1-19-2016

Information Fusion for Pattern Classification in Complex Interconnected Systems

Nayeff Najjar
nayeff.najjar@uconn.edu

Information Fusion for Pattern Classification in Complex Interconnected Systems

Nayeff Najjar, PhD
University of Connecticut 2017

Typical complex interconnected systems consist of several interconnected components with several heterogeneous sensors. Classification problems (e.g., fault diagnosis) in these systems are challenging because sensor data might be of high dimension and/or convey misleading or incomplete information. So, this thesis performs optimal sensor selection and fusion to lower computational complexity and improve classification accuracy.

First, the thesis presents the novel *Unsupervised Embedded* algorithm for optimal sensor selection. The algorithm uses the minimum Redundancy Maximum Relevance (mRMR) criterion to select the candidate list of sensors, then uses K-means clustering algorithm and entropy criterion to select the optimal sensors. The Unsupervised Embedded algorithm is applied to heat exchanger fouling severity level diagnosis in an aircraft.

Second, the thesis presents a fusion algorithm for classification improvement called the *Better-than-the-Best Fusion* (BB-Fuse) algorithm, which is analytically proven to outperform the best sensor Correct Classification Rate (CCR). Using confusion matrices of individual sensors, the BB-Fuse selects the optimal sensor-class pairs and organizes them in a tree structure where one class is isolated at each node. The BB-Fuse algorithm was tested on two human activity recognition data sets and showed CCR improvement as expected.

Using the confusion matrices is a limitation of the BB-Fuse algorithm because of high computation complexity and pertinence to specific classifiers. So, the thesis next presents the *Decomposed Sensor-Class Pair Tree with maximum Admissible-Relevance for Fusion* (D*-Fuse) algorithm, which is a novel sensor-class pair selection and fusion algorithm. Instead of using the confusion matrices, the D*-Fuse algorithm uses a novel classifier-independent information-theoretic criterion (i.e., *Admissible-Relevance* (AR) criterion) to obtain the sensor-class pair. As a result, any classifiers can be used with the D*-Fuse Tree.

In comparison to other sensor selection algorithms in literature, the novel AR criterion has two major advantages. First, the AR criterion ensures selection of non-redundant sensors by selecting optimal sensor-class pairs. This pair selection ensures that the selected sensors carry signatures about different classes; i.e., they convey non-redundant classification information. Second, the AR criterion outputs a general fusion tree that suggests an order in which the sensors should be used.

Information Fusion for Pattern Classification in Complex Interconnected Systems

Nayeff Najjar

B.S., King Fahd University of Petroleum and Minerals, 2005

M.S., Widener University, 2008

M.S., University of Connecticut, 2016

A Dissertation

Submitted in Partial Fullfilment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2017

Approval Page

Doctor of Philosophy Dissertation

Information Fusion for Pattern Classification in Complex Interconnected Systems

Presented by

Nayeff Najjar

Major Advisor

Prof. Shalabh Gupta

Associate Advisor

Prof. Peter Luh

Associate Advisor

Prof. Rajeev Bansal

University of Connecticut

2017

To King Abduallah bin Abdulaziz (R.I.P.)

Acknowledgments

I would like first to express my appreciations to the highness of King Abdullah bin Abdulaziz (RIP), the previous king of Saudi Arabia. King Abduallah initiates a scholarship program through which thousands of Saudi students have the opportunity to join top universities worldwide and; especially the United States of America. I been lucky to gain on of these scholarships using which I continued my higher education and for that I feel grateful.

Next, I would like to thank my advisor, Dr. Shalabh Gupta. Dr. Gupta's mentoring through my PhD work helped me to improve my academic skills. In his lab, he encouraged teamwork and gave me the opportunity to improve my leadership and communication skills. It was not possible to develop my PhD theory without his kind directions and advice. Thank you Dr. Gupta.

Besides that, I would like to express my gratitude to my mother, Thuraiyah Najjar, and father, Abdularahman Najjar, for all encouragements and support. My mother was my first teacher and her contribution to my personality is invaluable and so was my father.

Through my life, many of family members and friends have been generous with encouragements, kind wishes and advices. I would like to seize this opportunity to thank all of them; especially, my father in law, Jameel Itraji and my uncle, Ghazi Najjar.

Last but not least, I would like to thank my wife, Israa Itraji. The years I spent in my PhD work were hard and time consuming. Her patience and persistent support were vital for me to be successful in my PhD.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Classification problems arise in a plethora of modern applications including security, entertainment, surveillance, medical, engineering, geoscience and many other applications [1–9]. Most modern systems (e.g., aerospace, automobile and robotic systems) can be described as complex interconnected systems, where the components are interconnected via electrical, mechanical or wireless connections. Typically, several heterogeneous sensors are scattered all over these systems for monitoring and control purposes (e.g., fault diagnosis and prognosis, activity recognition and situation awareness). Accurate classification could be challenging in these systems due to various types of complexities including data, operational, and system complexities. First of all, the data could be complex because of the multitude of heterogeneous sensors which makes the analysis computationally expensive. Second, operational complexity stems from the fact that the system response may vary with respect to ambient conditions; in turn, class signatures vary from one ambient condition to another which may confuse the classifiers. Last but not the least, system complexity is due to the interconnections between the components; as a result, sensors may carry redundant, misleading or complementary classification information.

These complexities promote the need for automatically selecting an optimal sensor set, and then

obtaining a unified, robust and reliable classification decision using information fusion techniques. In this regard, there are three themes in this thesis as introduced next where the details are in Chapters 2,3 and 4.

1.2 Outline and Contributions

The first theme addresses the problem of optimal sensor set selection with application to heat exchanger fouling diagnosis in the Environmental Control System (ECS) of an aircraft. The ECS is a complex interconnected system that controls the temperature, humidity level and pressure of the cabin air of an aircraft. One of the major components in the ECS is the heat exchanger which exchanges the heat with the ram air. The heat exchanger is prone to the fouling phenomenon, which is the accumulation of debris on the surface of the heat exchanger. Fouling reduces the efficiency of the heat exchanger, might occur unexpectedly and may lead to cascading failures of other expensive components. Subsequently, it is required to perform periodic maintenance of the heat exchanger. Unfortunately, this maintenance incurs high financial and time costs. This motivates the design of an automated heat exchanger fouling diagnosis algorithm.

Heat exchanger fouling diagnosis is challenging because the ECS sensor readings vary with respect to ambient temperatures and altitudes of airports across the world. Furthermore, the ECS is prone to various sources of uncertainties such as measurement noises and vibrations. Moreover, fouling diagnosis is difficult and computationally expensive due to the large number of heterogeneous sensors. So, the thesis presents the *Unsupervised Embedded* sensor selection algorithm, which is an automated method that selects optimal, non-redundant sensor set for data reduction and reliable fouling diagnosis.

This algorithm is of two steps. In the first step, the algorithm uses the popular *minimum-Redundancy-Maximum-Relevance* (mRMR) [10] algorithm to select a candidate list of sensors. In the second step, the algorithm uses K-means clustering algorithm to cluster the data, and then ranks the sensors based on the average entropy of the clusters of each sensor.

The second theme addresses the challenge of sensor fusion to improve the Correct Classification

Rate (CCR). Most information fusion algorithms that are available in literature lack the analytical guarantee of CCR improvement. This thesis presents a novel algorithm called the *Better-than-the-Best Fusion* (BB-Fuse) algorithm, which is analytically proven to improve the CCR. The BB-Fuse algorithm exploits the fact that different sensors isolate different classes with different accuracies. Subsequently, the BB-Fuse algorithm uses the confusion matrices that result from different sensors to search for the optimal sensor-class pairs. Finally, the BB-Fuse organizes them in a fusion tree that isolates one class at a time.

The first and the second themes addressed the sensor selection problem and the fusion problem for CCR improvement; however, both of the above algorithms have certain limitations. In regards to the first theme, the Unsupervised Embedded algorithm is computationally efficient and performs better in comparison to existing sensor selection algorithms; however, it relies on the mRMR algorithm which is criticized in literature for subtracting the minimum redundancy criterion [11].

In regards to the second theme, the BB-Fuse algorithm guarantees CCR improvement; however, it relies on the confusion matrices of the sensors. The confusion matrices are not only expensive to compute but also pertain to specific classifiers. In other words, there would be a different optimal fusion tree if different classifiers are used because the optimal sensor-class pairs would differ.

To address the above limitations, the third theme presents the *Decomposed Sensor-Class Pair Tree with maximum Admissible-Relevance for Fusion* (D*-Fuse) algorithm, which is a novel sensor-class pair set selection and fusion algorithm. Similar to the BB-Fuse, the D*-Fuse algorithm utilizes the fact that sensors differ in the quality with which they can isolate individual classes. However, the D*-Fuse algorithm depends on an information-theoretic sensor-class pair selection criterion (called the *Admissible Relevance* (AR) criterion) instead of the confusion matrices of individual sensors. Subsequently, any set of classifiers can be trained at the nodes of the resultant sensor-class tree structure.

The D*-Fuse algorithm has two major improvements over the mRMR. First, the D*-Fuse algorithm selects optimal "sensor-class" pair set rather than selecting "sensor" set only. As a result, sensor redundancy between the optimal selected sensors does not exist since the selected sensors carry information that isolate different classes. Second, the D*-Fuse algorithm outputs a tree

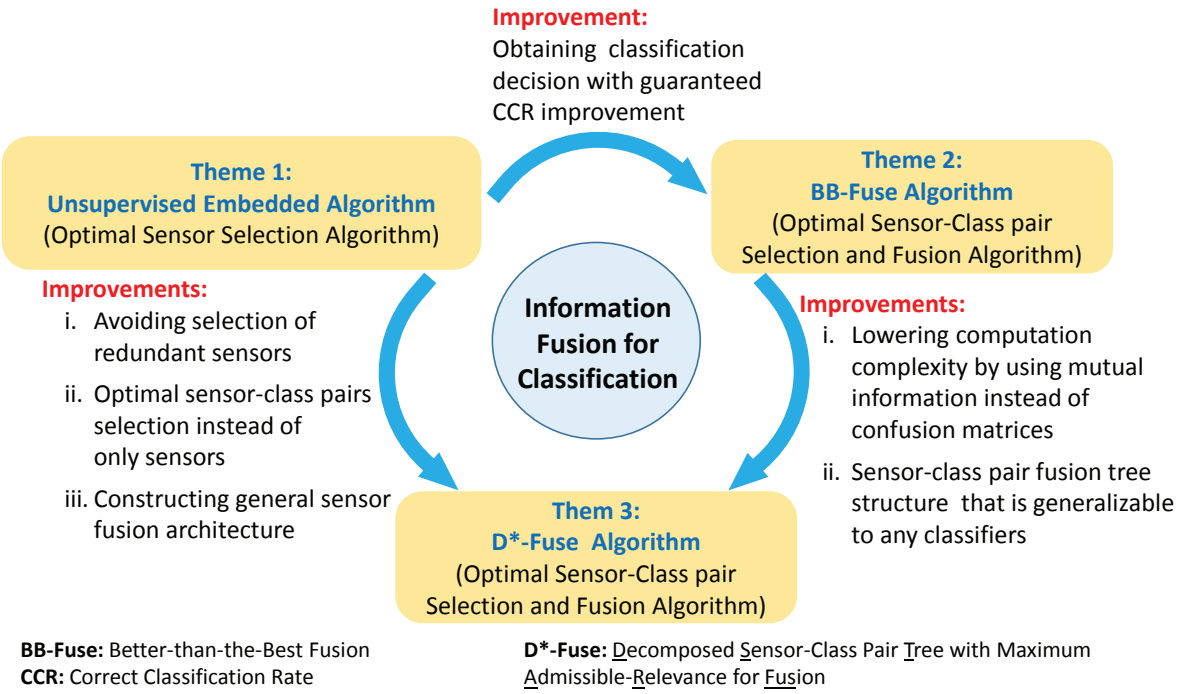


Figure 1.1: Research Themes Summary and Connections

structure to fuse the optimal sensors. Figure 1.1 summarizes the three research themes and the connections between them.

1.3 List of Publications

Journal Papers

1. **N. Najjar**, S. Gupta, J. Hare, S. Kandil, and R. Walthall, "Optimal sensor selection and fusion for heat exchanger fouling diagnosis in aerospace systems," *IEEE Sensors Journal*, vol. 16, no. 12, pp. 4866–4881, 2016.
2. **N. Najjar**, C. Sankavaram, J. Hare, S. Gupta, K. Pattipati, R. Walthal, and P. D'Orlando, "Health assessment of liquid cooling system in aircrafts: Data visualization, reduction, clustering and classification," *SAE International Journal of Aerospace*, vol. 5, no. 1, pp. 119–127, 2012.
3. **N. Najjar** and S. Gupta, "BB-Fuse: An information fusion algorithm for n-class classification problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Submitted.

Conference Papers

4. **N. Najjar** and S. Gupta, "Better-than-the-Best Fusion algorithm with application in human activity recognition," in *Proceedings of SPIE Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2015*, SPIE, 2015, pp. 949805-1 – 949805-10.

5. **N. Najjar**, J. Hare, P. D'Orlando, G. Leaper, K. Pattipati, A. Silva, S. Gupta, and R. Walthall, "Heat exchanger fouling diagnosis for an aircraft air-conditioning system," in *Proceedings of SAE 2013 AeroTech Congress and Exibition- Technical Paper 2013-01-2250*, SAE International, 2013.

6. **N. Najjar**, C. Sankavaram, J. Hare, S. Gupta, K. Pattipati, R. Walthall, and P. D'Orlando, "Health assessment of liquid cooling system in aircrafts: Data visualization, reduction, clustering and classification," in *Proceedings of SAE 2012 Aerospace Electronics and Avionics Systems Conference- Technical Paper 2012-01-2106*, SAE International, 2012, pp. 119–127.

7. J. Wilson, **N. Najjar**, J. Hare, and S. Gupta, "Human activity recognition using lzw-coded probabilistic finite state automata," in *Proceedings of 2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 3018–3023.

8. J. Hare, S. Gupta, **N. Najjar**, P. D'Orlando, and R. Walthall, "System-level fault diagnosis with application to the environmental control system of an aircraft," in *Proceedings of SAE 2015 AeroTech Congress and Exhibition- Technical Paper 2015-01-2583*, SAE International, 2015.

9. A. Silva, **N. Najjar**, S. Gupta, P. D'Orlando, and R. Walthall, "Wavelet-based fouling diagnosis of the heat exchanger in the aircraft environmental control system," in *Proceedings of SAE 2015 AeroTech Congress and Exhibition- Technical Paper 2015-01-2582*, SAE International, 2015.

Patents

10. **N. Najjar**, S. Gupta, J. Hare, G. Leaper, P. D'Orlando, R. Walthall, and K. Pattipati, *Optimal sensor selection and fusion for heat exchanger fouling diagnosis in aerospace systems*, 14/700,769, (Submitted).

11. J. Hare, S. Gupta, **N. Najjar**, P. D'Orlando, and R. Walthall, *System level fault diagnosis for the air management system of an aircraft*, 14/687,112, (Submitted).

12. A. Silva, **N. Najjar**, S. Gupta, P. D'Orlando, and R. Walthall, *Wavelet-based analysis for fouling diagnosis of an aircraft heat exchanger*, 14/689,467, (Submitted).

CHAPTER 2

UNSUPERVISED EMBEDDED SENSOR SELECTION FOR CLASSIFICATION ALGORITHM

2.1 Introduction

Optimal sensor set selection is essential for classification in complex interconnected systems. Primarily, optimal sensor set selection reduces the data and in turn computational complexity. Furthermore, it increases classification accuracy because some sensors might convey misleading or partial classification information. Optimal sensor set selection can sometimes be counter intuitive in complex interconnected systems because of direct or feedback interconnections between the components. Hence, automated data-driven optimal sensor selection is desirable in complex interconnected systems.

Recent literature has developed several sensor selection algorithms that are categorized into two main types based on their evaluation criteria. The first in this category are the *Wrapper Algorithms* that depend on the evaluation of the *Correct Classification Rate* (CCR)[1] for each sensor using a specified classifier [12–14]. Wrapper algorithms usually lead to a high CCR but are computationally expensive if the number of sensors is large because they rely on the *cross-validation* algorithm to calculate the CCR. Besides that, the wrapper algorithms cannot be generalized to any classifier.

[1] CCR is the ratio of correctly classified samples to the total number of testing samples

The second type are the *Filter Algorithms* that evaluate the performance of each sensor based on an evaluation function. Recently, many filter algorithms have been developed using the concepts of information theory [10, 15]. Filter algorithms do not depend on the classifier, are computationally less expensive, and may perform as good as the wrapper algorithms [13].

In addition, there exist the *Embedded Algorithms* that take advantage of both the wrapper and the filter algorithms. The embedded algorithms use a filter to select a candidate list of sensors and then apply a wrapper on this list to rank the optimal sensor set [10]. Embedded algorithms are less expensive than wrappers and more accurate than filters; yet they are pertinent to the specified classifier [16]. Several search methods have been suggested for the above algorithms, such as the forward and backward search [10, 15]. Dash and Liu [17] compared different such search methods.

This theme presents a modified embedded sensor selection algorithm which is called the Unsupervised Embedded algorithm. The first step of the algorithm uses *minimum Redundancy Maximum Relevance* (mRMR) criterion [10] as a filter to initiate a candidate list. Subsequently, the second step of the algorithm is *clustering based sensor ranking*, which relies on K-means clustering method instead of a specific classifier to rank the sensors in the candidate list. This method has low computational complexity, faster execution, and it does not depend on a specific classifier. Once the optimal set of sensors is selected, different machine learning tools can be applied for data analysis and fusion to make classification decisions.

2.1.1 Application

For applicatin, this theme uses the *Environmental Control System* (ECS) of an aircraft as an example of a complex interconnected system. The ECS is a system which regulates temperature, pressure and humidity of the cabin air of an aircraft. The ECS consists of various components such as the primary and secondary heat exchangers, turbines, compressor, condenser, and water extractor as schematically shown in Fig. 2.1 where the sensros are defined in Table 2.1. These components are interconnected through various mechanical and pneumatic connections. In addition, various sensing devices (more than 100 parameter recorded) such as temperature, pressure and flow sensors are mounted at different locations in the ECS [18, 19].

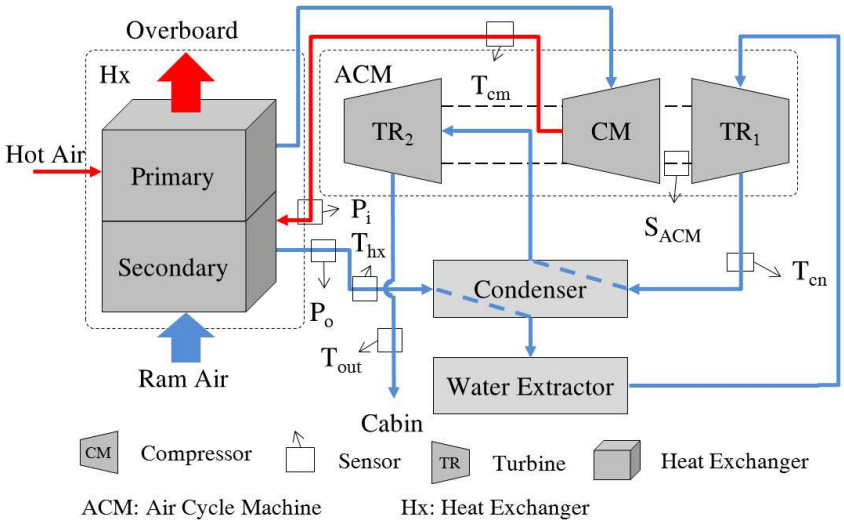


Figure 2.1: Environmental Control System (ECS) Schematics

Table 2.1: List of Critical Sensors in the ECS

S	Description	S	Description
$\dot{m}_1$	PD mass flow rate	$\dot{m}_2$	SD mass flow rate
T_1	PD air temperature	T_2	SD air temperature
P_1	PD air pressure	P_2	SD air pressure
P_i	SHX input pressure	P_o	SHX output pressure
T_{hx}	SHX output temperature	T_{cm}	Compressor output temperature
T_{out}	ECS output temperature	T_{cn}	Condenser output temperature
T_{FD}	Flight deck zone temperature	T_{Zj}	Zone Zj $\forall j = 1 \ldots n_z$ temperature

S: Sensor
PD: Primary bleed air duct
n_z: The number of zones in the cabin
SHX: Secondary heat exchanger
SD: Secondary bleed air duct

Heat exchangers are critical components of the ECS of an aircraft. Typically, several plate fin heat exchangers are used in an ECS, which consist of plates and fins stacked over each other as shown in Fig. 2.2. Plate fin heat exchangers are used in this application because of their compact design, light weight and high efficiency. Often, physical objects (e.g., debris) accumulate on the fins of the heat exchanger due to particulates and other contaminants present in the air stream. This phenomenon is known as *fouling* which obstructs the flow of the cooling medium through the heat exchanger and hence degrades its efficiency.

In absence of an automated fouling diagnosis methodology, heat exchanger necessitates periodic and expensive maintenance [20]. On top of maintenance costs, fouling diagnosis incurs financial losses that result from the aircraft flight interruption that last several days. Besides, fouling may occur unexpectedly and, in extreme cases, can lead to damage of other expensive components (i.e., cascading failures). As such, early fouling diagnosis is of utmost importance to facilitate *Condition Based Maintenance* (CBM)[2] and to avoid cascading failures.

2.1.2 Approach and Contributions

A high fidelity ECS Simulink model that is provided by our industry partner was used for data generation. To reduce system modeling uncertainty, the Simulink model was validated using actual flight test data operating under nominal conditions. System validation is an iterative process in which actual flight data is compared with Simulink output data; after that, the Simulink model is tuned to minimize the residuals. Next, the Simulink model output data is compared with actual flight data again and so on. The process terminates when the residuals are within acceptable tolerance.

