

INCREMENTAL PERFORMANCE IMPROVEMENT DURING DEVELOPMENT OF DIAGNOSTIC FUSION ALGORITHMS

Kai Goebel

GE Global Research, Information & Decision Technologies, K1-5C4A, 1 Research Circle, Niskayuna, NY 12309

KEY WORDS: Classification, Diagnostics, Correlation, Information Fusion, Redundancy.

Abstract

We examine the design of classifier fusion algorithm development where both the classifiers and the fusion algorithm are unknown. To properly guide algorithm development, it is desirable to evaluate algorithm performance throughout its design. Because the design space may be of considerable complexity, exhaustive simulation of the entire design space may not be practical. In addition, using the wrong classifier output distribution may lead to fusion tools that are either overly optimistic or otherwise distort the outcome. Either case may lead to a fuser that performs sub-optimal in practice. We study an algorithm development that is guided by a two-stroke design of experiment setup and evaluate different classifier distributions on the fused outcome. We show results from an application of diagnostic classifier fusion for gas path faults of aircraft engines.

INTRODUCTION

Classifier fusion attempts to aggregate the output from several classification tools as well as secondary information where available. Fusing different classifiers is mostly motivated by overcoming the performance ceiling of any single classifier. Fusion works only when different classifiers misclassify for different cases. While partial correlation is acceptable, even desired, completely redundant classifiers would not leave the fusion tool with any opportunity to improve the overall accuracy. If, for example, two classifiers in a three classifier fusion task are completely redundant, fusion schemes can easily be biased by the repetitive information. It is really the complementary information that allows the multi-classifier fusion to be successful. It is therefore necessary to investigate classifier correlation within a simulation context. The paper will first discuss classifier fusion, then address correlation analysis, followed by application to a fault detection example.

CLASSIFIER FUSION

Classifier development can go through several phases because class coverage increases or because classification performance needs to be higher. In some instances, several tools are being used in part because any one tool may not be able to deal with all classes of interest at the desired level of accuracy. Other reasons may include the historic use of particular tools that do not have expansion capabilities, or

limited resources to develop a comprehensive classification tool addressing all issues. In addition, tools' classification capabilities are impacted differently by environmental changes. The resulting multi-classifier patchwork approach might achieve optimization at a local level, but it might also blur the picture because inevitably, conflicting information is introduced. Taking a system-level view may have certain benefits because the results of different classification tools can be gathered and fused. The hope is to maximize the advantages of each one while at the same time minimizing the disadvantages. Such a fusion scheme holds the promise to deliver a result that is better than the best result possible by any one tool employed. In part this can be accomplished because redundant information is available, which when combined correctly improves the estimate of the better tool and compensates for the shortcomings of the less capable tool. The correlation between the classifiers to be fused needs to be small to enable performance improvement (Ho et al., 1994; Petrakos et al., 2000) in classifier fusion. This is not generally an invitation to use marginal-performance classifiers which in fact may worsen the outcome (Hall and Garga, 1999).

During the design of the fusion tool it is important to evaluate algorithm performance at an early stage. The impact of input variations on the algorithm needs to be assessed. It is desirable to develop a coarse algorithm during a first path and fine-tune during a second. This can be accomplished with a guided DoE simulation while the performance throughout the design of the fuser is quantified via a suitable numerical error metric.

FUSION ARCHITECTURE DESIGN

To facilitate the performance evaluation during the fusion design process, new algorithmic modules need to be subjected to an iterative cycle of conception, implementation, and testing. This is done to reduce the re-design necessary after completion of the whole fusion architecture which at that stage would be more difficult because the influence of individual modules are then masked in the overall architecture. At the onset, it is not clear what effect the addition of new modules of the algorithm will have on the overall performance and whether the chosen implementation will move the error into the desired direction. Similarly, the parameters are not known beforehand and at least coarse tuning needs to be carried out for each routine. To appraise the performance of the algorithm during development, a metric to assess incremental enhancements to the code is

needed. Where possible, it is advisable to establish an error metric using expert judgments. In cases where the design space is of rather large size (a common issue) it will be difficult to elicit expert opinions for the entire space. To cut down on the number of case permutations to look at and to at the same time cover the entire design space as exhaustively as possible, we propose to employ the Design-of-Experiment (DoE) approach (Fowlkes and Creveling, 1995) for a conceptual set of cases. Because the domain expert will be exposed only to the tool output without knowledge of the root causes for the cases, this step is called the “bottom-up DoE”. The purpose is to capture expert’s reasoning and to get a training set that can be used to design the reasoning tool. Using the experts’ heuristics, the DoE set, and the associated ratings, one can then implement successively the heuristics (or routines) for the fusion tool with constant evaluation of performance. Implementations of heuristics not leading to an error reduction can be redesigned and can also be rapidly re-evaluated. This cuts down significantly on development time and ensures a successful architecture. In a second step, one may run a full Monte Carlo simulations (Fishman, 1996; Sobol, 1994) with known inputs (“top-down Monte Carlo”). The Monte Carlo runs can be carried out in batch mode for all cases, i.e., a desired (typically large) number of runs can be employed. Results are returned in cumulative fashion and automatically converted into false positive (FP), false negatives (FN), and falsely classified (FC). This is particular useful for simulations that operate on the far ends of the probability tails, i.e., for events that do not occur very often. The Monte Carlo simulations are no longer constrained to the conceptual case set but are used on the actual cases and tools under consideration for the application. This step helps to understand the complex behavior of the various classifiers, establishes that the fusion tool works properly and aids in fine-tuning of the algorithm and parameters.

