

Fusing diverse monitoring algorithms for robust change detection

Kai F. Goebel*, Xiao Hu, Neil H. W. Eklund, Weizhong Yan
GE Global Research, One Research Circle, Niskayuna, NY, USA 12309

ABSTRACT

Change detection is an important task in remotely monitoring and diagnosing equipment and other processes. Specifically, early detection of differences that indicate abnormal conditions has the promise to provide considerable savings in averting secondary damage and preventing system outage. Of course, accurate early detection has to be balanced against the successful rejection of false positive alarms. In noisy environments, such as aircraft engine monitoring, this proves to be a difficult undertaking for any one algorithm. In this paper, we investigate the performance improvement that can be gained by aggregating the information from a set of diverse change detection algorithms. Specifically, we examine a set of change detectors that utilize a variety of different techniques such as neural nets, random forests, and support vector machines. The different techniques have different detection sensitivities and different false positive rates. For fusion, we consider the Dempster regression technique that operates well for time series as well as averaging schemes, and a meta-classifiers. We provide results using illustrative examples from aircraft engine monitoring.

Keywords: Change detection, diagnostics, abnormal condition detection, information fusion, classifier fusion

1. INTRODUCTION

Change detection is an important task in equipment monitoring because it is often the first sign of a fault that in turn may indicate failure at some future point. Equipment monitoring has enjoyed renewed attention in the last few years due to service providers' desire to perform the detection of faults as early as possible to avoid (often costly) secondary damage and to also avoid equipment downtime. The latter is undesired from the customers' perspective and guaranteed uptime is sometimes written into the service contract.

The quality of change detection depends first and foremost on the sensors and data acquisition used to collect information about the system state. If there are no sensors that collect data pertaining to a particular fault mode (either directly or indirectly), no assessment about that fault mode can be made. On legacy systems, one has often times only limited (if any) opportunity to place additional sensors. In those situations, reasoning has to be performed with data from the existing sensors. Because their information may be poor, the task is harder for the downstream algorithms to extract the relevant information. Typical steps in change detection include sensor validation, features extraction, classification, and perhaps post-processing. To boost the abnormal condition detection performance, it may be advantageous to employ – and fuse – an ensemble of diverse classifiers.

Classifier ensembles have been considered as one of the most active research directions in machine learning⁹. An ensemble of classifiers is a set of classifiers whose decisions or outputs are combined to arrive at a final classification decision. The individual classifiers in the ensemble can be any type of classifiers, preferably “unstable” classifiers like neural networks and decision trees. “Unstable” refers to the fact that a small change in the training samples will lead to a classifier with different classification performance³.

Several studies (both theoretical and empirical) suggest that classifier ensembles are more accurate than the individual classifiers in the ensemble^{14, 18}. As a result, classifier ensembles have been used for various application domains ranging from optical character recognition (OCR)²⁰ to seismic signal classification²⁸. One of the recent findings is that in order for ensembles to be more accurate than the constituent classifiers in the ensemble, individual classifiers need to be at least modestly accurate, but, more importantly, need to be diverse. That is, they need to make classification errors on different examples¹⁴. Therefore, one of the most active areas of research in ensembles has been centered on how to

* goebelk@crd.ge.com; phone 1 518 387-4194;

generating accurate and diverse individual classifiers for ensembles. Obviously, combining methods also play an important role in improving performance.

There are many methods to generate diverse individual classifiers for ensembles. These methods can be broadly categorized into three groups as follows.

- 1) Using different structure or architecture for individual classifiers. For neural network ensembles, the individual networks can have different number of hidden layers and different number of hidden neurons, and/or are trained with different initial weights and biases. For decision tree ensembles, different number of nodes may be used.
- 2) Using different training samples for individual classifiers. The two most popular methods in this group are bagging and boosting. In bagging (short for “bootstrap aggregating”) ⁴, training set for each classifier is a bootstrap replicate of the original training set, which is obtained by uniformly sampling, with replacement, the original training data. In boosting ^{27, 10}, individual classifiers are trained sequentially (in a series). The training set for a k^{th} classifier in the series is chosen based on the performance of classifiers 1 through $k-1$. The probability of selecting an example for k^{th} classifier depends on how often that example was misclassified by all $k-1$ classifiers before it.
- 3) Using different subsets of features for individual classifiers. Each individual classifier is trained with a different subset of features. The feature subsets can be obtained by random selection ¹⁶, or input decimation ³⁰, or by some optimal selection scheme (for example genetic algorithms ^{12, 23}).

