Dynamic Fusion of Classifiers for Fault Diagnosis

Satnam Singh, Kihoon Choi, Anuradha Kodali, Krishna R. Pattipati,
Setu Madhavi Namburu, Shunsuke Chigusa, Danil V. Prokhorov and Liu Qiao

***Abstract*—This paper considers the problem of temporally fusing classifier outputs to improve the overall diagnostic classification accuracy in safety-critical systems. Here, we discuss dynamic fusion of classifiers which is a special case of the dynamic multiple fault diagnosis (DMFD) problem [1]-[3]. The DMFD problem is formulated as a maximum *a posteriori* (MAP) *configuration* problem in tri-partite graphical models, which is NP-hard. A primal-dual optimization framework is applied to solve the MAP problem. Our process for dynamic fusion consists of four key steps: (1) data preprocessing such as noise suppression, data reduction and feature selection using data-driven techniques, (2) error correcting codes to transform the multiclass data into binary classification, (3) fault detection using pattern recognition techniques (support vector machines in this paper), and (4) dynamic fusion of classifiers output labels over time using the DMFD algorithm. An automobile engine data set, simulated under various fault conditions [4], was used to illustrate the fusion process. The results demonstrate that an ensemble of classifiers, when fused over time, reduces the diagnostic error as compared to a single classifier and static fusion of classifiers trained over the entire batch of data. The results for sliding window dynamic fusion are also provided.**

I. Introduction

CLASSifier fusion has been widely investigated in diverse fields, such as image segmentation, data mining from noisy data streams, credit card fraud detection, sensor networks, image, speech and handwriting recognition, fault diagnosis, to name a few. In the literature, classifier fusion is variously referred to as classifier ensembles, consensus aggregation, decision fusion, committee machines, classifier selection, mixture of experts, etc. The objective of classifier fusion is to achieve better classification accuracy by combining the results of individual classifiers.

Our focus here is on combining class labels from multiple classifiers over time. The key motivation for performing dynamic classifier fusion in our application context is to improve the on-board diagnostic accuracy of safety-critical systems, such as aircraft, automobiles, nuclear power plants and space vehicles. An accurate on-board diagnostic process will ensure performability, maintainability and survivability of safety-critical systems.

The study reported in this paper was supported, in part, by the Toyota Technical Center and the Office of Naval Research under contract no. 00014-00-1-0101. Any opinions expressed in this paper are solely those of the authors and do not represent those of the sponsor.

S. Singh, K. Choi, A. Kodali and K. R. Pattipati, are with the University of Connecticut, Storrs, CT 06269, USA (phone: 860-486-2890; fax: 860-486-5585; e-mail: krishna@engr.uconn.edu).

S. M. Namburu, S. Chigusa, D. V. Prokhorov and L. Qiao are with Toyota Technical Center (TTC-TEMA), Ann Arbor, MI 48105, USA.

It is generally believed that classifier fusion may enhance the diagnostic accuracy in situations when the constituent classifiers have low correlations among their classification errors or, equivalently, more diversity among their outcomes [5]. Multiple classifiers can avoid the risk of picking the output of a single classifier, and, consequently, overcome the weaknesses of individual classifiers. If the individual classifiers are already performing well, then fusion accuracy is not expected to increase significantly, but the variability in its classification performance decreases.

In this paper, we formulate the dynamic classifier fusion problem as one of maximizing the *a posteriori* probability of a hidden state sequence given uncertain classifier outcomes over time. For simplicity of classifier fusion, we transform the data into binary classes by selecting the individual classifiers to correspond to the columns of an error correcting code (ECC) matrix [6]. In fault diagnosis area, we refer to classes as components and classifiers as tests. Thus, the binary classifiers (binary tests) correspond to the columns of the ECC matrix, and the components correspond to the rows of the ECC matrix. The ECC matrix may be viewed as a diagnostic matrix (D-matrix, diagnostic dictionary, reachability matrix), which defines the cause-effect relationship among fault sources/components (rows) and tests (columns).

Though not demonstrated experimentally in this paper, a major advantage of our multistage fusion architecture is that it allows the use of a heterogeneous set of classifiers over time. For example, in a scenario with 30 time epochs, the first 10 epochs may employ a data-driven classifier such as the SVM, a knowledge-based classifier (e.g., TEAMS-RT [7]) for the next 10 epochs, and for the last 10 epochs a model-based classifier (e.g., [8]). Our fusion approach provides a flexible framework to optimize diagnostic systems with respect to data pre-processing, number and type of classifiers, the ECC matrix, as well as the temporal complexity measured in terms of time epochs used for fusion.

