier is relatively simple since only a few of design parameters need to be tuned to achieve optimal performance while other parameters can be set as default values. One tunable parameter is *mtry*, the number of features that are randomly selected for determining the best split at each node. *mtry* is the most sensitive parameter for most of problems. Empirical studies, e.g., [SLT, 03], have shown that for classification, *mtry* is directly related to total number of features by relation of $mtry = \sqrt{p}$, and that for most real-world problems, setting *mtry* to be square root of total number of features can yield a reasonable good (if not the best) performance. Other parameters that a designer usually likes to tune include the number of trees (*ntree*) and the minimum node size (*nnode*) defined as the size of nodes below which node splitting stops. Random forest performance is generally not too sensitive to these parameters. A good practice is to specify the number of trees (*ntree*) as large as possible (from a few hundreds to a few thousands) while keeping the minimum node size (*nnode*) to a small number (e.g, 1, 2, or 3). For aircraft engine fault diagnosis concerned in this paper, our efforts on designing random forest mainly focus on determining the optimal parameters through experimental studies (see Section IV).

Like in design of any other classifiers, an important design endeavor for random forest classifier is to estimate generalization error based on the given dataset. Random forest has an internal mechanism that does unbiased estimate of the generalization error as the forest building progresses. In forest building process, each tree is trained on bootstrap samples of the training set. On average, about one-third of examples are not in the bootstrap set, which are called by Breiman the "out-of-bag" (OOB) samples. These OOB samples can be used for estimating the generalization error of random forest classifier. According to Breiman [BRM, 01], the OOB estimate is as accurate as using a holdout test set, which removes the need for a set aside test set. As part of the design efforts, in this paper we compare the OOB error estimate with cross-validation error estimate. We are curious to know if there is any difference between the two error estimation methods in aircraft engine fault diagnosis.

C. Some efforts on improving random forest performance for the aircraft engine fault diagnosis problem

Parameter optimization and error estimation described in the previous section can be considered as the standard design for random forest classifiers. In this section, we attempt to explore some novel design strategies specifically for improving classification performance of AEFD. Our efforts will be on two areas. One is on multi-class decomposition, and another is on feature linear transformation. Details of the two efforts are described as follows.

Multi-class classification: The AEFD system concerned in this paper involves diagnosing 7 different engine conditions (1 normal condition and 6 different types of faults). That is, the classifier to be designed has 7 different outputs, which is typically referred as a multi-class classification problem [FJH, 96]. Except for a few classifiers (e.g., Support Vector Machine) that inherently can only handle binary classification, most of classifiers can work for both binary and multi-class classification problems. However, studies have shown that performance can generally be improved if the multi-class problem is

decomposed into a series of binary classification ones [HTT, 98]. Random forest classifiers are inherently flexible for both binary and multi-class classification problems and have been used for both problems by many researchers. In this paper, as part of our efforts on performance improvement, we like to answer the question: would multi-class decomposition help improve random forest classification performance? There are several ways to decompose a multi-class classification problem into a series of binary classification ones in order to achieve better performance [DTB, 95]. In this study, we take "one-vs-other" method since it is the simplest one. We decompose the AEFD classification problem that originally has 7 target classes into 7 binary classifiers. Each of the seven classifiers is trained to distinguish between patterns belonging to a class (C_i) and its complement ($\overline{C_i}$) (combining all data not belonging to class C_i). To classify an unknown input x, the outputs of the 7 binary classifiers form a 7-component vector, $O(x) = [O_1(x), O_2(x), ..O_7(x)]$ and the MAX-rule can be used to arrive at the final classification decision for the input x, i.e., $\varpi(x) = \arg\max_i O_i(x)$.