2. DETECTION

Continuous parametric information from sensors allows one in principle to reason over the system health status. In aircraft engines, for example, exhaust gas temperature (EGT), fuel flow (WF), engine oil pressure (OIL) and engine core speed (N2) are some of the key parameters measured from sensors. These raw measurements can be directly fed into diagnostic models for evaluating the normal/abnormal condition of the engine. Unfortunately, the raw data space (i.e., using the measurements without further processing) is seldom the best place to attempt to make diagnostic decision. Usually, some feature extraction methods are required on the raw feature space in order to generate suitable features, which can help capture the characteristic patterns (trends) in the data. Besides parametric data, there are non-parametric data sources (such as error logs) that can be used for diagnostics as well. To combine these two types, it may be advantageous to convert the non-parametric data into the parametric domain ¹⁷.

The primary goal of diagnostics is to identify the abnormal condition of the system as early, accurate and consistent as possible. In most cases, the abnormal signatures of the engine usually will not change after fault initiation. Therefore, the condition assessment from the model should remain unchanged. In this paper, we build several diagnostic models on the features extracted from parametric and non-parametric data ¹⁷. In a forward mode, given the data accessible at each flight, the diagnostic models make an assessment (normal or abnormal) about the condition of the engine. At the end, we compare the decisions from the three models against each other and with the assessment by domain experts.

3. FUSION

There are numerous approaches such as bagging and boosting ¹⁰, Dempster-Shafer ²⁹, model-based approaches ²², fuzzy fusion ¹⁹ or statistics-based approaches ²⁶ that attempt to address the core aggregation functions. However, it has to be realized that the aggregation itself is only one function of the overall reasoner. In addition to combining information, it has to be ensured that the information that is being used provides the maximum information content. There are a number of issues that need to be dealt with prior to the actual aggregation. Specifically, the information needs to be checked for consistency, and it needs to be cleaned of outliers, noise, faulty or otherwise bad sensor information, it needs to be conditioned and formatted to allow a proper comparison. In addition to that, special cases need to be taken into account that, depending on the situation, should be done either before or after the actual aggregation step. To assist in these tasks, and to improve performance, it may be advantageous to incorporate domain expertise into the fusion strategy. Elements from such a strategy have been proven successful in diagnostic fusion environments within project IMATE ¹. There, a

hierarchical, multi-layer architecture¹³ was demonstrated that implemented some of these concepts. Information from various diagnostic models and evidential information sources was combined and manipulated through a series of steps that increased and decreased the weight given to the information sources according to the strategies implemented in the respective layers of the fusion process.

Where domain expertise is harder to come by, one has to rely on data driven approaches alone. Such a scenario is considered here. To that end, several fusion methods were evaluated. The fusion methods used are described below.

Averaging.

Breiman³ and Krogh et al.¹⁸ have shown that a simple average of the outputs of individual classifiers is an effective combining scheme. Here, classifiers were normalized as a preprocessing step using a non-linear regression of the form

$$x_n = \frac{1}{1 + e^{-\alpha x + bias}} \quad (1)$$

In addition, to account for some of the temporal effects, an exponential filter over the normalized data was employed, using $x_{n_{f_i}}(k) = \alpha_i \cdot x_{n_{f_i}}(k-1) + (1 - \alpha_i) \cdot x_{n_i}(k)$ where $\alpha = 0.8$. The individual classifiers were then simply averaged.

Linear neural net

The basic concept of regression is to determine a functional relationship between two or more correlated variables that is often empirically derived from data and is used especially to predict values of one variable when given values of the others. A linear neural network, which essentially performs linear regression, can also be used as a decision fuser to integrate outputs from different classifiers. A linear network has one layer of S neurons connected to R inputs through a matrix of weight \mathbf{W} . The network output is $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$, where the input \mathbf{x} is R by 1 vector, weight \mathbf{W} is S by R matrix, bias \mathbf{b} is R by 1 vector and \mathbf{y} is S by 1 vector. Since the diagnostic model only needs to give a classification of engine condition as output, S equals to 1 here. The output of the linear neural network needs to be appropriately thresholded to yield the classification decision.

