NOVEL CLASSIFIER FUSION APPROAHCES FOR FAULT DIAGNOSIS IN AUTOMOTIVE SYSTEMS

Kihoon Choi, Satnam Singh, Anuradha Kodali, and Krishna R. Pattipati
Dept. of ECE
University of Connecticut
Storrs, CT 06269, U.S.A.
krishna@engr.uconn.edu

John W. Sheppard
Dept. of CS
The Johns Hopkins University
3400 N. Charles St.
Baltimore, MD 21218, U.S.A

Setu Madhavi Namburu, Shunsuke Chigusa, Danil V. Prokhorov, and Liu Qiao
Toyota Technical Center
1555 Woodridge, RR #7
Ann Arbor, MI 48105, U.S.A.
setumadhavi.namburu@tema.toyota.com

Abstract – Faulty automotive systems significantly degrade the performance and efficiency of vehicles, and oftentimes are the major contributors of vehicle breakdown; they result in large expenditures for repair and maintenance. Therefore, intelligent vehicle health-monitoring schemes are needed for effective fault diagnosis in automotive systems. Previously, we developed a data-driven approach using a data reduction technique, coupled with a variety of classifiers, for fault diagnosis in automotive systems [1]. In this paper, we consider the problem of fusing classifier decisions to reduce diagnostic errors. Specifically, we develop three novel classifier fusion approaches: class-specific Bayesian fusion, joint optimization of fusion center and of individual classifiers, and dynamic fusion. We evaluate the efficacies of these fusion approaches on an automotive engine data. The results demonstrate that dynamic fusion and joint optimization, and class-specific Bayesian fusion outperform traditional fusion approaches. We also show that learning the parameters of individual classifiers as part of the fusion architecture can provide better classification performance.

INTRODUCTION

Modern automobiles are being equipped with increasingly sophisticated electronic systems. Operational problems associated with degraded components, failed sensors, improper installation, poor maintenance, and improperly implemented controls affect the efficiency, safety, and reliability of the vehicles. Failure frequency increases with age and leads to loss of comfort, degraded operational efficiency, and increased wear and tear of vehicle components. An intelligent on-board fault detection and diagnosis (FDD) system can ensure uninterrupted and reliable operation of vehicular systems, and aid in vehicle health management. In particular, faulty engine systems significantly degrade the performance and efficiency of vehicles, and oftentimes are the major causes of vehicle breakdown; these result in large expenditures for repair and maintenance. Therefore, intelligent vehicle health-monitoring schemes are needed for effective fault diagnosis in automotive systems.

What does one do when a mathematical model (for model-based approach) or a cause-effect graph model of system failures and their manifestations (for a knowledge-based approach) is not available for fault diagnosis? Data-driven approach for FDD is an alternative when system-monitoring data is available. Owing to its simplicity and adaptability, customization of a data-driven approach does not require an in-depth knowledge of the system. A data-driven approach to FDD has a close relationship with pattern recognition, wherein one learns classification rules directly from the data, rather than using analytical models or a knowledge-based approach. This approach is attractive when one has difficulty in developing an accurate system model.

In our previous research [1], we considered a data-driven approach to fault diagnosis in an automotive engine system. In order to diagnose the faults of interest in the engine system, we employed several classifiers for fault isolation. These include: multivariate statistical techniques exemplified by the principal component analysis (PCA) ([2], [3]) and linear/quadratic discriminant

analysis (L/QDA) [4], and pattern classification techniques epitomized by the support vector machines (SVM) [5], probabilistic neural network (PNN), and the *k*-nearest neighbor (KNN) classifier [4]. We employed multi-way partial least squares (MPLS[1]) as a data reduction technique to accommodate the processed information (i.e., transformed data) within the limited memory space available in the electronic control units (ECUs) for control and diagnosis. Adaptive boosting (AdaBoost) [6] was used to improve the classifier performance. We validated and compared the accuracies and memory requirements of various fault diagnosis schemes. We successfully applied the FDD scheme to an automotive engine, and showed that it resulted in significant reductions in computation time and data size without loss in classification accuracy.

It has been well recognized that typical pattern recognition techniques, which focus on finding the best single classifier, have one major drawback [7]: any complementary discriminatory information that other classifiers may provide is not tapped. Classifier fusion appears to be a natural step when a critical mass of knowledge for a single classifier has been accumulated [8]. The objective of classifier fusion is to improve the classification accuracy by combining the results of individual classifiers. The fusion also allows analysts to use the strengths and weaknesses of each algorithm to reduce the overall classification error. Classifier fusion has been widely studied in such diverse fields as image segmentation, data mining from noisy data streams, credit card fraud detection, sensor networks, image/speech/handwritten recognition, fault diagnosis, etc [9].

In this paper, we propose three novel approaches to classifier fusion. These were motivated by our previous work on multi-target tracking and distributed *M*-ary hypothesis testing [10-11] and on dynamic multiple fault diagnosis (DMFD) [12-13]. The first approach led to a class-specific Bayesian approach to classifier fusion which exploits the fact that different classifiers can be good at classifying different fault classes. The second approach, motivated by the fact that the decision rules of fusion center and individual classifiers are coupled [11], involves the joint optimization of fusion center and of individual classifier decision rules. Given the classifiers, we develop the necessary conditions for the optimal fusion rule, taking into account costs of decisions. The SVM, PNN, KNN, and PCA classifiers are used to obtain the *posterior* probabilities of class labels. These probabilities are then combined by our proposed fusion approaches. The third approach involves dynamic fusion of classifiers, where classifier outputs are combined over time. In this paper, our primary focus is on evaluating how effectively our proposed classifier fusion approaches can reduce the diagnostic error, as compared to traditional fusion methods and the individual classifiers.

The paper is organized as follows. A brief overview of previous fusion research is provided in the next section. Then, an overview of diagnostic and fusion process, including our proposed approaches, follows. Simulations and results are discussed in the next two sections. Finally, we conclude the paper with a summary and directions for future research.