II. Previous Research

In the literature, many techniques are proposed for classifier fusion. They can be divided into two categories: classifier combination and classifier selection. Classifier combination is an effective technique for combining independent classifiers with high accuracy and high diversity. Classifier combination can be applied to class labels, class rankings or confidence estimates on class labels. In [9]-[11], several methods are proposed for classifier combination, such as the hierarchical mixture of experts, voting methods,

behavior-knowledge space method, Borda count method, Bayesian fusion, fuzzy integrals, Dempster-Shaffer combination, and artificial neural networks, to name a few.

Classifier selection, as its name implies, chooses the best classifier for each test sample. A static classifier selection method decides on the best classifier *a priori* during training. The input patterns are partitioned, and the best classifier is nominated for each partition. While a static fusion method employs constant weights for each classifier based on training, a dynamic fusion method changes the weights of each classifier based on the observed test pattern [12]. For example, the distance of a test pattern to its nearest neighbor for each individual classifier may be used to compute the dynamic weights.

A prototypical classifier selection method is the decision templates approach. Decision templates are the estimated averages of the decision profiles (DP) of the samples of each class in the training set. The DP is a matrix of classifier outcomes where i^{th} row of the DP matrix contains output of i^{th} classifier and j^{th} column refers to the support from all the classifiers for class j. The fusion is performed by comparing the decision profile of a test set with the stored decision templates of the classes. A classifier is selected based on the elements of DP matrix. The process involves selecting the minimum value in each column of the DP matrix, and then declaring a classifier with the maximum value of these minimum values in each column as the best classifier. This method is used in [12] to classify meeting events.

Genetic algorithms are employed to select features in multiple classifier systems in [13]. Another approach is to estimate a local regression model for each partition of input data, and to dynamically decide on the combination function [9]. In [14], classification is performed by selecting the classifier with the highest classifier local accuracy (CLA) in a local region of the feature space. *A priori* and *a posteriori* selection methods are proposed to estimate CLA. In an *a priori* selection method, CLA is estimated as the ratio of the numbers of patterns correctly classified in the neighborhood to the total number of patterns in the neighborhood of the unknown test pattern. In an *a posteriori* selection method, CLA is estimated as the probability that a classifier assigns the test pattern to a particular class (e.g., as in a k-nearest neighbor method). A dynamic classifier selection (DCS) algorithm is proposed by selecting a CLA threshold, and rejecting classifiers below the threshold.

Kim et al. [15] proposed a dynamic integration system, which selects the best classifier from multiple base classifiers. The system focuses on learning the local region in which the classifier is the best. In [16], a feature-oriented dynamic classifier selection method is proposed for noisy data streams. Here, the evaluation set is split into subsets based on feature values of each pattern. Then, the classification accuracy of each base classifier is evaluated using these subsets during training.

TABLE I
ERROR CORRECTING CODE (ECC) MATRIX

	C_1	C_2	C_3	C_4
x_1	0	1	0	0
x_2	1	1	1	1
x_3	0	0	1	0
x_4	0	1	1	0
x_5	1	1	0	1

During testing, the feature values of a test pattern are used to select a subset and the concomitant best classifier for that subset. Next, we describe our dynamic fusion method.

III. Dynamic Fusion Process Overview

Our approach to dynamic fusion is shown in Figure 1. It involves four key steps: (1) data preprocessing (noise suppression, data reduction and feature selection) using data-driven techniques, such as multi-way partial least squares (MPLS) to perform data reduction, computing statistical moments, etc., (2) error correcting codes to transform the multiclass data into dichotomous choice situations (binary classification), (3) fault detection using pattern recognition techniques (e.g., support vector machines), and (4) fault isolation via dynamic fusion of classifier output labels over time using the DMFD algorithm. Next, we discuss each step of the dynamic fusion process in detail.

A. Feature Extraction or Data Pre-processing

Feature extraction involves signal processing methods such as wavelets, fast Fourier transforms (FFT) and statistical techniques to extract relevant information for diagnosing faults. In our experiments, we perform pre-processing using data reduction techniques, such as MPLS to transform the data to low-dimensional structures for implementation in limited memory electronic control units (ECUs) of an automotive system [4].