Feature linear transformation: In standard random forests, at each node features are evaluated using Gini index as the evaluation measure. Gini index evaluation is computationally fast, but it assumes the conditional independence of features and evaluates each feature independently, which does not take into account feature interaction, an important part in feature selection. Realizing this issue, Robnil-Sikonja [MRS, 04] proposed using multiple feature evaluation measures to replace the Gini index measure and showed improved performance. Their approach, however, requires modifications of the standard random forests package. In this paper, we address the issue by a novel approach, that is, to transform the features from the original space to a new space and, in the transformed space, the features are uncorrelated so that the Gini index evaluation in the standard random forests can be effective. The feature transformation simply serves as a preprocessing step before applying the standard random forests. There are several linear transformation methods available, including principal component analysis (PCA) [FUK, 90], linear discriminant analysis (LDA) [FUK, 90], and independent component analysis (ICA) [LEE, 98]. All these linear feature transformation have the same format. They differ only in the criterion function used in determining the transformation matrix or coefficients. We use principal component analysis (PCA) for feature transformation to obtain uncorrelated features for design of the random forest classifier. PCA transforms a *n*-dimensional data to a n-dimensional space whose basis vectors are defined by orthonormal eigenvectors of the covariance matrix of the original data. Such transformation is variance preservation, i.e., maximally preserve the variance in the transformed space. Most importantly, PCA ensures the transformed variables are uncorrelated in the transformed space. PCA is typically used as feature reduction where a small number of top principal components are selected. In this paper, we use all of the principal components, which results in the number of variables unchanged before and after PCA transformation.

IV. RESULTS AND DISCUSSION

A. The design data

The design data used in this paper is the simulated engine data. The simulation model is a full-bloom nonlinear thermodynamic model of GEAE CFM56-7B engine used for commercial aircrafts, which allows for simulating different levels of faults for each of the major components, such as, fan, low and high pressure compressors, low and high pressure turbines. The simulated data contains all sensed engine performance parameters for both normal condition engines and engines with 6 gas path faults, totaling of 7 classes. Each class has 2805 data points, covering entire flight regime. The simulated data also includes engine-to-engine variation, sensor bias, and 5 levels of engine deterioration. Total number of data points is 19635.

B. Performance measures

To evaluate classification performance of different designs, three performance indices (overall error rate, false positive rate, and false negative rate) calculated from confusion matrices are used. For multi-class classification, the three performance indices are defined as follows.

Let CM be the confusion matrix and each cell $CM(i, j)$, i, j=1,...C represents the number of times class i cases are misclassified as class j, where C is the number of classes. And assume Class 1 represents normal (fault-free) engine condition.

Overall error rate:

$$OER = 1 - \sum_{i=1}^{C} CM(i,i) \Big/ \sum_{i,j=1}^{C} CM(i,j) \qquad (1)$$

False positive rate:

$$FPR = \sum_{j=2}^{C} CM(1,j) \Big/ \sum_{j=1}^{C} CM(1,j) \qquad (2)$$

False negative rate:

$$FNR = \sum_{i=2}^{C} CM(i,1) \Big/ \sum_{i=2,j=1}^{C} CM(i,j) \qquad (3)$$

C. Results for standard designs:

All results of random forests shown in this paper are obtained by running Matlab® version of random forests implemented by T. Wang at Merck & CO. For standard design of random forest, we simply run experiments for determining the optimal design parameters, namely, *mtry*, *ntree*, and *nnode*. To find the optimal number of features (*mtry*), we run experiments by varying *mtry* from 1 to 9, i.e., *mtry={1,2,3,4,5,6,7,8,9},* while maintaining number of trees (*ntree*) to be 500 and the minimum node size (*nnode*) to be 1. Figure 1 shows the relationships between the three performance indices and different *mtry* values. For graphical presentation convenience, each of the three performance indices is normalized by its performance index value corresponding to *mtry* of 3. Figure 1 shows that random forest design gives the best performance (smallest values for all three errors) when number of features (*mtry*) equals to 3. Given that the total number of features we are using for random forest is 11, the recommended *mtry* is also 3, which is equal to the square root of 11. Hence, for aircraft engine fault diagnosis concerned here, taking number of features as the recommended one can be an optimal design for random forest.