PREVIOUS FUSION RESEARCH

The area of classifier fusion has been investigated extensively over the past fifteen years. Fusion involves manipulating an ensemble of learners to improve the overall classification accuracy. This work has produced numerous techniques, which can be decomposed into five categories: classifier selection, combination of classifier outputs, sampling of classifier training data, manipulation of classifier outputs, and classifier feature selection. Classifier fusion techniques are categorized in Table 1, and a brief explanation of each technique follows.

Classifier Selection

Classifier selection endeavors to choose the best classifier for a given task. This method assumes that a classifier is an expert on a subset of the feature space. The two approaches to classifier selection, viz., static and dynamic, differ in terms of how the subset is calculated. Static selection determines *a priori* a competent classifier for each region based on training data. The more popular dynamic classifier selection chooses the classifier according the certainty of the decision [15]. An adaptive *k*-nearest neighbor rule for dynamic classifier selection is suggested in [16]. Kuncheva suggested a hybrid between dynamic classifier

[1] This algorithm was used to convert the 3D matrix data (samples x measurements x time), which we collected from CRAMAS® [35] into a 2D matrix (samples x features) as a data reduction technique.

Table 1. Overview of Fusion Techniques

Fusion Category	Techniques
Classifier Selection	Static, Dynamic
Combining Classifier Outputs	• Single Label Fusion: Voting, naïve Bayes, Decision Trees • Class Rankings: Borda Count, Highest Rank • *Posterior* Probability: Sum, Min, Max, Product, [20] Median, Decision Templates [21], Dempster-Schafer [22]
Sampling Training Data	Bagging, Boosting, AdaBoost
Manipulating Classifier Outputs	ECC
Classifier Feature Selection	Random Selection, Non-random Selection, GEFS, Hill Climbing

selection and combination of classifier outputs [17].

Combining Classifier Outputs

Combining classifier outputs is the most frequently used fusion technique. It combines the outputs of each classifier in the ensemble to determine a final decision. There are three types of classifier outputs: single class label, ranking of classes, and soft-decisions or *posterior* probabilities. Techniques using single class labels include voting, naïve Bayes, and decision trees. Kuncheva investigated several combination techniques in [8], including a discussion of voting techniques. In [18], the accuracy of naïve Bayes fusion for dependent classifiers is studied. An application of decision tree combination in the context of gas turbine fault isolation is provided in [19]. Combinational techniques operating on a set of ranked classes are discussed in [20]. Soft decision combination techniques operate on classifiers' *posterior* probabilities.

Sampling Training Data

Sampling training data has seen considerable success in various applications [21]. These algorithms create an ensemble of classifiers by training the classifiers on different samples of the training data. The two most common forms are bagging and boosting [25]. Bagging randomly samples the data set with replacement to create different training sets for each ensemble classifier. AdaBoost, the most popular rendition of boosting, uses weights for each training pattern that are updated at each iteration of the algorithm to focus on the more difficult patterns to classify [26]. A comparison of these techniques is provided by Bauer [25].

Manipulation of Classifier Outputs

In this category, the class labels are used to train an ensemble of classifiers to improve the performance. A representative example is the error correcting codes (ECC) matrix used by Dietterich and Bariki to improve the separation of output class signatures [26]. Each row of ECC, termed the codeword, represents a class, and the collection of words constitutes the code matrix. There exist many techniques for the development of the code matrices [8, 26]. The dependence among code words' bit errors is discussed in [27].

Classifier Feature Selection

Different subsets of features may offer different performance on the same data. This class of classifiers exploit this fact to create diversity in a classifier ensemble and, consequently, to improve classifier performance. Numerous techniques aimed at feature selection are outlined in [8]. In [28], the genetic ensemble feature selection (GEFS) algorithm is introduced using entropy as a diversity measure. Several search strategies, including ensemble forward sequential selection (EFSS), ensemble backward sequential selection (EBSS), a genetic algorithm (GA), and hill climbing, are discussed in [29] and are evaluated on data from the UCI repository.

DIAGNOSTIC AND FUSION PROCESS OVERVIEW

A block diagram of the diagnostic and classifier fusion schemes in our proposed approach is shown in Figure 1. The proposed fusion scheme is a three-step process: data reduction (raw data is also considered), fault isolation via individual classifiers, and classifier fusion with and without parameter optimization.

Data Reduction

Due to memory-constrained ECUs of automotive systems, intelligent data reduction techniques for on-board implementation of data-driven classification techniques are needed. Traditional

methods of data collection and storage capabilities often become untenable because of the increase in the number of observations (measurements), but mainly because of the increase in the number of variables associated with each observation ("dimension of the data") [30]. Using data reduction techniques, the entire data is projected onto a low-dimensional space, and the reduced space often gives information about the important structure of the high-dimensional data space. Among widely used dimension reduction techniques, an MPLS-based data reduction technique was examined, and we found that it is very efficient in reducing the data size (by a factor of 2000). Consequently, the classification algorithm could be embedded in existing ECUs with limited memory [1].

Fault Isolation

The following classification techniques are used as individual classifiers for fault isolation:

- Pattern classification techniques: SVM, PNN, and KNN
- Multivariate statistical technique: PCA

A brief explanation of each technique follows.

Support Vector Machines (SVM)

Support vector machines transform the data to a higher dimensional feature space, and find an optimal hyperplane that maximizes the margin between the classes [31]. There are two distinct features of SVM. One is that it is often associated with the physical meaning of data, so that it is easy to interpret, and the other one is that it requires only a small amount of training data. A kernel function is used for fitting non-linear models by transforming the data into a higher dimension before finding the optimal hyperplane. In this paper, we employed a radial basis function to transform the data into the feature space.