B. Error Correcting Codes (ECC) Matrix

The next step involves fault detection using binary classifiers corresponding to the columns of an ECC matrix. Error correcting codes are widely used in communications to decode messages sent over noisy channels by exploiting the redundancy in the transmitted code. We use an ECC matrix to project the data into a binary orthogonal space. Each column of the ECC matrix represents a classifier, and each row depicts a component or class. In the context of fault diagnosis, the ECC matrix can be viewed as a diagnostic matrix (D-matrix), which provides the cause-effect relationships between the faults (rows) and tests (columns). The ECC matrix provides a flexible and robust framework for combining the classifiers. For example, we can choose the

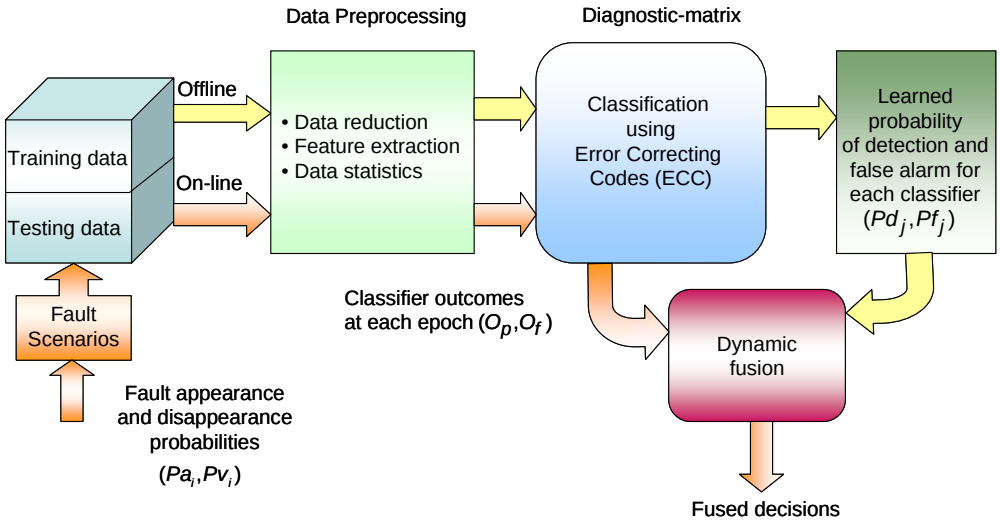

Fig.1. Overview of dynamic fusion process

TABLE II
CONFUSION MATRIX

		Estimated class	
		0	1
True class	0	N_{00}	N_{01}
	1	N_{10}	N_{11}

first column to be any model-based classifier, a second column as any data-driven classifier, and third column to be any knowledge-based classifier, and so on. Table I shows an example of the ECC matrix. A column and a row of the ECC matrix represents a classifier (C_j) and a component (x_i), respectively. For example, classifier C_1 considers the data corresponding to faults in components 2, and 5 as class 1, and data for other faulty components as class 0.

C. Fault Detection using the Support Vector Machine (SVM) Classifiers

The SVM introduced in [17], [18] is applied in areas such as handwritten digit recognition, anomaly detection in computers, text classification, fault diagnosis, etc. The SVM principle is to find a hyperplane that maximizes the separation between classes. SVM uses nonlinear pre-processing techniques ("kernel") to project the data from a low-dimensional space (input space) to a high-dimensional space (feature space). The linear operation in the feature space is equivalent to non-linear operation in the input space. It finds an optimal hyperplane in the high dimensional space using quadratic programming. To obtain the SVM, we need to specify the kernel parameter γ and cost relaxation parameter C . In this paper, these parameters are empirically computed. The probabilities of detection and false alarm (Pd_j , Pf_j) for all the SVM classifiers are learned from the training data by constructing the confusion matrix.