Non-linear neural net meta-classifier

The generalized regression neural network (GRNN) is a universal approximator and typically used for function approximation. It has been shown that, given a sufficient number of neurons, GRNN can approximate a continuous function to an arbitrary accuracy. GRNN has two layers of artificial neurons. While the first layer consists of radial basis neurons (whose transfer function is a Gaussian with a spreading factor), the second layer consists of neurons with a linear transfer function. In this paper, the inputs to the GRNN are the outputs of individual classifiers/models and the single outputs are the target class labels. We set the spreading factor of Gaussian function as 1.5.

Random forest meta-classifier

For the random forest (RF) fusion approach, the raw classifier output from all 12 classifiers were first individually smoothed (within each case) using the Epanechnikov kernel with scaling parameter set to 11. Random forests were trained using a leave one out procedure for all time series cases using a new realization of balanced data (produced in the manner described in the RF Classifier section, below). For each forest, three variables were tested at each node, and 1000 trees were generated per forest. It took approximately 64 minutes to train all of the RF fusers on a 3 GHz Pentium IV.

Dempster-Shafer regression.

Dempster-Shafer-based regression²⁴ has the advantage to allow incorporation of classifier uncertainty estimates. Classical regression techniques (kernel, MLP, RBF, splines, linear, etc.) assume perfect knowledge of y (both precise and certain). However, these techniques do not work optimally if knowledge of classifier output y is imprecise due to limited precision and accuracy of classifiers. The issue is exacerbated when there are multiple classifiers with different sensitivities and reliabilities. In situations where the probe point is very different from that employed in the training set it might be desirable to have mechanisms to cast doubt on the validity of the output.

Dempster Shafer regression²⁴ (DSR) provides a prediction of the output in form of a fuzzy belief assignment. The output is computed using a nonparametric, instance-based approach: evidence samples $e_i = (x_i, m_i)$ in the neighborhood of the input vector x are sources of partial information on the response variable. The evidence samples can be represented by a fuzzy belief assignment $m_{y|[x, e_i]}$. Relevance of the evidence with respect to y is assumed to be dependent on the

dissimilarity to y . If x is “close” to x_i according to a given metric $\|\cdot\|$, y is expected to be close to y_i , which makes example e_i quite relevant to predict the value of y . Therefore, neighborhood evidence input elements are discounted as a function of their distance to x . They are then pooled using Dempster’s rule of combination. While the method can cope with heterogeneous training data, the more important characteristics in this context is the formalism for modeling both unreliable and imprecise information provided by multi-classifier systems whose outputs are time series.

DSR determines the value of sensor measurement y at a given time by discounting the belief mass of each observation by:

$$\phi(|x - x_i|) = \gamma e^{-\frac{(x-x_i)^2}{\Theta^2}} \tag{2}$$

where:

γ is a tuning parameter (usually ≥ 0.9)

Θ is a scale parameter, commonly set using cross validation on training data

Next, the discounted belief masses are combined using appropriate version of DS combination. When there are many data points, the computational overhead can become considerable. A remedy is using only the k-nearest neighbors to reduce the complexity of the calculation with little loss of accuracy.

4. PERFORMANCE EVALUATION

Overall classification accuracy (or alternatively overall classification error) is the most popular measure of classifier performance. Accuracy has been almost exclusively used in design and evaluation of classifier ensembles. The underlying assumption of the accuracy as a classifier performance measure is that the classification errors for all classes have equal cost consequences – which rarely holds for real-world applications. In real applications, different misclassification incur different costs. However, exact costs associated with different misclassification are rarely known and are difficult to estimate. We define the two types of classification errors (false positive and false negative) and ROC curves (short for Receiver Operating Characteristic or Relative Operating Characteristic) that show tradeoffs between the two types of errors for a classifier below. We restrict our discussion to two-class classification problems where the two classes are referred to as the positive and negative classes.

		Classes assigned by Classifier	
		Negative	Positive
True Classes	Negative	N⁰⁰	N⁰¹
	Positive	N¹⁰	N¹¹

Figure 1 - A typical confusion matrix for 2-class classification problems

Consider a classification problem specified by a test set, $\{\bar{x}_i, y_i\}$, $i = 1, 2, \dots, l$, where $\bar{x}_i \in R^D$ is the i^{th} input vector and $y_i \in \{0, 1\}$ (“0” stands for negative class and “1” for positive class) is the corresponding class label. Outputs of the classifier to the test data set are typically summarized in a *confusion matrix* as shown in Figure 1.