Probabilistic Neural Networks (PNN)

The probabilistic neural network (PNN) is a supervised method to glean distribution functions from data. In the recall mode, these functions are used to estimate the likelihood of an input measurement vector being part of a learned category, or class. The learned patterns can also be weighted, with *a priori* probability (relative frequency) of each category and misclassification costs, to determine the most likely class for a given input vector. If the relative frequency of the

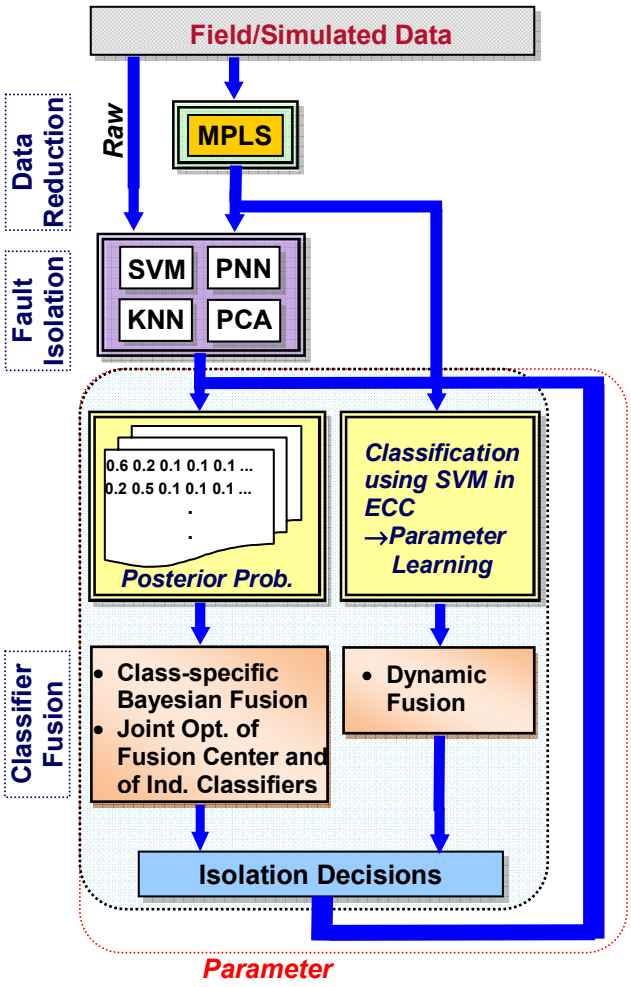

Figure 1. Block Diagram of Proposed FDD Fusion Scheme

categories is unknown, then all the categories are assumed to be equally likely, and the determination of category is solely based on the closeness of the input feature vector to the distribution function of a class.

k-Nearest Neighbor (KNN)

The *k*-nearest neighbor classifier is a simple non-parametric method for classification. Despite the simplicity of the algorithm, it performs very well, and is an important benchmark method [32]. KNN-based classifier requires a metric *d* and a positive integer *k*. A new input vector $\underline{x} \in \Re^n$ is classified using a subset of *k*–feature vectors that are closest to $\underline{x}$ with respect to the given metric *d*. The input vector $\underline{x}$ is then assigned to the class that appears most frequently within the *k*–subset. Ties can be broken by choosing an odd number for *k* (e.g., 1, 3, 5, etc.). Mathematically this can be

viewed as computing the *a posteriori* class probabilities $P(c_j|\underline{x})$ via,

$$P\left(c_j \mid \underline{x}\right) = k_j p\left(c_j\right) \Big/ k \tag{1}$$

where k_j is the number of vectors belonging to class c_j within the subset of k vectors. A new input vector $\underline{x}$ is assigned to the class c_j with the highest *a posteriori* class probability $P(c_j|\underline{x})$.

Principal Component Analysis (PCA)

Principal component analysis transforms correlated variables into a smaller number of uncorrelated variables, called principal components. PCA calculates the covariance matrix of the training data, and the corresponding eigenvalues and eigenvectors. The eigenvalues are then sorted, and the vectors (called scores) with the highest values are selected to represent the data in a reduced dimensional space. The number of principal components is determined by cross-validation [33]. The model of PCA is:

$$X = \sum_{f=1}^{U} \underline{t}_f \underline{p}_f^T + E \tag{2}$$

where E is the residual matrix, and U is the number of principal components. The loading vectors ($\underline{p}_f$) are orthonormal and provide the directions with maximum variability. The score vectors ($\underline{t}_f$) from the different principal components are the coordinates of the objects in the reduced space. Nonlinear iterative partial least squares (NIPALS) [33] algorithm is used to perform the PCA. A classification of a new test pattern is done by obtaining its predicted scores and residuals. If the test pattern is similar to a specific class in the training data, the scores will be located near the origin of the reduced space, and the residual should be small. The distance of test data from the origin of the reduced space is measured by the Hotelling statistic [34].

Fusion Approaches

We discuss three approaches to fusion: Class-specific Bayesian fusion, joint optimization of fusion center and of individual classifiers, and dynamic fusion.

Class-specific Bayesian Fusion

We let d_k (k = 1,2, $\cdots$,L) be the set of models (classifiers) with C classes, and let the targets be $\{\underline{t}_i\}$ (i=1, 2 $\cdots$,N). Let Z ($\underline{z}_i$: i = 1, 2, $\cdots$, N) and $\underline{x}$ be the set of training patterns and the test pattern, respectively. The *posterior* probability of class j from classifier k for the pattern $\underline{x}$ is denoted by $e_{jk}(\underline{x})$. Here, we make use of the fact that different classifiers are good at classifying different fault classes. Formally,

$$\begin{aligned} P(c_j \mid \underline{x}, Z) &= \sum_{M_k} P(c_j, d_k(c_j) \mid \underline{x}, Z) \\ &= \sum_{M_k} P(c_j \mid \underline{x}, Z, d_k(c_j)) P(d_k(c_j) \mid Z) \\ &= \sum_{M_k} e_{ij}(\underline{x})\, P(d_k(c_j) \mid Z) \end{aligned} \tag{3}$$