Table II shows the confusion matrix, where N_{ab} shows the number of patterns having true class $a \in \{0,1\}$ but having estimated class $b \in \{0,1\}$. The probabilities of detection and false alarm are computed from the sample statistics as

$$Pd_j = \frac{N_{11}}{N_{10} + N_{11}} \quad \text{and} \quad Pf_j = \frac{N_{01}}{N_{00} + N_{01}} .$$

D. Dynamic Fusion

During the testing phase, we generate fault scenarios and the corresponding test patterns according to component failure and recovery rates. We assume that each fault is intermittent. The testing data is processed through the classifiers (columns of the ECC matrix) to obtain the classifier outcomes, i.e., sets of passed and failed classifier outcomes $(O_p(k), O_f(k))$. These test outcomes are fed to the dynamic fusion block, along with the probability pairs (Pd_j, Pf_j) to obtain the fault isolation decisions.

IV. DYNAMIC MULTIPLE FAULT DIAGNOSIS (DMFD) PROBLEM

Our 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 dynamic multiple fault diagnosis problem (DMFD) [1]-[3]. 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 (component states, as in Fig. 2), and a set of binary classifier outcomes that are observed at each sample (observation, decision) epoch. Evolution of each component states is assumed to be independent. 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/false-alarm probabilities associated with them.

Formally, we represent the dynamic fusion problem as $DF = \{S, \kappa, C, O, ECC, P, A\}$, where $S = \{s_1, ..., s_m\}$ is a finite set of m components (failure sources) associated with the system. The state of component s_i is denoted by $x_i(k)$ at epoch k, where $x_i(k) = 1$ if failure source s_i is present; $x_i(k) = 0$, otherwise. Here, $\kappa = \{0, 1, ..., k, ...K\}$ is the set of discretized observation epochs. The status of all component states at epoch k is denoted by $\underline{x}(k) = \{x_1(k), x_2(k), ..., x_m(k)\}$. We assume that the initial state $\underline{x}(0)$ is known (or its probability distribution is known). The observations at each epoch are subsets of binary outcomes of classifiers $O = \{O_1, O_2, ..., O_n\}$, i.e., $O_j(k) \in \{pass, fail\} = \{0, 1\}$.

Figure 2 shows the dynamic fusion problem as a tri-partite graph at epoch k. Component states, classifiers and classifier outcomes represent the nodes of the digraph. Here, the true states of the component states are hidden. The true states of classifiers are also hidden because the classifiers are imperfect. We also define the ECC matrix $ECC = [e_{ij}]$ as the diagnostic matrix (D-matrix), which represents the full-order dependency among failure sources and classifiers. 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 k, $A = (Pa_i(k), Pv_i(k))$ denotes the set of fault appearance probability $Pa_i(k)$ and fault disappearance probability $Pv_i(k)$ defined as $Pa_i(k) = \Pr(x_i(k) = 1 \mid x_i(k-1) = 0)$ and $Pv_i(k) = \Pr(x_i(k) = 0 \mid x_i(k-1) = 1)$. Figure 3 shows fault appearance and disappearance mechanisms of the two-state HMM.

Here, $C = \{C_1, C_2, ..., C_n\}$ is a finite set of n available binary classifiers, where the integrity of the system can be ascertained. At each observation epoch, k, $k \in \kappa$, classifier outcomes upto and including epoch K are available, i.e., we let $O^K = \{O(k) = (O_p(k), O_f(k))\}_{k=1}^K$, where O^K is the set of observed classifier outcomes upto and including epoch K, with $O_p(k) (\subseteq O(k))$ and $O_f(k) (\subseteq O(k))$ as the sets of passed and failed classifier outcomes at epoch k, respectively. The classifiers are partially observed in the sense that outcomes of some classifiers may not be available, i.e., $(O_p(k) \bigcup O_f(k)) \subset O(k)$. In addition, classifiers exhibit missed detections and false alarms. $P = \{Pd_j, Pf_j\}$ represents a set of probabilities of detection and false alarm, which is associated only with each classifier C_j. Formally, $Pd_j = \Pr(o_j(k) = 1 \mid C_j(k) = 1)$ and $Pf_j = \Pr(o_j(k) = 1 \mid C_j(k) = 0)$. Figure 4 illustrates these probabilities.