All modules are evaluated on a stand-alone basis, where possible to ensure that only modules were added that had a positive impact on reducing the overall error.

CLASSIFIER PERFORMANCE EVALUATION

We consider classifier problems where a feature vector $x \in \mathfrak{R}^P$ is to be labeled into one or more of c classes. In order to achieve high overall performance of the fault detection function, the performance of each individual classifier has to be optimized prior to using it within any fusion schemes. The optimization is performed with respect to selecting appropriate parameters and – where applicable – structure that govern the performance.

For each classifier, a confusion matrix M can be generated using labeled training data (Goebel, 2001a). The confusion matrix lists the true classes c versus the estimated classes \hat{c} . Because all classes are enumerated, it is possible to obtain information not only about the correctly classified states (N^{00} and N^{11}), but also about the false positives (N^{01}) and false negatives (N^{10}). The top-left entry of the confusion matrix is

dedicated to the normal case N^{00} . The first row – except the first entry – contains the N^{01} . The off-diagonal elements – except the first row – contains the N^{10} . Sometimes a further distinction is made between false negatives and false classifieds where the false classifieds are defined to be the off-diagonal elements of the confusion matrix except the first row and the first column. A typical two-class confusion matrix M is shown in Figure 1.

		Classes assigned by classifier	
		0	1
True Classes	1	N^{00}	N^{01}
	0	N^{10}	N^{11}

Figure 1: Typical 2-class confusion matrix

From the confusion matrix of each classifier, the false positive (FP) error, the false negative (FN) error, the total error rate (TER), and the total success rate (TSR) can be calculated for the classifier. All of these error rates are defined as in Equations 1 – 4. The total error rate (TER) or the total success rate (TSR) is typically used as a simple measure for overall performance of a classifier:

$$FP = \frac{N^{01}}{N^{00} + N^{01}} \tag{1}$$

$$FN = \frac{N^{10}}{N^{10} + N^{11}} \tag{2}$$

$$TER = \frac{N^{01} + N^{10}}{N^{00} + N^{11} + N^{01} + N^{10}} \tag{3}$$

$$TSR = 1 - TER \tag{4}$$

Classifier design is in practice most of the times guided by particular performance requirements. That is, there are particular specifications about the false positive and true positive rates (or false negative or true negative rates). The actual specification is driven by the application and the domain where the classifier is executed. For example, it might be extremely undesirable to issue false positives in certain domains while (even for the same application) it is more important to avoid false negatives in a different domain (such as in some military vs. civilian applications). At any rate, the classifier designer would follow a path of data pre-processing, feature extraction and selection, classifier design and optimization, and classifier evaluation (Yan et al., 2002) If the classifier performance is not satisfactory, the steps of classifier selection and optimization as well as feature selection and extraction may be revisited to improve on the results. Classifier fusion design adds a similar (outer) cycle of selecting the classifiers, then designing and optimizing the fuser, evaluating the result, and, if necessary, returning back to investigate additional classifiers (see Figure 2).

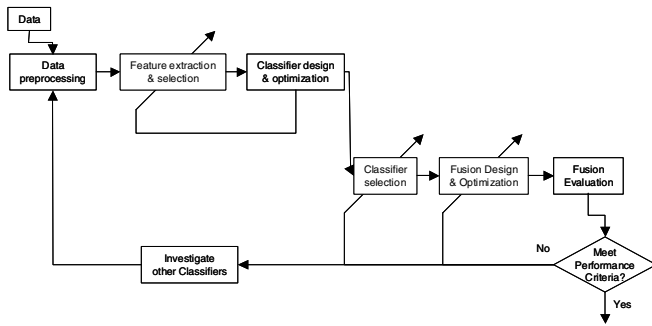


Figure 2 - Fusion Design

Although there are a number of classifiers that might be considered for a given problem, each classifier will exhibit a performance ceiling. This ceiling might prevent the overall goals of the problem to be achieved. A convenient way to display the classifier performance over the range of performance criteria is the receiver operating characteristic (ROC) curve. ROC analysis is an established method of measuring classification performance in various domains such as medical imaging. Originated from the field of signal detection to depict tradeoffs between hit rate and false alarm rate (Egan, 1975) ROC analysis and its associated indices have been extended for use in evaluating performance of 2-class classifiers (Bradley, 1997; Downey et al., 1999) The ROC space is a coordinate system that is used for visualizing classifier performance. In ROC space, the true positive rate (TPR) is plotted on the Y-axis and the false positive rate (FPR) is plotted on the X-axis. The TPR is sometimes also referred to as “sensitivity” while (1-FPR) is called “specificity”. By varying the decision threshold of the classifier, a series of points (FPR, TPR) can be obtained and a ROC curve can be generated in ROC space by connecting these points. Typical ROC curves are shown in Figure 3 where the three ROC curves represent three different classifiers. 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 them.