A classifier can make two types of errors, namely, false positive error and false negative error. In the field of statistical hypothesis testing, they are referred as Type I and Type II errors. $P(\text{false positive}) = P(\text{classified positive} | \text{negative})$, and $P(\text{false negative}) = P(\text{classified negative} | \text{positive})$. From the confusion matrix defined above, the accuracy can be calculated as follows.

$$acc = \frac{N^{00} + N^{11}}{N^{00} + N^{01} + N^{10} + N^{11}} \quad (3)$$

It is worthwhile to note that, for a given classifier, the accuracy above varies with decision actions. In fact, by varying the decision threshold, one can reduce one type of errors to any arbitrarily small number through sacrificing another type of errors.

The tradeoff between the two types of errors for a classifier can be better depicted by ROC curves. ROC curves represent the behavior of a classifier without regard to class distribution or error cost. Hence ROC curves are a true representative of classifier performance. In ROC space, the true positive rate, TPR (TPR=1-FNR), is plotted on the Y-axis and the false positive rate, FPR, is plotted on the X-axis, where the TPR is commonly referred to as “sensitivity” while (1-FPR) is called “specificity”. A point in ROC space corresponds to a (FPR, TPR) pair of a classifier. Classifiers with ROC curves located in the upper-left corner in ROC space are better because they represent classifiers that have lower false positive rate and higher true positive rate than the classifiers below them³². ROC curves are typically generated by varying either some parameters of the classifier (e.g. threshold in neural networks) or costs (e.g. in decision trees). A summary measure is the area under the curve (AUC) of the ROC.

5. DIAGNOSTIC MODELS

A suite of diagnostic models was built to integrate selected features to make a diagnostic decision. These models are linear regression, neural network (NN), several random forests (RF), and support vector machine (SVM). Each model has its advantages and disadvantages. Since they are quite different in principle, they give different decisions about the condition of the engine thus meeting one requirement for a successful fusion. Specifically, by providing the diversities in the diagnostic models, it creates the opportunities of fusing the decisions from each model together to make a better assessment.

5.1. Neural networks

Neural networks, usually composed of simple elements (called neurons) operating in parallel, are well known for their function approximation and pattern recognition performance^{2, 15}. They have been successfully applied across an wide range of domains, such as finance, engineering, medicine, physics, etc. In this application of aircraft engine fault diagnostics, we employ neural networks as a classifier for the condition of the engine. We designed 2 different neural network models, namely, a multiple-layer feed-forward network (FFN) and a linear regression network (LRN).

For FFN, we set the network architecture as two layers (i.e., one hidden layer) with two hidden neurons and one output node. The activation function for both hidden and output nodes are the logistic sigmoid function. The Levenberg-Marquardt learning algorithm was used for training the network.

5.2. Support vector machine

We also consider here a Support Vector Machine (SVM) as the classification engine^{8, 31}. A SVM is a maximum margin linear classifier in a high dimensional feature space $\Phi(x)$ defined by a positive definite kernel function $k(x, x')$ specifying an inner product in the feature space $\Phi(x)$. $\Phi(x') = k(x, x')$ given a labeled data set as in Eq. (5). The Gaussian radial basis function

$$k(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}} \quad (5)$$

can be used as a kernel function. Input patterns for feasible Lagrange multipliers of the optimization scheme are called support vectors²⁵.

5.3. Random forest

Random Forest ⁵ is a classification method that applies bagging ⁴ to a variation of classification trees ⁶. A standard classification tree is constructed by splitting the data on the best of all possible features at each node. For RF, only a randomly selected subset (chosen always from the full set) of features is eligible to split each node. Moreover, each individual tree is constructed on a bootstrap sample of the data. Finally, in contrast to standard classification trees, the individual RF trees are not pruned; rather they (typically) are grown to 100% node purity. Although hundreds of trees may be developed, RF's are very quick to train (e.g., much faster than neural networks for a given data set and processor). Predictions are made by aggregating the predictions of the ensemble (majority vote for classification or averaging for regression). Random forest generally exhibits a substantial performance improvement over the single tree classifier such as CART and C4.5. It yields generalization error rate that compares favorably to Adaboost, yet is more robust to noise. The RF implementation used here was by Breiman et al. ⁷.