The Simulink model inputs include the ambient temperature, altitude, occupant count and aircraft state (on-ground or flying). In addition, heat exchanger fouling condition can be injected into

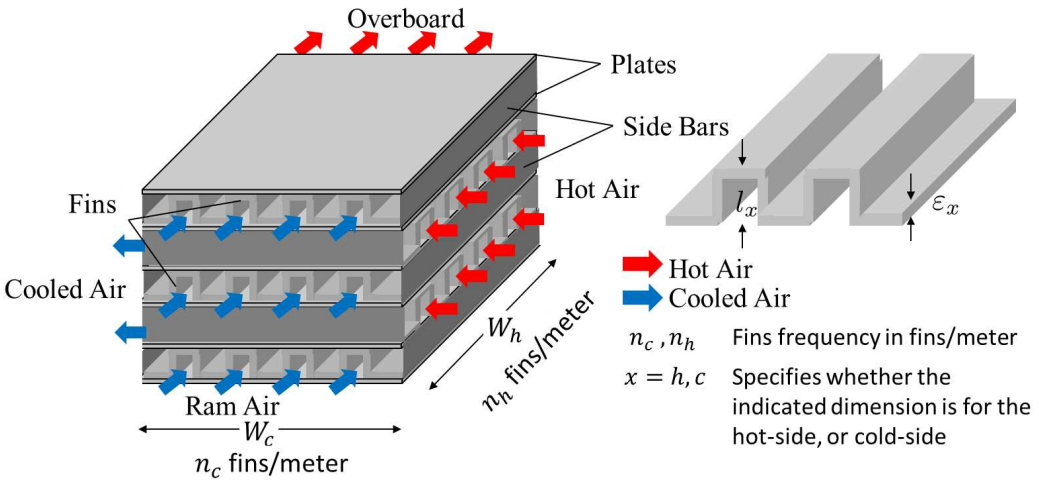


Figure 2.2: An Illustration of the Plate Fin Heat Exchanger

[2]Condition Based Maintenance (CBM) is to perform maintenance only when necessary

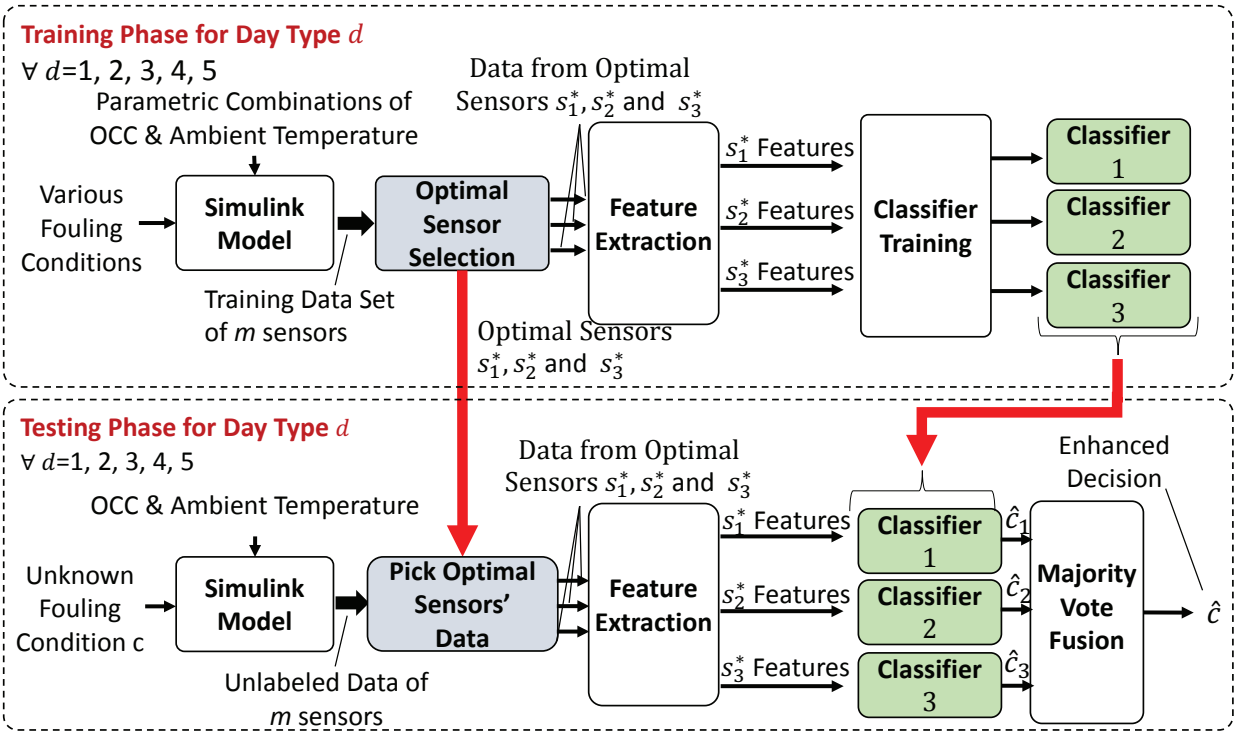


Figure 2.3: Heat Exchanger Fouling Diagnosis Algorithm where $d = 1, 2 \ldots 5$

the Simulink model by varying the flow impedance through the heat exchanger. The Simulink model outputs time series data of more than 100 sensor parameters.

To accommodate the system behavior under various input conditions, the ambient temperatures and passenger loads are tabulated as discussed in the Section 2.3.2. After that, fouling classes are defined based on the air-flow through the heat exchanger. Then, the validated Simulink model of the ECS is used to generate sensor data for nominal and different fouling conditions of the heat exchanger while considering various sources of uncertainties in the system. After data generation, machine learning algorithms are applied to obtain a fouling decision. The heat exchanger fouling diagnosis algorithm is shown in Fig. 2.3.

The main contributions are below:

- Optimal sensor set selection for fouling diagnosis using a novel *Unsupervised Embedded Algorithm*, that uses the mRMR criteria as a filter and the K-means clustering method for ranking.

- Application of the *Maximum Entropy Principle* for data partitioning to compute probability

distributions and to estimate the mutual information in the mRMR criteria.

2.2 Literature Review

2.2.1 Sensor Selection Algorithms

Sensor/feature selection for classification has gained interest in literature. Han et al. [21] studied feature selection problem for chillers. Namburu et al. [22] used genetic algorithm for sensor selection and applied SVM, PCA, and Partial Least Squares (PLS) for fault classification in HVAC systems. Optimal sensor selection for discrete-event systems with partial observations was performed by Jiang and Kumar [23]. Gupta et al. [24] discussed stochastic sensor selection with application to scheduling and sensor coverage. Joshi and Boyd [25] used *convex optimization* to perform sensor selection. Hero and Cochran [26] provided a review of the methods and applications of sensors management. Xu et al. [27] used sensor configuration, usage and reliability costs for sensor selection for PHM of aircraft engines. Shen, Liu and Varshney [28] considered the problem of multistage look-ahead sensor selection for nonlinear dynamic systems.

2.2.2 Fault Diagnosis Algorithms

Several techniques have been proposed in recent literature for fault detection, diagnosis and prognosis (FDDP) of air-conditioning systems, in particular, *Heating, Ventilating and Air Conditioning* (HVAC) systems [29–31]. Katipamula and Brambley introduced a two-part survey of FDDP of HVAC systems [32, 33]. Buswell and Wright [34] accounted for uncertainties in model-based approaches to minimize false alarms in fault diagnosis of HVAC systems. Fault diagnosis of *Air Handling Units* (AHU) was presented in [35–37]. Pakanen and Sundquist [38] developed an *Online Diagnostic Test* (ODT) for fault detection of *Air Handling Units* (AHU). Qin and Wang [39] performed a site survey on hybrid fault detection and isolation methods for *Variable Air Volume* (VAV) air conditioning systems. Rossi and Braun [40] designed a classifier that uses temperature and humidity measurements for fault diagnosis of the *Vapor Compression Air Conditioners*. Zhao et al. [41] utilized exponentially-weighted moving average control charts and support vector re-

gression for fault detection and isolation in centrifugal chillers. Najjar et al. [42] developed a tool for data visualization, reduction, clustering, and classification of the actual data obtained from flight test reports of the *Liquid Cooling System* (LCS) in aircrafts. Shang and Liu [43] used the *Unscented Kalman Filter* (UKF) to diagnose sensor and actuator faults in the *Bleed Air Temperature Control System*. Gorinevsky and Dittmar [44] addressed fault diagnosis of the *Auxiliary Power Unit* (APU) using a model-based approach. Isermann [45] provided a review of model-based fault detection and diagnosis methods.

Heat exchanger fouling diagnosis has become a critical research issue in recent years. Lingfang et al. [46][47] developed a method of fouling prediction based on SVM. Najjar et al. [48] presented the fouling severity diagnosis of the *Plate Fin Heat Exchanger* using the principal component analysis (PCA) and the k-nearest neighbor classification (k-NN). Kaneko et al. [49] introduced a statistical approach to construct predictive models for thermal resistance based on operating conditions. Shang and Liu [50] proposed a method to detect heat exchanger fouling based on the deviation of valve commands from the actual valve positions. Riverol and Napolitano [51] used *Artificial Neural Networks* (ANN) to estimate the heat exchanger fouling. Garcia [52] used *Neural Networks* and *rule based* techniques to improve heat exchanger monitoring. Adili at al. [53] used genetic algorithms to estimate the thermophysical properties of fouling.

2.3 System Description

The Environmental Control System (ECS) is an air conditioning system that regulates temperature, pressure and humidity of the cabin air. In order to meet the health and comfort requirements of the passengers, the ECS supplies air to the cabin at moderate temperatures and pressures [54]. Figure 2.1 shows a simplified system diagram of the main ECS components, namely: i) *primary heat exchanger*, ii) *secondary heat exchanger*, iii) *air-cycle machine* (ACM), iv) *condenser*, and v) *water extractor*.

The ACM in turn consists of a *compressor* and two *turbines*: a) *first stage turbine* and b) *second stage turbine*. The compressor and the turbines rotate on the same shaft [54–57]. In addition,

various sensing devices such as temperature and pressure sensors [54] are mounted at different locations of the ECS. Table 2.1 shows a list of critical sensors, as also shown in Fig. 2.1.

The primary heat exchanger is supplied with hot bleed air through two ducts, namely, the primary bleed air duct and the secondary bleed air duct, where air flow in each duct is controlled by a valve (not shown in Fig. 2.1). These ducts are then merged together to drive the bleed air to the primary heat exchanger. As shown in Fig. 2.1, hot bleed air is cooled in the primary heat exchanger, using ambient ram air as a sink, to a temperature below the auto-ignition temperature of fuel as a safety measure in case of a fuel leak. Air that comes out of the primary heat exchanger flows into the compressor section of the ACM where it gets compressed and thus heated. Air then flows out of the compressor into the secondary heat exchanger where it is cooled again using ram air as the sink. Air then flows through the hot side of the condenser heat exchanger where moisture is condensed out of the air-flow and collected by the water extractor. Air then flows into the first stage turbine where it gets expanded and cooled. Cold air out of the turbine flows through the cold side of the condenser heat exchanger into the second stage turbine where it gets further expanded and cooled providing the air at the desired cabin supply temperature and pressure [55].

2.3.1 Primary and Secondary Heat Exchangers

The heat exchangers used in the ECS under consideration are the *cross-flow plate fin heat exchangers* that are built from light weight plates and fins stacked over each other, as shown in Fig. 2.2. By definition, the direction through which the hot-air flows is called the *hot-side* while the direction through which the ram air flows is called the *cold-side* of the heat exchanger. The fins are placed alternatively in parallel to the hot air flow and the cold air flow, hence the name *cross-flow plate fin heat exchanger*. Plate fin heat exchangers are desirable for their compact sizes, high efficiency, and light weight. The function of the heat exchanger is to transfer heat from the hot air to the ram air. The temperature can be set to the desired value by controlling the flow of the ram air in the cold-side of the heat exchanger. Debris accumulates on the fins of the heat exchangers due to several factors including chemical reactions, corrosion, biological multiplications and freezing. This phenomenon is known as *fouling* and it obstructs the ram air flow. Fouling lowers the heat

efficiency of the heat exchanger because the deposited material has low thermal conductivity and hinders the transfer of heat [43, 53]. A detailed description of fouling substances and cleaning methods can be found in [43, 58]. In this regard, this chapter focuses on the fouling diagnosis of the secondary heat exchanger. The heat transfer rate $\dot{\Psi}$ $(Watts)$ through the heat exchanger [59] is given by Eq. (2.1) as follows

$$\dot{\Psi} = \kappa \cdot A_h \cdot (T_{avg,h} - T_m) = \kappa \cdot A_c \cdot (T_m - T_{avg,c}) \tag{2.1}$$

where κ is the overall heat transfer coefficient $(W/(m^2K))$, T_m is the metal temperature (K), and $T_{avg,x}$ and A_x are the average air temperature (K) and the total heat transfer area (m^2) at the x-side, respectively. The subscript x is either h for the hot-side or c for the cold-side. The total heat transfer areas of the hot and cold sides are calculated as follows

$$A_x = W_h W_c N_x [1 + 2n_x(l_x - \varepsilon_x)] \tag{2.2}$$

where W_h, W_c, l_x, and ε_x are the fin dimensions (m) as shown in Fig. 2.2, and N_x and n_x are the number of fin layers and fin frequency per unit length at the x-side, respectively [60, 61].

The heat transfer is also calculated as a function of the input and the output temperatures of the heat exchanger, as follows

$$\dot{\Psi} = \dot{m}_h c_{p,h}(T_{i,h} - T_{o,h}) = \dot{m}_c c_{p,c}(T_{o,c} - T_{i,c}) \tag{2.3}$$

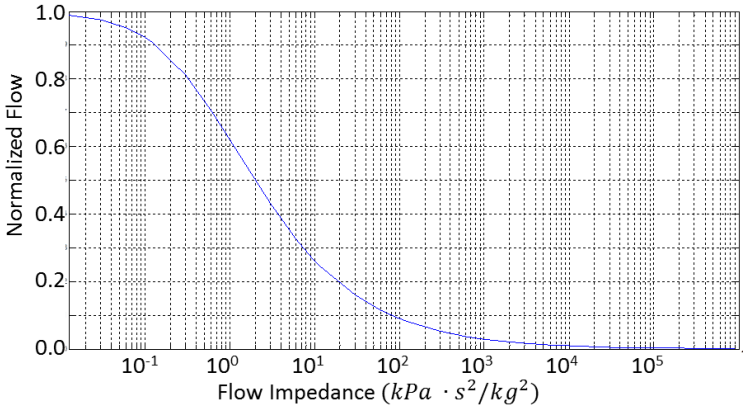


Figure 2.4: Normalized Flow vs Secondary Heat Exchanger Impedance

where $\dot{m}_x$, $c_{p,x}$, and $T_{i,x}$ and $T_{o,x}$ are the mass flow rate (kg/s), specific heat ($J/(kgK)$), and the input and output temperatures of the x-side, respectively [59].

The input-output pressure drop at the cold-side of the heat exchanger is modeled as

$$\begin{aligned} \Delta P &= P_{in,c} - P_{out,c} \\ &= \frac{1}{\beta} z_c \dot{m}_c^2 \end{aligned} \tag{2.4}$$

where $P_{in,c}$ and $P_{out,c}$ are the input and the output pressures (kPa) for the cold-side, β is a dimensionless correction factor and z_c is the flow impedance ($kPa \cdot s^2/kg^2$) which is varied in the simulation to represent different fouling conditions. The plot of the flow vs the flow impedance is shown in Fig. 2.4. A change in z_c affects $\dot{m}_c$ and thus affects the heat transfer and the output temperatures of hot and cold air streams as computed using Eq. (2.1)-(2.3). This also affects the sensor readings of all other sensors in the ECS.

2.3.2 Data Generation Process

This chapter utilizes an experimentally validated high-fidelity Simulink model of the ECS provided by an industry partner. The model is used to generate dynamic data for various sensor locations around the ECS system for fouling diagnosis. It is important to note that the model represents the ECS performance for a specific aircraft and has been validated to match experimental results from lab testing and flight data for this specific ECS. For this chapter, the model is exercised to generate time series of sensor data for various ground operating conditions (e.g., ambient temperature, occupant count, etc.). Ground operating conditions are chosen because typically more debris exists in the aircraft vicinity while on the ground as opposed to in-flight operation. Data generated for this study includes a large number (> 100) of sensor outputs.

Figure 2.5 shows the stochastic time series data plots of three critical sensors under various uncertainties for different day types. The structure of the data is explained below. Let us denote the sensor suite by a set $\mathcal{S} = \{s_1, \ldots s_m\}$, where m is the total number of sensors. For each sensor s_j, the time series data are collected for 600 seconds at the sampling rate of 1 sample/sec, thus

generating a data sequence $\mathbf{z}_j = [z_j(1), ...z_j(600)]$, $\forall j = 1, 2, \ldots m$. The system reaches steady state after 300 seconds, thus the data from 301 to 600 seconds is used for analysis; however this interval could be reduced for higher sampling rates.

It is to be noted that the system behavior and the sensor data are affected by several input parameters, which affect the accuracy of fouling diagnosis. This chapter considers variations in two main input parameters: the ambient temperature for different day types and the load corresponding to different occupant counts on the aircraft. Besides the heat exchanger fouling itself results in variations in sensor data. Thus the objective is to capture the effects of fouling under different input conditions. Specifically, the data is generated by varying the aforementioned parameters as described below.

• **Ambient Temperature (T_A)**

As expected, the ambient temperature is the most critical external parameter that affects sensor readings. The effect of ambient temperature could be misinterpreted and could lead to false diagnosis of heat exchanger fouling. Thus, to incorporate the effect of ambient temperature, sensor data is categorized into five different day types: i) extremely cold, ii) cold, iii) medium, iv) hot and v) extremely hot day types. The temperature ranges for each day type are shown in Table 2.2(a). Each day type is further partitioned into eight uniformly spaced temperature values.

• **Occupant Count (OCC)**

The occupant count also affects ECS sensor readings due to passengers adding heat load that the ECS reacts to in order to maintain the desired cabin conditions. The number of occupants is grouped into four categories: i) Low Load, ii) Medium Load, iii) Heavy Load, and iv) Very Heavy Load based on the percentage occupancy in the cabin. Table 2.2(b) shows these four load categories. Since OCC has relatively less influence on the sensor readings, only the middle point of each category is used for data generation.

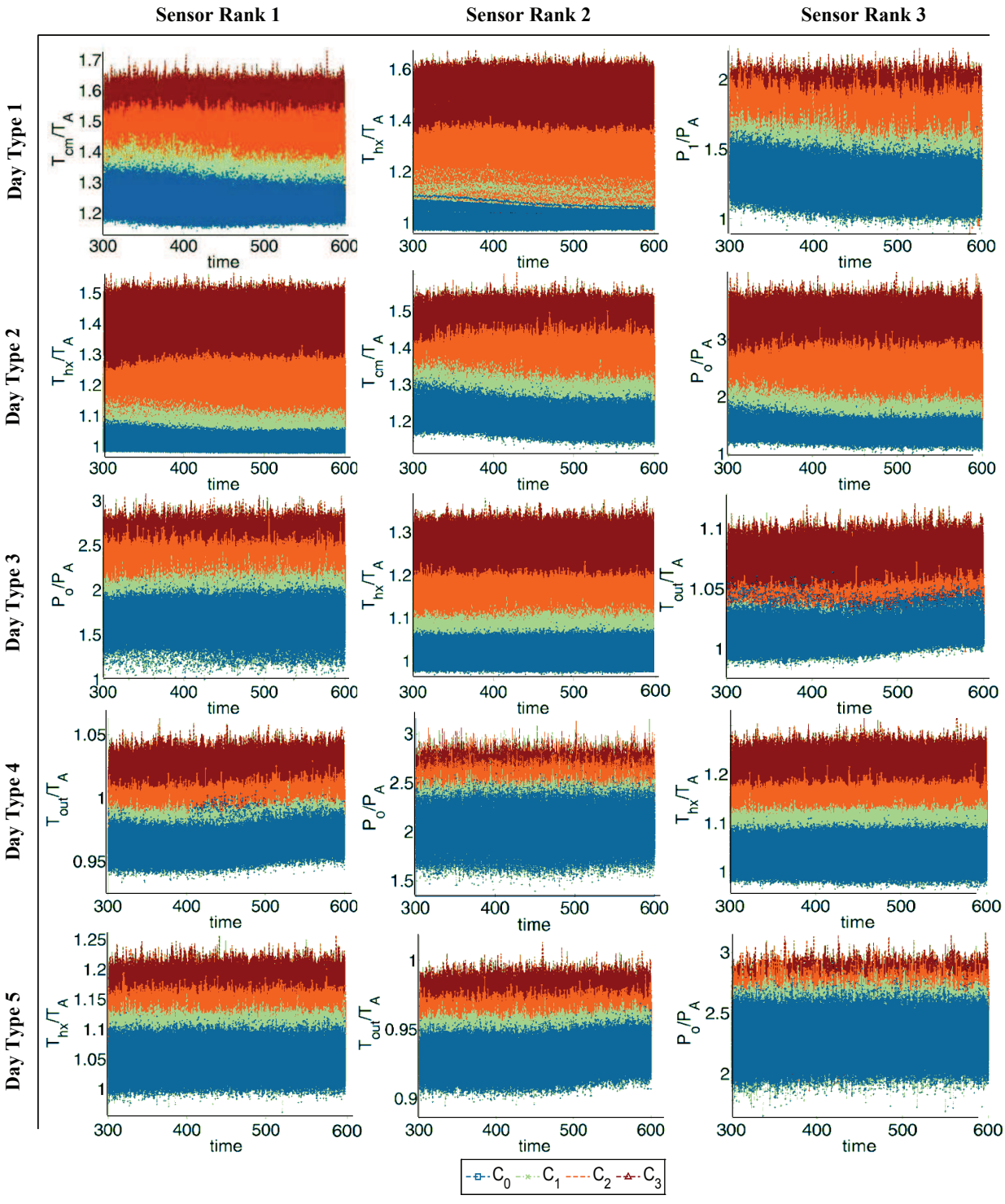


Figure 2.5: Stochastic time series data of three critical sensors for five different day types

Table 2.2: Input Parameters and Fouling Classes

(a) Day Types

Day Type	$\mathbf{T_{amb}}$ (^oF)
Extremely cold	-30 – 0
Cold	0 – 30
Medium	30 – 60
Hot	60 – 90
Extremely hot	90 – 120

(b) Passenger Load Categories

Load Type	**OCC**
Low Load	0% – 60%
Medium Load	60% – 75%
Heavy Load	75% – 95%
Very Heavy Load	95% – 100%

(c) Fouling Classes

Class	**Flow**
Green (c_0)	80% – 100%
Yellow (c_1)	60% – 80%
Orange (c_2)	40% – 60%
Red (c_3)	0% – 40%

• **Heat Exchanger Fouling** (z_c)

Fouling of the secondary heat exchanger is modeled as an increase in the ram air-flow impedance (z_c) on the cold side of the heat exchanger. When the flow impedance is increased, the air-flow decreases simulating blockage due to heat exchanger fouling. This lowers the effectiveness of the heat exchanger. For this chapter, four fouling classes have been defined based on the flow through the cold-side of the secondary heat exchanger as follows: i) *Green Class* (c_0)- i.e., 80-100 $\%$ flow, ii) *Yellow Class* (c_1)- i.e., 60-80 $\%$ flow, iii) *Orange Class* (c_2)- i.e., 40-60 $\%$ flow, and iv) *Red Class* (c_3)- i.e., 0-40 $\%$ flow. The reason to introduce Yellow and Orange classes is to avoid direct confusion between the Green and Red classes. The model is run for different values of flow impedance and the resulting flow through the heat exchanger is observed. The plot in Fig. 2.4 is used to determine the range of impedance values for each of the above classes that are defined based on the flow. Table 2.2(c) shows the impedance intervals associated with each class. Each class is further partitioned into eight uniformly spaced flow values for data generation.