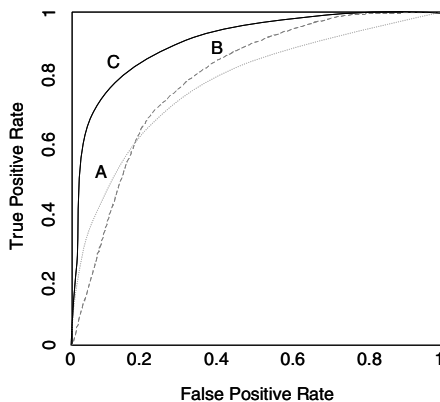


Figure 3 - ROC curves for classifiers A, B, and C.

ROC curves are a valuable technique for visualizing classifier behavior over a range of decision rules, therefore ROC curves are frequently used for selecting a suitable operating point, or decision threshold, for the task at hand. However, when used for comparing or ranking classifiers based on their performance, evaluation based on ROC curves becomes more involved when the curves overlap. In Figure 3 the three ROC curves correspond to three different classifiers A, B, and C. One can easily conclude that classifier C is better than or at least as good as the other two classifiers for all possible cost and class distributions since curve C dominates others over the entire range. Determining which of the two classifiers (A and B) is better, on the other hand, would not be so straightforward unless a specific performance requirement is given. For example, given the maximum acceptable false positive rate, one would simply draw a vertical line at the specified maximum FPR, and rank the classifiers based on TPR at the intersection of ROC curves with this vertical line. Similarly, one would use a horizontal line to rank the classifiers if the maximum required true positive rate is given. There is always a trade-off between false FPR and TPR and typically, only one parameter can be set. However, there has been a general desire to increase the overall accuracy of classifiers which has in part driven the development of classifier fusion systems. ROC curves are a valuable tool in visualizing the classifier performance and can aid in finding the classifier performance ceiling

CLASSIFIER CORRELATION ANALYSIS

It has been well understood that each individual classifier’s performance is very important to the performance of a classifier fusion. In a similar fashion, the dependency between the classifiers to be fused also affects the fusion results (Petraikos et al., 2000). Therefore, classifier correlation analysis is important for classifier fusion design.

Two-class correlation analysis

Petraikos et al. (Petraikos et al., 2000) describe a classifier correlation analysis for two classifiers. Based on the classifier outputs on the labeled training data, a 2x2 matrix *N* as shown in Figure 4 can be generated for each classifier pair. The off-diagonal numbers directly indicate the correlation degree of the two classifiers. The smaller the two off-diagonal numbers are, the higher the correlation between the two classifiers will be. The proportion of specific agreement is the correlation index, ρ_2 . It is defined as (Petraikos et al., 2000)

$$\rho_2 = \frac{2 \times N^{FF}}{N^{TF} + N^{FT} + 2 \times N^{FF}} \tag{5}$$

where N^{TT} implies that both classifiers classified correctly, N^{FF} means both classifiers classified incorrectly, N^{TF} represents the case of the 1st classifier classified correctly and 2nd classifier classified incorrectly, and N^{FT} stands for the 2nd classifier classified correctly and 1st classifier classified

incorrectly as further shown in Figure 4. In order for classifier fusion to be effective in performance improvement, the correlation index, ρ_2 , has to be small (low correlation).

		Classifier #2	
		T	F
Classifier #1	T	NTT	NTF
	F	NFT	NFF

Figure 4 - Correlation analysis matrix

n-Class correlation analysis

We propose an extension of the 2 class correlation coefficient to n different classifiers (Goebel et al., 2002). The notion that redundancy is described by the individual true and false answers of the classifiers is retained from the 2 class correlation analysis. The larger the correlation index, the larger the redundancy. In particular, the correlation index goes to zero if the individual incorrect answers are disjoint for all answers. In other words there is always at least one correct answer for any class. The correlation coefficient gets larger as the number of wrong answers are the same for many answers. Let N be the number of experiments where all tools had a wrong answer, N_i^c be the number of experiments with combinations of correct and incorrect answers; c is the combination of correct and incorrect answers (for 2 tools: $c \in \{wr, rw\}$; for 3 tools: $c \in \{wwr, wrw, rww, wrr, rwr, rrw\}$, etc.); n is the number of tools. The correlation coefficient is

$$\rho_n = \frac{nN^f}{\sum_{i=1}^{2^n-2} N_i^c + nN^f} \quad (6)$$

If N is the number of experiments and N^r is the number of experiments for which all tools had a right answer to, equation 6 can more conveniently be rewritten by:

$$\rho_n = \frac{nN^f}{N - N^r - N^t + nN^f} \quad (7)$$

It is interesting to note that the correlation index does not record redundancy with any particular tool (for $n > 2$) but with a set of tool only. The calculation of the correlation factor can be performed on multi-class scenarios as well because the factor is only concerned with the correctness of the outcome.