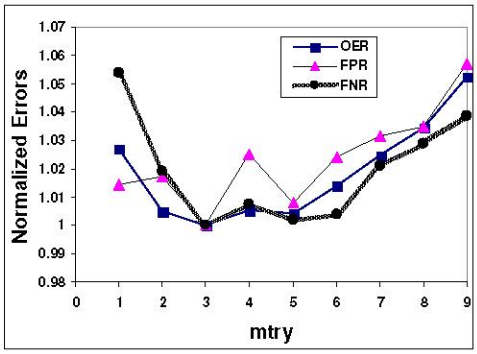

Figure 1: Normalized errors vs. mtry

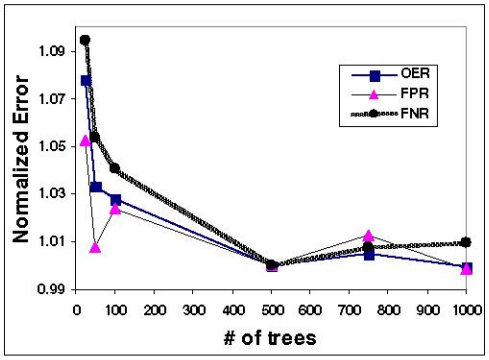

Figure 2: Normalized errors vs. ntree

We run another experiment for determining optimal number of tress. In this experiment we keep *mtry* to be 3 (the optimal # determined above) and the minimum node size *(nnode)* to be 1, while varying number of trees from 20 to 1000, i.e. *ntree*={20, 50, 100, 500, 750, 1000}. Figure 2 shows how the three errors (performance indices) change as number of trees increases. Once again, the three errors are normalized with respect to the errors corresponding to the design with *ntree* of 500. We can see from figure 2 that the three errors reduce as number of trees increases. The three errors reach the minimal at 500 trees and using tree size larger than 500 does not lead to a further reduction for the three errors. Therefore, setting tree size to 500 may be the best design of the random forest for AEFD.

We also did similar experiments for the minimum node size (*nnode*). We obtain the three performance indices (errors) for the designs with different minimum node sizes, i.e., *nnode*={1,2,3,4,5,10}, while *mtry* and *ntree* are fixed at 3 and 500, respectively, for all designs. Figure 3 shows the normalized errors (normalized with respect to the errors corresponding to *nnode* of 1). As can be seen from Figure 3, unlike in the other 2 experiments above, the three errors reach their own minimum at different design (*nnode*). While *nnode* of 1 gives the smallest overall error and false negative rates, in terms of false positive error, using *nnode* larger than 1 (e.g. 4) can be better. Determining *nnode* requires knowing the error costs for the three errors. Since error costs associated with the three errors are not available, we have to assume they are equal, which means *nnode* of 1 is the best design for AEFD random forest classifier concerned here.

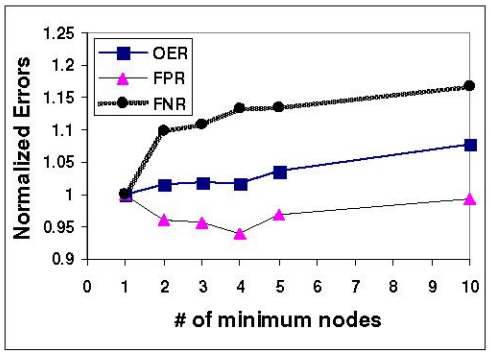

Figure 3: Normalized errors vs. nnode

In summary, for aircraft engine fault diagnosis concerned in the paper, our experimental studies indicate that *mtry*=3, *ntree*=500, and *nnode*=1 can be a good design configuration for the random forest classifier.

It is important to note that the above results of the 3 experiment studies are all based on 5-fold cross-validation error estimation. To do that, the entire data set was divided into 5 mutually exclusive subsets of equal size.

The forest is then trained with the 4 subsets and tested on the remaining set. This is repeated 5 times, each with different test set, and the final errors are the average of the 5 runs of the test errors and are shown in Figures 1 – 3 above.

We also evaluate the performance using OOB error estimation. For that purpose, we set design parameters of random forest as the optimal ones obtained above, i.e., *mtry*=3, *ntree*=500, and *nnode*=1, and we compare OOB errors and cross-validation errors for that parameter setting only. We run random forest 5 times, each with different random seeds. The three performance indices obtained by OOB estimation are compared in Table 2 to those from 5-fold cross validation for all 5 runs. Mean and standard deviation of each performance index are also calculated for each of the estimation methods. Generally, OOB estimation not only gives smaller mean errors than cross-validation does, but also shows smaller variation of the errors. The reason for that may be that, unlike in cross-validation where test cases over 5 runs are mutually exclusive, some OOB test cases may appear in all 5 runs, which reduces variation. To find out if the results of the two methods are statistically different, we use Wilcoxon rank sum test [GJD, 85]. While the Wilcoxon rank sum test is generally considered as a nonparametric alternative to the conventional two-sample *t*-test, it has one advantage that it does not require the assumption that the differences between the two samples are normally distributed. The Wilcoxon rank sum test results indicate that the difference of the overall error rates (OER) between CV and OOB is significant at 15% level, while for other 2 performance indices (FPR and FNR) the differences between the 2 error estimate methods (CV and OOB) are not statistically significant.