Thus, for each day type stochastic time series data are generated for various combinations of the above parameters to represent each fouling class. The model is run for different combinations of the values of ambient temperature (8) (within each day type), occupant counts (4), and impedance values (8) (within each fouling class), resulting in a set consisting of a total number of $8 \times 4 \times 8$ = 256 runs of time series data. Furthermore, for each day type, similar data sets are generated for all the fouling classes, thus leading to a total of 4×256 = 1024 runs of time series data. Subsequently, the above data sets are generated for all five day types. Let $\Gamma = \{\gamma_1, ...\gamma_{1024}\}$ denote the set of parametric combinations and let $t \in \{1, 2, ...T\}$ denote the set of discrete time indices, where $T = 600$ is the length of the time series data. Then, for each day type the entire data for each

sensor $s_j \in \mathcal{S}$ is arranged in a $|\Gamma| \times T$ matrix Z_j, where the element at the r^{th} row and t^{th} column of Z_j is the reading of s_j at time index $t \in \{1, 2, ...T\}$ with parametric input $\gamma_r \in \Gamma$. Figure 2.5 shows the stochastic time series data plots of three critical sensors for each day type. For the purpose of data analysis, the fluctuations in OCC and the variations of impedance values within each class are considered as uncertainties. Other sources of uncertainties such as measurement noise, mechanical vibrations, and fluctuations in valve positions have been considered by adding white Gaussian noise with 25 dB SNR to the data. The variations in ambient pressure have not been considered in this chapter.

2.3.3 Heat Exchanger Fouling Diagnosis Architecture

Figure 2.3 shows the Heat exchanger fouling diagnosis architecture that consists of a training and a testing phase. The training phase consists of generating stochastic data for each sensor in the ECS (total 109 sensors) as described above. This sensor data is labeled with the fouling class information and is used for optimal sensor selection for each day type separately, as described in Section 2.4. From the data of optimal sensors, some useful features are extracted using PCA and GMM methods and classifiers (k-NN) are trained to identify the fouling classes, as described in Section 2.5.

In the testing phase, an unlabeled time series data is generated for an unknown parametric condition $\gamma \in \Gamma$ where the fouling severity is also considered as unknown. Subsequently, the optimal sensors identified in training phase are used for feature extraction and classification using trained classifiers. To further improve the classification accuracy, the results of the top three optimal sensors are fused using the majority vote.

2.4 Optimal Sensor Selection Methodology

Since a large number of sensors are available in the ECS mounted at different locations, the underlying processes of data generation, storage, and analysis become cumbersome. Therefore, an optimal sensor selection methodology is needed to rank the most relevant sensors in terms of the

best classification performance for heat exchanger fouling diagnosis. This is formally stated in the following problem statement.

Optimal Sensor Selection Problem: *Given the sensor set $\mathcal{S} = \{s_1, s_2, \ldots s_m\}$, with m sensors, and the class set $\mathscr{C} = \{c_1, c_2, \ldots c_n\}$, with n classes, the optimal sensor selection problem is to select a set $\mathcal{S}^* \subseteq \mathcal{S}$, where $|\mathcal{S}^*|$=m^*, $m^* < m$, that consists of sensors with maximum classification accuracy and are ranked accordingly in decreasing order.*

As discussed in the introduction, two commonly used sensor selection methods are: i) the *wrapper method* and ii) the *filter method*. Since the wrapper algorithms rank the sensors based on their correct classification rate (CCR), a feature extractor and a classifier have to be designed, trained, and applied to all sensors in order to compute their CCRs, thus making the whole process computationally expensive. Furthermore, the wrapper algorithms cannot be generalized to any classifier [12–14]. On the other hand, the filter algorithms evaluate the performance of each sensor based on an information theoretic measure [10, 15]. Filter algorithms are computationally less expensive and do not depend on the choice of a classifier, but they may not perform as good as the wrapper algorithms [13].

To circumvent this difficulty, the *embedded algorithms* take advantage of both the wrapper and the filter algorithms by using a filter to select a candidate list of sensors and then applying a wrapper on this list to rank and select the optimal set of sensors [10]. Embedded algorithms are less expensive than wrappers and more accurate than filters; yet they are pertinent to the specified classifier [16]. In this regard, this section presents a detailed description of the optimal sensor set selection methodology based on the *embedded algorithm*. In addition, a novel algorithm for sensor selection is presented, called the *unsupervised embedded algorithm*, that relies on the K-means clustering approach. This method has the advantage that it does not depend on the choice of a classifier and enables faster execution with very low computational complexity.

Both the embedded and the unsupervised embedded algorithms are based on the *minimum Redundancy Maximum Relevance* (mRMR) [10] criteria for the filter algorithm as a precursor step before applying the wrapper. The filter step facilitates fast execution of the first round of data re-

duction and produces a candidate list of top ranked sensors. Before describing the optimal sensor selection techniques, some useful information-theoretic quantities are defined below.

2.4.1 Information-theoretic Measures

Definition 2.4.1 (Entropy) *Entropy $H(X)$ is defined as a measure of uncertainty in a random variable X such that*

$$H(X) = -\sum_{i=1}^{|\mathscr{X}|} p_i \ln p_i \tag{2.5}$$

where X is a random variable whose outcomes belong to the set $\mathscr{X} = \{x_1, x_2, \ldots x_{|\mathscr{X}|}\}$ with the associated probability distribution defined as $p(X = x_i) = p_i \ \forall\, i = 1, 2, \ldots |\mathscr{X}|$.

According to Shannon [62], the entropy $H(X)$ qualifies to be a measure of uncertainty because it satisfies the following three conditions:

- $H(X)$ is a continuous function of p_i.
- If the random variable X is uniformly distributed (i.e., $p_i = \frac{1}{|\mathscr{X}|}, \forall\, i = 1, 2, \ldots |\mathscr{X}|$), then $H(X)$ is a monotonically increasing function of $|\mathscr{X}|$.
- If an event $X = x_i$ is split into two posterior sub-events, then the original entropy can be expressed as a weighted sum of the entropies of the sub-events.

The higher the entropy is, the higher is the uncertainty in the random variable. On the other hand, the entropy reaches its lowest value, $H(X) = 0$, when $p(X)$ is a delta distribution.

Suppose now that we have two random variables: X defined as above, and Y whose outcomes belong to the set $\mathscr{Y} = \{y_1, \ldots y_{|Y|}\}$ with probabilities $p(Y = y_j) = q_j$ for all $j = 1, 2, \ldots |\mathscr{Y}|$. Furthermore, suppose that the joint probability distribution is defined as $p_{i,j} = p(X = x_i, Y = y_j)$ for all $i = 1, 2, \ldots |\mathscr{X}|,\ j = 1, 2, \ldots |\mathscr{Y}|$. Then the joint and conditional entropies are defined as

follows:

$$H(X,Y) = -\sum_{i=1}^{|\mathscr{X}|}\sum_{j=1}^{|\mathscr{Y}|} p_{i,j} \ln p_{i,j} \tag{2.6}$$

$$\begin{aligned} H(X|Y) &= -\sum_{i=1}^{|\mathscr{X}|}\sum_{j=1}^{|\mathscr{Y}|} p_{i,j} \ln p(X = x_i|Y = y_j) \\ &= H(X,Y) - H(Y) \end{aligned} \tag{2.7}$$

Definition 2.4.2 (Mutual Information) *The mutual information between two random variables X and Y is defined as*

$$\begin{aligned} I(X,Y) &= H(X) - H(X|Y) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned} \tag{2.8}$$

The subtraction of $H(X|Y)$ from $H(X)$ represents the information gained about the random variable X given the information about the random variable Y [15]. The next section presents a partitioning approach for transformation of the continuous data to the symbolic domain for computation of the information-theoretic quantities as needed in the filter method.

2.4.2 Data Partitioning for Symbol Sequence Generation

Consider the data matrix Z_j of size $|\Gamma| \times T$ for any particular sensor s_j, $j = 1, 2, ...m$, generated under different parametric conditions as described in Section 2.3.2. The encoding of the underlying dynamics of this sensor data is achieved by partitioning [63] of the sensor observation space using an appropriate partitioning method. Let $R_j \subset \mathcal{R}$ be the compact (i.e., closed and bounded) region within which the observed sensor data Z_j is circumscribed. Let $\Sigma = \{\alpha_1, \alpha_2, \ldots \alpha_{|\Sigma|}\}$ be the *symbol alphabet* that labels the partition segments, where the number of segments is $|\Sigma|$, where $2 \leq |\Sigma| < \infty$. Then, the symbolic encoding of R_j is accomplished by introducing a partition $\{\varphi_j^1, \cdots, \varphi_j^{|\Sigma|}\}$ consisting of $|\Sigma|$ mutually exclusive (i.e., $\varphi_j^\ell \cap \varphi_j^k = \emptyset, \ \ \forall \ell \neq k$), and exhaustive (i.e., $\bigcup_{r=1}^{|\Sigma|} \varphi_j^r = R_j$) cells. Each cell is encoded with a symbol from the alphabet Σ. For each input condition, as the system evolves in time, the state trajectory (i.e., sensor readings) fall within

different cells of the partition, accordingly the corresponding symbol is assigned to each point of the trajectory. Let $\mathbf{z}_j(\gamma_r) \triangleq [Z_j(r,1), Z_j(r,2), \ldots Z_j(r,T)]$ be the r^{th} row of the data matrix Z_j for a given $\gamma_r \in \Gamma$. Then, for each sensor $s_j \in \mathcal{S}$ and for each $\gamma_r \in \Gamma$, the time series data $\mathbf{z}_j(\gamma_r)$ are transformed into a symbol sequence [64] $\tilde{\mathbf{z}}_j(\gamma_r) \triangleq [\tilde{Z}_j(r,1), \ldots \tilde{Z}_j(r,T)]$ as

$$[Z_j(r,1), \ldots Z_j(r,T)] \rightarrow [\tilde{Z}_j(r,1), \ldots \tilde{Z}_j(r,T)] \tag{2.9}$$

where T is the data length, $\tilde{Z}_j(r,t) \in \Sigma$, $\forall t = 1, ...T$ and $\tilde{Z}_j$ is the symbolic data matrix. Note: As mentioned earlier, this chapter uses only the steady state part of the data for fouling diagnosis analysis (i.e., $\forall\, t = 301, 302, \ldots 600$).

To do the above symbolization, this chapter uses the *maximum entropy principle* [65] based partitioning to create a partition of the observed sensor data space, which is finer in the information dense regions and coarser in the low information regions as described below.

Definition 2.4.3 (Maximum Entropy Principle, Jaynes [65]) *The maximum entropy principle states that the probability distribution that unbiasedly estimates the distribution of a random variable X under a given set of constraints is the distribution that maximizes the entropy $H(X)$.*

In other words, the unbiased estimate of the probability distribution of a random variable X should satisfy the following optimization problem:

$$\begin{aligned} \mathbf{P}^* = \begin{bmatrix} p_1^* \\ \vdots \\ p_{|\mathcal{X}|}^* \end{bmatrix} &= \arg\max_{\mathbf{P}} H(X); \quad H(X) = -\sum_{i=1}^{|\mathcal{X}|} p_i \ln p_i \\ \text{subject to:} \qquad & \sum_{i=1}^{|\mathcal{X}|} p_i = 1 \end{aligned} \tag{2.10}$$

The optimization problem in Eq. (2.10) can be solved using the Lagrange multiplier as shown in Appendix A.1. As a result, the entropy is maximized for the uniform distribution (i.e., $p_i^* = \frac{1}{|\mathcal{X}|}, \ \forall i = 1, ...|\mathcal{X}|$).

The Maximum Entropy Principle generates the unbiased distribution for each of the sensor

readings. Considering the data matrix $Z_j = [Z_j(r,1), Z_j(r,2), \ldots Z_j(r,T)]_{r=1,\ldots|\Gamma|}$ for sensor s_j, the goal is to find the partition that results in maximum entropy distribution (i.e., the uniform distribution). The partition cells are defined by the partitioning levels $\mathcal{L} = \{L_j^0, L_j^1, \ldots L_j^{|\Sigma|}\}$, such that $\varphi_j^r = [L_j^{r-1}, L_j^r)\ \forall r = 1, 2, \ldots |\Sigma|$. To compute the maximum entropy partition for the sensor data Z_j, the first step is to calculate the number of samples in each cell (i.e., $\eta^* = \eta^r$ = floor($|\Gamma| \times T/|\Sigma|$), $\forall r = 1, 2, ...|\Sigma|$). The second step is to sort the entire data into a vector $\mathbf{w}_j = [w_j(1), w_j(2), \ldots w_j(|\Gamma| \times T)]$, such that

$$w_j(\ell) \in \{Z_j(r,t)\ \forall\ r = 1, 2 \ldots |\Gamma|, t = 1, 2, \ldots T\}\ \ \forall \ell = 1, 2, \ldots |\Gamma| \times T \tag{2.11}$$

$$\&\ \ w_j(1) \leq w_j(2) \ldots \leq w_j(|\Gamma| \times T) \tag{2.12}$$

Then the partitioning levels are defined as follows:

$$L_j^0 = w_j(1) \tag{2.13}$$

$$L_j^q = w_j(q \cdot \eta^*)\ \forall q = 1, \ldots |\Sigma| - 1, \ \text{ and} \tag{2.14}$$

$$L_j^{|\Sigma|} = w_j(|\Gamma| \times T) \tag{2.15}$$

The algorithm counts the samples from the bottom and defines the partitioning levels at the multiples of η^* while setting the first and the last levels at the min and the max of the original data. This procedure generates a partition that is finer in the regions of high data density and coarser in the regions of low data density, as shown by an illustrative example in Fig. 2.6. Subsequently, a unique symbol from the alphabet Σ is assigned to all the data points in each cell of the partitioning. This process transforms each data matrix in Z_j into a symbol matrix $\tilde{Z}_j$, as shown in Eq. (2.9).

In the above manner, the maximum entropy partitioning is constructed for all sensors and the corresponding data are transformed into symbol sequences. Subsequently, the candidate list of sensors is selected and ranked according to the filter criteria as described next.

2.4.3 minimum Redundancy Maximum Relevance (mRMR)

Based on mutual information, the mRMR criterion [10] evaluates and ranks the sensors that best describe the classes and simultaneously avoid sensors that provide redundant information by means

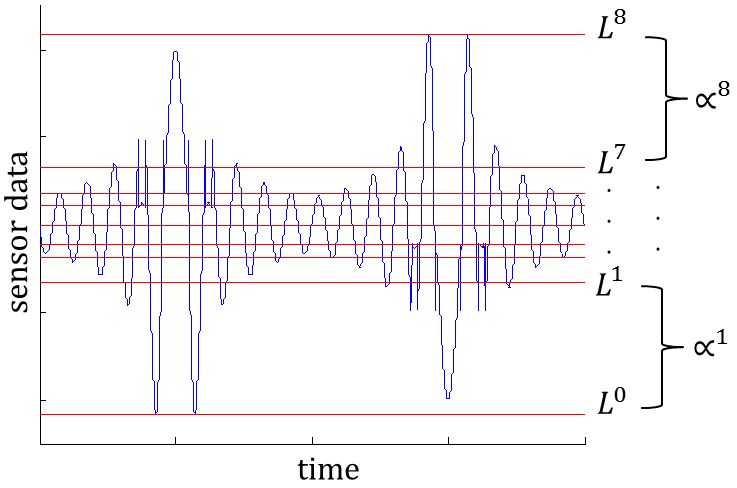


Figure 2.6: An illustration of the Maximum Entropy Partitioning

of the following two conditions: i) *Maximum Relevance* and ii) *Minimum Redundancy*, as described below.

Let us define the random variables C and S_j, $j = 1, ...m$, as follows:

- C: A random variable whose sample space is the set of all symbol sequences and its outcome belongs to the class set $\mathscr{C} = \{c_1, \ldots c_n\}$, and
- S_j: A random variable whose sample space is the symbolized data matrix $\tilde{\mathbf{z}}_j$ for sensor s_j and its outcome belongs to Σ.

Then the Maximum Relevance criteria is defined as follows.

Definition 2.4.4 (Max Relevance) *The Maximum Relevance criterion aims to find the set $\mathscr{U}_1^* \subseteq \mathcal{S}$, where $|\mathscr{U}_1^*| = m^*$, $m^* < m$, that has the maximum average mutual information between its sensors and the random variable C, such that*

$$\begin{aligned} \mathscr{U}_1^* &= \arg \max_{\mathscr{U}_1 \subseteq \mathcal{S}, |\mathscr{U}_1| = m^*} J_1(\mathscr{U}_1, C); \\ J_1(\mathscr{U}_1, C) &= \frac{1}{m^*} \sum_{s_j \in \mathscr{U}_1} I(S_j, C) \end{aligned} \tag{2.16}$$

The Maximum-Relevance criterion does not account for the information redundancy between sensors. Thus, the Minimum Redundancy criteria is defined as follows.

Definition 2.4.5 (Min Redundancy) *The Minimum Redundancy criterion aims to find the set $\mathscr{U}_2^* \subseteq$*

$\mathcal{S}$, where $|\mathcal{U}_2^*| = m^*$, $m < m$, that minimizes average mutual information between its sensor pairs, such that*

$$\mathcal{U}_2^* = \arg \min_{\mathcal{U}_2 \subseteq \mathcal{S}, |\mathcal{U}_2| = m^*} J_2(\mathcal{U}_2),$$
$$J_2(\mathcal{U}_2) = \frac{1}{(m^*)^2} \sum_{s_\ell, s_j \in \mathcal{U}_2} I(S_\ell, S_j) \tag{2.17}$$

The mRMR criterion combines the above two criterion as follows.

Definition 2.4.6 (minimum Redundancy Maximum Relevance) *The minimum Redundancy Maximum Relevance (mRMR) criterion aims to find the set $\mathcal{S}^* \subseteq \mathcal{S}$, where $|\mathcal{S}^*| = m^*$, $m^* < m$, to optimize J_1 and J_2 simultaneously, such that*

$$\mathcal{S}^* = \arg \max_{\mathcal{U} \subseteq \mathcal{S}, |\mathcal{U}| = m^*} J(\mathcal{U}, C),$$
$$J(\mathcal{U}, C) = J_1(\mathcal{U}, C) - J_2(\mathcal{U}) \tag{2.18}$$

The evaluation of the mRMR criteria requires: a) computation of $I(S_j, C)$ and $I(S_\ell, S_j)$, $\forall \ell, j = 1, 2, \ldots m$, and b) finding the solution of the optimization function in Eq. (2.18). The mutual information quantities are computed from the symbol sequences of each sensor data, as described in Appendix A.2. The optimization problem based on the mRMR criterion is a combinatorial problem, which can be solved using the *Forward Selection* search method [10]. Note: this information-theoretic method of sensor selection is more efficient and several orders of magnitude faster as compared to the full wrapper method that requires computation of the CCRs for all sensors.

Forward Selection Search

The forward selection search is a greedy search algorithm that is used to find a (sub)optimal solution of the mRMR optimization problem in Eq. (2.18). To be specific, the algorithm starts with an empty set of sensors, then keep adding sensors that maximize the mRMR criteria until the desired number of m sensors is obtained. The details [66] of the algorithm are shown in Algorithm 1.

Result: An optimal set of sensors $\mathcal{S}^* = \{s_\ell^*,\ \forall\, \ell = 1, 2, \ldots m^*\}$.
Initialization: $\mathcal{S} = \{s_1, s_2, \ldots s_m\}$, $\mathcal{S}^* = \emptyset$, $\ell = 1$
while $\ell \leq m^*$ **do**

- Step 1: Find the sensor $s_\ell^* \in \mathcal{S}$ that maximizes the criterion in Eq. (2.18) for a single sensor
- Step 2: Update $\mathcal{S} \rightarrow \mathcal{S} - s_\ell^*$
- Step 3: Update $\mathcal{S}^* \rightarrow \mathcal{S}^* \cup s_\ell^*,\ \ell \rightarrow \ell + 1$

end

Algorithm 1: *The Forward Selection Search Algorithm*

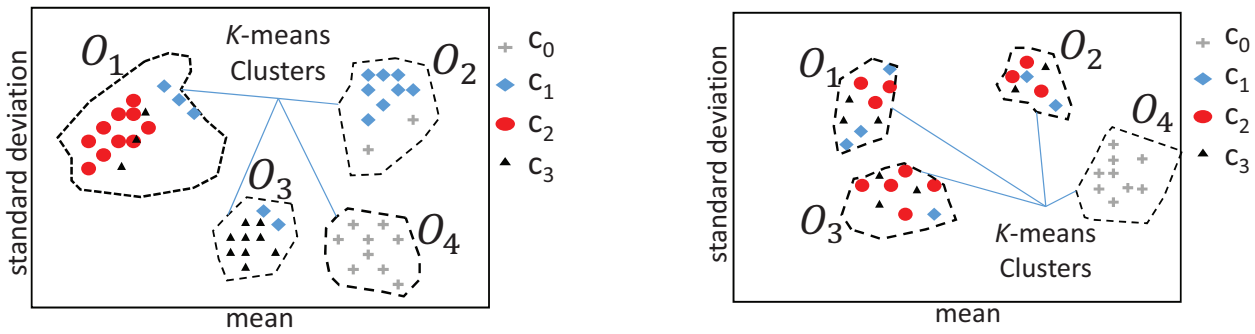
2.4.4 Embedded Algorithm

As mentioned earlier, an *embedded wrapper and filter* algorithm is used to tradeoff between the low complexity of filter algorithms and the accuracy of wrapper algorithms in the optimal sensor set selection procedure. In other words, an embedded algorithm uses a filter algorithm first to select a *candidate list* (CL) of sensors; subsequently, a wrapper algorithm (which uses a specific classifier) is deployed to select or rank the *optimal set of sensors*[10] from the candidate list. The embedded algorithms also have several deficiencies including being classifier specific and requiring tuning the classifier beforehand for each sensor separately. To circumvent these disadvantages, the paper proposes the unsupervised embedded algorithm as described next.