Note that

$$\begin{aligned} P(d_k(c_j) \mid Z) &= P(Z \mid d_k(c_j)) P(d_k(c_j)) \Big/ P(Z) \\ &= \frac{P(Z \mid d_k(c_j)) P(d_k(c_j))}{\sum_{l=1}^{L} P(Z \mid d_l(c_j)) P(d_l(c_j))} \end{aligned} \tag{4}$$

We can initialize $P(d_k(c_j))=1/L$ assuming all classifiers perform equally or make it proportional to the overall accuracies of classifiers, i.e.,

$$P(d_k(c_j)) = a_k(c_j) \Big/ \sum_{l=1}^{L} a_l(c_j) \tag{5}$$

where $a_k(c_j)$ is the accuracy of j^{th} class in classifier k. Evidently,

$$\ln P(Z \mid d_k(c_j)) = \sum_{i=1}^{N} t_{ij} \ln e_{jk}(\underline{z}_i) \tag{6}$$

This simply sums the logs of *posterior* probabilities for the target class over all the samples corresponding to class c_j. Larger this number (closer to zero because the sum is negative), better is the classifier. So, the numerator of $P(d_k(c_j)|Z)$ can be evaluated in log form as:

$$\begin{aligned} n_{kj} &= \ln P(Z \mid d_k(c_j)) + \ln P(d_k(c_j)) \\ &= \sum_{i=1}^{N} t_{ij} \ln e_{jk}(\underline{z}_i) + \ln P(d_k(c_j)) \end{aligned} \tag{7}$$

$$\text{Then,} \quad P(d_k(c_j) \mid Z) = \exp(n_{kj}) \Big/ \sum_{l=1}^{L} \exp(n_{lj}) \tag{8}$$

Joint Optimization of Fusion Center and of Individual Classifiers

Assume that there are L classifiers, a fusion center, C fault classes, and the N targets $\{\underline{t}_i\}$. The decisions of individual classifiers are denoted by $\{u_k\}^L_{k=1}$while that of fusion center by u_0: The classification rule of k^{th} classifier is $u_k \in \{1, 2, ..., C\} = \gamma_k(\underline{x})$ and that of fusion is center $u_0 \in \{1, 2, ..., C\} = \gamma_0(u_1, u_2, ..., u_L)$. The fusion center must decide which one of the classes is correct based on its own data and the evidence from the L classifiers. The prior probabilities of each class c_j is $P_j=N_j/N$,

where N is number of training patterns of class c_j. We let $J(u_0,c_j)$ be the cost of decision u_0 by the committee of classifiers when the true class is c_j. The problem is to find the joint committee strategy $\gamma_c=(\gamma_0, \gamma_1, \gamma_2, ..., \gamma_L)$ such that the expected cost $E\{J(u_0,c_j)\}$ is a minimum, where E denotes expectation over $\{\underline{x}, u_k, 0 \le k \le L\}$ and $\{c_j, 1 \le j \le C\}$. The necessary conditions of optimality for the optimal decision rules are given by [11]:

$$\gamma_0 : u_o = \arg \min_{d_0 \in \{1, 2, \cdots, C\}} \sum_{j=1}^{C} \left\{ P\left(u_1, u_2, \cdots, u_L \mid c_j\right) P_j J\left(d_0, c_j\right) \right\} \quad (9)$$

The key problem here is to obtain $P(u_1, u_2, ..., u_L|c_j)$. For C classes, this would involve building a table of C^L entries, which is clearly intractable in practice. The probability computation is simplified by correlating each classifier with respect to only the best classifier from the training data. The best classifier may change during iterations.

$$P\left(u_1, u_2, \cdots, u_L \mid c_j\right) = \prod_{k=1}^{L} P\left(u_k \mid c_j, u_1, u_2, \cdots, u_{L-1}\right) \approx P\left(u_{best} \mid c_j\right) \prod_{\substack{k=1 \\ k \neq best}}^{L} P\left(u_k \mid c_j, u_{best}\right) \quad (10)$$

We can estimate $P(u_k|c_j, u_{best})$ by

$$P\left(u_k = d_k \mid c_j, u_{best} = h\right) = e_{d_k jh} \quad (11)$$

where e_{dkjh} is the proportion of time classifier k makes decision d_k when the true class is c_j and the best classifier makes decision h for d_k=1,2, ⋯ C; j=1,2, ⋯,C; h=1,2, ⋯,C. Given the classifiers, one can also fuse the classifier decisions taking into account costs of decisions [10]. The key here is that the decision rules of fusion center and individual classifiers are coupled. Given that the fusion rule is fixed and the decision strategies of other classifiers are fixed, the k^{th} classifier makes its decision as follows:

$$\gamma_k : u_k = \arg \min_{d_k \in \{1, 2, \cdots, C\}} \quad (12)$$

$$\sum_{j=1}^{C} \sum_{u_0=1}^{C} P\left(u_0 \mid \underline{x}, u_k = d_k, c_j\right) p\left(\underline{x} \mid c_j\right) P_k J\left(u_0, c_j\right)$$

Since $p(\underline{x})$ is independent of j,

$$= \arg \min_{d_k \in \{1, 2, \cdots, C\}} \quad (13)$$

$$\sum_{j=1}^{C} \sum_{u_0=1}^{C} P\left(u_0 \mid \underline{x}, u_k = d_k, c_j\right) P_k\left(c_j \mid \underline{x}\right) J\left(u_0, c_j\right).$$

Eq. (13) can be simplified as:

$$= \arg \min_{d_k \in \{1, 2, \cdots, C\}} \sum_{j=1}^{C} \sum_{u_0=1}^{C} P_k\left(c_j \mid \underline{x}\right) \hat{J}\left(u_0, c_j\right) \quad (14)$$

where

$$\hat{J}\left(u_0, c_j\right) = \sum_{u_0=1}^{C} P\left(u_0 \mid \underline{x}, u_k = d_k, c_j\right) J\left(u_0, c_j\right)$$

$$\approx \sum_{u_0=1}^{C} P\left(u_0 \mid u_k = d_k, c_j\right) J\left(u_0, c_j\right). \quad (15)$$

$P(u_0|u_k=d_k, c_j)$ is computed from the training data by counting the fraction of times the fusion center made the decision u_0 when the local classifier k made the decision d_k and the true class is c_j in the previous iteration of the fusion rule. Also $P(c_j|\underline{x})$ is the estimate of *posterior* probability class c_j given the data $\underline{x}$.