One of the beneficial features of the RF algorithm is its use in variable selection. To estimate the importance of a particular variable, the values of that variable are permuted, and predictions of the out-of-bag cases for each tree are made using the permuted data. For a particular case, the margin is the proportion of votes for its true class minus the maximum of the proportion of votes for each of the other classes. The RF importance measure for a particular variable is the average lowering of the margin across all cases when that variable is randomly permuted. This value can be calculated quite quickly for each variable in turn.

The Random Forest (RF) classifiers were each composed of 100 trees that used all of the nine features. The number of variables examined at each split was varied from 1 to 3 (“m”, Table 1) – this, for a small fixed number of trees and in conjunction with the inherent bootstrap sampling inherent to RF, will tend to promote classifier diversity. The data were substantially unbalanced – 79% class 0, 21% class 1. For the RF labeled “balanced”, the data were resampled, keeping 100% of class 1, and sampling randomly from class 0 such that the number of cases in the training set was equal for both classes.

5.4. Linear regression classifier

Two varieties of regression-based classifier were used. The “linear regression” classifier (LRC) fit one random realization of the balanced (using the scheme described in the RF classifier, above) class data as a linear function of the nine features, plus a constant.

The “bootstrap linear classifier” (BLC) uses one random realization of the balanced data. Fifty bootstrap samples (i.e., a sample of the same size as the original data, sampled with replacement) were taken of the data, and a linear fit of class by the 9 features plus a constant was made for each sample. The BLC is the mean of the resulting 50 estimates.

While both of these approaches are rather simple minded (and arguably statistically unsound) the resulting classifiers were among the top individual performers (Table 1), and required by far the least amount of computation time (except for the linear NN, which required a comparable amount of time).

6. RESULTS

We show firsts the results from single classifiers followed by results from the fusion methods.

6.1. Single classifier results

For the experiment, we used data from 32 time series of engine data and error logs. At the beginning, the full set of 26 data features (from parametric and non-parametric data) were examined using RF. By using RF variable importance measure (averaged over the 32 time series), the original feature space was reduced from 26 to the most important 9 features, which include the raw measurements from 3 sensors, the rank permutation test feature value from the 3 sensor measurements and the features from 3 types of error log messages ¹⁷. Then, all the diagnostic models used the same set of 9 features as input for classification. A leave-one-out (LOO) approach was employed, where 31 time series were used for training and one time series were used for testing. This one test time series and the training set were then rotated such that 32 different classifiers were built. The performance was averaged across the different test runs.

Table 1 – Performance of individual classification approaches

classifier	best accuracy	AUC
linear NN	0.833	0.875
feedforward NN	0.848	0.881
SVM	0.837	0.789
RF01 (m=1)	0.821	0.853
RF02 (m=1, repeat)	0.827	0.860
RF03 (m=2)	0.821	0.861
RF04 (m=3)	0.817	0.851
RF05 (m=1, balanced)	0.813	0.864
RF06 (m=2, balanced)	0.819	0.877
RF07 (m=3, balanced)	0.834	0.878
linear regression	0.843	0.881
bootstrap linear regression	0.841	0.878

Table 1 compares the performance of the models in terms of their best overall accuracy on the 32 time series of engine data using the LOO procedure. Each model has different performance as expressed by the best accuracy and the AUC. Depending on the primary goal of the diagnostic system, one can weight the performance of the model differently based on the confusion matrix and choose the most appropriate model. The ROC curves of the three models are shown in Fig. 2.

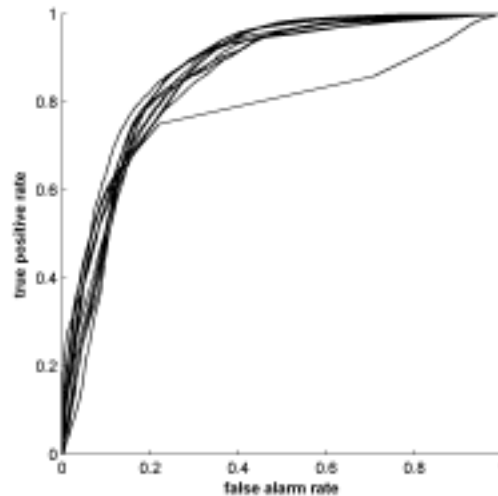


Figure 2 – ROC curves of the diagnostic models.