Classifiers with continuous output

For classifiers that give continuous output such as confidences, partial class membership etc. we propose a slightly different operator ρ_{n_c} (Goebel et al., 2002). Let

N_c^f be the sum of all false tool outputs that are greater than

threshold t_c ; o_i^f is the number of aggregated false output per

$$\text{case } i, \quad o_i^f = \frac{\sum_{j=1}^{n_{tools}} o_{j,i}}{n_{tools}} \quad \text{and} \quad o_{j,i} = \begin{cases} o_{j,i}^{raw} & \text{if } o_{j,i}^{raw} > t_c \\ 1 - o_{j,i}^{raw} & \text{otherwise} \end{cases}; \quad t_c \text{ is}$$

the decision threshold for class membership ($t_c = 0.5$); N_f is the

$$\text{number of cases that are false per threshold } t. \quad N_c^f = \sum_{i=1}^{N_f} o_i^f$$

Then the correlation index ρ_{n_c} is

$$\rho_{n_c} = \frac{nN_c^f}{N - N_c^f - N_c^t + nN_c^f} \quad (8)$$

Use of Correlation Index

The correlation coefficient can be used for different purposes, in particular for 1.) tool selection; 2.) tool simulation; and 3.) fusion estimate refinement

Tool selection

Tool selection should be carried out such that the least redundancy is maintained. Starting with the best performing tool using false positives, false negatives, and false classified states for evaluation, the next best tool will be added. This process is repeated until the desired number of classifiers has been reached or until the correlation index increases.

Tool Simulation

Establishing the degree of correlation is also important for simulation purposes. When testing and validating fusion systems, it is important to take into account the degree of correlation. If systems are treated as quasi-independent, incorrect conclusions might be drawn about their performance. Therefore, a simulation must always consider the interdependence of classifiers. One way to accomplish that is to perform a guided Monte Carlo simulation. This guided Monte Carlo simulation is carried out such that output for the prime classifier is generated according to performance criteria as found in the confusion matrix. In particular, values are simulated with values greater than 0.5 for the percentage indicated in the diagonal entries of the confusion matrix and values less than 0.5 in the remaining cases. Appropriate distributions must be used such as monotonic increasing Gaussian centered around 1 or Weibull distributions centered around 0.5 (Goebel, 2003).

The secondary classifier, that is, the classifier that is ranked second using some overall performance measure such as false positives and false negatives (Goebel, 2001a) must be simulated to account for the correlation established. This can be done by guiding the Monte Carlo simulation such that based on the value of the primary classifier, the value of the secondary classifier is modeled.

Next, the value of the tertiary classifier is modeled based on the combination of the values of the first two classifiers. The

information used for that purpose is encoded in the truth tables amended by the correlation information.

CLASSIFIER OUTPUT SIMULATION

There are different ways to simulate the classifier output. Among the many, we considered in this study 1.) A one-sided Gaussian distribution; 2.) A bi-variate uniform distribution; 3.) A bi-variate Gaussian distribution..

One-sided Gaussian distribution

Diagnostic tool output can be simulated by assuming a Gaussian distribution of fault recognition. That is, if the reliability of the tool for a specific fault was listed as 0.8 in the diagonal entry of the confusion matrix, then 80% of the faults were simulated between confidence 0.5 and 1. The other 20% were simulated between 0 and 0.5. Specifically, for a given case, the associated z-score is computed by interpreting the associated entries in the confusion matrix as the area under the curve of a Gaussian distribution. Next, random values based on the z-score are assigned. Figure 5 illustrates the example.

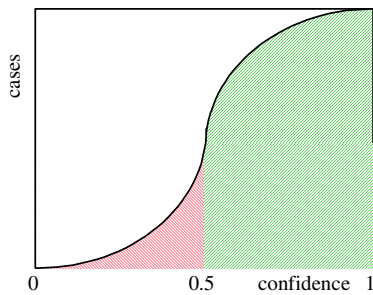


Figure 5 - Simulation of diagnostic tool output

This distribution is somewhat idealized in the sense that it assumes that most of the cases have confidence values that lie closer to the extremes than at the decision intersection. On the other hand, there is a continuous transition from no fault decision to fault decision. In the results section below we will illustrate the performance of this simulation model in comparison with other models.

Bi-variate uniform distributions

This distribution assumes that the output of the classifiers is uniform across the confidence space. This distribution is depicted in Figure 6 below. While this model would in most cases not represent a real world situation, there are certain advantages such as the ease of modeling. In addition, this distribution might still be useful in providing a bound to fusion performance.