Dynamic Fusion of Classifiers

Dynamic fusion process is based on an optimization framework that computes the most likely fault sequence over time. The dynamic fusion problem is a specific formulation of the DMFD problem [12-14]. In the DMFD problem, the objective is to isolate multiple faults based on test (classifier) outcomes observed over time. The dynamic fusion problem consists of a set of possible fault states in a system, and a set of binary classifier outcomes that are observed at each time (observation, decision) epoch. Evolution of fault states is independent, i.e., there is no direct coupling among the component states. The component states are coupled via classifier outcomes. Each classifier outcome provides information on a subset of the fault states (the entries with ones in the corresponding column of the ECC matrix). At each sample epoch, a subset of classifier outcomes is available. Classifiers are imperfect in the sense that the outcomes of some of the classifiers could be missing, and classifiers have missed-detection and false-alarm probabilities associated with them. Let the set of passed classifier outcomes be O_p and that of failed classifiers be O_f. Formally, we represent the dynamic fusion problem as DF={s, κ; D, O, ECC, P, A}, where S={s_1, s_2, …, S_M} is a finite set of M components (failure sources, classes) associated with the system. The state of a component m is denoted by $s_m(t)$ at discrete time epoch t, where $s_m(t)$=1if failure source s_m is present; $s_m(t)$=0, otherwise. Here κ = {0, 1, …, t,…,T} is the set of discretized observation epochs. The status of all component states at epoch t is denoted by $\underline{s}(t)$={$s_1(t)$, $s_2(t)$, ..., $s_M(t)$}. We assume that the probability distribution of initial state is known. The observations at each epoch are subsets of binary outcomes of classifiers O = {o_1, o_2, ..., o_L}, i.e., o_n(t) ∈ {*pass*, *fail*} = {0, 1}. Figure 2 shows the DMFD problem viewed as a FHMM. The hidden system fault state of m^{th} HMM at discrete time epoch t is denoted by $s_m(t)$. Each fault state $s_m(t)$ is modeled as a two-state HMM. Here, the true

states of the component states and of classifiers are hidden.

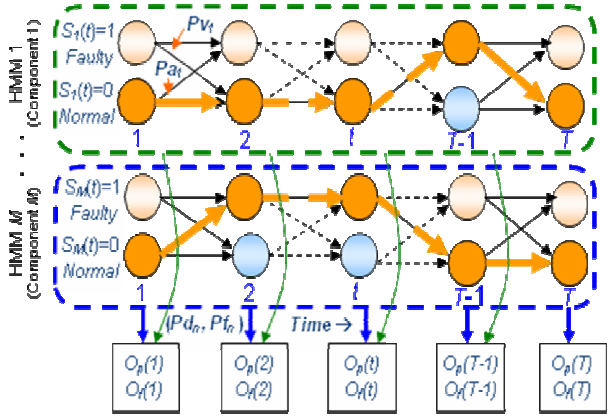

Figure 2. Dynamic Fusion Problem Viewed as a Factorial Hidden Markov Model (FHMM)

We also define the ECC matrix $ECC = [e_{mn}]$ as the diagnostic matrix (*D*-matrix), which represents the full-order dependency among failure sources and classifiers. Table 2 shows an ECC matrix as an illustrative example (8 failure sources and 4 classifiers).

Table 2. An Error Correcting Codes (ECC) Matrix

	d_1	d_2	d_3	d_4
s_1	0	1	0	0
s_2	1	1	1	1
s_3	0	0	1	0
s_4	0	1	1	0
s_5	1	1	0	1
s_6	0	0	1	1
s_7	1	1	1	0
s_8	1	0	0	1

Each component state is modeled as a two-state non-homogenous Markov chain. For each component state, e.g., for component s_i at epoch t, $A = \{Pa_m(t), Pv_m(t)\}$ denotes the set of fault appearance probability $Pa_m(t)$ and fault disappearance probability $Pv_m(t)$ defined as $Pa_m(t) = P(s_m(t) = 1 | s_m(t\text{-}1) = 0)$ and $Pv_m(t) = P(s_m(t) = 0 | s_m(t\text{-}1) = 1)$. For static classes, $Pa_m(t) = Pv_m(t) = 0$. When these probabilities are non-zero, the model provides the capability to handle intermittent faults. Here, $D = \{d_1, d_2, ..., d_L\}$ is a finite set of L available binary classifiers, where the integrity of the system can be ascertained. $P = \{Pd_n, Pf_n\}$ represents a set of probabilities of detection and false alarm, which is associated only with each classifier n. Formally, $Pd_n = P(o_n(t) = 1 | d_n(t) = 1)$ and $Pf_n = P(o_n(t) = 1 | d_n(t) = 0)$. The dynamic fusion problem is one of finding, at each decision epoch t, the most likely fault state candidates $\underline{s}(t) \in \{0, 1\}^m$, i.e., the fault state evolution over time, $S^T = \{\underline{s}(1), \underline{s}(2), ..., \underline{s}(T)\}$ that best explains the observed classifier outcome sequence O^T. We formulate this as one of finding the maximum *a posteriori* (MAP) configuration:

$$\hat{S}^T = \underset{S^T}{\operatorname{argmax}}\ P(S^T \mid O^T) \qquad (16)$$

This problem is solved using a primal-dual optimization framework as discussed in [12-14]. Note that the details of the dynamic fusion are provided in [12, 14]. In this work, we assume that the faults evolve independently, but they are coupled through the test outcomes via the D-matrix denoting fault-test dependencies.