Table 2: Comparison between CV and OOB error estimation methods (CV=cross validation error estimation and OOB=out-of-bag error estimation)

run	OER		FPR		FNR	
	CV	OOB	CV	OOB	CV	OOB
1	16.81	15.80	21.57	21.50	5.50	5.10
2	16.40	15.81	21.03	21.64	5.73	5.16
3	16.09	15.81	23.71	21.82	4.69	5.19
4	15.76	15.86	22.64	22.25	5.08	5.06
5	16.12	15.91	23.71	21.89	4.87	5.11
mean	*16.24*	*15.84*	*22.53*	*21.82*	*5.17*	*5.12*
std	*0.393*	*0.047*	*1.222*	*0.285*	*0.433*	*0.051*

D. Results for performance improvement efforts

For the convenience of comparing our results to those from other classifiers, we use 5-fold cross-validation to estimate classification errors in this section.

Multi-class classification: For each cross-validation run, we build 7 forests, each of which is a binary classifier to distinguish class (C_i) and its complement ($\overline{C_i}$) (combining all data not belonging to class C_i). As a result, there are 6 times as many cases in $\overline{C_i}$ class as in C_i class, an imbalanced data set for each forest. Random forest, however, is inherently weak in dealing with imbalanced data since bootstrap samples are used in forest building process. To overcome this, we over-sample the C_i class to obtain a balanced training set for training each forest. The parameters of each of the forests are set to be the optimal ones determined from the previous subsection, namely, *mtry*=3, *ntree*=500, and *nnode*=1. Table 3 shows the confusion matrix of the multi-class decomposition design. It is the sum of matrices of 5 cross-validation runs. Last column of Table 3 shows per-class classification rate in percentage. The three performance indices are 16.23%, 23.46%, and 5.94% for overall error, false positive, and false negative rates, respectively. The three error rates are also recorded in Table 5 and will be compared to those from other designs.

Table 3: Confusion matrix for multi-class decomposition

	NF	Fan	Comp	HPT	LPT	CDP	VBV	pcAC
NF	2147	0	105	201	191	55	106	76.54
Fan	0	2803	2	0	0	0	0	99.93
Comp	92	1	2522	45	91	48	6	89.91
HPT	236	0	117	2168	253	13	18	77.29
LPT	263	0	111	394	1785	210	42	63.64
CDP	186	0	76	9	188	2320	26	82.71
VBV	54	0	3	2	22	20	2704	96.40

Linear feature transformation: Prior to constructing forest, PCA is performed on the dataset of each cross-validation run. All 11 principal components are used as the inputs to forest. Once again, the optimal parameters (*mtry*=3, *ntree*=500, and *nnode*=1) obtained from previous subsection are chosen for the forests. Confusion matrix of this design is shown in Table 4, with last column being the per-class classification rates in percentage. The three error rates (overall error, false positive, and false negative) are 15.74%, 22.17%, and 5.01%, respectively, for this design.

Table 4: Confusion matrix for linear feature transformation

	NF	Fan	Comp	HPT	LPT	CDP	VBV	pcAC
NF	2183	2	121	155	181	51	112	77.83
Fan	0	2800	5	0	0	0	0	99.82
Comp	104	11	2512	45	61	62	10	89.55
HPT	192	1	88	2223	265	15	21	79.25
LPT	248	0	88	360	1880	172	57	67.02
CDP	214	0	82	4	164	2308	33	82.28
VBV	86	0	5	8	43	24	2639	94.08

For performance comparison convenience, we present the three error rates (performance indices) from all designs into a single table, Table 5. Design 1 represents the standard RF design with cross-validation error estimations. Designs 2 and 3 designated as RF_mc and RF_pc in Table 5 are for multi-class decomposition and feature linear transformation designs, respectively, presented in this subsection. It is noted that all 3 RF designs use the same forest configuration, i.e., *mtry*=3, *ntree*=500, and *nnode*=1 and the same cross-validation error estimation method. In addition, we also include in Table 5 the results from other classifiers designed in our previous work [YAN, 06], which include 3 individual classifiers and fusion. The 3 individual classifiers were neural network (NN), support vector machine (SVM), and decision tree (CART). The error rates of the 3 classifiers are shown in Designs 4 through 6, while the error rates for fusion are shown in Design 7 in Table 5.