2.4.5 Unsupervised Embedded Algorithm

The unsupervised embedded algorithm also relies on a filter algorithm (e.g., the mRMR) to select the candidate list (CL) of m sensors. Then the data Z_j corresponding to each sensor $s_j \in$ CL, which consists of the data of all classes, are clustered into n clusters using the K-means clustering algorithm [67], where n is equal to the number of fouling classes (for this paper $n = 4$). Lets call these clusters as $\{\mathcal{O}_1, \mathcal{O}_2, \ldots \mathcal{O}_n\}$. Lets now define a random variable O_i that is drawn on the cluster $\mathcal{O}_i$ and whose outcome belongs to the set of classes $\mathscr{C} = \{c_1, \ldots c_n\}$. Subsequently, the entropy $H(O_i), i = 1, \ldots n$, of the class distribution within each cluster is computed using Eq. (2.5). Then, the weighted entropy sensor s_j is calculated as

$$H_j = \sum_{i=1}^{n} \frac{|\mathcal{O}_i|}{|\Gamma| \cdot T} \cdot H(O_i). \tag{2.19}$$



(a) High-ranking, low-entropy sensor (b) Low-ranking, high-entropy sensor

$\mathbf{O_i}$: a cluster $\forall\ i = 1, 2, 3, 4$ $\mathbf{c_i}$: classes $\forall\ i = 0, 1, 2, 3$

Figure 2.7: An example of using the K-means clustering algorithm and the entropy criterion to rank sensors

where $|\Gamma| \cdot T = \sum_{i=1}^{n} |\mathcal{O}_i|$ is the total number of training data for sensor $s_j \forall s_j \in CL$.

Finally, the sensors are ranked according to their entropies such that the sensor that has the lowest entropy is ranked the highest and so on. In this fashion the candidate list is re-ranked and a possible list of top ranked sensors is selected for further analysis. This process ranks the sensors in the order such that the sensors that have the least uncertainty between classes in their data clusters are ranked the highest, thus facilitating a better classification decision.

Figures 2.7(a) and 2.7(b) show examples of a high and low rank sensors, respectively. As seen in the figures, when the average entropy H_j for a sensor $s_j \in \mathcal{S}$ is low, the *inter-class distances* between samples that belong to different classes are high while the *within-class distances* (i.e., distances between samples that belong to the same class) are low. Therefor, low entropy sensors are expected to achieve higher CCRs.

2.5 Data Analysis for Fouling Diagnosis

Once an optimal sensor set is obtained, different machine learning methods are applied for analysis of sensor data for fouling diagnosis. These methods consists of the feature extraction and the classification steps as described below.

2.5.1 Feature Extraction

Two methods of feature extraction are applied in this chapter; namely, the *Principal Component Analysis* (PCA) and the *Gaussian Mixture Model* (GMM) as described below.

Principal Component Analysis (PCA)

The Principal Component Analysis (PCA) is a data reduction method. Given a data matrix $\mathbf{X}$ of dimension $L \times m$, where $L > m$, whose columns are data vectors (e.g., sensor data). The objective of PCA is to transform the data matrix $\mathbf{X}$ into a matrix $\mathbf{X}'$ of size $L \times m'$, where $m' < m$. The columns of $\mathbf{X}'$ hold the *Score Vectors* (also known as the *Principal Components*). This transformation is accomplished using the *Karhunen-Loéve* (KL) algorithm which is summarized in Appendix B.1. For implementation to the heat exchanger data, consider the sensor data Z_j for a specific sensor s_j for a specific day type. The nominal data corresponding to class c_0 is extracted from Z_j and averaged to get a time series data $\bar{\mathbf{z}}_j^0$ as follows:

$$\bar{\mathbf{z}}_j^0 = \frac{1}{256} \sum_{\gamma \in \{\gamma_1 \cdots \gamma_{256}\}} \mathbf{z}_j^0(\gamma) \tag{2.20}$$

Thereafter, the steady state part of $\bar{\mathbf{z}}_j^0$ is partitioned into $m = 10$ segments, each of length $L = 30$. These data segments are organized to form a $L \times m$ data matrix X^0. Following the steps of the KL algorithm, the $m \times m'$ transformation matrix Ω is then obtained, where $m' = 2$ and is kept fixed. Subsequently, the scores (or principal components) of any observation sequence $\mathbf{z}_j(\gamma)\ \forall\ \gamma = 1, 2, \ldots |\Gamma|$ are generated by reorganizing the sequence into an $L \times m$ matrix $\mathbf{X}$ by breaking $\mathbf{z}_j(\gamma)$ into m segments of length L each. The scores of this sequence are then computed using Eq. (B.1) in Appendix B.1. These scores consist of L points each of which are $m' = 2$ dimensional. In the training phase these scores are labeled with fouling class and are sent to the classifier for training, while in the testing phase they are unlabeled and are sent to the trained classifier for decision on the fouling class.

Gaussian Mixture Model (GMM)

Consider sensor data $\mathbf{z}_j(\gamma_r) = [Z_j(r,1), Z_j(r,2), \dots Z_j(r,T)]$ for some $\gamma_r \in \Gamma$. The Gaussian Mixture Model (GMM) is a statistical model of $\mathbf{z}_j(\gamma_r)$ represented as a sum of R different Gaussian distributions as

$$p(\mathbf{z}_j(\gamma_r)|\mathcal{M}) = \sum_{\ell=1}^{R} \rho^\ell \cdot \mathcal{G}(z; \mu^\ell, \sigma^\ell) \tag{2.21}$$

where $M = \{\rho^1, \rho^2, \dots \rho^R, \mu^1, \mu^2, \dots \mu^R, \sigma^1, \sigma^2, \dots \sigma^R\}$ is the set of weights ρ^ℓs, means μ^ℓs and variances σ^ℓs, $\ell = 1, \dots R$, and z is a random variable. The function $\mathcal{G}(z, \mu^\ell, \rho^\ell)$ is a Gaussian distribution given as

$$\mathcal{G}(z; \mu^\ell, \sigma^\ell) = \frac{1}{(2\pi)^{1/2}\sqrt{\sigma^\ell}} \times e^{-\frac{1}{2}(z-\mu^\ell)'(\sigma^\ell)^{-1}(z-\mu^\ell)} \tag{2.22}$$

The parameters ρ^ℓ, μ^ℓ and $\rho^\ell, \forall \ell = 1, 2, \dots R$, are estimated from the data using the *Expectation Maximization* (EM) algorithm [68]. Subsequently, $\mathcal{M}$ is used as a feature for the classifier. For implementation to the heat exchanger data, a GMM is constructed from sensor data with $R = 2$. For each observation sequence $\mathbf{z}_j(\gamma_r)\ \forall\ r = 1, 2, \dots |\Gamma|$ in Z_j, the feature set $\mathcal{M}_j(\gamma_r)$ is computed as

$$\mathcal{M}_j(\gamma_r) = [\rho_j^1, \rho_j^2, \mu_j^1, \mu_j^2, \sigma_j^1, \sigma_j^2](\gamma_r) \tag{2.23}$$

2.5.2 Classification

Once the features are obtained as the principal components or the parameter set of the GMM, they are processed by a classifier to make a decision on the heat exchanger fouling severity. This section describes the k-*Nearest Neighbor* (k-NN) algorithm that is used as the classification technique.

k-Nearest Neighbor (k-NN)

The k-Nearest Neighbor (k-NN) classification algorithm is popular for its simplicity, efficiency and low complexity. The k-NN classifier is explained in Appendix B.3. For implementation,

the k-NN classifier is applied on the feature space generated by PCA and GMM for each sensor in the candidate list and each day type. For each observation sequence $\mathbf{z}_j(\gamma_r)$, the PCA based features (i.e., principal components) are $L = 30$ dimensional vectors in $m' = 2$ dimensional feature space while the GMM based features (i.e. the parameter set of GMM) are 1 dimensional in the 6 dimensional feature space. Since the PCA based features are vectors, the k-NN classifier produces L decisions one for each point in the vector. Then, a single decision is obtained from these decisions using the simple majority rule. In the training phase, the optimal k is selected that results in the highest CCR using the cross-validation algorithm for different values of $k = 1, 3, 5, \ldots 21$.

For the cross-validation method, 30 data sequences are hold-out from each class and are used for testing while the remaining are used for training. This process is repeated 50 times, where in each run 30 data sequences are randomly selected from each class. This generates a total of 1500 testing samples for each class. The results are summarized into a *confusion matrix* [69], whose columns contain the predicted classes while the rows contain the actual classes.

2.5.3 Sensor Fusion

Sensor fusion is performed for further improvement in the classification performance. Suppose for a given day type the top three optimal sensors are $\{s_1^*, s_2^*, s_3^*\}$, where $s_j^* \in \mathcal{S}^* \subseteq \mathcal{S}, \forall j = 1, 2, 3$. Moreover, lets say that the classifiers have generated the following three decisions $\{\hat{c}^1, \hat{c}^2, \hat{c}^3\}$, corresponding to the above three sensors, where $\hat{c}^j \in \mathscr{C}, \forall j = 1, 2, 3$. Then, the fusion decision $\hat{c}^F$ is obtained using the majority fusion method. Appendix C.3 presents a mathematical description of the majority fusion method.

2.6 Results and Discussion

This section presents the results for optimal sensor selection and the classification for heat exchanger fouling diagnosis. As shown in the methodology in Fig. 2.3, the first step is training of the classifiers. Therefore, sensor data is generated using a Simulink model that has been experimentally validated by our industry partner. For each sensor $s_j \in \mathcal{S}$ the data is stored in a $|\Gamma| \times T$

Table 2.3: Decreasingly Sorted Candidate List and Optimal Sensor Sets

	Day Type 1				Day Type 2				Day Type 3				Day Type 4				Day Type 5			
Rank	**CL**	**UE**	**EP**	**EG**	**CL**	**UE**	**EP**	**EG**	**CL**	**UE**	**EP**	**EG**	**CL**	**UE**	**EP**	**EG**	**CL**	**UE**	**EP**	**EG**
1	T_{cm}	T_{cm}	T_{hx}	T_{hx}	T_{hx}	T_{hx}	T_{hx}	T_{hx}	T_{hx}	P_o	T_{hx}	T_{hx}	T_{hx}	T_{out}	T_{hx}	T_{out}	T_{hx}	T_{hx}	T_{out}	T_{out}
2	T_{out}	T_{hx}	T_{cm}	T_{cn}	T_{out}	T_{cm}	T_{cm}	P_o	T_{cn}	T_{hx}	T_{cm}	T_{cm}	T_{cn}	P_o	T_{out}	T_{hx}	T_{cm}	T_{out}	T_{hx}	T_{hx}
3	T_{cn}	P_1	T_{cn}	P_2	P_2	P_o	P_o	P_2	T_{out}	T_{out}	P_o	P_o	T_{out}	T_{hx}	P_o	P_o	P_o	P_o	P_o	P_o
4	$\dot{m}_1$	P_2	P_o	P_1	T_{cn}	P_1	P_2	P_1	P_2	T_{cm}	T_{out}	T_{out}	P_o	T_{cn}	T_{cm}	T_{cm}	T_{out}	P_2	T_{cm}	T_{cm}
5	P_2	P_o	P_1	T_{cm}	T_{cm}	P_2	T_{cn}	T_{cm}	$\dot{m}_2$	P_1	P_2	P_2	T_{cm}	T_{cm}	P_2	P_2	P_2	P_1	P_2	P_2
6	T_{hx}	T_{cn}	P_2	P_o	$\dot{m}_1$	T_{out}	P_1	T_{cn}	P_1	P_2	P_1	P_1	T_{FD}	P_1	P_1	P_1	P_1	T_{cm}	P_1	P_1
7	$\dot{m}_2$	T_{out}	$\dot{m}_1$	T_{out}	P_1	T_{cn}	T_{out}	$\dot{m}_1$	P_o	T_2	$\dot{m}_2$	T_2	P_1	P_2	T_{cn}	T_{cn}	T_{FD}	T_{cn}	T_{FD}	T_{cn}
8	P_1	$\dot{m}_1$	$\dot{m}_2$	$\dot{m}_2$	P_o	$\dot{m}_1$	$\dot{m}_2$	$\dot{m}_2$	T_2	T_1	T_2	T_1	P_2	T_{FD}	T_{FD}	T_{FD}	T_{cn}	T_{FD}	T_{cn}	T_{FD}
9	P_o	$\dot{m}_2$	T_{out}	$\dot{m}_1$	$\dot{m}_2$	$\dot{m}_2$	$\dot{m}_1$	T_{out}	T_{cm}	T_{cn}	T_1	$\dot{m}_2$	T_{Z1}	T_{Z1}	T_{Z1}	T_{Z1}	T_{Z2}	T_{Z2}	T_{Z1}	T_{Z2}
10	T_{Z5}	T_{Z5}	T_{Z5}	T_{Z5}	T_{Z5}	T_{Z5}	T_{Z5}	T_{Z5}	T_1	$\dot{m}_2$	T_{cn}	T_{cn}	T_{Z5}	T_{Z5}	T_{Z5}	T_{Z5}	T_{Z1}	T_{Z1}	T_{Z2}	T_{Z1}

CL Candidate List **UE** Unsupervised Embedded

EP, **EG** Embedded with PCA+k-NN and GMM+k-NN

matrix Z_j which includes the data for all four fouling classes as described in Section 2.3.2. The rows of the matrix consist of various parametric combinations of the ambient temperature (T_A) within a specific day type, the occupant count (OCC), and the impedance values (z_c) within each class, as shown in Tables 2.2(a)-2.2(c). As described in Section 2.3.2, for each class there are 256 data sequences each of length $T = 600$, thus resulting in a total of $4 \times 256 = 1024$ sequences for each day type. Similar data are generated for each day type. To include the effect of other uncertainties beyond the above parametric uncertainties, the data is corrupted by $25dB$ additive white Gaussian noise, and the noisy data are plotted in Fig. 2.5. The noise is filtered out using 5-levels wavelet denoising technique [70] using 6-taps Daubechies wavelet with *soft-thresholding* on the detail coefficients of the wavelet. The thresholds are determined using *Stein Unbiased Risk Estimator* (SURE). The fouling diagnosis methodology uses only the last 300 seconds of steady state data for analysis.

Subsequently, the data of all sensors for one day type are taken and the mRMR technique is applied to find the candidate list of top 10 sensors. To compute the information theoretic quantities in the mRMR criterion, the sensor data is transformed into symbol sequences using the Maximum Entropy Partitioning as described in Section 2.4.2. The above process is repeated for each day type. The resultant candidate lists (CL) are shown in Table 2.3.

Once the candidate list is generated for each day type using the mRMR criteria, the unsupervised embedded algorithm is applied on the candidate list, which includes a two-step process: i) K-means clustering of each sensor data, ii) computation of weighted average cluster entropies and sensor ranking as described in Section 2.4.5. The resultant sensor ranking is presented in Table 2.3. For comparison, the embedded algorithm is used to rank the sensors which used two different wrappers: i) PCA as the feature extractor and k-NN as the classifier, and ii) GMM as the feature extractor and k-NN as the classifier. As seen in Table 2.3, the results of unsupervised embedded algorithm improved the candidate list rankings which were purely based on information-theoretic quantities, such that the updated rankings are similar to the embedded wrapper results. However, the unsupervised embedded algorithm does not depend on the feature extraction and classification methods and is computationally much more efficient.

Using the PCA procedure for feature extraction as described in Section 2.5.1, a feature vector consisting of 30 feature points are extracted from each data sequence in the data matrix Z_j. Each feature point is composed of the elements of the first and second principal components. These principal components are computed from the eigenvectors whose corresponding eigenvalues contain more than 90% of the energy. Since ambient temperature can be measured and known, it is used to augment the feature space as the third axis where the other two axis are formed by the two principal components. As a result, we have 30 feature points for each data sequence plotted in a 3D feature space. This resulted in excellent clustering of the classes as shown in Fig. 2.8.

Using the GMM procedure for feature extraction as described in Section 2.5.1, the second order GMM was used to extract a feature vector of size 1×6 from each data sequence which is composed of the weights, means, and variances of the 2 mixtures as explicitly stated in Eq. (2.23). Similar to the PCA procedure, the GMM feature space is also augmented with the ambient temperature. Due to its high dimension, GMM feature vectors are difficult to visualize; however, it leads to high CCR in conjunction with k-NN classifier as discussed below.

The confusion matrices of the PCA + k-NN and the GMM + k-NN methods are presented in Tables 2.4 and 2.5, respectively. An interesting observation on the confusion matrices is that false-alarms and miss-detections mostly occur between adjacent fouling classes. Table 2.6 presents the

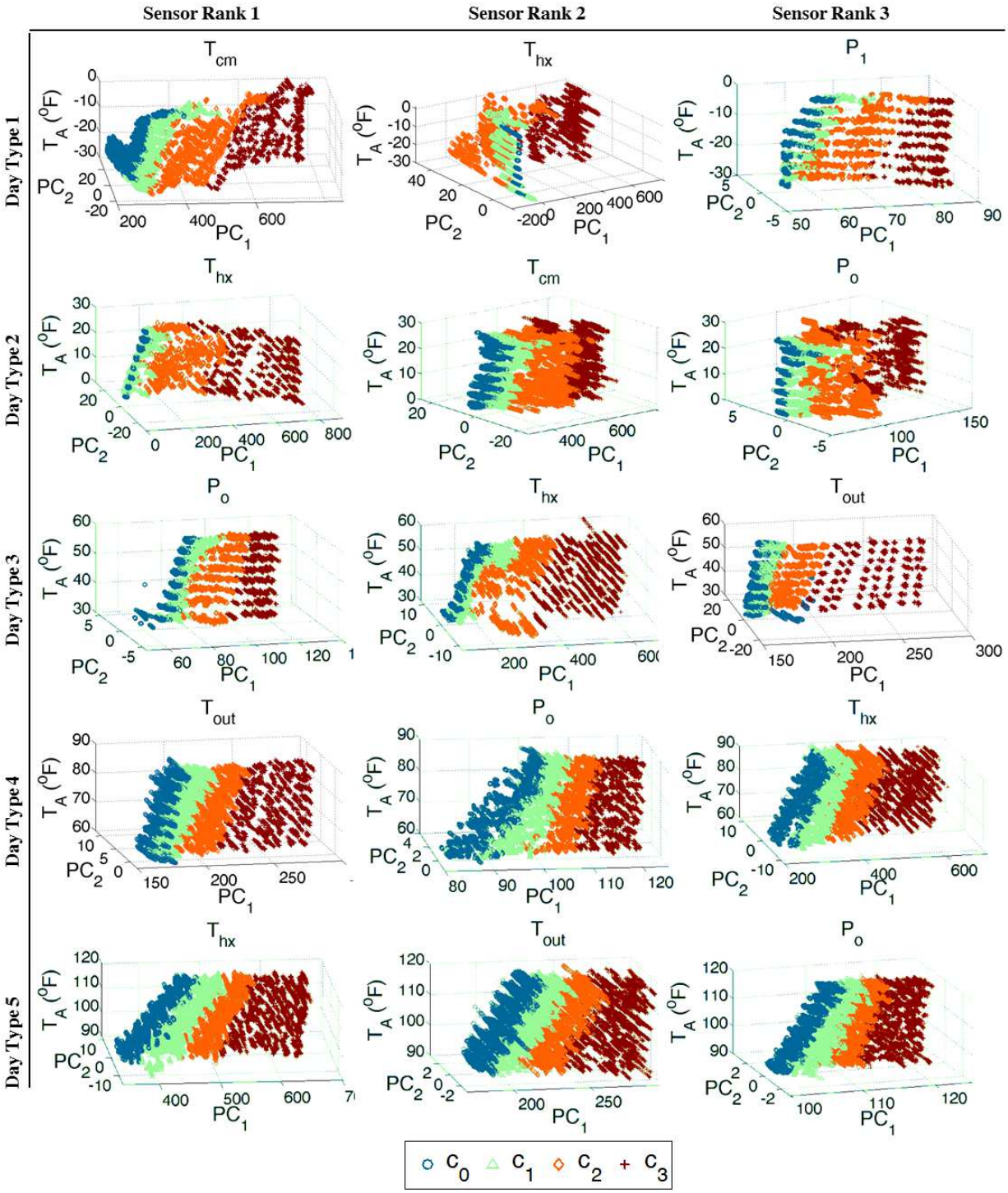


Figure 2.8: Top three unsupervised embedded optimal sensors' principal components 1 and 2 vs the ambient temperature for day types $1, 2, \ldots 5$

CCR results which indicate that the performances of PCA and GMM are comparable, with a slight lead for the GMM.