SIMULATIONS

The proposed schemes are evaluated on an engine data set. A realistic automotive engine model is simulated under various fault conditions in a custom-built ComputeR Aided Multi-Analysis System (CRAMAS®) [35]. CRAMAS®, a vehicle engine simulator, which is used to develop vehicular ECUs, is a high-speed, multi-purpose, and expandable system. The system is subject to the following 8 faults: air flow sensor fault (misreading the air flow mass), leakage in air intake manifold (a hole in the intake tube), blockage of air filter (blocking the incoming air), throttle angle sensor fault (misreading the throttle angle), air/fuel ratio sensor fault (misreading the A/F ratio), engine speed sensor fault (misreading the engine speed), less fuel injection (delivering less fuel from the fuel pump), and added friction (increasing the friction in cylinders). It also contains the following 5 sensors: air flow meter reading, air/fuel ratio, vehicle speed, turbine speed, and engine speed. We collected observations of the 5 sensor readings (measurements) from the CRAMAS® hardware-in-the-loop simulator. The operating condition for the simulation was as follows: 2485 rpm (engine speed), 18° (pedal angle), and 86° (water temperature). For each class (fault), we performed simulations for 40 different severity levels (0.5 % ~ 20 %); each run is sampled at 2,000 time points with a 0.005-sec sampling interval. We apply our proposed fusion techniques to the data set and compare them to individual classifier performance. Here, 10 randomized data sets of 2-fold cross-validation were used to assess the classification performance.

RESULTS

We implemented and experimented with the three proposed fusion approaches on the CRAMAS® engine data. Widely used fusion techniques, the majority voting and the naïve Bayes were also evaluated and compared to our approaches. The diagnostic results, measured in terms of classification errors and testing times for the 8 faults, are shown in Table 3. Testing time was computed using Matlab® Software on a 2.3 GHz Intel Pentium 4 processor with 1 GB of RAM. We assume that time shown could be further reduced by a factor of 10 by implementing in the *C* language. As shown in Table 3, we not only achieved smaller fault isolation error, but also obtained significant data reduction (25.6 → 12.8 KB). Since data reduction improved classifier performance, the proposed fusion approaches were mainly evaluated on reduced data set.

Although majority voting, naïve Bayes, proposed Bayesian fusion and dynamic fusion (with no parameter optimization) approaches for this problem helped marginal classifiers (PNN, KNNs, and PCA) in reducing the correct isolation error, they were unable to overcome the best single classifier (SVM). However, the proposed Bayesian and dynamic fusion are comparable to the best single classification error, and outperformed the majority voting and naïve Bayes. Initially the results from joint optimization approach and class-specific Bayesian fusion were quite similar to that from best single classifier (virtually tied statistically). However, we were able to improve the classification performance using the joint optimization approach on marginal classifiers (PNN, KNN (*k*=3) and PCA) and then applying the majority voting on the fused decision and those from SVM and KNN (*k*=1). Specifically, the *posterior* probabilities from PNN, KNN (*k*=3), and PCA were fed to the joint optimization algorithm, and then SVM and KNN (*k*=1) were used for the majority voting with fused decisions from the joint optimization algorithm. Use of only majority voting provided poor isolation results; this implies that the joint optimization approach was definitely a contributor to the reduced error. We believe that this is because the joint optimization of fusion center and of individual classifiers increases the diversity of the classifier outputs, which is a vital requirement for reducing the diagnostic error. The Bayesian fusion result was also improved by optimally selecting classifiers. For the result shown in Table 3, three classifiers, SVM and KNNs (*k*=1,3) were selected and fused for the

Table 3. Comparison of Individual and Fusion Classification on Raw and Reduced Data

CRAMAS®	Method	Classification Error ± Std Dev in % [Training Time ± Std Dev / Testing Time ± Std Dev in sec]				
		SVM	KNN (*k*=1)	KNN (*k*=3)	PNN	PCA
Raw Data (25.6 MB)	Individual Classification	8.76 ± 2.54 [3.17 ± 0.19/ 3.21 ± 0.23]	12.94 ± 2.23 [20.68 ± 0.28/ 19.05 ± 2.41]	15.01 ± 2.42 [20.73 ± 0.11/ 19.13 ± 2.25]	14.83 ± 2.19 [23.53 ± 0.11/ 23.40 ± 3.15]	22.5 ± 2.32 [9.66 ± 0.35/ 8.32 ± 1.73]
Reduced Data via MPLS (12.8 KB)	Individual Classification	8.19 ± 2.53 [0.02 ± 0.04/ 0.02 ± 0.01]	12.75 ± 2.07 [0.05 ± 0.01/ 0.05 ± 0.02]	14.13 ± 2.54 [0.05 ± 0.02/ 0.05 ± 0.02]	14.06 ± 2.13 [0.35 ± 0.04/ 0.25 ± 0.02]	21.06 ± 3.73 [0.95 ± 0.03/ 0.77 ± 0.06]
	Majority Voting	12.06 ± 1.89 [NA / 0.03 ± 0.01]				
	Naïve Bayes	11.81 ± 1.96 [0.01 ± 0.01 / 0.19 ± 0.02]				
	Bayesian Fusion	8.62 ± 2.85 [0.20 ± 0.04 / 0.22 ± 0.01]				
	Joint Optimization	8.39 ± 2.02 [4.22 ± 0.38 / 3.38 ± 0.22]				
	Dynamic Fusion	9.9 ± 1.9 [8635.69 ± 2949.88 / 0.12 ± 0.01]				
	Bayesian Fusion with Classifier Selection	6.25 ± 2.29 [0.18 ± 0.01 / 0.19 ± 0.01]				
	Joint Optimization with Majority Voting	5.87 ± 2.04 [3.80 ± 0.4 / 3.4 ± 0.24]				
	Dynamic Fusion with Parameter Optimization	4.5 ± 1.6 [8635.69 ± 2949.88 / 0.12 ± 0.01]				

Bayesian fusion approach. For the dynamic fusion approach, we ran the fusion algorithm with a sampling interval of 0.5 seconds in order to suppress the noise in the data. Thus, we used a down sampling rate of 100, and obtained 20 time epochs for the dynamic fusion process. The results in Table 3 were obtained using 15 SVM classifiers, which are represented by the columns of the ECC matrix. The ECC matrix was generated using the Hamming code generation method [36]. The dynamic fusion achieved low isolation error results as compared to single classifier results. We experimented with two different approaches for *Pd* and *Pf* in the dynamic fusion process. The first approach used *Pd* and *Pf* learned from the *training data of individual classifiers*, while coarse optimization was applied to learn *Pd* and *Pf*, and the optimal parameters were *Pd* = 0.5~0.6 and *Pf* = 0~0.02 *when they are part of the dynamic fusion*. We found that the dynamic fusion approach with parameter optimization significantly reduces diagnostic error by 45.1% (as compared to single best classifier on reduced data).