Table 5: Performance comparison for different designs

Design		OER	FPR	FNR
1	RF	16.24	22.53	5.18
2	RF_mc	16.23	23.46	4.94
3	RF_pc	15.74	22.17	5.01
4	NN	16.35	23.74	5.19
5	SVM	18.42	31.84	5.00
6	CART	25.47	36.79	6.97
7	Fusion	14.49	21.21	4.63

Comparing RF_mc and RF (Designs 2 and 1) in Table 5, we can see that decomposing multi-class classification into multiple binary classifiers (Design RF_mc in Table 5) shows almost no improvement on the three performance indices (errors) for the random forest classifier for AEFD problem concerned in this paper. This result is not intuitive and is different from what has been observed from other classifiers. A rational explanation is required. More work on this topic needs to be performed before a concrete conclusion can be made.

Applying linear feature transformation (PCA) to data prior to training forests slightly improves the classification performance, which can be seen by comparing Design RF_pc to design RF in Table 5. This result is expected since PCA uncorrelates features and makes random forests more effective.

Among the three individual classifiers (Designs 4, 5, and 6 in Table 5) designed in our previous work, neural network (NN) has the best performance. Random forest, even in a standard design (Design 1), performs as well as the neural network (NN) and greatly outperforms other classifiers. However, random forest is computationally much efficient than neural networks. It is worthy noting that the NN classifier was designed using multi-class decomposition scheme discussed in Section III, which required longer training time.

Fusion (Design 7 in Table 5) of the three individual classifiers conducted in our previous work not only show performance improvement over the best constituent classifier (NN here), but also performs better than random forests of all designs studied in this paper. Both fusion and random forests are ensemble learning methods. Results obtained here for the AEFD problem seem to indicate that fusion is more effective than random forests on improving classification performance. One explanation of this can be that different types of classifiers used in fusion design provide higher degree of diversity among classifier outputs, thus more complementary information, which helps classification performance. Random forests, on the other hand, tend to use a large number of trees that are individually weak and with certain diversity obtained from the randomness injected. However, random forests are computationally efficient and have their own unique features as listed in Subsection III-A of this paper.

V. CONCLUSIONS

Driven by great desire of improved AEFD performance, in this paper we explore random forest, a recently emerged machine learning technique, for aircraft engine fault diagnosis. Our primary objective is to find out how effective of random forests on aircraft engine fault diagnosis and investigate some design strategies as well.

We first design random forest by experimentally determining optimal configurations of random forest. We also examined/investigated cross-validation error estimation and compared it against the RF built-in OOB estimate method. Additionally, we made some efforts on improving RF performance, specifically for the AEFD problem. Realizing that AEFD is a multi-class classification problem, we investigated multi-class decomposition, a technique that has proved to be effective on classification performance for other classifiers, for designing forest even though RF is inherently flexible for both binary and multi-classification problems. More importantly, we propose a novel approach to address the issue that standard RF does not take into account feature interaction in feature evaluation. Our approach essentially un-correlates the features by linear feature transformation (PCA).

Our design results show that the AEFD problem concerned in this paper is indeed a complex classification problem. For such complex problem, random forest, even a standard design, outperforms conventional decision tree classifier (CART) and support vector machine (SVM) and can achieve classification performance comparable to neural network (NN), the best individual classifier considered. Surprisingly, our two performance improvement efforts do not improve classification performance as significantly as we expected for the AEFD

problem. Another interesting observation is that RF seems not as effective as fusion for the AEFD problem, even though both are ensemble learning methods. Our studies conducted here, especially on the two performance improving efforts, are preliminary. Further investigations on random forest design and performance improvement efforts are required before we can draw any concrete conclusions. A natural extension of our current work is to fuse RF outputs with other 3 classifiers (NN, SVM, and CART) to see if a better performance (better than fusion of the 3 classifiers only) can be achieved.