Table 2.6 shows that not only do optimal sensors selected using the unsupervised embedded algorithm lead to high CCRs (above 85% in most of the cases), but also the majority vote fusion of resultant sensors lead to superior results (above 94% for most of the day types). Besides that, testing the PCA requires majority vote among the 30 feature vectors (as described in Section 2.5.1) unlike the GMM; this makes the GMM much faster to train and test. Nonetheless, training the GMM requires the use of the Expectation Maximization (EM) algorithm, which is more expensive than the KL algorithm. The computation times on 32-bit MatLab running on a 3.10 GHz Intel(R) Core(TM) $i5-2400$ processor, 16 GB ram and Windows 7 Operating System are shown

Table 2.4: Confusion Matrices of the Optimal Sensors with PCA + k-NN for Classification

Day Type		**Optimal Sensors Selected Using UE Algorithm**												**Majority Vote Fusion**			
		s_1^*				s_2^*				s_3^*				**of s_1^*, s_2^* and s_3^***			
		Classifier Output				**Classifier Output**				**Classifier Output**				**Classifier Output**			
		c_0	c_1	c_2	c_3	c_0	c_1	c_2	c_3	c_0	c_1	c_2	c_3	c_0	c_1	c_2	c_3
Day Type 1	c_0	1378	122	0	0	1354	146	0	0	1284	216	0	0	1439	61	0	0
Actual	c_1	129	1261	110	0	105	1295	100	0	311	1112	77	0	130	1332	38	0
	c_2	6	145	1303	46	0	108	1373	19	0	151	1300	49	2	65	1419	14
	c_3	0	0	60	1440	0	0	41	1459	0	0	50	1450	0	0	12	1488
Day Type 2	c_0	1352	148	0	0	1325	175	0	0	1314	186	0	0	1435	65	0	0
Actual	c_1	146	1286	68	0	213	1196	91	0	401	1015	84	0	173	1305	22	0
	c_2	0	118	1330	52	0	131	1278	91	0	112	1357	31	0	79	1414	7
	c_3	0	0	47	1453	0	4	80	1416	0	0	33	1467	0	0	5	1495
Day Type 3	c_0	1223	277	0	0	1326	174	0	0	1065	412	7	16	1349	151	0	0
Actual	c_1	287	1119	94	0	162	1248	90	0	400	1023	77	0	229	1259	12	0
	c_2	0	99	1374	27	0	89	1411	0	25	107	1368	0	1	34	1465	0
	c_3	0	0	9	1491	0	0	1	1499	6	0	0	1494	0	0	0	1500
Day Type 4	c_0	1269	231	0	0	1332	168	0	0	1359	141	0	0	1419	81	0	0
Actual	c_1	218	1271	11	0	205	1223	72	0	119	1360	21	0	97	1399	4	0
	c_2	0	1	1499	0	0	102	1356	42	0	41	1459	0	0	9	1491	0
	c_3	0	0	0	1500	0	0	46	1454	0	0	2	1498	0	0	0	1500
Day Type 5	c_0	1413	87	0	0	1433	67	0	0	1396	104	0	0	1480	20	0	0
Actual	c_1	84	1414	2	0	94	1392	14	0	106	1330	64	0	40	1460	0	0
	c_2	0	11	1489	0	0	17	1483	0	0	73	1351	76	0	4	1496	0
	c_3	0	0	18	1482	0	0	0	1500	0	0	73	1427	0	0	2	1498

k Number of nearest neighbors for the k-NN algorithm **UE** Unsupervised Embedded

Table 2.5: Confusion Matrices for the Optimal Sensors with GMM + k-NN for Classification

Day Type			Optimal Sensor Selected Using UE Algorithm s_1^*				s_2^*				s_3^*				Majority Vote Fusion of s_1^*, s_2^* and s_3^*			
			Classifier Output				Classifier Output				Classifier Output				Classifier Output			
			c_0	c_1	c_2	c_3	c_0	c_1	c_2	c_3	c_0	c_1	c_2	c_3	c_0	c_1	c_2	c_3
Day Type 1	Actual	c_0	1289	210	1	0	1385	115	0	0	1303	197	0	0	1438	62	0	0
		c_1	194	1195	111	0	113	1334	53	0	288	1205	7	0	111	1373	16	0
		c_2	11	131	1322	36	0	129	1326	45	0	170	1303	27	2	60	1419	19
		c_3	0	0	89	1411	0	0	81	1419	0	0	63	1437	0	0	14	1486
Day Type 2	Actual	c_0	1371	129	0	0	1243	257	0	0	1354	146	0	0	1438	62	0	0
		c_1	158	1286	56	0	254	1140	106	0	385	1080	35	0	183	1303	14	0
		c_2	0	121	1352	27	1	126	1279	94	0	103	1353	44	0	53	1436	11
		c_3	0	0	64	1436	0	0	100	1400	0	0	41	1459	0	0	14	1486
Day Type 3	Actual	c_0	1317	183	0	0	1366	134	0	0	962	493	21	24	1384	116	0	0
		c_1	339	1122	39	0	118	1363	19	0	437	1028	35	0	187	1312	1	0
		c_2	0	97	1387	16	0	89	1411	0	14	141	1345	0	4	20	1476	0
		c_3	0	0	35	1465	0	0	5	1495	0	0	27	1473	0	0	1	1499
Day Type 4	Actual	c_0	1344	127	29	0	1398	102	0	0	1447	53	0	0	1479	21	0	0
		c_1	141	1349	10	0	203	1249	48	0	158	1335	7	0	90	1409	1	0
		c_2	0	40	1454	6	0	99	1366	35	0	98	1396	6	0	18	1478	4
		c_3	0	0	11	1489	0	0	59	1441	0	0	50	1450	0	0	3	1497
Day Type 5	Actual	c_0	1412	88	0	0	1411	89	0	0	1413	87	0	0	1475	25	0	0
		c_1	93	1392	15	0	82	1402	16	0	122	1342	36	0	30	1470	0	0
		c_2	0	31	1464	5	0	50	1443	7	0	100	1378	22	0	10	1490	0
		c_3	0	0	36	1464	0	0	7	1493	0	0	112	1388	0	0	4	1496

k Number of nearest neighbors for the k-NN algorithm **UE** Unsupervised Embedded

in Table 2.7 for training and testing phases.

2.7 Conclusions

The chapter presented a methodology for fouling diagnosis of the Secondary Heat Exchanger in the *Environmental Control System* of an aircraft that regulates temperature, pressure and humidity of the cabin air as well as the air used to cool electronics onboard the aircraft. Since the ECS contains a large number of sensors, an optimal sensor selection methodology is presented to select the most useful sensors that provide the best diagnosis results. The results of unsupervised embedded (UE)

Table 2.6: Classification Results for the Top Three Optimal Sensors Obtained using the UE Algorithm

Day Type	UE		PCA + k-NN			GMM + k-NN		
	Sensor	**NE**	**CCR (%)**	**FA (%)**	**MD (%)**	**CCR (%)**	**FA (%)**	**MD (%)**
Day Type 1	T_{cm}	0.50	89.7	8.1	3.0	87.0	14.1	4.6
	T_{hx}	0.51	91.4	9.7	2.3	91.1	7.7	2.5
	P_1	0.52	85.8	14.4	6.9	87.5	13.1	6.4
	MVF	-	94.6	4.1	2.9	95.3	4.1	2.5
Day Type 2	T_{hx}	0.42	90.4	9.9	3.2	90.8	8.6	3.5
	T_{cm}	0.45	86.9	11.7	4.7	84.4	17.1	5.7
	P_o	0.45	85.9	12.4	8.9	87.4	9.7	8.6
	MVF	-	94.2	4.3	3.8	94.4	4.1	4.1
Day Type 3	P_o	0.44	86.8	18.5	6.4	88.2	12.2	7.5
	T_{hx}	0.46	91.4	11.6	3.6	93.9	8.9	2.6
	T_{out}	0.50	82.5	29.0	9.6	80.1	35.9	10.0
	MVF	-	92.9	10.1	5.1	94.5	7.7	4.2
Day Type 4	T_{out}	0.44	92.3	15.4	4.8	93.9	10.4	3.1
	P_o	0.46	89.4	11.2	4.6	90.9	6.8	4.5
	T_{hx}	0.47	94.6	9.4	2.6	93.8	3.5	3.5
	MVF	-	96.8	5.4	2.2	97.7	1.4	2.0
Day Type 5	T_{hx}	0.49	96.6	5.8	1.9	95.5	5.9	2.1
	T_{out}	0.49	96.8	4.5	2.1	95.8	5.9	1.8
	P_o	0.51	91.7	6.9	2.4	92.0	5.8	2.7
	MVF	-	98.9	1.3	0.9	98.9	1.7	0.7

UE Unsupervised Embedded Algorithm
CCR Correct Classification Rate (%)
NE Normalized Entropy
MVF Majority Vote Fusion
FA False Alarm Rate (%)
MD Miss-Detection Rate (%)

algorithm for sensor selection are compared with embedded wrapper algorithms. It is shown that the sensors ranked by UE algorithm yield excellent classification results with significant improvement in computational complexity. Subsequently, the data of the top ranked sensors are analysed using the k-NN classifier in combination with either PCA or GMM as feature extractors and results are compared. The data is generated from an experimentally validated high-fidelity Simulink model of the ECS provided by our industry partner and included various uncertainties generated by parametric fluctuations in ambient temperature, occupant count, and flow impedance. Finally, the majority vote algorithm is applied as a simple fusion technique to improve the results.

The following areas are envisioned for future research:

Table 2.7: Computation Time for Various Procedures Used for Heat Exchanger Fouling Diagnosis

Procedure	**Training Time (Per Day Type)**	**Testing Time (Per Observation)**
Maximum Entropy Partitioning	1 s	-
mRMR	5.2 s	-
Clustering + Entropty Based Ranking	4.8 s	-
GMM Feature Extraction	7.55 min	-
PCA Feature Extraction	1.06 s	-
PCA + k-NN	7.55 min	50 ms
GMM + k-NN	36.1 s	0.7 ms
PCA + k-NN + Majority Vote	22.65 min	150 ms
GMM + k-NN + Majority Vote	108.3 s	2.1 ms

- Real-time implementation of the heat exchanger fouling diagnosis methodology on actual aircraft data
- Testing and validation of different sensor fusion methods
- Utilization of different machine learning tools for improving the classification performance
- Development of a similar fouling diagnosis methodology for aircraft cruise operating conditions.

CHAPTER 3

BB-FUSE: OPTIMAL SENSOR-CLASS PAIR SELECTION AND FUSION ALGORITHM FOR CLASSIFICATION

3.1 Introduction

Information fusion is the art of combining information from different sources to enhance *information quality*, where the information quality can be evaluated using different measures depending on the application (e.g., *Correct Classification Rate* (CCR), tracking accuracy, etc.). Often the data collected from different sensors is uncertain, conflicting, incomplete and/or redundant, thus making information fusion a challenging task [71]. The need for information fusion has emerged in diverse applications such as biometrics identification [4, 7, 72], geoscience [1, 73], target tracking [74, 75], audio processing [9, 76], text classification [8, 77], medical data [3, 78], surveillance sensor networks [19, 79], facial recognition [80–82], human activity recognition [**Najjar2015**, 83–85], image analysis [86–88], robotics [89, 90], and complex systems diagnostics [18, 91]. In general for n-class classification problems, one of the main objective of information fusion is to improve the CCR of the sensing system with multiple sensors.

However, if information fusion is not performed properly, it can sometimes degrade the information quality. This can happen if fusion is performed on sensors that do not all contain useful information pertaining to the underlying classification problem. Typically, it is observed that dif-

ferent sensors might have different classification accuracies in separating different classes. In other words, a sensor might be extremely good in classifying a certain subset of classes, but yield a poor performance in classifying others. Thus, an efficient fusion algorithm must capitalize on this fact to identify the best *sensor-class* pairs and fuse them effectively to yield an overall improved CCR.

In this regard, this chapter introduces a fusion algorithm, called *Better-than-the-Best Fusion* (BB-Fuse), which systematically fuses relevant information derived from optimal sensor-class pairs, to guarantee that the resulting CCR is better than or equal to the best CCR obtained by any single sensor. In the training phase, the algorithm constructs a fusion tree, called the *BB-Fuse Tree*, by finding the optimal sensor-class pairs and placing them on the nodes of the tree. This is done by constructing n binary confusion matrices for each sensor by training a classifier, where each confusion matrix corresponds to one particular class and evaluates the classifier performance in separating that class from the rest. Subsequently, the best sensor is selected for isolating each class, thus generating the optimal *sensor-class* pairs. The process is done in a sequential manner to built the BB-Fuse Tree which isolates one class at each node with maximum achievable CCR. In the testing phase, the BB-Fuse Tree is used on unlabeled sensor data by sequentially applying the sensor-class pairs designed at each level of the tree to obtain the final decision at a terminal node.

The BB-Fuse algorithm is validated on experimental data sets from two different applications. The first application is focused on human activity recognition where the objective is to classify basic hand movements in a kitchen environment using different sensors placed at multiple locations. The second application is focused on people identification from walking activity using acceleration sensor data collected from Android phones placed in their chest pockets.

The main contributions of this chapter are as follows:

- Development of the BB-Fuse algorithm for n-class classification problems, which produces higher CCR than that of the best sensor with provable guarantees.
- Testing and validation of the BB-Fuse algorithm on two different data sets and performance comparison with other fusion algorithms in literature.

The chapter is organized as follows. Section 3.2 presents the relevant literature review on fusion

methods for classification. Section 3.3 lays out the mathematical foundation for the BB-Fuse algorithm while Section 3.4 presents its details. Section 3.5 discusses the performance of BB-Fuse from optimality and complexity perspectives. Section 3.6 presents the testing and validation results and Section 3.7 concludes the chapter with recommendations for future work. Additionally, Appendix C describes other fusion algorithms that are used for comparative analysis with the BB-Fuse algorithm.

3.2 Literature Review

Several fusion methods have been proposed in literature with diverse applications. While some technical details are discussed in Appendix C, a brief review is presented below.

3.2.1 Voting Methods

Voting-based fusion methods are the simplest where the classifiers are the voters and the classes are the candidates. Three popular voting methods are reviewed below:

• Majority Voting

The simplest form of voting is the Majority Voting where the decision of each classifier is treated as its vote while the voted class earns a point. Then the class that scores the maximum points wins the election [92]. Majority Voting has been used in a variety of applications. For example, Kim et al. [93] used Majority Voting to develop a diagnostic system for an air-operated valve system that is used in nuclear power plants. Specifically, they used Majority Voting to fuse logistic regression, artificial neural network, and support vector machine classifiers. Orrite et al. [94] addressed the problem of selecting the smallest set of classifiers that maximizes recognition and then fused them using Majority Voting. Kuncheva et al. [95] derived the upper and lower bounds of the accuracy of the Majority Voting.

Despite the simplicity and effectiveness of Majority Voting, there is a possibility of ties. Further it has a drawback that a class may win even though it received a small number of votes; especially when the number of classes is large [92].

• Condorcet Count Voting

Condorcet Count Voting decreases the chances that a candidate wins with a small number of votes. Here instead of voting for all classes, the classifiers vote for all 2-class pairwise competitions, thus comparing each class with every other class. The class that scores the largest number of points wins the overall election [92]. Condorcet Count Voting has appeared in a number of applications. Ahmed et al. [96] applied this method to search a database of chemical compounds to select a compound that matches a desired biological activity. Wu [97] presented an improvement by using Linear Discriminant Analysis to design weights for the Condorcet Count Voting, and then applied *Weighted Condorcet Count Voting* for information retrieval.

Repeated pairwise class elections can make the Condorcet Count Voting computationally expensive [92].

• Borda Count Voting

Another improvement of the Majority Voting is the Borda Count Voting. Here each classifier expresses its confidence on its classification decision by ranking the classes using a function called the *Borda Count* [98]. Borda count for a class is defined as the number of classes ranked below it by the classifier. Then the class that has the maximum Borda count summed from all classifiers wins. Borda Count Voting has been applied in many applications [99]. An example is in fault diagnosis of induction motors for elevator traction machines where Niu et al. [100] fused support vector machine, linear discriminant analysis, k-Nearest Neighbors, random forests and adaptive resonance theory-Kohonen neural network classifiers using Majority Voting, Bayesian Belief, and modified Borda Count Voting methods.

Borda Count Voting is effective and easy to implement; however, its performance could degrade when erratic classifiers outnumber the accurate ones [101].

3.2.2 Bayesian Fusion Methods

Bayesian fusion methods [101] require the classifiers' outputs to be expressed as posterior probabilities. Posterior probabilities can be estimated using various techniques, e.g., using confusion

matrices or using posterior probabilities of *Bayes classifiers* [101]. Bayes Belief Integration is a member of the Bayesian fusion family that uses confusion matrices of individual classifiers to calculate the posterior probabilities of the classes given the decisions of all classifiers, then picks the class that maximizes it [101]. Qu et al. [102] proposed a method for mechanical fault diagnosis using dual-tree wavelet packet transform and classifier fusion using Bayes Belief method. Kabir et al. [103] developed a Bayesian Belief network based data fusion model for failure prediction in water mains. Jin et al. [104] presented a fuzzy ARTMAP ensemble approach using improved Bayesian Belief method for reliable fault diagnosis.

The implementation of Bayes Belief Integration method in the testing phase requires the acquisition of all classifiers decisions, which makes it computationally expensive.

3.2.3 Boosting Methods

Boosting algorithms [105] find a linear combination of a set of weak classifiers to obtain an enhanced decision. In the training phase, a classifier is added at each iteration which minimizes the error on previously miss-classified training samples. This is done according to a distribution which increases (decreases) weight of miss (correct) classified examples. *Adaptive Boosting* (Adaboost) is a popular boosting method that is based on the minimization of the *exponential loss* function and it achieves better than random guessing under certain conditions [105]. Adaboost has been used in many applications. Guo et al. [106] proposed a pedestrian detection method that combines Adaboost and SVM algorithms. Cheng and Jhan[107] proposed an AdaBoost-SVM classifier for triaxial accelerometer-based fall detection.

The popular classifiers (e.g., k-NN and *Support Vector Machines*) need to be slightly modified to accept training data distribution before the boosting algorithm is applied.

3.2.4 Decision Tree Methods

Several variants of decision trees [108] have been proposed in literature for classification and fusion. The most popular classification decision trees apply thresholds to different features at each node, to split the training data into two (or more) groups. The optimal feature is then selected

at each node using an *impurity measure* (e.g., entropy [109, 110], information index, Gini index [109], etc.) of the resultant groups. The process is repeated for all nodes and the algorithm terminates when the terminal nodes contain only single classes. The tree is then *pruned* to reduce the computational complexity [109, 110]. Three popular classification decision trees are Classification and Regression Trees (CaRT)[3, 109, 111], ID3 [110, 112], and C4.5 [113, 114].

In lieu of the above review, the BB-Fuse algorithm is positioned in the category of decision trees. The algorithm constructs a binary tree by selecting an optimal sensor-class pair at each node. The decision-making is not threshold based and it applies a classifier at each node to isolate one class at a time. Thereby, it provides an analytical guarantee to yield a CCR which is better than the best individual sensor. Furthermore, BB-Fuse has low-computational complexity because it sequentially uses only a selected subset of classifiers that maximize the isolation of one class at a time.

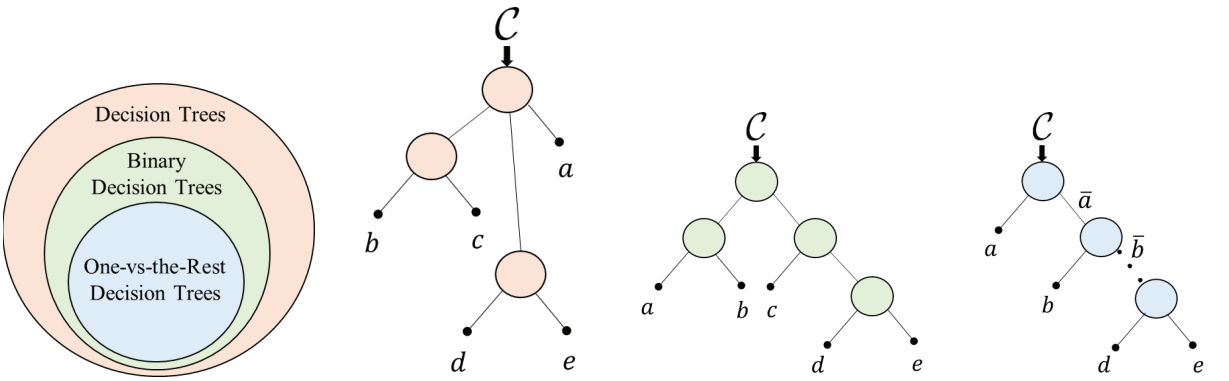
3.3 Mathematical Preliminaries

This section builds the mathematical foundation necessary for constructing the BB-Fuse algorithm.

Definition 3.3.1 (Graph [115]) *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes (or vertices) $\mathcal{V} = \{v_1, v_2, \ldots v_{|\mathcal{V}|}\}$ and a set of edges $\mathcal{E} = \{e_1, e_2, \ldots e_{|\mathcal{E}|}\}$, where each edge is a node pair of the form (v_i, v_j). If the order of the pairs (v_i, v_j) is specified (i.e., $(v_i, v_j) \neq (v_j, v_i)$, $\forall i \neq j$), then the graph $\mathcal{G}$ is called a directed graph; otherwise it is called an undirected graph.*

Further, the following are defined for a graph $\mathcal{G}$:

- A sequence $\{e_{(k)}\}_{k=1,2,\ldots}$ of edges $e_{(k)} = (v(k-1), v(k)) \in \mathcal{E}$, is called a *trail* if all edges are distinct.
- A *path* is a trail where all nodes are distinct.
- A *cycle* is a closed trail where only the first and the last nodes are the same.
- If the graph $\mathcal{G}$ has a cycle, then it is called a *cyclic graph*; otherwise, it is called an *acyclic graph*.



(a) Decision Trees Types (b) Decision Trees (c) Binary (d) One-vs-the-Rest

Figure 3.1: Decision Tree Types; $\mathscr{C} = \{a, b, c, d, e\}$ is the Class Set.

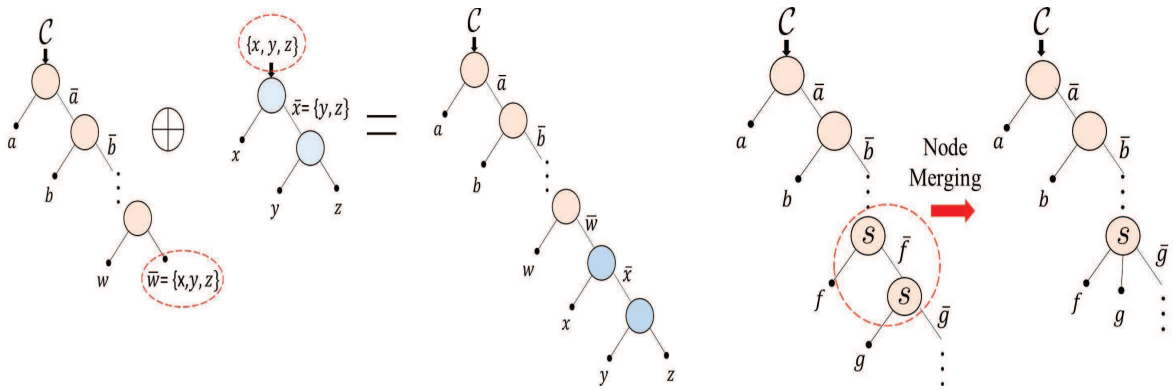
- For any directed edge $e_k = (v_i, v_j) \in \mathcal{E}$, v_i is called the *parent* of v_j, and v_j is called a *child* of v_i.