6.2. Fusion results

As a baseline fusion method, we employed the simple averaging scheme. The averaging scheme does a very good job, improving the best accuracy to 0.858 and the AUC to 0.899. This result is only beaten by one other fusion method, the RF meta-classifier. The fusion results are summarized in Fig. 2 and Table 2.

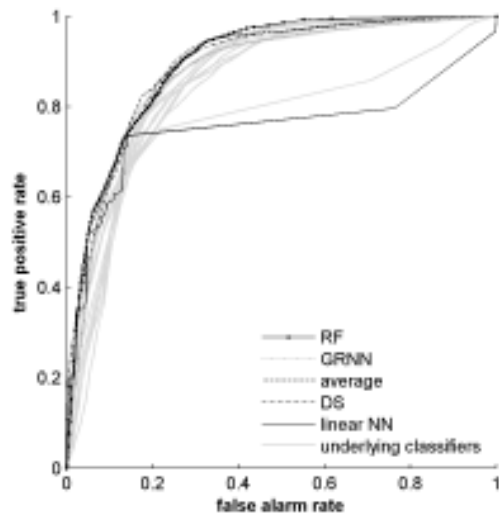


Figure 3 - ROC curves for different fusion methods (black) and the underlying classifiers (grey).

Table 2 - Performance of individual fusion approaches

fusion method	best accuracy	AUC
RF	0.859	0.902
GRNN	0.849	0.882
averaging	0.858	0.899
Dempster-Shafer	0.846	0.894
linear NN	0.858	0.758

Figure 3 shows a zoomed version of the same ROC for better examination. In particular, there are several fusion methods that perform best in certain regions of the FP/TP space. Moreover, there is a region where all fusion methods perform worse than the best classifier. In general, the performance of the fusion methods is very close.

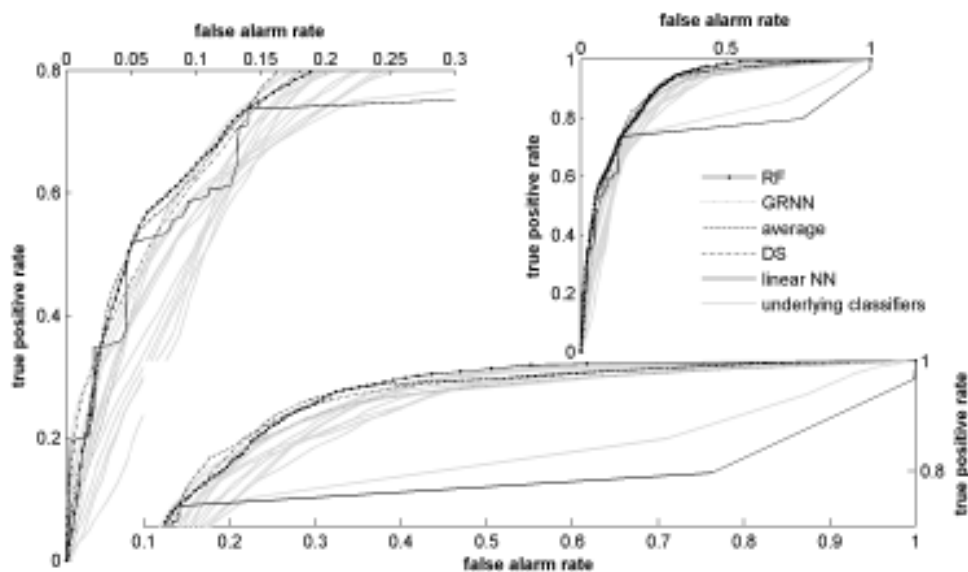


Figure 4 - ROC curves for different fusion methods (black) and the underlying classifiers (grey). Top right axes shows overall ROC. Left and bottom sets are rescaled to show detail.