Figure 3 shows that dynamic fusion with parameter optimization provided the most significant improvement and also the best in classification accuracy. Note that the training time for the dynamic fusion method depends on how many classifiers (ECC columns) are used. The more the number of classifiers, the larger is the training time.

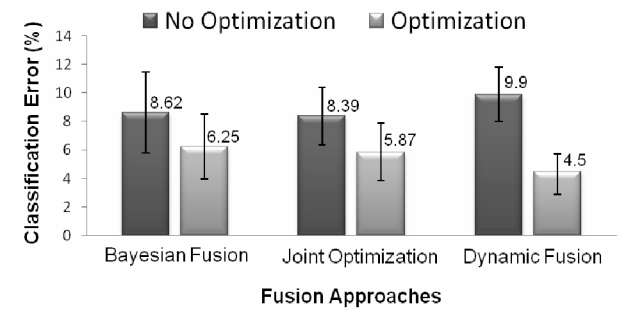

Figure 3. Comparison of Fusion Approaches with No Parameter Optimization and Optimization

Figure 4 illustrates all the approaches we considered in this paper and Pareto efficiency (dash line) [37] in terms of testing time vs. classification error. By indicating all of the potentially optimal approaches to the problem, analysts can obtain focused tradeoffs within the constrained set of classification error and testing time, rather than considering the full ranges of fusion approaches. Since our primary focus is on diagnostic error, the lower values are preferred to the higher values. The figure clearly shows that our dynamic fusion with parameter optimization is superior to all other approaches.

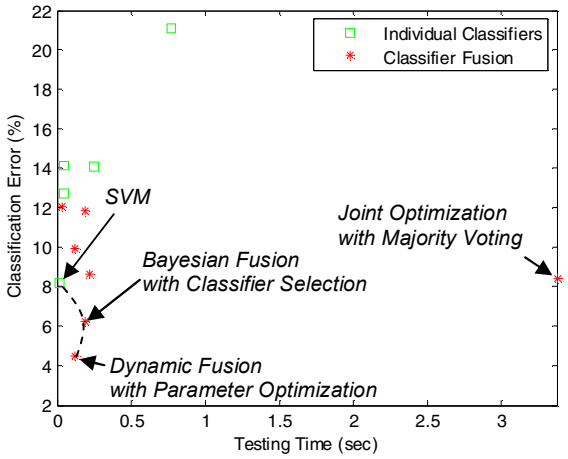

Figure 4. Comparison of Individual and Fusion Classifiers on Reduced Data

CONCLUSIONS

In this paper, we have proposed and developed three new approaches to classifier fusion. In addition to individual classifiers, such as the SVM, PNN, KNN, and PCA for fault isolation, *posterior* probabilities from these classifiers were fused by class-specific Bayesian fusion, joint optimization of fusion center and individual classifiers, and dynamic fusion. All the approaches were validated on the CRAMAS® engine data (raw and reduced data sets). Although in terms of diagnostic error, the results of the proposed approaches before applying optimization were quite close to the best single classifier performance, they are generally better than the majority voting and the naïve Bayes approaches. It confirms again that fusing marginal classifiers can increase the diagnostic performance substantially. We showed that classifier selection in the context of class-specific Bayesian fusion, majority voting among the best classifiers and fused marginal classifies, and parameter optimization in dynamic fusion significantly reduced the overall diagnostic error. The key empirical result here is that one needs to learn parameters as part of the fusion architecture (not standalone) to obtain the best classification performance from a team of classifiers. This is consistent with the finding in distributed detection that the individual sensors (classifiers in our case)

operate at a different operating point when part of a team (fusion in our case) than when they operate alone [38].

Our future research involves evaluating the proposed classifier fusion techniques on various data sets, such as real automotive field data, UCI repository, etc. Furthermore, we plan to explore and compare other fusion techniques that can be applied to fault diagnosis in automotive systems. For the dynamic fusion research, we also plan to explore relaxation of the independence assumption and solve the dynamic fusion problem when faults are dependent. Coupled hidden Markov models offer a promising platform for the solution of this problem [39]. We also plan to extend the joint optimization approach to correlated faults via Bayesian network/influence diagram framework.

ACKNOWLEDGEMENT

This work was supported by Toyota Technical Center at the University of Connecticut under agreement AG030699-02.

REFERENCES

[1] K. Choi, J. Luo, K. R. Pattipati, S. M. Namburu, L. Qiao, and S. Chigusa, "Data reduction techniques for intelligent fault diagnosis in automotive systems," *Proc. of the IEEE Autotestcon*, Anaheim, CA, September 2006.

[2] P. Nomikos and K. F. MacGregor, "Monitoring batch processes using multiway principal component analysis," *Amer. Inst. Chem. Eng.*, vol. 40, no. 8, pp. 1361-1375, 1994.

[3] J. Chen and K-C. Liu, "On-line batch process monitoring using dynamic PCA and dynamic PLS models," *Chem. Eng. Sci.*, vol. 57, no. 1, pp. 63-75, 2002.

[4] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, second edition, New York: Wiley-Interscience, 2001.