The dynamic fusion problem is one of finding, at each decision epoch k, the most likely fault state candidates $\underline{x}(k) \in \{0,1\}^m$, i.e., the fault state evolution over time, $X^K = \{\underline{x}(1), ..., \underline{x}(K)\}$, that best explains the observed classifier outcome sequence O^K. We formulate this as one of finding the maximum *a posteriori* (MAP) configuration:

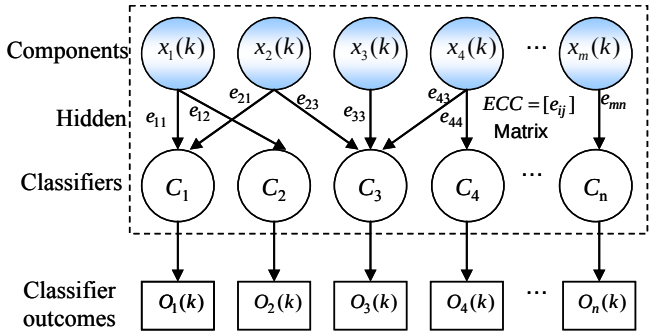

Fig. 2. Tri-partite graph for dynamic fusion problem

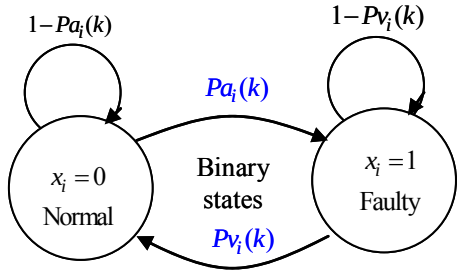

Fig. 3. Fault appearance and disappearance probabilities

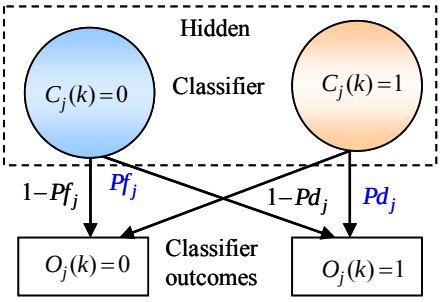

Fig. 4. Detection and false alarm probabilities

$$\hat{X}^K = \arg\max_{X^K} \Pr(X^K \mid O^K)$$

The NP-hard nature of the primal dynamic fusion problem motivates us to decompose it into a primal-dual problem using a Lagrangian relaxation approach. By defining new variables and constraints, the dynamic fusion problem reduces to a combinatorial optimization problem with a set of equality constraints. The constraints are relaxed via Lagrange multipliers. The relaxation procedure generates an upper bound for the objective function. The procedure of minimizing the upper bound via a subgradient or surrogate subgradient optimization produces a sequence of dual feasible and the concomitant primal feasible solutions to the dynamic fusion problem. Details of the DMFD algorithm, subgradient method and dynamic programming are provided in our previous papers [2],[3].

During the online monitoring of the system, the observations and potential fault sequences are usually very long hence in order to reduce the amount of computation and storage, the DMFD problem is solved using the sliding window method. The sliding window DMFD method solves the diagnostic problem over a set of observations. The window size is selected based on the performance criteria such as low diagnostic error and low false alarm rate.

V. Simulations and Results

A realistic automotive engine model of Toyota Camry 4-cylinder engine is simulated under various fault conditions in a custom-built ComputeR Aided Multi-Analysis System (CRAMAS®) simulator and controlled via a prototype ECU (Rtype) [4]. We simulated the engine model under eight faults conditions inserted manually via the CRAMAS® control panel. Eight faults inserted were: air flow sensor fault, fault leakage in air intake system, blockage of air filter, throttle angle sensor fault, less fuel injection, added engine friction, air/fuel sensor fault and engine speed sensor fault. We collected measurements from five sensors: air flow meter, air/fuel ratio, vehicle speed, turbine speed and engine speed. For each fault component, we performed simulations for 40 different severity levels (0.5 % to 20 %); each run is sampled at 5 ms sampling interval with 2,000 time points (10 s of data). The fault severity level refers to deviation of the sensor value from its nominal value e.g. 10% severity level of air flow sensor fault refers to the change in the air flow from its nominal value by 10%.