Definition 3.3.2 (Rooted-Tree [108]) *A rooted-tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is a (directed) acyclic graph that has one node, called the root, which has no parent, and there exists a unique path from the root to each node, i.e. every node except the root has only one parent. A node without children is called a terminal node (or a leaf).*

Definition 3.3.3 (Decision Tree) *A decision tree is a rooted tree where every non-terminal node consists of a hypothesis and an input class set such that the hypothesis divides the class set into multiple subsets forming inputs to its children. A unique classification decision for a testing pattern is made by sequential application of the hypotheses on nodes, thus tracing a path on the tree from the root to a leaf node.*

A *binary decision tree* is a tree where every non-terminal node has only two children. A *one-vs-the-rest decision tree* is a binary tree, which isolates one class from the rest of the classes available at each node. Fig. 3.1 shows the different types of decision trees. Finally, two tree operations, *grafting* and *node merging*, as shown in Fig. 3.2, are defined below.

Definition 3.3.4 (Grafting) *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two decision trees, where $\diamond$ is a leaf node of $\mathcal{T}_1$ and $\bullet$ is the root of $\mathcal{T}_2$, and the class sets at the node $\diamond$ and the root $\bullet$ match. Then the grafting operation $\mathcal{T}_1 \oplus \mathcal{T}_2$ is defined as grafting of $\mathcal{T}_2$ to $\mathcal{T}_1$ at node $\diamond$ by replacing the node $\diamond$ with the root $\bullet$.*



(a) Grafting Example (b) Node Merging Example

Figure 3.2: An Illustration of the Grafting and Node Merging Operations on One-vs-the-Rest Trees; where $\mathscr{C} = \{a, b, ...z\}$.

Definition 3.3.5 (Node Merging) *Let $\mathcal{T}$ be a decision tree and let v_i and v_j be any two of its non-terminal nodes such that v_j is a child of v_i. Then the node merging operation, $v_i \prec v_j$, is defined as merging of v_j with v_i if the following conditions hold:*

- *An M-ary hypothesis h (e.g. a M-class classifier) is applied at node v_i but only partially used to divide the input class set at v_i into $M' < M$ class subsets.*
- *The same hypothesis h is again partially applied at v_j to further divide the input class subset at v_j.*

3.4 BB-Fuse Algorithm

This section describes the details of the BB-Fuse algorithm. First, the objective of the algorithm is discussed, then its training and testing phases are explained.

3.4.1 Objective

Let $\mathcal{P}$ be the classification problem under consideration that aims to separate n classes from the class set $\mathscr{C} = \{c_1, c_2, \ldots c_n\}$. Let $\mathcal{F} = \{f_1, f_2, \ldots f_{|\mathcal{F}|}\}$ be the set of sensing devices (or modalities) which produce time series data for the observed phenomenon which has to be classified into the above classes. Further, let $\mathcal{H} = \{h_1, h_2, \ldots h_{|\mathcal{H}|}\}$ be the set of different clas-

sifiers. Now define a set $\mathcal{S} = \{s_1, s_2, ...s_m\}$ which is the product set of $\mathcal{F}$ and $\mathcal{H}$ such that $\mathcal{S} = \mathcal{F} \times \mathcal{H} = \{s \triangleq (f, h) : f \in \mathcal{F}, h \in \mathcal{H} \text{ and } s \in \mathcal{S}\}$. Thus each $s_j \in \mathcal{S}$ is a combo of a sensing device and a classifier trained on it such that it outputs a decision $\hat{c}_j \in \mathscr{C}$, $\forall j = 1, ...m$. For the sake of simplicity, we will just call each sensor-classifier combo as a '*sensor*' where it is understood that a classifier is trained and attached to it.

The objective of the fusion problem is to combine the decisions $\{\hat{c}_j \in \mathscr{C}, j = 1, ...m\}$ of m sensors, to produce a unique decision $\hat{c} \in \mathscr{C}$ with improved accuracy. As discussed in the introduction, different sensors might have different accuracies in separating different classes. Thus the objective of the BB-Fuse algorithm is to identify the best *sensor-class* pairs and fuse them by constructing a decision tree, called the BB-Fuse Tree, which yields a higher classification accuracy than the best individual sensor.

3.4.2 BB-Fuse Training Phase

In the training phase, the BB-Fuse algorithm constructs a *one-vs-the-rest decision tree* $\mathcal{T}_{BB}$ using a greedy algorithm that isolates one class at each node from the rest until all classes are isolated at terminal nodes.

BB-Fuse Tree Structure

Since the class set $\mathscr{C} = \{c_1, c_2, \ldots c_n\}$ consists of n classes, the BB-Fuse tree $\mathcal{T}_{BB}$ consists of n levels denoted by $\ell = 1, ...n$, as shown in Fig. 3.3. While level 1 consists of the root node, level n consists of the two bottom most leaf nodes with exit classes. Every other level $2 \leq \ell \leq (n-1)$ consists of two nodes: i) a *terminal*, i.e. a *leaf node*, which contains the single isolated class at the previous level, and ii) a *non-terminal* node, whose input is the class set available after removing all previously isolated classes. Every non-terminal node splits into a leaf node and a non-terminal node at a level below. Also, every non-terminal node contains a sensor $s^\ell \in \mathcal{S}$ which makes the underlying binary classification decision. Fig. 3.3 shows the structure of the BB-Fuse tree consisting of n leaf nodes and $n-1$ non-terminal nodes.

To build the recursive iteration process to construct the tree, lets denote $\bar{c}^0 \triangleq \mathscr{C} = \{c_1, c_2, ...c_n\}$

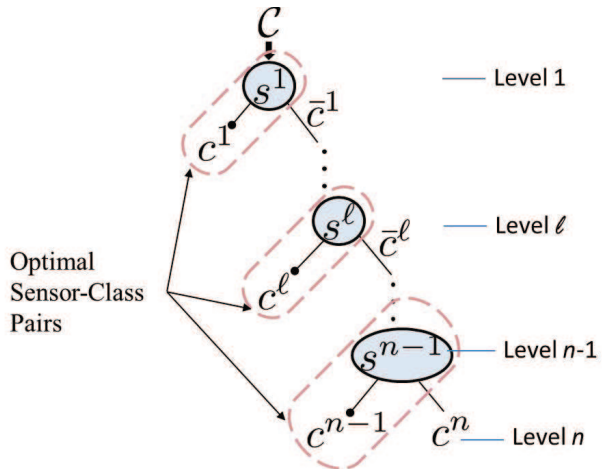


Figure 3.3: The Structure of a BB-Fuse Tree ($\mathcal{T}_{BB}$).

to form the entering class set at the root node of the tree at level 1. The root then splits the class set $\bar{c}^0$ into two subsets at level 2 below, one consisting of a class $c^1 \in \bar{c}^0$ which goes to the leaf node as the isolated class, and the other consisting of the subset $\bar{c}^1 = \{\bar{c}^0 \backslash c^1\}$, which forms the entering class set for the corresponding non-terminal node. Then $\bar{c}^1$ is further split into two subsets and this process is repeated at all levels until all classes are isolated. In general, let $\bar{c}^{\ell-1}\ \forall\ \ell = 1, ...n-1$ be the entering class set at the non-terminal node at level ℓ, such that $|\bar{c}^{\ell-1}| = n - \ell + 1$. Then $\bar{c}^{\ell-1}$ is split into two subsets, one consisting of the class $c^\ell \in \bar{c}^{\ell-1}$, i.e. the isolated class which goes to the leaf node below, and the other consisting of the set $\bar{c}^\ell = \{\bar{c}^{\ell-1} \backslash c^\ell\}$ which forms the entering class set to the following non-terminal node, until it contains only a single class.

Selection of the Optimal Sensor-Class Pairs

Now we describe the procedure to find the optimal *sensor-class pairs* $\{(s^\ell, c^\ell), \ell = 1, ...n-1\}$ to construct the BB-Fuse tree. To do so we first define one-vs-the-rest classification subproblems at every non-terminal node of the tree. Consider the non-terminal node at level ℓ whose entering class set is $\bar{c}^{\ell-1}$ as described above. Further let $x \in \bar{c}^{\ell-1}$ be a dummy class variable. Lets now define its complement set $\bar{x} \triangleq \{\bar{c}^{\ell-1} \backslash x\}$, i.e. the remaining available classes at that node, to generate a partition of the class set $\bar{c}^{\ell-1}$ as $\{x, \bar{x}\}$. Then, a one-vs-the-rest classification subproblem $\mathcal{P}_{(x,\bar{x})}$ is defined, which aims to separate the two meta-classes x and $\bar{x}$. Thus the total number of subproblems at any level ℓ is equal to the size of the entering class set $|\bar{c}^{\ell-1}|$. Given all such

subproblems, the following questions arise:

Q.1 Which subproblem produces the best results, i.e. which class is the best to isolate, at level ℓ?

Q.2 Which sensor is the best to isolate it?

To answer the above questions, we need to first establish our performance measures to select the optimal sensor-class pairs. One good performance measure is the CCR of a classifier, also called classification accuracy, which measures its efficacy in making correct decisions. However, CCR alone may not be enough since a classifier can produce an overall high CCR even though it poorly isolates a certain class. Another measure is sensitivity, which is defined with respect to a certain class as its correct classification probability. Thus, we consider these two performance measures together; however, sometimes it may not be possible to obtain a feasible solution where these two measures are maximized together. This is because the sensor-class pair which has the highest CCR may not have the highest sensitivity and vice versa. Thus the optimization problem is formulated as follows:

• **Optimization Problem:** Let $s^* \in \mathcal{S}$ be the optimal sensor that generates the highest CCR for the full classification problem $\mathcal{P}$. Our aim is to construct a fusion scheme that will achieve a CCR which is higher than that of s^* used alone. Let $\pi(s_j, \mathcal{P}_{(x,\bar{x})})$ denote the CCR generated by sensor $s_j \in \mathcal{S}$ for the classification subproblem $\mathcal{P}_{(x,\bar{x})}$, whose objective is to separate class x from $\bar{x}$. Then, at each level $\ell = 1, ...n-1$ of the tree, we consider the following problem:

$$(s^\ell, c^\ell) = \max_{\forall s_j \in \mathcal{S}, \forall x \in \bar{c}^{\ell-1}} \pi\big(s_j, \mathcal{P}_{(x,\bar{x})}\big),$$

subject to the constraint

$$p(x|s)p(x|x,s) \geq p(x|s^*)p(x|x,s^*), \tag{3.1}$$

which generates the optimal sensor-class pair (s^ℓ, c^ℓ) for each level ℓ. The sensor s^ℓ is placed at the non-terminal node of level ℓ which is used to separate the class c^ℓ which goes to the leaf node at the level below. The term in the inequality is the sensitivity of sensor s, for separating class x

from the rest, weighted by the probability of class x.

We will show later that the BB-Fuse tree with optimal sensor-class pairs selected as above yields a CCR higher than s^*. Here we define the *Sensor-Class Pair Accuracy Matrix* matrix as:

$$\Pi^{\ell} = \left[\pi\left(s_j, \mathcal{P}_{(x,\bar{x})}\right)\right]_{\forall s_j \in \mathcal{S}, \forall x \in \bar{c}^{\ell-1}} ; \ \forall \ell = 1, ...n-1 \tag{3.2}$$

whose rows and columns correspond to sensors and classes, respectively.

• **Solution Procedure:** To solve the optimization problem, $\pi\left(s_j, \mathcal{P}_{(x,\bar{x})}\right)$ are obtained as follows. Let $A^1(s_j) = [a_{i,i'}]_{i,i'=1,2,...n}$, be the confusion matrix resulting from sensor $s_j \in \mathcal{S}$ for the original n-class classification problem $\mathcal{P}$. Here $a_{i,i'}$ represents the number of testing samples that are actually of class $c_i \in \mathscr{C}$ but are classified as $c_{i'} \in \mathscr{C}$; the diagonal elements $a_{i,i}$ represent the correctly classified samples. Then *binary* confusion matrices $B_{(x,\bar{x})}(s_j) = [b_{i,i'}]_{i,i'=1,2}$, $\forall x \in \bar{c}^0$, are derived from $A^1(s_j)$ as follows:

$$\begin{aligned} b_{1,1} &= a_{\alpha,\alpha} \quad , \ b_{1,2} = \sum_{k=1,k\neq\alpha}^{n} a_{\alpha,k} \\ b_{2,1} &= \sum_{r=1,r\neq\alpha}^{n} a_{r,\alpha} \ , \ b_{2,2} = \sum_{r=1,r\neq\alpha}^{n} \sum_{k=1,k\neq\alpha}^{n} a_{r,k} \end{aligned} \tag{3.3}$$

where α is the index of class x and $B_{(x,\bar{x})}(s_j)$ evaluates the performance of sensor s_j in solving the subproblem $\mathcal{P}_{(x,\bar{x})}$. Subsequently, these binary matrices are constructed for all sensors $s_j \in \mathcal{S}$. From these, $\pi\left(s_j, \mathcal{P}_{(x,\bar{x})}\right)$=$\frac{b_{1,1}+b_{2,2}}{\sum_{i,j} b_{i,j}}$ can be computed $\forall s_j \in \mathcal{S}$ and $\forall x \in \bar{c}^0$, to generate the matrix Π^1. Similarly, the sensitivity terms are also computed which are equal to $\frac{b_{1,1}}{\sum_{i,j} b_{i,j}}$. The optimal sensor-class pair (s^1, c^1) is then selected as per the optimization problem in Eq. (3.1). To do so, first the elements of Π^1 are sorted and ranked. Then following down the rank, the sensor-class pair with the highest rank and which satisfies the constraint below is selected. Note that the selected pair will always have higher CCR for the underlying subproblem as well as higher sensitivity than the sensor s^*; in the worst case, s^* will be selected itself.

After the optimal class c^1 is found, the class set is reduced to $\bar{c}^1 = \{\bar{c}^0 \backslash c^1\}$ and the confusion matrices $A^2(s_j)$ are obtained from $A^1(s_j)$, $\forall s_j \in \mathcal{S}$, by deleting the row and column corresponding to class c^1. Then again binary confusion matrices are constructed from $A^2(s_j)$ as described above

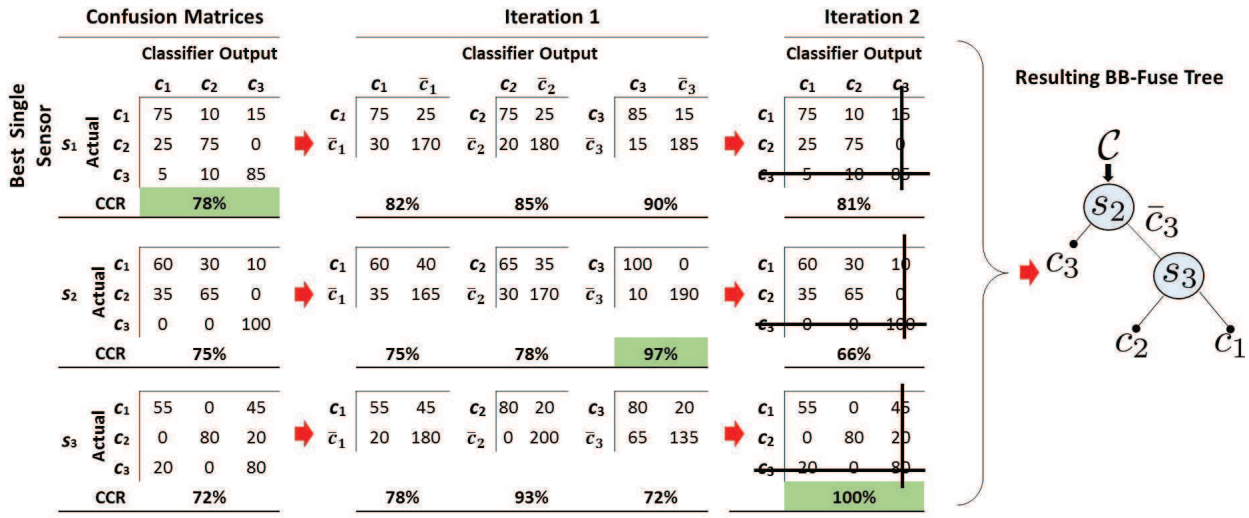


Figure 3.4: An Example of the BB-Fuse Tree Construction

and the optimal sensor-class pair (s^2, c^2) is selected at level 2 and so on. In general, $A^\ell(s_j)$ is the resulting $(n-\ell+1)\times(n-\ell+1)$ confusion matrix of sensor s_j at level ℓ of the tree which is formed by deleting the rows and columns corresponding to all classes that have been separated so far. The above process is repeated at all levels to find all the optimal sensor-class pairs of the BB-Fuse tree. Finally, in order to reduce the size of the tree, any parent and child nodes which contain the same optimal sensor, are merged together as described in Definition 3.3.5.

Example 3.4.1 *Figure 3.4 shows an example of the BB-Fuse tree construction where the input class set is* $\mathscr{C} = \{c_1, c_2, c_3\}$. *The confusion matrices for* 3 *sensors* s_1, s_2 *and* s_3 *are shown in the left column. As seen, the highest CCR of* 78% *is obtained by sensor* s_1. *The first step is to calculate the binary confusion matrices for isolating one class from the rest, for each of the three sensors, as shown in Iteration 1. Here, class* c_3 *is isolated best using sensor* s_2 *with a CCR of* 97% . *Thus, the optimal sensor-class pair at the root node is* (s^1, c^1)=(s_2, c_3). *Next, the class set is reduced to* $\bar{c}^1 = \{c_1, c_2\}$, *and the third rows and columns are deleted from the confusion matrices, as illustrated in Iteration 2. Again, it is seen that* s_3 *separates the classes* c_1 *and* c_2 *with the highest CCR of* 100%. *Thus, the optimal sensor-class pair* (s^2, c^2)=(s_3, c_2) *is selected and the BB-Fuse tree*

is constructed. The CCR of the BB-Fuse tree is computed as

$$\begin{aligned}\pi(\mathcal{T}_{BB},\mathcal{P}) &= p(c_3)p(c_3|c_3,s_2)+p(\bar{c}_3)p(\bar{c}_3|\bar{c}_3,s_2)\Big[p(c_2)p(c_2|c_2,s_3)+p(c_1)p(c_1|c_1,s_3)\Big] \\ &= \frac{100}{300}\cdot\frac{100}{100}+\frac{200}{300}\cdot\frac{190}{200}\Big[\frac{80}{135}\cdot\frac{80}{80}+\frac{55}{135}\cdot\frac{55}{55}\Big]=96\%\end{aligned}$$

which is higher than the CCR of any single sensor.

3.4.3 BB-Fuse Testing Phase

In the testing phase, unlabeled time series data are collected only from the optimal sensors $s^\ell;\ \ell = 1,\dots n-1$, found in the training phase. Then, the execution moves down the BB-Fuse tree starting from the root node by using the optimal sensor-class pairs embedded in the tree. For example, at level ℓ sensor s^ℓ is used to obtain a decision $\hat{c}^\ell \in \mathscr{C}$. If $\hat{c}^\ell$ is equal to the a priori determined optimal class c^ℓ, then the algorithm terminates at the corresponding leaf node, where it outputs the classification decision as $\hat{c} = c^\ell$; otherwise, the algorithm moves down the tree. The above process is repeated until a terminal node is reached.

3.5 BB-Fuse Performance

This section discusses the performance of the BB-Fuse algorithm from two perspectives; namely, the optimality and the computational complexity.

3.5.1 BB-Fuse Optimality

First we show that the node merging operation as per Definition 3.3.5 does not affect the overall CCR of the tree. We will need the following lemma.

Lemma 1 *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two decision trees such that they both have the same input class A and the same exit class B at the last terminal node. Also, $\mathcal{T}_1$ and $\mathcal{T}_2$ yield the same CCR for the classification problem $\mathcal{P}$ and the same probability of correct detection for class B. Furthermore, let $\mathcal{T}_\alpha$ be a decision tree whose last exit class is A and let $\mathcal{T}_\beta$ be a decision tree whose input class*

is B. Then,

- $\pi(\mathcal{T}_\alpha \oplus \mathcal{T}_1, \mathcal{P}') = \pi(\mathcal{T}_\alpha \oplus \mathcal{T}_2, \mathcal{P}')$
- $\pi(\mathcal{T}_1 \oplus \mathcal{T}_\beta, \mathcal{P}'') = \pi(\mathcal{T}_2 \oplus \mathcal{T}_\beta, \mathcal{P}'')$
- $\pi(\mathcal{T}_\alpha \oplus \mathcal{T}_1 \oplus \mathcal{T}_\beta, \mathcal{P}''') = \pi(\mathcal{T}_\alpha \oplus \mathcal{T}_2 \oplus \mathcal{T}_\beta, \mathcal{P}''')$

where $\mathcal{P}'$, $\mathcal{P}''$ and $\mathcal{P}'''$ represent the corresponding classification problems for the above grafting operations; and $\pi(\diamond, \circ)$ denote the CCR obtained from the tree $\diamond$ for the classification problem $\circ$.

Proof: The proof is straightforward and results from graphical visualization of the grafting operations. □

Theorem 3.5.1 *Let $\mathcal{T}$ be a one-vs-the-rest decision tree such that two of its non-terminal nodes, say u and v, where u is the parent of v, carry the same sensor $s \in \mathcal{S}$ for classification. Let $\mathcal{T}'$ be the tree obtained by merging these two nodes to form a node $w = u \prec v$ which carries the same sensor s. Then,*

$$\pi(\mathcal{T}, \mathcal{P}) = \pi(\mathcal{T}', \mathcal{P})$$

where $\pi(\mathcal{T}, \mathcal{P})$ and $\pi(\mathcal{T}', \mathcal{P})$ denote the CCRs obtained from the trees $\mathcal{T}$ and $\mathcal{T}'$ for the classification problem $\mathcal{P}$, respectively.