7. DISCUSSION

Although one can use the best accuracy or the AUC as a performance guide, these measures do not necessarily give the full picture. Using the best accuracy of the linear regression NN as an example, one can see that although the best accuracy is close to optimal, the general performance of this fusion method is very poor. In addition, if a Neyman-Pearson criterion were used as a performance requirement, the fusion method chosen might be different based on the specific requirements. The fusers seem to make up performance mostly in the extreme regions of the FP and TP space. This may be due in part to the ability of the fusers to weed out outliers that decrease the FP rate. In the mid-region, no performance gains are obtained compared to the best classifier. Fundamentally, the fuser gets dragged down by the poorer performing classifiers.

When using the linear neural network, non-linear neural network and random forest as the meta-classifier for fusion, the leave-one-out approach was employed again, just like during classifier training. This testing scheme has a potential problem because the decisions from any two time series are not completely independent. That is, they were generated by the two classifiers which were trained on the same 31 time series. A possible solution might be to use the 31 time series to build the individual classifiers and simultaneously train the fusion models. Testing of the individual classifiers and the fusion scheme would then be performed together on one the left-out time series.

The Dempster-Shafer fusion was limited to fusing four classifiers due to rapidly increasing computational requirements that prevented a practical use beyond that number. Some of these shortcomings could be overcome by using a limited-sized rolling window of data. In contrast, the averaging fusion compels due to its simplicity, performance, and low computational overhead. Because the linear network fuser was not very good when it was built upon the original continuous outputs of each classifier, the output of each classifier was converted into binary decisions using the best threshold obtained from best overall accuracy. Then, the linear network was constructed to integrate the decisions from each classifier. Variations of the thresholds of each classifier was used as a training knob to accomplish best overall fusion performance. The linear network fuser achieved best accuracy when each classifier used other than their respective best thresholds.

When we evaluated the performance of each model, we counted each individual incorrect classification as either a false positive or false negative. This may lead to overly harsh assessment because an early prediction – while technically desired – is counted as a false positive. It would be more reasonable to use a different metric to assess the diagnostic decisions from the models, which does not take the early false positives close to the start of the engine abnormal condition as misclassification. Such an evaluation metric should also give relative small penalties for decisions which do not recognize the abnormal condition immediately as long as the model catches the fault soon after.

In general, we did not employ time dependent techniques outright. This might be an area for possible improvement. Schemes that employed implicit (or explicit, as in the case of the averaging scheme) time processing, even as simple as filtering, did reasonably well.

Other areas of improvement include increasing the diversity of the classifiers through the use of better and more varied features. We constrained the investigation to a set of limited features which may explain some of the limited performance gains.

8. SUMMARY

We present here the results of fusing different classifiers for change detection in time series. Challenges here are the sequential nature of the data, limited information content of the features, unbalanced training sets, and inconsistent labeling of training data. Fusers of varying complexity can improve over single classifiers in most of the FP/TP space. However, for this particular problem, complex fusion schemes do not provide much gain over simpler ones. Areas of single individual outliers are effectively weeded out while areas of joint (false) agreement may point more to inconsistent labels. Classifiers were largely designed without taking advantage of temporal concepts. Incorporating both temporal aspects as well as relaxing the stringent labeling will be subject of future work.