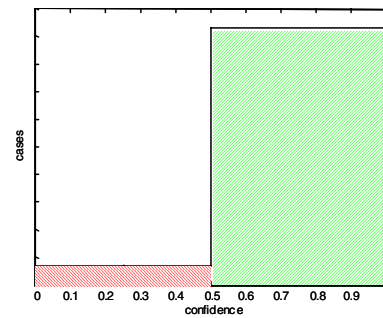


Figure 6 - Bi-variate uniform confidence distribution

Bi-variate Gaussian distribution

This distribution does more closely represent typical classifier output. For example, the histogram in Figure 7 shows the output of a 2-class classifier that was trained to detect whether or not features represented corrosion in pipelines. Figure 7a shows the distribution for corrosion, Figure 7b represents no-corrosion, and Figure 7c shows the overlap of the two classes over the confidence space.

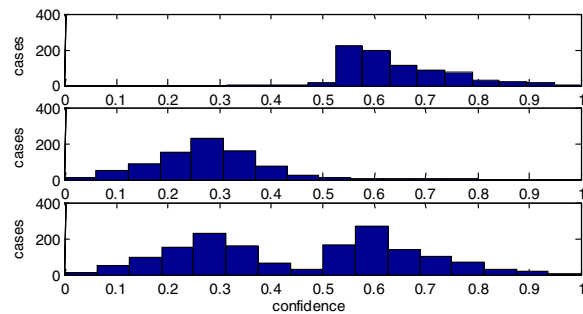


Figure 7 - Histogram of 2-class classifier

Figure 8 shows the corresponding modeled 2-class Gaussian model for the classifier output. Although this model seems to reflect the output above to some degree, caution must be advised against generalizing from this observation. In the case below, the mean and standard deviation for both classes was known. Generally, this is not the case. Results may differ when the distributions are assumed to have a larger or smaller standard deviation (and different means) than in reality. Generally, we postulate that the larger the difference of the mean of the two classes, the easier it is for the classifier (and the fuser) to make a correct decision. The same holds true to some degree for a tighter standard deviations. As a bounding agent at one end of the spectrum, the bi-variate uniform distribution represents the case where the standard deviation is infinitely large. Table 2 shows the results for the 3 distributions using the fusion scheme explained below.

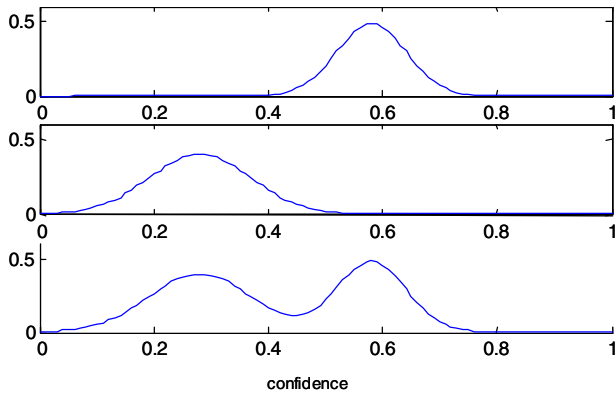


Figure 8 - Model for 2-class classifier: 6(a): class 1 (corrosion);

Table 2 also shows an index which is a weighted accuracy measure reflecting the relative goodness of the algorithm. As can be seen, the one-sided Gaussian distribution performs the best, while the bi-variate uniform distribution performs worst. The bi-variate Gaussian performs quite well but not as well as the one-sided Gaussian. The results of the performance index were bounded on average by $\pm 0.03\%$ within the 95% confidence interval.

Table 2 – False Positives, False Negatives, and False Classified Rates for fusion scheme using different classifier output distributions

	FP	FN	FC	index
One-sided Gaussian	0.0001	0.0019	0.0026	99.91%
Bi-variate uniform	0.0094	0.0113	0.0315	98.78%
Bi-variate Gaussian	0.0025	0.0035	0.0141	99.60%

APPLICATION AND RESULTS

We simulated the distributions for a 7-class classifier suite that dealt with gas path faults for an aircraft engine (Ashby and Scheuren, 2000) and used a fusion engine that was developed specifically for that domain. We postulate that the observations for the classifier output model is applicable also to other fusion engines.

In the sections below we will discuss the information used for the fusion tool, the architecture of the proposed fusion scheme, and the design of the tool.

Information used in fusion scheme

The fusion tool uses a priori information and the output coming from the diagnostic and non-diagnostic information sources. The proposed scheme relies heavily on information about tool performance implying that this information is attainable through experiments or simulations.

IFM Input

Primary input to the information fusion is the output of the classifiers, i.e., the fault vector of each diagnostic tool for the respective faults considered. The information fusion tool is built on the premise that it can utilize information that led to the classification. In other words, it will not only consider the final fault assignment but also the underlying relevant fault strength. Depending on the diagnostic tool employed this can be a distance measure (for example for a k-means classifier), probability (for example for a Bayesian Belief Net), weight (for example for a neural net), membership (for example for a fuzzy knn), etc. This individual assignment criterion is then scaled between zero and one using an appropriate classifier specific non-linear function. The implicit interpretation is that a level closer to one means that the fault is increasingly more likely while a confidence level less than 0.5 is increasingly not likely. Thereby we avoid the step of needing a parametric model for fusing heterogeneous data (Hathaway et al., 1996) and instead impose this task on the designer of the diagnostic tools who has to provide the mapping from diagnostic output to confidence level. Figure 9 shows the diagnostic output to information fusion mapping.