Proof: Let $\mathcal{T}_{uv}$ be the subtree consisting of the non-terminal nodes u and v, such that the class x is separated at node u, while classes y and z are separated at node v. Let $\mathcal{T}_\alpha$ and $\mathcal{T}_\beta$ be the subtrees grafted above the node u and below the node v respectively, to form the tree $\mathcal{T}$, such that the class set $\{x, y, z\}$ exits from T_α and forms the input class set for $\mathcal{T}_{uv}$, while the class z exists from $\mathcal{T}_{uv}$ and forms the input class set for T_β. Thus, $\mathcal{T}$ is decomposed as follows

$$\mathcal{T} = \mathcal{T}_\alpha \oplus \mathcal{T}_{uv} \oplus \mathcal{T}_\beta. \tag{3.4}$$

Since u and v contain the same sensor s, they can be merged as per Definition 3.3.5, to form a node $w = u \prec v$. Let $\mathcal{T}_w$ be the tree consisting of the single non-terminal node w, which separates

the three classes x, y and z. As above, lets graft $\mathcal{T}_\alpha$ and $\mathcal{T}_\beta$ above and below $\mathcal{T}_w$ respectively, to form the tree $\mathcal{T}'$, such that

$$\mathcal{T}' = \mathcal{T}_\alpha \oplus \mathcal{T}_w \oplus \mathcal{T}_\beta. \tag{3.5}$$

Now let the confusion matrix of sensor s to separate the classes x, y and z be represented as

$$\mathcal{A}^1(s) = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

The CCR of $\mathcal{T}_{uv}$ is obtained as

$$\pi(\mathcal{T}_{uv}, \mathcal{P}_{(x,y,z)}) = p^1(x|s)p^1(x|x,s) + p^1(\bar{x}|s)p^1(\bar{x}|\bar{x},s) \cdot \Big\{ p^2(y|s)p^2(y|y,s) + p^2(z|s)p^2(z|z,s) \Big\} \tag{3.6}$$

where $\bar{x} = \{y, z\}$ and $\pi(\mathcal{T}_{uv}, \mathcal{P}_{(x,y,z)})$ is the CCR of the tree $\mathcal{T}_{uv}$ for the classification subproblem $\mathcal{P}_{(x,y,z)}$ which separates classes x, y and z. The quantities $p^1(x|s)$, $p^1(\bar{x}|s)$, $p^1(x|x,s)$, and $p^1(\bar{x}|\bar{x},s)$ are computed from $\mathcal{A}^1(s)$ as

$$p^1(x|s) = \frac{\sum_{j=1}^{3} a_{1,j}}{\sum_{i=1}^{3}\sum_{j=1}^{3} a_{i,j}}, \quad p^1(\bar{x}|s) = \frac{\sum_{i=2}^{3}\sum_{j=1}^{3} a_{i,j}}{\sum_{i=1}^{3}\sum_{j=1}^{3} a_{i,j}}, \quad p^1(x|x,s) = \frac{a_{1,1}}{\sum_{j=1}^{3} a_{1,j}}, \quad p^1(\bar{x}|\bar{x},s) = \frac{\sum_{i=2}^{3}\sum_{j=2}^{3} a_{i,j}}{\sum_{i=2}^{3}\sum_{j=1}^{3} a_{i,j}}.$$

At level 2 of the tree $\mathcal{T}_{uv}$, the confusion matrix $\mathcal{A}^1(s)$ is updated to $\mathcal{A}^2(s)$ by deleting the row and column corresponding to the class x. Thus,

$$\mathcal{A}^2(s) = \begin{bmatrix} a_{2,2} & a_{2,3} \\ a_{3,2} & a_{3,3} \end{bmatrix}$$

from where the quantities $p^2(y|s)$, $p^2(z|s)$, $p^2(y|y,s)$, and $p^2(z|z,s)$ are computed as

$$p^2(y|s) = \frac{a_{2,2}+a_{2,3}}{\sum_{i=2}^{3}\sum_{j=2}^{3} a_{i,j}}, \quad p^2(z|s) = \frac{a_{3,2}+a_{3,3}}{\sum_{i=2}^{3}\sum_{j=2}^{3} a_{i,j}}, \quad p^2(y|y,s) = \frac{a_{2,2}}{a_{2,2}+a_{2,3}}, \quad p^2(z|z,s) = \frac{a_{3,3}}{a_{3,2}+a_{3,3}}.$$

By substituting all the above values in Eq. (3.6), we get

$$\begin{aligned}\pi(\mathcal{T}_{uv}, \mathcal{P}_{(x,y,z)}) =& \frac{a_{1,1}}{\sum_{i=1}^{3}\sum_{j=1}^{3} a_{i,j}} + \frac{\sum_{i=2}^{3}\sum_{j=2}^{3} a_{i,j}}{\sum_{i=1}^{3}\sum_{j=1}^{3} a_{i,j}} \left\{ \frac{a_{2,2}+a_{3,3}}{\sum_{i=2}^{3}\sum_{j=2}^{3} a_{i,j}} \right\} \\ =& \frac{a_{1,1}+a_{2,2}+a_{3,3}}{\sum_{i=1}^{3}\sum_{j=1}^{3} a_{i,j}} \\ =& \pi(\mathcal{T}_w, \mathcal{P}_{(x,y,z)}). \end{aligned} \tag{3.7}$$

From Eqs. (3.4), (3.5) and (3.7), and using Lemma 1, we obtain $\pi(\mathcal{T}, \mathcal{P}) = \pi(\mathcal{T}', \mathcal{P})$. Thus, the node merging operation does not affect the CCR. Hence Proved. □

Next we present the following theorem which will be needed to prove the main result of this chapter.

Theorem 3.5.2 *Let $\mathcal{T}$ be a one-vs-the-rest decision tree such that all its non-terminal nodes contain the same sensor $s \in \mathcal{S}$. Then*

$$\pi(\mathcal{T}, \mathcal{P}) = \pi(s, \mathcal{P})$$

Proof: First consider a subtree $\mathcal{T}_1$ which has only one non-terminal node containing sensor $s \in \mathcal{S}$ to separate classes $c^1 \in \mathscr{C}$ and $\bar{c}^1 \subset \mathscr{C}$. Let the CCR of this subtree be denoted as $\pi(\mathcal{T}_1, \mathcal{P}_{(c^1,\bar{c}^1)})$. Trivially,

$$\pi(\mathcal{T}_1, \mathcal{P}_{(c^1,\bar{c}^1)}) = \pi(s, \mathcal{P}_{(c^1,\bar{c}^1)}) \tag{3.8}$$

Now graft a subtree $\mathcal{T}_2$ below $\mathcal{T}_1$, where $\mathcal{T}_2$ consists of a single node containing sensor s and which separates classes $c^2 \in \bar{c}^1$ and $\bar{c}^2 \subset \bar{c}^1$, such that $\bar{c}^1$ forms the entering class set of $\mathcal{T}_2$, as shown in Fig. 3.5 with $s^1 = s^2 = s$.

The resulting subtree is denoted as $\mathcal{T}_1 \oplus \mathcal{T}_2$ which separates class c^1 at level 1, c^2 at level 2,

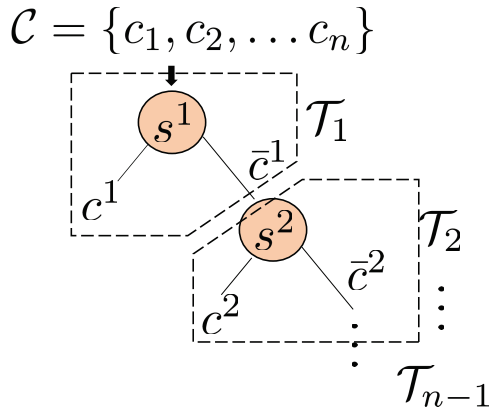


Figure 3.5: The BB-Fuse Tree

and $\bar{c}^2$ is the remaining class set. The CCR of $\mathcal{T}_1 \oplus \mathcal{T}_2$ for the subproblem $\mathcal{P}_{(c^1,c^2,\bar{c}^2)}$ is denoted as $\pi(\mathcal{T}_1 \oplus \mathcal{T}_2, \mathcal{P}_{(c^1,c^2,\bar{c}^2)})$. Using Theorem 3.5.1,

$$\pi(\mathcal{T}_1 \oplus \mathcal{T}_2, \mathcal{P}_{(c^1,c^2,\bar{c}^2)}) = \pi(s, \mathcal{P}_{(c^1,c^2,\bar{c}^2)}) \tag{3.9}$$

Now we continue the process of grafting subtrees below and prove the result by induction. Consider a tree which is composed by grafting $k-1$ subtrees $\mathcal{T}_1$, $\mathcal{T}_2$, ...,$\mathcal{T}_{k-1}$, where each $\mathcal{T}_j,\ j = 1, ...k-1$ consists of a single non-terminal node containing sensor s, which solves the subproblem $\mathcal{P}_{(c^j,\bar{c}^j)}$. Thus, the resulting subtree is denoted as $\mathcal{T}_1 \oplus ...\mathcal{T}_{k-1}$ which solves the subproblem $\mathcal{P}_{(c^1,...c^{k-1},\bar{c}^{k-1})}$. We assume that the following is true

$$\pi(\mathcal{T}_1 \oplus ...\mathcal{T}_{k-1}, \mathcal{P}_{(c^1,...c^{k-1},\bar{c}^{k-1})}) = \pi(s, \mathcal{P}_{(c^1,...c^{k-1},\bar{c}^{k-1})}) \tag{3.10}$$

and then show that

$$\pi(\mathcal{T}_1 \oplus ...\mathcal{T}_k, \mathcal{P}_{(c^1,...c^k,\bar{c}^k)}) = \pi(s, \mathcal{P}_{(c^1,...c^k,\bar{c}^k)}) \tag{3.11}$$

From Eq. (3.10) it is observed that we can replace the subtree $\mathcal{T}_1 \oplus ...\mathcal{T}_{k-1}$ with a tree $\mathcal{T}'$ with a single node containing the sensor s, which separates the classes $c^1, ...c^{k-1}, \bar{c}^{k-1}$ to get the same CCR. Now graft the subtree $\mathcal{T}_k$ below the subtrees $\mathcal{T}_1 \oplus ...\mathcal{T}_{k-1}$ and $\mathcal{T}'$, such that $\bar{c}^{k-1}$ forms the entering class set for $\mathcal{T}_k$, which results in subtrees $\mathcal{T}_1 \oplus ...\mathcal{T}_k$ and $\mathcal{T}' \oplus \mathcal{T}_k$ respectively. Then using

Lemma 1 we get

$$\pi(\mathcal{T}_1 \oplus ...\mathcal{T}_k, \mathcal{P}_{(c^1,...c^k,\bar{c}^k)}) = \pi(\mathcal{T}' \oplus \mathcal{T}_k, \mathcal{P}_{(c^1,...c^k,\bar{c}^k)}) \tag{3.12}$$

where $\mathcal{T}' \oplus \mathcal{T}_k$ is a subtree with two non-terminal nodes, which separate class set $\{c^1, ...c^{k-1}\}$ at level 1 and classes $c^k, \bar{c}^k$ at level 2. Since these two nodes contain the same sensor s, they can be merged and following the procedure similar to that of Theorem 3.5.1, it can be shown that

$$\pi(\mathcal{T}' \oplus \mathcal{T}_k, \mathcal{P}_{(c^1,...c^k,\bar{c}^k)}) = \pi(s, \mathcal{P}_{(c^1,...c^k,\bar{c}^k)}) \tag{3.13}$$

Thus Eq. (3.11) is true for arbitrary k. Since $\mathcal{T} = \mathcal{T}_1 \oplus ...\mathcal{T}_{n-1}$ and $\mathcal{P} \triangleq \mathcal{P}_{(c^1,...c^{n-1},c^n)}$, the result follows. Hence Proved. □

Corollary 1 *Let $\mathcal{T}$ and $\mathcal{T}'$ be two decision trees where each of them address the same classification problem $\mathcal{P}$ and all their non-terminal nodes contain the same sensor $s \in \mathcal{S}$, however they separate the classes in a different order. Then $\pi(\mathcal{T}, \mathcal{P}) = \pi(\mathcal{T}', \mathcal{P})$.*

Proof: Using Theorem 3.5.2, $\pi(\mathcal{T}, \mathcal{P}) = \pi(s, \mathcal{P})$. Also $\pi(\mathcal{T}', \mathcal{P}) = \pi(s, \mathcal{P})$. Hence proved. □

Now we show the main result which states that the CCR of the BB-Fuse tree is better than the CCR obtained by the best sensor. We will need the following lemma.

Lemma 2 *If the numbers a_i, b_i, c_i and $d_i \in [0, 1]$, such that $a_i + b_i \leq 1$, $c_i + d_i \leq 1$, $a_i \geq c_i$ and $a_i + b_i \geq c_i + d_i$, $\forall i = 1, 2, ...n-1$, then the following are true:*

i. $a_i + b_i\lambda_i \geq c_i + d_i\mu_i$, *where* $0 \leq \mu_i \leq \lambda_i \leq 1, \forall i$, *and*

ii. $a_1 + b_1(a_2 + b_2(...(a_{n-1} + b_{n-1}))) \geq c_1 + d_1(c_2 + d_2(...(c_{n-1} + d_{n-1})))$

Proof: i) There are two possible cases as follows.

• $b_i \geq d_i$: Since $a_i \geq c_i$ and $\lambda_i \geq \mu_i$, this implies that

$$a_i + b_i\lambda_i \geq c_i + d_i\mu_i \tag{3.14}$$

• $b_i < d_i$: Since $a_i + b_i \geq c_i + d_i$

$$\Rightarrow \quad a_i - c_i \geq d_i - b_i \tag{3.15}$$

$$\text{Now} \quad 1 - \mu_i \geq 1 - \lambda_i \Rightarrow \quad d_i(1 - \mu_i) \geq b_i(1 - \lambda_i)$$

$$\Rightarrow \quad d_i - b_i \geq d_i\mu_i - b_i\lambda_i \tag{3.16}$$

From (3.15) and (3.16) we get $a_i - c_i \geq d_i\mu_i - b_i\lambda_i$. This implies that $a_i + b_i\lambda_i \geq c_i + d_i\mu_i$.

ii) Lets start from the innermost term. Choose $\lambda_{n-1} = \mu_{n-1} = 1$. Since $a_{n-1} + b_{n-1} \geq c_{n-1} + d_{n-1}$,

$$\underbrace{a_{n-1} + b_{n-1}\lambda_{n-1}}_{\lambda_{n-2}} \geq \underbrace{c_{n-1} + d_{n-1}\mu_{n-1}}_{\mu_{n-2}}$$

Since $a_{n-2} + b_{n-2} \geq c_{n-2} + d_{n-2}$, part i) implies that

$$\underbrace{a_{n-2} + b_{n-2}\lambda_{n-2}}_{\lambda_{n-3}} \geq \underbrace{c_{n-2} + d_{n-2}\mu_{n-2}}_{\mu_{n-3}}$$

Repeating the above procedure we get

$$\underbrace{a_{n-k+1} + b_{n-k+1}\lambda_{n-k+1}}_{\lambda_{n-k}} \geq \underbrace{c_{n-k+1} + d_{n-k+1}\mu_{n-k+1}}_{\mu_{n-k}}$$

where $\lambda_{n-k} \triangleq a_{n-k+1} + b_{n-k+1}\lambda_{n-k+1}$ and $\mu_{n-k} \triangleq c_{n-k+1} + d_{n-k+1}\mu_{n-k+1}$, $\forall k = 2, ...n-1$. Finally,

$$a_1 + b_1\lambda_1 \geq c_1 + d_1\mu_1 \tag{3.17}$$

Now plugging the values of λs and μs, we get

$$a_1 + b_1(a_2 + b_2(...(a_{n-1} + b_{n-1}))) \geq c_1 + d_1(c_2 + d_2(...(c_{n-1} + d_{n-1})))$$

Hence Proved. □

Theorem 3.5.3 *Let $\mathcal{T}_{BB}$ be the BB-Fuse Tree with optimal sensor-class pairs $\{(s^\ell, c^\ell),\ \forall \ell = 1, \ldots n-1\}$. Then*

$$\pi(\mathcal{T}_{BB}, \mathcal{P}) \geq \pi(s^*, \mathcal{P})$$

where s^* is the sensor that achieves the highest CCR for the classification problem $\mathcal{P}$ when used alone.

Proof: The BB-Fuse tree $\mathcal{T}_{BB}$ consists of $n-1$ non-terminal nodes to separate n classes. Thus $\mathcal{T}_{BB}$ is constructed by grafting $n-1$ subtrees $\mathcal{T}_1$, $\mathcal{T}_2$, ...,$\mathcal{T}_{n-1}$, where each $\mathcal{T}_j,\ j = 1, ...n-1$ consists of a single non-terminal node containing sensor s^j, which solves the subproblem $\mathcal{P}_{(c^j, \bar{c}^j)}$, as shown in Fig. 3.5. Therefore,

$$\mathcal{T}_{BB} = \mathcal{T}_1 \oplus \mathcal{T}_2 \oplus ...\mathcal{T}_{n-1}. \tag{3.18}$$

where

$$\begin{aligned}\pi(\mathcal{T}_{BB}, \mathcal{P}) \triangleq & \pi(\mathcal{T}_1 \oplus ...\mathcal{T}_{n-1}, \mathcal{P}_{(c^1, ...c^{n-1}, c^n)}) \\ = & p^1(c^1|s^1)p^1(c^1|c^1, s^1) + \ p^1(\bar{c}^1|s^1)p^1(\bar{c}^1|\bar{c}^1, s^1)\Big[p^2(c^2|s^2)p^2(c^2|c^2, s^2) + p^2(\bar{c}^2|s^2)p^2(\bar{c}^2|\bar{c}^2, s^2) \\ & \Big[\ldots\Big[p^{n-1}(c^{n-1}|s^{n-1})p^{n-1}(c^{n-1}|c^{n-1}, s^{n-1}) + p^{n-1}(c^n|s^{n-1})p^{n-1}(c^n|c^n, s^{n-1})\Big]\ldots\Big]\Big]\end{aligned} \tag{3.19}$$

Similarly, consider another tree $\mathcal{T}'$ which is constructed by grafting $n-1$ subtrees $\mathcal{T}'_1$, $\mathcal{T}'_2$, ...,$\mathcal{T}'_{n-1}$, where each $\mathcal{T}'_j,\ j = 1, ...n-1$ consists of a single non-terminal node containing sensor s^*, which solves the subproblem $\mathcal{P}_{(c^j, \bar{c}^j)}$. Therefore, we get

$$\mathcal{T}' = \mathcal{T}'_1 \oplus \mathcal{T}'_2 \oplus ...\mathcal{T}'_{n-1}. \tag{3.20}$$

where

$$\begin{aligned}\pi(\mathcal{T}',\mathcal{P}) \triangleq & \pi(\mathcal{T}'_1 \oplus ...\mathcal{T}'_{n-1}, \mathcal{P}_{(c^1,...c^{n-1},c^n)}) \\ =& p^1(c^1|s^*)p^1(c^1|c^1,s^*) + \ p^1(\bar{c}^1|s^*)p^1(\bar{c}^1|\bar{c}^1,s^*)\Big[p^2(c^2|s^*)p^2(c^2|c^2,s^*) + p^2(\bar{c}^2|s^*)p^2(\bar{c}^2|\bar{c}^2,s^*) \\ & \Big[\dots\Big[p^{n-1}(c^{n-1}|s^*)p^{n-1}(c^{n-1}|c^{n-1},s^*) + p^{n-1}(c^n|s^*)p^{n-1}(c^n|c^n,s^*)\Big]\dots\Big]\Big] \end{aligned} \tag{3.21}$$

Using Theorem 3.5.2, we get $\pi(\mathcal{T}',\mathcal{P}) = \pi(s^*,\mathcal{P})$. Now, lets define the following quantities:

$$a_i = p^i(c^i|s^i)p^i(c^i|c^i,s^i),\ \ b_i = p^i(\bar{c}^i|s^i)p^i(\bar{c}^i|\bar{c}^i,s^i),\ \ c_i = p^i(c^i|s^*)p^i(c^i|c^i,s^*),\ \ d_i = p^i(\bar{c}^i|s^*)p^i(\bar{c}^i|\bar{c}^i,s^*)$$

We know that $a_i + b_i \leq 1$ and $c_i + d_i \leq 1$ and from the optimality conditions in Eq. (3.1), we get $a_i \geq c_i$ and $a_i + b_i \geq c_i + d_i$, $\forall i = 1, ...n-1$. Thus, applying Lemma 2 we get the result

$$\pi(\mathcal{T}_{BB},\mathcal{P}) \geq \pi(s^*,\mathcal{P}). \tag{3.22}$$

Hence Proved. □

3.5.2 BB-Fuse Complexity

Generally, brute force searching for the optimal decision tree (i.e., running full cross-validation over each tree) is computationally very expensive because the number of possible decision trees could be very large. To illustrate, the following theorem enumerates the total number of one-vs-the rest trees.

Theorem 3.5.4 *The number of one-vs-the-rest decision trees for separating n classes using m sensors is $m^{n-1}n!/2$.*

Proof: In one-vs-the-rest decision trees one class is isolated at each non-terminal node using one of the m sensors. Thus such trees should have $n-1$ non-terminal nodes to separate n classes. At every level we have m sensors to choose from but the class set keeps reducing as we isolate them. In particular, at level 1 there are n classes, at level 2 there are $n-1$ classes and so on, until at level

$n-2$, there are 3 classes to choose from. However, at level $n-1$, only 2 classes are left, whose order does not matter. Thus the total number of trees are $nm \cdot (n-1)m \cdots 3m \cdot 1m = m^{n-1}n!/2$.

□

Remark: Theorem 3.5.4 does not account for the node merging operation which reduces the number of trees.

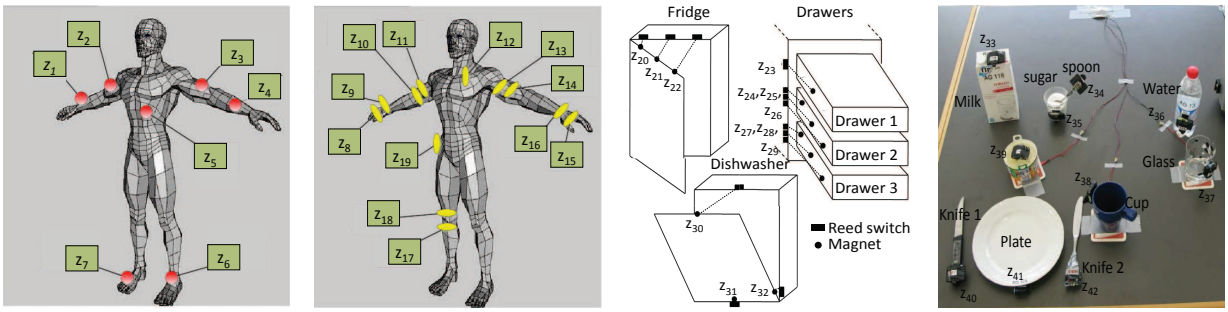
Furthermore, the number of binary and non-binary trees is even higher than the number one-vs-the-rest trees. Therefore, brute force searching for the optimal tree could become very tedious as the number of classes and sensors increase. In contrast, this paper is using a simple algorithm that builds the BB-Fuse Tree node by node by selecting optimal sensor-class pairs using binary confusion matrices, as described in Section 3.4. These binary matrices are easy to compute and the algorithm has fairly low computational complexity with guarantees of a CCR higher than the best sensor. Numerical values of computational times for the BB-Fuse tree construction are shown in the Results section for different applications.

3.6 Results and Discussion

This section presents the results of testing and validation of the BB-Fuse algorithm on experimental data sets of two different applications. These applications are discussed below.