REFERENCES

1. Ashby, M. and Scheuren, W. (2000), "Intelligent Maintenance Advisor for Turbine Engines (IMATE)", Proceedings of the IEEE Aerospace Conference, 11.0309, 2000.
2. Bishop, C.M. (1995), Neural Networks for Pattern Recognition, Oxford University, Press.
3. Breiman, L. (1996a), "Stacked regression", Machine Learning, Vol. 24, No. 1, pp 49-64.
4. Breiman, L. (1996b), "Bagging predictors", Machine Learning, Vol.24, No.2, pp 123-40.
5. Breiman, L. (2001), "Random forest", Machine Learning, 45(1), pp. 5-32.
6. Breiman, L.; Friedman, J.; Olshen, R. and Stone, C. (1984), *Classification and Regression Trees*, Wadsworth, Belmont, CA.
7. Breiman, L. and Cutler, A. (2004), Random Forest Toolbox Version 5.1, http://www.stat.berkeley.edu/users/breiman/RandomForestes/cc_software.htm, University of California at Berkeley.
8. Cawley, G.C. (2000), Support Vector Machine Toolbox v0.50 beta, <http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox>, University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K. NR4 7TJ.
9. Dietterich, T.G. (1999), "Machine learning research: four current directions", AI Magazine, Vol. 18, No. 4, pp97-136.
10. Freund, Y. and Schapire, R. (1999), "A Short Introduction to Boosting", J. Japanese Society Artificial Intelligence, Vol. 14, No.5, pp. 771-780.
11. Freund, Y. and Schapire, R.E. (1996), "Experiments with a new boosting algorithm", In Saitta, L. (Ed.), Proceedings of the 13th International Conference on Machine Learning, pp148-156.
12. Gerra-Salcedo, C. and Whitley, D. (1999), "Genetic approach to feature selection for ensemble creation", Proceedings of Genetic and Evolutionary Computation Conference, pp236-243.
13. Goebel, K. (2001), "Architecture and Design of a Diagnostic Information Fusion Tool", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 15 (4), Sept. 2001, pp. 335-348.
14. Hansen, L. and Salamon, P. (1990), "Neural network ensembles", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 10, pp 993-1001.
15. Haykin, S. (1999), Neural Networks: a comprehensive foundation, 2nd Ed., Prentice Hall, New Jersey, 1999.
16. Ho, T.K. (1998), "The random subspace method for constructing decision forests", IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8), pp832-844.
17. Hu, X.; Eklund, N. and Goebel, K. (2006), "Using Rank Permutation for Aircraft Engine Fault Diagnostics", Proceedings of 60th MFPT, 2006, to appear.
18. Krogh, A. and Vedelsby, J. (1995), "Neural network ensembles, cross validation, and active learning", In Tesauro, G., Touretzky, D. & Leen, T. (Eds), Advances in Neural Information Processing Systems, Vol. 7, pp231-238, Cambridge, MA, MIT Press.
19. Loskiewicz-Buczak, A. and Uhrig, R. (1994), "Decision Fusion by Fuzzy Set Operations", Proc. third IEEE Conf. Fuzzy Systems, Vol. 2, pp.1412-1417.
20. Mao, J. (1998), "A case study on bagging, boosting, and basic ensembles of neural networks for OCR", Proceedings of IEEE World Congress on Computational Intelligence, Vol. 3, pp1828-33
21. Nadaraya, E. (1964), "On estimating regression". Theory of Probability and its Applications. 10, 186-190.
22. Nelson, M. and Mason, K. (1999), "A Model-Based Approach to Information Fusion". Proc. Information, Decision, and Control, pp. 395-400.
23. Opitz, D.W. (1999), "Feature selection for ensembles", Proceedings of 16th International Conference on Artificial Intelligence, pp379-384.
24. Petit-Renaud, S. and Denoeux, T. (2004), "Nonparametric Regression Analysis of Uncertain and Imprecise Data Using Belief Functions", International Journal of Approximate Reasoning, Vol. 35, No. 1, pp. 1-28.
25. Platt, J.C. (1999), "Fast training of support vector machines using sequential minimal optimization", Advances in Kernel Methods - Support Vector Learning, (Eds) B. Scholkopf, C. Burges, and A. J. Smola, MIT Press, Cambridge, Massachusetts, chapter 12, pp. 185-208.
26. Rao, N.S.V. (2000), "Finite Sample Performance Guarantees of Fusers for Function Estimators, Information Fusion, Vol. 1, no. 1, pp. 35-44.
27. Schapire, R.E. (1990), "The strength of weak learnability", Machine Learning, Vol.5, No.2, pp197-227.
28. Shimshoni, Y. and Intrator, N. (1998), "Classification of seismic signals by integrating ensembles of neural networks", IEEE Transactions of Signal Processing, Vol. 46, No. 5, pp1194-1201.

29. P. Smets (1994), "What is Dempster-Shafer's model?" *Advances in the Dempster-Shafer Theory of Evidence*, Yager, R., Fedrizzi, M., and Kacprzyk, J., (Eds.), John Wiley & Sons, New York, pp. 5-34.
30. Tumer, K. and Oza, N.C. (1999), "Decimated input ensembles for improved generalization", *Proceedings of the International Joint conference on Neural Networks*, Washington DC.
31. Vapnik, V.N. (1995), *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.
32. W. Yan, K. Goebel, and J. Li (2002), "Classifier Performance Measures in Multi-Fault Diagnosis for Aircraft Engines", *Proceedings of SPIE, Component and Systems Diagnostics, Prognostics, and Health Management II*, vol. 4733, pp. 88-97.