ACKNOWLEDGMENTS

The author is grateful to his colleague, Dr. Neil Eklund, for insightful discussion on design of random forests.

REFERENCES

[YAN, 03] Yan, W.Z. (2003), "Diagnostic system configuration optimization and its application to aircraft engine fault diagnosis", PhD Dissertation, Department of Mechanical Engineering, Rensselaer Polytechnic Institute, Troy, New York

[MKG, 91] Merrington, G., Kwon, O-K, Goodwin, G., and Carlsson, B. (1991), "Fault detection and diagnosis in gas turbines", ASME Journal of Engineering for Gas Turbines and Power, Vol. 113, No. 4.

[LIY, 02] Li, Y.G. (2002), "Performance-analysis-based gas turbine diagnostics: A review", I. Mech.E. Proceedings of the Institute of Mechanical Engineering, Part A: Journal of Power and Energy, Vol. 216, No. A5.

[GKS, 00] Goebel, K., Kirok, M. and Sutherland, H. (2000), "Diagnostic information fusion: requirements flowdown and interface issues", Proceedings of the IEEE 2000 Aerospace Conference – Advanced Reasoner and Information Fusion Techniques, vol. 6, pp.155-162.

[VBL, 03] Volponi, A., Brotherton T., Luppold, R., and Simon, D.L., (2003), "Development of an information fusion system for engine diagnostics and health management", NASA TM—2004-212924, ARL–TR–3127, JANNAF Conference, 2003

[RKS, 01] Roemer, M., Kacprzynski, G., and Schoeller, M. (2001), "Improved diagnostic and prognostic assessment using health management information fusion", Proceedings of AUTOTESTCON, IEEE Systems Readiness Technology Conference, August 20-23, 2002, pp365-377.

[OSS, 03] Ogaji, S. and Singh, R. (2003), "Gas path fault diagnosis framework for a three-shaft gas turbine", Proceedings of the Institution of Mechanical Engineers, Vol. 217, Part A, pp. 149-157.

[BRM, 01] Breiman, L., (2001), "Random forests", Machine Learning, 45:5-32

[MRS, 04] Robnik-Sikonja, M. (2004), "Improving random forests", Proceedings of ECML, in J. F. Boulicaut et al. (eds): Machine Learning

[FUK, 90] Fukunaga, K. (1990), "*Introduction to Statistical Pattern Recognition*", 2nd Edition, Academic Press, Inc.

[FJH, 96] Friedman, J. H. (1996), "Another approach to polychotomous classification", Technical Report, Department of Statistics, Stanford University

[HTT, 98] Hastie, T. & Tibshirani, R. (1998), "Classification by pairwise coupling", The Annals of Statistics, Vol. 26, Issue 2, pp451-471.

[DTK, 95] Dietterich, T.G. & Bakiri, G. (1995), "Solving multiclass learning problems via error-correcting output codes", Journal of Artificial Intelligence Research, 2, pp263-286.

[GJD, 85] Gibbons, J. D., "*Nonparametric Statistical Inference*", 2nd edition, M. Dekker, 1985.

[SLT, 03] Svetnik, V., Liaw, A., Tong, C., Culberson, J.C., Sheridan, R.P., Feuston, B.P. (2003), "QSAR modelling using random forest, an ensemble learning tool for regression and classification", Journal of Chem. Inf. Computational Science, Vol. 43, pp1947-58

[LEE, 98] Lee, T.W. (1998), "*Independent Components Analysis: Theory and Applications*", Kluwer Academic Publishers, Boston, 1998.

Dr. Weizhong Yan is a Research Engineer in the Computing and Decision Sciences of GE Global Research Center, Niskayuna, New York. He has conducted applied research in the areas of pattern recognition, classification, data modeling, and soft computing. His specialties include application of soft computing technologies to condition monitoring, diagnosis, and prognosis of gas turbine engines and other mechanical systems. Prior to joining GE Global Research Center, Dr. Yan was a Lead Test Engineer at GE Power Systems, Schenectady, NY. In addition, Dr. Yan had 15 years working experience in structural engineering. Dr. Yan is an Adjunct Professor of the Mechanical Engineering Department at Rensselaer Polytechnic Institute, Troy, New York, where he teaches Control and Modeling. Dr. Yan was a Professional Engineer registered in the states of NY and MA.