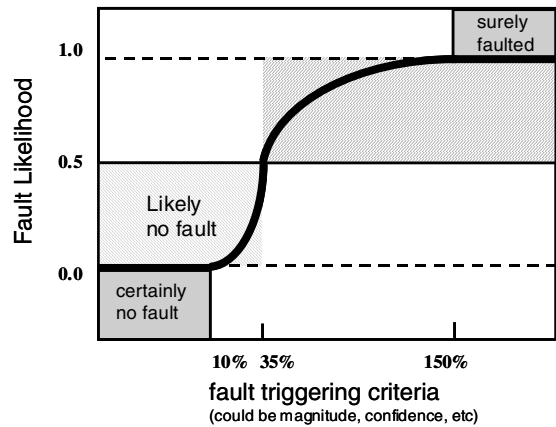


Figure 9 - Mapping of diagnostic tool output into 0-1 domain

Other system information not diagnostic in nature (that can be used to support a diagnostic opinion) is also provided as input for the information fusion tool. This is information that would not in itself give rise to an action but helps the diagnostician in understanding and confirming a diagnostic opinion.

Fusion algorithm architecture

Generally, our fusion tool is divided into three components: 1.) pre-processing, 2.) analysis or core fuser, and 3.) post-processing. We have designed the architecture in such a manner that a maximum amount of external information can be integrated. In addition, we attempted to keep the design modular to allow for later addition of domain-specific modules. Each component consists of several modules that are designed to improve the fusion task at hand. We have explored different schemes that can deal with the fusion

scheme. Data are first conditioned in the data pre-processing component.

This includes changes to the classifier output through smoothing, outlier eliminations, capturing temporal effects, and integrating a priori classifier performance information. The outputs of the pre-processing component are modified class estimates. Next, the modules of the hierarchical core fuser aggregate the modified inputs. Finally, the results are polished – where necessary – in the post-processing component to allow for better user interpretation and to account for unequal fault representation. Elsewhere (Goebel et al., 2000), we reported about a hierarchical architecture (“scheme 1”) as shown in Figure 10. We will now briefly introduce the components of this scheme.

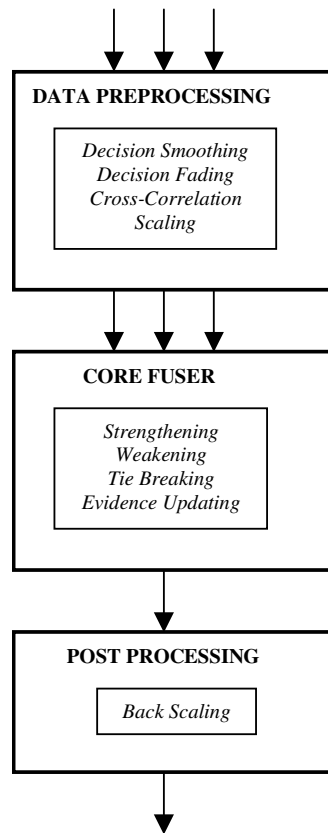


Figure 10: Fusion Components

Preprocessing - Averaging

This stage deals with temporal information aggregation. Although plain averaging ensures smoothing of information with time, it can stifle the influence of new information. Hence adaptive averaging is used employing an adaptive filter parameter that is adjusted to be low when ‘changes’ are high and vice-versa (Goebel, 2000).

Preprocessing – Fading

This module serves to aggregate conflicting information across tools and simultaneously acts on the temporal information also. For instance, if tool X indicates class A at time t_1 and tool Y indicates class B at time t_2 , ($t_2 > t_1$), then we need to account for the fact that B might have occurred in the time interval $t_2 - t_1$. So we need to fade the past information with a fading factor that is a function both of the time elapsed and the confidence in the earlier tool’s decision (Goebel, 2000)..

Cross-Correlation

This module makes use of the preferred misclassification of tools (off-diagonal entries of the confusion matrix) to discount the tool output for each class. The purpose is to factor out cross correlation effects. In this scheme, information about preferred misclassifications is used in a manner that discounts the output of a certain class based on the entries in the association matrix (Goebel and Mysore, 2001).

Scaling

The inequity in the representation of faults by tools is addressed here by boosting the diagonal entries of the confusion matrix. In this process, the module uses a ‘relevance matrix’ $[r_{t,i}]$, where $r_{t,i}$ is 1 if tool t is built to recognize fault class i and 0 otherwise. Once this is done, the diagonal entries are used to scale the tool outputs so that more reliable tools are ‘trusted’ more (Goebel, 2001b).

Strengthening

If tools agree on a certain class, then this module strengthens the output for that class (by a simple addition operation) (Goebel, 2001b).