Each fault class contains 40 patterns over a time period of 2000 time epochs. This data is divided into training and testing data using 10 randomized datasets of 2-fold cross-validation. The training and testing data is reshaped into 2 dimensional (2D) so that a support vector machine (SVM) classifier can be used. In order to suppress the noise in the data, we run the fusion algorithm with a sampling interval of 0.5 seconds. Thus, we use a down sampling rate of 100, and obtain 20 time epochs for the dynamic fusion process. We employed multi-way partial least squares (MPLS) method to perform data reduction over a window of 100 samples. The MPLS-based data reduction technique achieves high classification accuracy on high-dimensional datasets and is also computationally efficient [4]. This reduced data set was used as features to train the SVM classifiers. In the SVMs, we empirically computed the kernel parameter γ and cost relaxation parameter C as $9*10^{-5}$ and $8*10^{6}$ respectively. We used 15 classifiers, which are represented by the columns of the ECC matrix. The ECC matrix was generated using the Hamming code generation method [19].

Table III shows the results of a single classifier, static fusion and dynamic fusion methods. In all the methods, the SVM was used as the base classifier. The static fusion method was performed using the ECC matrix and the final decision was made using Hamming distance between outputs and rows of the ECC matrix and binary weighted voting. In the dynamic fusion, Pd_j and Pf_j were learned using a coarse optimization technique and the optimal parameters were $Pd_j = 0.4 \sim 0.5$ and $Pf_j = 0.0 \sim 0.02$ when the classifiers are part of the dynamic fusion. The dynamic fusion algorithm achieves the lowest diagnostic error and lowest standard deviation of the diagnostic error for CRAMAS® data. Figure 5 shows the box plot shows the dispersion of the diagnostic error rate for 10 datasets. The whiskers are shown by extending the lines from each end of the box and the maximum length of the line is a function of the inter-quartile range.

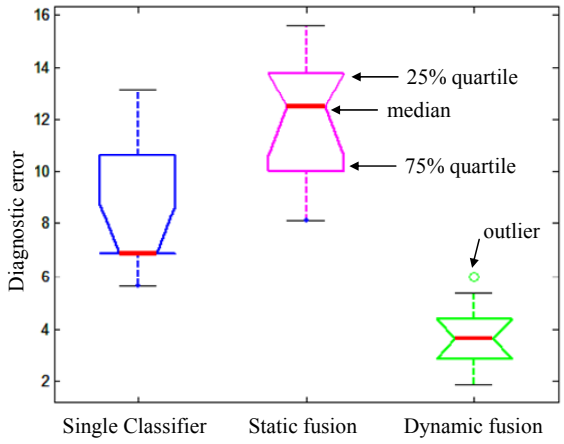

Fig.5. Comparison of diagnostic error among various methods

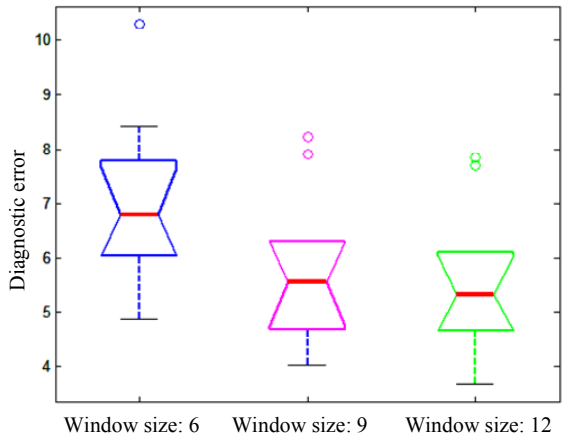

Fig.6. Effect of window size on the diagnostic error rate

TABLE III
RESULTS ON CRAMAS® DATA

	Single classifier	Static fusion	Dynamic fusion
Diagnostic error rate ± std dev in %	8.19±2.52	12±2.26	3.7±1.3
Computation time ± std dev in sec	0.02±0.01	0.03±0.005	0.12±0.001
False alarm rate ± std dev in %	--	--	5.13±0.83

The box plot demonstrates that there is consistent reduction in the diagnostic error rate by temporally fusing the classifiers as compared to static fusion of classifiers trained over the entire batch of data. The static fusion achieves higher diagnostic error rate as compared to a single classifier because the classifiers are already performing very well and they are not diverse.

Next, we discuss the results of sliding window dynamic fusion method. Figures 6 and 7 show the effect of window size on the diagnostic error and false alarms. A window size of 9 is a good candidate to perform on-line dynamic fusion because it not only achieves the diagnostic error similar to window size of 12 and but also it achieves the low false alarm rate as compared to window size of 12.