[5] C-W. Hsu and C-J. Lin, "A comparison of methods for multi-class support vector machines," *IEEE Tran. on Neural Networks*, vol. 13, no. 2, pp. 415-425, 2002.

[6] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," *Machine Learning: Proc. of the Thirteenth Inter. Conf.*, 1996.

[7] J. Kittler, "A framework for classifier fusion: Is it still needed?" [Online]. Available: http://www.ph. tn.tudelft.nl/Organisation/TC1 /pda/kittlerpda2001.pdf.

[8] L. I. Kuncheva, *Combining Pattern Classifiers*, John Wiley, 2004.

[9] D. Ruta and B. Gabrys, "An overview of classifier fusion methods," *Computing and Information Systems*, pp. 1-10, 2000.

[10] K. R. Pattipati and N. R. Sandell, Jr., "A unified view of state estimation in switching environments," *Proc. of the American Control Conference*, pp. 458-465, San Francisco, CA, June 1983.

[11] Z. B. Tang, K. R. Pattipati, and D. L. Kleinman, "A distributed M-ary hypothesis testing problem with correlated observations," *IEEE Tran. on Automatic Control*, vol. 37, no. 7, pp. 1042-1046, July 1992.

[12] S. Singh, K. Choi, A. Kodali, K. Pattipati, J. Sheppard, S. M. Namburu, S. Chigusa, D. V. Prokhorov and L. Qiao, "Dynamic multiple fault diagnosis problem formulations and solution techniques," *DX-07 International Workshop on Principles of Fault Diagnosis*, Nashville, TN, May 2007.

[13] S. Singh, S. Ruan, K. Choi, K. Pattipati, P. Willett, S. M. Namburu, S. Chigusa, D. V. Prokhorov and L. Qiao, "An optimization-based method for dynamic multiple fault diagnosis problem," *IEEE Aerospace Conference*, Big Sky, Montana, March 2007.

[14] S. Singh, K. Choi, A. Kodali, K. Pattipati, S. M. Namburu, S. Chigusa, D. V. Prokhorov, and L. Qiao, "Dynamic fusion of classifiers for fault diagnosis," *IEEE SMC Conference*, Montreal, Canada, October 2007.

[15] G. Giacinto and F. Roli, "Methods for dynamic classifier selection," *10th ICAP*, pp. 659, 1999.

[16] L. Didaci and G. Giacinto, "Dynamic classifier selection by adaptive k-nearest neighbor rule," [Online]. Available: http://ce.d lee.unica.it/en/publications/papers-prag/MCS-Conference-19.pdf.

[17] L. I. Kuncheva, "Switching between selection and fusion in combining classifiers: An experiment," *IEEE Trans. on Systems, Man and Cybernetics*: Part B, vol. 32, no. 2, April 2002.

[18] H. Altincay, "On naïve Bayesian fusion of dependent classifiers," *Pattern Recognition Letters*, vol. 26, pp. 2463-2473, 2005.

[19] W. Donat, K. Choi, W. An, S. Singh, and K. R. Pattipati, "Data visualization, data reduction and classifier fusion for intelligent fault detection and diagnosis in gas turbine engines," *Proc. of ASME Turbo Expo*, Montreal, Canada, May 2007.

[20] O. Melnik, Y. Vardi, and C-H. Zhang, "Mixed group ranks: Preference and confidence in classifier combination," *IEEE Tran. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, August 2004.

[21] [Online]. Available: http://www.cscs.umich.edu/~c rshalizi/notebo oks/ensemble-ml.html.

[22] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Tran. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, March 1998.

[23] L. I. Kuncheva, "Decision templates for multiple classifier fusion: An experimental comparison," *Pattern Recognition*, vol. 24, no. 2, pp. 299-314, 2001.

[24] L. Xu, A. Krzyzak, and CY Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Tran. on System, Man and Cybernetics*, vol. 22, no. 3, pp. 418-435, May/June 1992.

[25] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, variants," *Machine Learning*, vol. 36, pp. 105-142, 1999.

[26] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problem via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263-286, 1995.

[27] F. Masulli and G. Valentini, "An experimental analysis of the dependence among codeword bit errors in ECOC learning machines," *Neurocomputing*, vol. 57, pp. 189-214, 2004.

[28] D. W. Opitz, "Feature selection for ensembles," *Proc. of 16th National Conference on Artificial Intelligence*, 1999.

[29] P. Cunningham and J. Carney, "Diversity versus quality in classification ensembles based on feature selection," *11th European Conference on Machine Learning*, 2000.

[30] I. K. Fodor, " survey of dimension reduction techniques," [Online]. Available: http://www.llnl.gov/CASC/sapphire/p ubs/148494.pdf

[31] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167, 1998.

[32] J. K. Shah, *Sequential k-NN Pattern Recognition for Usable Speech Classification – A Revised Repor*t, Speech Processing Lab, Temple Univ., 2004.

[33] S. Wold, P. Geladi, K. Esbensen, and J. Ohman, "Principal component analysis," *Chemometrics and Intell. Lab. Sys.*, vol. 2, no. 1-3, pp. 37-52, 1987.

[34] P. Nomikos, "Detection and diagnosis of abnormal batch operations based on multi-way principal component analysis," *ISA Tran.*, vol. 35, no. 3, pp. 259-266, 1996.

[35] Fukazawa et al., "Development of PC-based HIL simulator CRAMAS," *FUJITSU TEN Technical Journal*, no. 20, 2003.

[36] R. W. Hamming, "Error detecting and error correcting codes," *Journal of Bell Sys. Tech.*, 29, 1950.

[37] M. J. Osborne and A. Rubenstein, *A Course in Game Theory*, MIT Press, 1994.

[38] A. Pete, K.R. Pattipati and D.L. Kleinman, "Optimal team and individual decision rules in uncertain dichotomous situations," *Public Choice,* 75, 1993, pp. 205-230.

[39] M. Brand, "Coupled hidden Markov models for modeling interacting processes," *Neural Computation*, November 1996.