Weakening

This module performs conflict resolution by discounting the entries of the classes in conflict (Goebel, 2001b).

Tie-breaking

Fault criticality and fault frequency information are used to break ties.

Evidence updating

Any evidentiary information available is used to modify the fused output. We note that evidence plays only a supporting role and is used only to reinforce a decision, not to weaken it. Evidence information is available in the form of the evidence vector and it is multiplied by the evidence matrix (a binary matrix which captures the relevance of an evidence item to a fault class).

Back scaling

This is the final module and it converts the internally coherent information to a form that is externally interpretable. A $[0,1]$

normalization and a dilation operation also constitute this stage.

Bottom up DoE

To deal with the combinatorial explosion, we started out with a conceptual problem of 2 diagnostic tools, 1 evidential tool, and 2 possible fault states. Each one of these states could take on 3 possible values (confidences). This amounts to 729 solutions. Even for this very constrained problem, 729 was a number too large for evaluation and hence had to be reduced. This was accomplished by performing a half factorial DoE (Fowlkes and Creveling, 1995) which cuts that number to 45 experiments as can be seen in Table 16. The result of the expert evaluation is seen in the columns labeled “Target 1” and “Target 2” which represent the diagnosis for Fault 1 and Fault 2, respectively. This set was then used to evaluate the first generation fusion algorithm in an iterative fashion as seen in Figure 11 where the SSE was used as error metric. The highlighted section of the algorithm (with labels provided next to it for better legibility) corresponds to a particular heuristic added (or re-visited) as outlined earlier.

Top-Down Monte Carlo

Next, we simulated the actual system fault states (“top down”) for the full input set with 3 diagnostic tools, 5 evidential tools, and 7 fault states to fine tune the algorithm and parameters. Each fault state was allowed to take on any value between 0 and 1 per architecture design requirements. For achieve realistic tool behavior, we utilized the output distributions described earlier to create the system response.

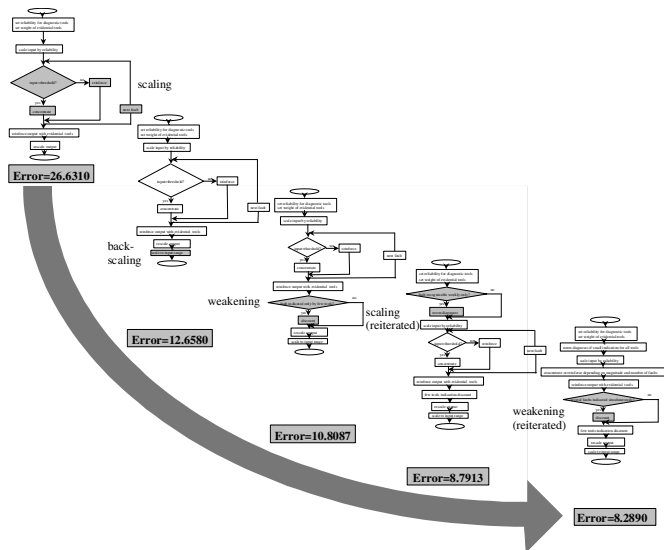


Figure 11 - Error reduction during sequential implementation phase of heuristics using DoE set

The procedure described is repeated many times and the results are compiled in confusion matrix format. The next step involves the evaluation of the results. If necessary, the parameters and/or algorithm for a particular heuristic are adjusted, and the process is repeated until desired system response is obtained.

CONCLUSIONS

We examined the design of classifier fusion algorithm development where both the classifiers and the fusion algorithm are unknown. To properly guide algorithm development, it is desirable to evaluate algorithm performance throughout its design, to guide development by a two-stroke design of experiment setup and to evaluate how different classifier distributions impact the fused outcome.

Specifically, for the design, we employed a dual approach that first extracted and used expert knowledge to design the individual layers of the fusion tool and to ensure that error reduction was achieved for each layer. Then, a large-scale simulation was exercised to exhaustively test the input scenarios and the fusion tool’s response. The two approaches, here called bottom-up DoE and top-down Monte Carlo, do not always need be engaged in the dual fashion. In some situations, the bottom-up DoE may be more appropriate in isolation, in others the top-down Monte Carlo simulation. The test set obtained from the bottom-up DoE has the potential to be usable for validation of a wide variety of diagnostic fusion situations because it is not application dependent. In addition, we view the IFM modules developed as applicable not only to decision fusion. Rather, they might prove equally useful for processing of information on the feature or data level.

We also described a correlation coefficient that measures the correlation between n classes for crisp as well as for soft class assignment. The correlation coefficient can be used for classifier selection, classifier simulation, and fusion. As expected, the bi-variate uniform classifier model performs worst while the one-sided Gaussian performs best. If possible, one should attempt to get an understanding of the output distribution (which is likely bi-variate Gaussian) and model it accordingly. Where this is not possible, the bi-variate uniform distribution gives a reasonable lower bound for the performance while the one-sided Gaussian tends to be somewhat too optimistic. One caveat with using the bi-variate model revolves around the possibly unknown spread and mean. Larger variations here may distort true performance to some degree. Other classifier output models can of course be used such as the uni-variate Gaussian distribution. Such a distribution is expected to produce results that are even more conservative than the bi-variate uniform distribution since the decision threshold is in an area with poor partition properties.