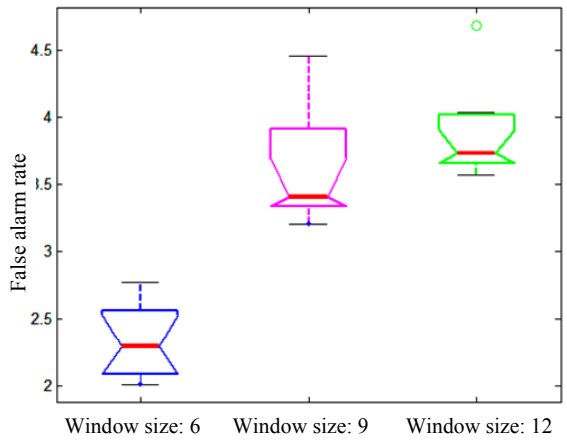

Fig.7. Effect of window size on the false alarm rate

The window size of 6 achieves high diagnostic error so it is not a good choice to perform on-line dynamic fusion. The sliding window method improves the computation time of the dynamic fusion algorithm by using the Lagrange multipliers from the previous window computation in the DMFD algorithm. The on-line dynamic fusion reduces the computation time (in MATLAB on a 3.0 GHz clock speed processor machine) to 0.016 sec per window as compared to 0.12 sec. per epoch in off-line dynamic fusion. These numbers are attractive practically, and they can be further reduced significantly by a careful implementation in C language.

VI. Conclusion

This paper presented a systematic process to perform temporal fusion of classifier outputs. We presented the dynamic fusion algorithm which is implemented as a special case of the dynamic multiple fault diagnosis method. We validated the algorithm using an automotive system dataset. The dynamic fusion algorithm achieves lowest diagnostic error and lowest standard deviation in the diagnostic error estimate as compared to a single classifier and static fusion of classifiers, which verifies that fusing classifier outputs over time improves the diagnostic accuracy. On-line version of the dynamic fusion algorithm was performed using a sliding window method to illustrate significant reduction of computation time without much sacrifice in accuracy.

References

[1] S. Ruan, Y. Zhou, F. Yu, K. R. Pattipati, P. Willett and A. Patterson-Hine, "Dynamic multiple fault diagnosis and imperfect tests," *IEEE Trans. on Systems, Man and Cybernetics*: Part A, under review, June 2006.

[2] 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.

[3] 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.

[4] 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 IEEE Autotestcon* Anaheim, CA, September 2006.

[5] T.G. Dietterich, "Ensemble methods in machine learning," *Multiple Classier Systems*, Cagliari, Italy, 2000.

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

[7] Qualtech Systems Inc., www.teamqsi.com.

[8] J. Luo, K. R. Pattipati, L. Qiao and S. Chigusa, "An integrated diagnostic development process for automotive engine control systems," to appear in *IEEE System, Man, and Cybernetics – Part C*, 2007.

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

[10] L. I. Kuncheva, J. C. Bezdek, and R .P. W. Duin, "Decision templates for multiple classifier fusion: An experimental comparison," *Pattern Recognition*, 34, 2001.

[11] S. Reiter and G. Rigoll, "Segmentation and classification of meeting events using multiple classifier fusion and dynamic programming," *Proceedings of the Pattern Recognition*, 17th International Conference on (ICPR'04), 2004.

[12] R. M. Vadovinos, J. S. Sanchez, and R. Barandela, "Dynamic and static weighting in classifier fusion," *Iberian conference on Pattern recognition and image analysis*, Portugal, 2005.

[13] L. I. Kuncheva and L. C. Jain, "Designing classifier fusion systems by genetic algorithms," *IEEE Trans. on Evolutionary Computation*, vol. 4, no. 4, September 2000.

[14] G. Giacinto and F. Roli, "Dynamic classifier selection," *Proc. of the First Int. Workshop on Multiple Classifier Systems*, Lecture Notes In Computer Science, Springer, Berlin, pp. 177-189, Cagliari, Italy, June 2000.

[15] E. Kim and J. Ko, "Dynamic classifier integration method," *Multiple Classifier Systems*, 97-107, 2005

[16] X. Zhu, X. Wu and Y. Yang, "Dynamic classifier selection for effective mining from noisy data streams," *4th IEEE conference on Data mining*, 2004.

[17] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144-152, 1992.

[18] V. N. Vapnik, "The nature of statistical learning theory," 2nd edition, *Springer*, 1999.

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