For the classifier fusion system, we proposed a layered weight manipulation scheme. This approach helps in situation with conflicting classifier information, temporal information discord (where the estimate of different tools is considerably separated in time), differences in information updates (where the classifiers are updated at different rates), fault coverage discrepancies, and integration of a priori performance specifications, among other things. If a hierarchical model is not desired or if some layers are not applicable, individual

layers can be used separately or in any combination for fusion tasks. Approach useful for sw development which can be expressed in some kind of (error) metric.

While the overall fusion output achieves substantial gains with all models used, the resulting fused performance may differ to some degree depending on the classifier output model used for simulation. Whereas the variation shown here does not seem to be very large, it must be noted that such differentiation in the higher accuracy ranges can make a substantial difference with respect to either meeting or not meeting specific performance goals, in particular since any gain at this level is typically exponentially harder to achieve than gains at the lower performance level.

ACKNOWLEDGMENTS

This research was in part supported by DARPA project MDA 972-98-3-0002.

REFERENCES

- Ashby, M., and Scheuren, W. (2000), "Intelligent Maintenance Advisor for Turbine Engines (IMATE)", Proceedings of the IEEE Aerospace Conference, 11.0309, 2000.
- Bradley, A.P., (1997), "The use of the area under the ROC curve in the evaluation of machine learning algorithms", Pattern Recognition, Vol.30, No.7, pp1145-1159.
- Downey, T.J., Meyer, D.J., Price, R.K., and Spitznagel, E.L., (1999), "Using the receiver operating characteristic to assess the performance of neural classifiers", IJCNN '99 – International Joint Conference on Neural Networks, Vol.5, pp3642-3646.
- Egan, J.P. (1975), *Signal detection theory and ROC analysis*, Series in Cognition and Perception. New York: Academic Press.
- Fowlkes, W. and Creveling, C. (1995), *Engineering Methods for Robust Product Design*, Addison-Wesley Publishing Company, Reading, MA.
- Fishman, G. (1996), *Monte Carlo : concepts, algorithms, and applications*. Springer-Verlag, New York.
- Goebel, K., (2000), "Decision Forgetting and Decision Smoothing for Diagnostic Information Fusion in Systems with Redundant Information", *Proc. SPIE: Sensor Fusion: Architectures, Algorithms, and Applications IV*, Vol. 4051, pp. 438-445.
- Goebel, K., Krok, M., and Sutherland, H., (2000), "Diagnostic Information Fusion: Requirements Flowdown and Interface Issues", *Proc. IEEE Aerosense Conference*, 11.0303.
- Goebel, K. (2001a) "Architecture and Design of a Diagnostic Information Fusion Tool", *AIEDAM: special edition AI in Equipment Service*, vol. 15 no. 4, pp. 335-348.
- Goebel, K., (2001b) "Conflict Resolution using Strengthening and Weakening Operations in Decision Fusion", *Proc. 4th Annual Conf. Information Fusion, Fusion 2001*, pp. ThA1-19 - ThA1-25, 2001.
- Goebel K., and Mysore, S., (2001). "Taking Advantage of Misclassifications to Boost Classification Rate in Decision Fusion", *Proc. SPIE: Sensor Fusion: Architectures, Algorithms, and Applications V*, pp. 11-20.
- Goebel, K., Yan, W. and Cheetham, W. (2002), "A Method to Calculate Classifier Correlation for Decision Fusion", *Proceedings of IDC 2002*, Adelaide, Australia. pp. 135-140.
- Goebel, K. (2003), "Sensitivity of Fusion Performance to Classifier Model Variations", *Proceedings of SPIE, Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications VII*, pp. 39-46.
- Hall, D. and Garga, A. (1999), "Pitfalls in Data Fusion (and How to Avoid Them)". *Proc. Second Int. Conf. Information Fusion (Fusion '99)*, pp. 429-436, 1999.
- Hathaway, R., Bezdek, J., and Pedrycz, W. (1996), "A Parametric Model for Fusing Heterogeneous Fuzzy Data", *IEEE Trans. on Fuzzy Systems*, Vol. 4, No. 3, pp. 270-281.
- Ho, T., Hull, J., and Srihari, S. (1994), "Decision Combination in Multiple Classifier Systems", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No.1, 1/1994.
- Petrakos, M., Kannelopoulos, I., Benediktsson, J., and Pesaresi, M. (2000), "The Effect of Correlation on the Accuracy of the Combined Classifier in Decision Level Fusion", *Proceedings of IEEE 2000 International Geoscience and Remote Sensing Symposium*, Vol. 6, 7/2000.
- Sobol, I. (1994), *A primer for the Monte Carlo method*. CRC Press, Boca Raton, FL.
- Yan, W. Goebel, K., and Li, J. (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.