3.6.1 Application I: Human Activity Recognition

In this application, the Opportunity data set [116, 117] for human activity recognition was used to classify basic hand movements in a kitchen environment. In the kitchen, there were 2 doors, 3 drawers, a fridge, a dishwasher, a lazy chair, and a variety of kitchen items which include a cup, salami slices, a water bottle, cheese, a bread loaf, 2 knives, a milk carton, a spoon, sugar, a plate and a glass. Four subjects were asked to perform 20 repetitions of daily activities in the kitchen such as opening and closing of the fridge, the dishwasher, the drawers, and the doors; locking and unlocking the doors; turning the lights on and off; cleaning the table; and drinking while standing and sitting; and others. The objective is to classify the hand-environment interaction into one of 10



(a) (b) (c) (d)

Figure 3.6: Placement of (a) Inertial Measurement Units (IMUs), (b) Subject Acceleration Devices, (c) Reed Sensors and (d) Object Acceleration Devices in Application I Dataset [116, 117]

.

classes: unlocking (c_1), locking (c_2), closing (c_3), reaching (c_4), opening (c_5), sipping (c_6), cleaning (c_7), releasing (c_8), moving (c_9), and none of the above (c_{10}).

Time series data were collected using a variety of sensors that were placed at different places in the kitchen. There were a total of 56 different sensing devices and 42 out of them are shown in Fig. 3.6. Table 3.1 shows the details of various sensing devices and their locations. The data was collected at a sampling rate of 30.3 Hz and the total length of the experiment was ~ 2 hours. Further details can be found in [116, 117]. For feature extraction, a section of 1000 seconds of time series data (i.e. 250 seconds from each subject covering the various activities) was used. This data was recorded for each sensing device z_j and arranged into a data matrix $O^j \in \mathcal{R}^{L \times m_j}$, where $T = 30300$ is the number of samples and m_j is the number of sensing modalities. For example, each acceleration device has three modalities for 3 dimensions, which were arranged into the 3 columns of the data matrix. The data from reed switches were arranged in a similar fashion by putting them along different columns of the data matrix. The data matrix of each sensing device was then reduced by applying two different dimension reduction or feature extraction techniques, namely, *Principal Component Analysis* (PCA) and *Linear Discriminant Analysis* (LDA) (see Appendices B.1 and B.2 for details). Specifically, features vectors $F^j \in \mathcal{R}^{T \times d_j}$ and $G^j \in \mathcal{R}^{T \times q_j}$ were extracted from PCA and LDA; respectively, where $d_j \leq m_j$ and $q_j \leq m_j$, which varied from a sensing device to another. In the training phase the labels of these features are known and marked.

Table 3.1: Description of Sensing Devices, Modalities, and Locations for Application I Dataset

Sensing Devices	Types	Modalities	Locations
$z_1 - z_7$	Inertial Measurement Unit (IMU)	• 3D linear acceleration • 3D angular orientation • 3D magnetic field	Subject body
$z_8 - z_{19}$	Acceleration Device	3D linear acceleration	Subject body
$z_{20} - z_{32}$	Reed Sensor	0 (open) or 1 (closed)	Fridge, drawer and dishwasher
$z_{33} - z_{44}$	Acceleration & Gyroscope Device	• linear acceleration • 2D angular location	Kitchen items
$z_{45} - z_{52}$	Acceleration Device	3D linear acceleration	Fridge, 2 doors, dishwasher, 3 drawers and lazy chair
$z_{53} - z_{56}$	Localization Device	3D location	Subject and Ceiling of the room

After constructing the feature space for each sensing device, the k-*Nearest Neighbor* (k-NN) classifier was trained on the features F^j and G^j separately, to find the optimal k for each sensor for each feature space, and the confusion matrices were generated using the standard 10-fold cross validation process. Then these sensor-classifier combos (i.e. the sensing device plus the trained classifier which is referred to as 'sensors' in this paper for simplicity) were stored and used for further analysis. It was found that a localization device located on the subject, with LDA as feature extractor and the k-NN classifier with $k = 3$, yielded the highest CCR of 93% when used alone. The confusion matrix of this best individual sensor is shown in Table 3.2.

Next, the BB-Fuse tree is constructed given the confusion matrices of all sensors by computing the optimal sensor-class pairs as described in Section 3.4. The resultant BB-Fuse Tree is shown Fig. 3.7. The description of the optimal sensors for the BB Fuse tree is provided in Table 3.3. The CCR obtained by the BB-Fuse tree was 96%, which is a 3% improvement over the best sensor. The confusion matrix for the BB-Fuse tree is also shown in Table 3.2. Finally, the performance of the BB-Fuse tree is compared with several other methods; namely, CaRT, C4.5, Bayes Belief

Table 3.2: Application I: Confusion Matrices for Different Methods

BB-Fuse

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	399	1	0	0	0	0	0	0	0	0
c_2	3	855	2	0	19	0	0	5	0	16
c_3	5	16	2628	2	34	3	1	41	2	68
c_4	3	0	1	854	2	0	0	2	25	13
c_5	0	14	18	2	3037	2	12	8	2	105
c_6	9	7	0	15	2	4411	9	0	39	8
c_7	0	4	1	3	7	8	995	0	30	52
c_8	6	3	29	3	19	10	1	2455	2	72
c_9	13	5	0	1	0	61	18	0	1983	19
c_{10}	4	18	62	17	76	33	67	61	38	11424

Best Sensor

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	399	1	0	0	0	0	0	0	0	0
c_2	3	854	2	0	19	0	0	5	0	17
c_3	4	17	2626	3	32	4	2	45	10	57
c_4	3	0	0	853	1	20	0	7	2	14
c_5	0	15	23	2	3023	17	8	6	7	99
c_6	9	7	6	20	11	3942	82	35	170	218
c_7	0	4	1	3	5	45	982	0	28	32
c_8	6	6	25	8	18	35	0	2419	13	70
c_9	10	6	6	3	12	166	21	11	1775	90
c_{10}	4	18	59	13	67	191	39	59	46	11304

CaRT

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	388	2	3	0	1	0	1	1	0	4
c_2	5	843	10	1	7	1	0	5	1	27
c_3	5	10	2609	9	44	2	3	37	4	77
c_4	0	0	6	842	11	1	0	12	0	28
c_5	4	8	46	3	2993	5	5	20	3	113
c_6	2	0	0	2	1	4401	8	3	63	20
c_7	0	0	1	0	3	5	1037	5	9	40
c_8	3	4	51	7	25	8	3	2423	3	73
c_9	0	0	2	0	0	67	10	7	1998	16
c_{10}	11	12	77	32	98	33	41	77	22	11397

C4.5

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	374	5	4	0	2	0	0	1	0	14
c_2	4	847	23	1	3	0	1	10	0	11
c_3	2	19	2605	7	55	0	3	28	0	81
c_4	0	3	3	848	13	0	0	2	0	31
c_5	0	8	48	12	2990	1	3	20	1	117
c_6	0	0	0	0	1	4416	4	4	45	30
c_7	0	0	1	0	5	6	1046	2	6	34
c_8	1	5	21	2	21	4	7	2462	4	73
c_9	2	0	1	0	0	133	6	2	1932	24
c_{10}	6	17	87	17	128	32	26	54	19	11414

Bayes Belief Integration

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	254	92	25	0	10	0	0	0	0	19
c_2	9	826	37	0	7	0	0	14	0	7
c_3	0	7	2749	1	6	0	0	14	0	23
c_4	0	0	46	720	51	0	0	57	0	26
c_5	0	2	58	3	3073	0	0	25	0	39
c_6	0	0	0	0	2	4419	11	31	25	12
c_7	0	0	1	0	0	2	977	0	7	113
c_8	0	1	22	4	9	1	0	2556	0	7
c_9	0	0	0	0	0	57	71	2	1925	45
c_{10}	0	9	9	3	10	2	40	11	0	11716

Majority Voting

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	265	0	0	0	1	0	0	0	0	134
c_2	0	584	0	0	0	0	0	1	0	315
c_3	0	3	1715	0	2	1	0	5	0	1074
c_4	0	0	2	440	2	0	0	10	0	446
c_5	0	0	1	0	2194	0	0	11	0	994
c_6	0	0	0	0	0	4387	0	0	4	109
c_7	0	0	0	0	0	18	587	0	1	494
c_8	0	0	0	0	7	3	0	1871	0	719
c_9	0	0	0	0	0	108	0	0	1791	201
c_{10}	0	0	0	0	0	2	0	0	0	11798

Adaboost

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	368	1	1	0	4	0	0	0	0	26
c_2	1	816	7	0	3	1	0	18	0	54
c_3	0	15	2365	5	15	4	0	39	1	356
c_4	0	0	13	712	7	1	0	21	0	146
c_5	1	3	10	2	2798	2	0	25	1	358
c_6	0	0	0	0	0	4456	0	0	10	34
c_7	0	0	2	0	3	3	931	0	4	157
c_8	0	2	19	7	21	8	0	2381	0	162
c_9	0	0	0	0	1	77	17	0	1953	52
c_{10}	0	2	9	0	1	4	21	6	0	11757

Borda Count Voting

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	76	0	0	0	0	0	0	0	0	324
c_2	0	148	0	0	0	0	0	1	0	751
c_3	0	1	629	0	4	7	0	2	0	2157
c_4	0	0	1	51	3	6	0	10	2	827
c_5	0	0	0	0	1553	2	0	1	0	1644
c_6	0	0	0	0	0	4405	0	0	4	91
c_7	0	0	0	0	0	45	49	0	0	1006
c_8	0	0	0	0	6	40	0	836	0	1718
c_9	0	0	0	0	0	186	0	0	1652	262
c_{10}	0	0	0	0	0	3	0	0	0	11797

Condorcet Voting

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	49	21	0	0	8	0	0	0	0	322
c_2	0	314	1	0	1	0	0	0	0	584
c_3	0	1	1134	0	7	0	0	8	0	1650
c_4	0	0	11	94	6	0	0	31	1	757
c_5	0	1	3	0	1343	0	0	1	0	1852
c_6	0	0	0	0	0	4135	0	0	6	359
c_7	0	0	0	0	0	39	22	0	0	1039
c_8	0	0	9	0	10	10	0	984	0	1587
c_9	0	0	0	0	0	157	0	0	1448	495
c_{10}	0	0	0	0	0	2	0	0	0	11798

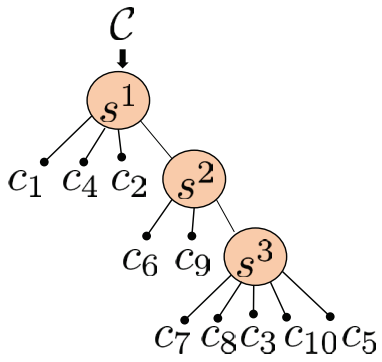


Figure 3.7: The BB-Fuse Tree for Application I.

Table 3.3: Application I: Description of the Optimal Sensors

ℓ	s^ℓ	**Sensing Device**	**Classifier**	k
1	s^1	Localization	LDA + k-NN	3
2	s^2	IMU on the Lower Arm	LDA + k-NN	3
3	s^3	Localization	LDA + k-NN	3

Integration, Majority Voting, Adaboost, Borda Counting, and Condorcet Counting (see appendix for details). The resulting confusion matrices and CCRs for these different methods are shown in Tables 3.2 and 3.4, respectively. Table 3.5 shows a comparison of execution times of different methods on a 64-bit Windows 7 operating system, with a Intel(R) Core(TM) i5-2400, 3.10GHz

Table 3.4: Application I: CCRs for Different Methods

Method	CCR	Method	CCR
BB-Fuse	**96%**	**Majority Voting**	85%
Best Sensor	93%	**Adaboost**	94%
CaRT	95%	**Borda Count Voting**	70%
C4.5	95%	**Condorcet Voting**	70%
Bayes Belief Integration	96%		

Table 3.5: Application I: Execution Times

Method	Training (s)	Testing (s)
Compute Confusion Matrices	603.49	**-**
BB-Fuse	10.06	7.31E-05
Best Sensor	5.09	4.10E-06
CaRT	37.35	5.34E-06
C4.5	3017.58	5.17E-06
Bayes Belief Integration	-	2.08E-02
Majority Voting	-	1.64E-05
Adaboost	467.96	8.80E-04
Borda Count Voting	0.02	3.07E-05
Condorcet Voting	-	4.76E-02

processor, with 16GB RAM.

It can be observed that for this application, the BB-Fuse algorithm achieved higher CCR than all other methods except the Bayes Belief Integration method. Both the BB-Fuse and Bayes Belief Integration algorithm achieved a CCR of 96%; however, it should be noted that the testing phase of the BB-Fuse algorithm is of lower complexity than the Bayes Belief Integration method. This is because the BB-Fuse algorithm sequentially tests the pre-determined classifiers at the non-terminal nodes, and it terminates once it reaches a terminal node; on the other hand, the Bayes Belief Integration method utilizes all the available classifiers for all devices in parallel for each testing data set. For example, to classify a test data as class c_2, the BB-Fuse algorithm only needs the classifier at node 1; however, the Bayes Belief Integration requires the decisions of all of the classifiers to conclude that the testing data belongs to class c_2.

3.6.2 Application II: User Identification

In this application, the Walking Activity Dataset available on the University of California, Irvine (UCI) Machine Learning Repository was used for User Identification. Here, the time series data

was collected using Android smart phones placed in the chest pockets of 10 users who were made to walk in the wild over a predefined path. These users formed the 10 classes, $c_i, i = 1, 2, \ldots 10$. The data contain 3D accelerations (m/s^2) with sampling rate of 33.3Hz, where the $x-, y-$ and $z-$ axes are illustrated in Fig. 3.8. The objective is to identify and authenticate the people using the 3D acceleration time series data [118].

Figure 3.8: The x, y and z axes for the Android Phones.

Since only one sensing device is available here, therefore for fusion purposes the time series data for the $x-$, $y-$ and $z-$ axes are treated as 3 different sensors. For each axis, the data for each user is partitioned into blocks of length 1000 each, thus forming a total of 50 blocks for all users. Subsequently, each block was again partitioned into 20 chunks, each of length 50 samples, which were then reshaped to form a 50×20 matrix. These matrices were then stacked below each other for all users to form the data matrix $O^j \in \mathcal{R}^{L \times 20}$, where $L = 2500$ corresponds to the total number of sample points for 50 blocks. Next, as before feature vectors F^j and G^j were extracted from O^j using PCA and LDA, respectively. Subsequently, the k-NN classifier is applied and the confusion matrices were generated for each axis data using the standard 10-fold cross validation process. It was found that the x-axis data, with PCA as feature extractor and the k-NN classifier with k=4, yielded the highest CCR of 97% when used alone.

Figure 3.9 shows the BB Fuse tree constructed on this application and the selected sensors are described in Table 3.6. Again the BB-Fuse performance is compared with other methods whose CCRs and execution times are listed in Tables 3.8 and 3.9, respectively. Table 3.7 shows the

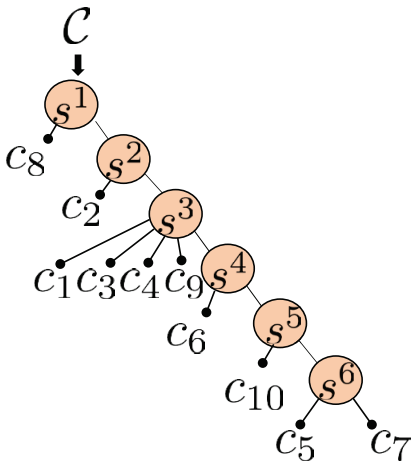


Figure 3.9: The BB-Fuse Tree for Application II.

Table 3.6: Application II: Description of the Optimal Sensors.

ℓ	s^ℓ	**Sensor**	**Classifier**	k
1	s^1	x	LDA + k-NN	1
2	s^2	z	LDA + k-NN	3
3	s^3	x	PCA + k-NN	4
4	s^4	x	LDA + k-NN	1
5	s^5	z	PCA + k-NN	6
6	s^6	x	PCA + k-NN	4

confusion matrices obtained for the different methods including the best sensor. As seen, the BB-Fuse tree produced a CCR of 99% which is 2% improvement over the best sensor.

3.7 Conclusions

This chapter presented a novel information fusion algorithm, called the *Better-than-the-Best Fusion* (BB-Fuse), for the purpose of improving the classification accuracy in multi-sensor multi-class systems. The BB-Fuse algorithm utilizes the fact that some sensors have better classification accuracy in isolating certain classes than others. Therefore, the algorithm constructs a one-vs-the-rest decision tree, called the BB-Fuse tree, which systematically selects the optimal sensor-class pairs at each level to maximize the classification accuracy. The BB-Fuse algorithm is analytically proven to yield a CCR that is higher than or at least equal to the best individual sensor. Both the training and testing phases of the BB-Fuse algorithm are shown to have low computational complexity. The application of the BB-Fuse algorithm is not limited to sensor fusion, but can be easily generalized

Table 3.7: Confusion Matrices for the User Identification from Walking Activity Dataset

BB-Fuse

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	247	0	0	0	0	2	0	0	1	0
c_2	0	600	0	0	0	0	0	0	0	0
c_3	0	0	150	0	0	0	0	0	0	0
c_4	0	0	0	50	0	0	0	0	0	0
c_5	0	0	0	0	149	0	1	0	0	0
c_6	0	0	2	0	0	448	0	0	0	0
c_7	0	0	0	0	7	0	193	0	0	0
c_8	0	0	0	0	0	0	0	150	0	0
c_9	0	0	0	0	0	1	0	0	149	0
c_{10}	0	0	0	0	0	0	0	0	0	350

Best Sensor

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	241	0	0	0	0	7	0	0	2	0
c_2	0	597	1	0	1	0	0	0	0	1
c_3	0	8	142	0	0	0	0	0	0	0
c_4	0	1	0	47	0	2	0	0	0	0
c_5	0	0	0	0	143	0	0	0	0	7
c_6	0	6	1	0	0	442	0	0	1	0
c_7	0	0	0	0	4	0	190	0	0	6
c_8	0	0	0	0	0	0	0	150	0	0
c_9	0	0	0	0	0	2	0	0	148	0
c_{10}	0	5	0	0	11	0	2	0	0	332

CaRT

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	207	0	0	0	0	14	0	0	29	0
c_2	0	600	0	0	0	0	0	0	0	0
c_3	1	0	143	2	2	2	0	0	0	0
c_4	0	0	2	42	0	3	0	0	3	0
c_5	0	0	1	0	130	0	12	0	0	7
c_6	15	0	5	5	0	409	0	0	16	0
c_7	0	0	0	0	7	0	192	1	0	0
c_8	0	0	0	0	1	0	0	149	0	0
c_9	29	0	0	2	0	16	0	0	103	0
c_{10}	0	0	1	0	3	0	0	0	0	346

C4.5

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	208	0	0	2	0	14	0	0	26	0
c_2	0	598	1	0	0	1	0	0	0	0
c_3	1	0	138	1	2	8	0	0	0	0
c_4	1	0	2	37	0	7	0	0	3	0
c_5	0	0	0	0	133	0	8	0	0	9
c_6	15	0	4	10	0	402	0	0	18	1
c_7	0	0	0	0	9	0	188	1	0	2
c_8	0	0	0	0	0	0	0	150	0	0
c_9	46	0	0	2	0	15	0	0	87	0
c_{10}	0	0	0	0	6	0	5	0	0	339

Bayes Belief Integration

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	245	0	0	0	1	2	0	0	2	0
c_2	0	600	0	0	0	0	0	0	0	0
c_3	0	0	148	0	0	0	1	1	0	0
c_4	0	0	0	50	0	0	0	0	0	0
c_5	0	0	0	0	148	0	1	0	0	1
c_6	2	0	0	0	0	447	0	0	1	0
c_7	0	1	0	0	5	0	194	0	0	0
c_8	1	0	0	0	0	0	0	149	0	0
c_9	2	0	0	0	0	2	0	0	146	0
c_{10}	0	0	0	0	0	0	0	0	0	350

Majority Voting

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	246	1	0	0	0	3	0	0	0	0
c_2	0	600	0	0	0	0	0	0	0	0
c_3	0	15	134	0	1	0	0	0	0	0
c_4	0	2	1	45	0	1	0	0	1	0
c_5	3	0	0	0	141	3	0	0	0	3
c_6	5	3	0	0	0	440	0	0	0	2
c_7	0	0	0	0	3	0	193	0	0	4
c_8	1	1	0	0	0	4	0	144	0	0
c_9	13	10	0	0	0	15	1	0	111	0
c_{10}	0	4	0	0	3	4	0	0	0	339

Adaboost

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	225	4	0	0	2	14	2	1	2	0
c_2	0	600	0	0	0	0	0	0	0	0
c_3	0	3	140	0	0	2	4	0	0	1
c_4	0	2	1	43	0	3	0	0	1	0
c_5	3	0	0	0	137	4	3	1	0	2
c_6	5	0	1	1	2	431	0	0	1	9
c_7	2	0	3	0	4	1	187	0	0	3
c_8	2	1	0	0	0	8	2	135	1	1
c_9	14	4	0	0	0	8	0	0	124	0
c_{10}	1	1	0	0	2	8	0	0	0	338

Borda Count Voting

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	205	29	0	0	0	16	0	0	0	0
c_2	0	600	0	0	0	0	0	0	0	0
c_3	0	12	137	0	0	1	0	0	0	0
c_4	0	0	1	48	0	1	0	0	0	0
c_5	1	10	3	0	78	15	6	0	0	37
c_6	1	5	0	0	0	432	0	0	0	12
c_7	0	0	0	0	0	0	192	0	0	8
c_8	0	0	0	0	0	0	0	150	0	0
c_9	3	22	0	0	0	17	0	0	107	1
c_{10}	0	2	0	0	0	4	0	0	0	344

Condorcet Voting

Actual Classes \ Classifier Output	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	227	0	0	0	0	23	0	0	0	0
c_2	0	600	0	0	0	0	0	0	0	0
c_3	0	1	144	0	0	5	0	0	0	0
c_4	0	0	0	33	0	7	0	0	10	0
c_5	0	0	0	0	143	5	1	0	0	1
c_6	2	0	0	0	0	448	0	0	0	0
c_7	0	0	0	0	4	0	194	0	0	2
c_8	0	0	0	0	0	2	14	130	0	4
c_9	29	0	0	0	0	23	0	0	98	0
c_{10}	0	0	0	0	0	1	0	0	0	349

Table 3.8: Application II: CCRs for Different Methods

Method	CCR	Method	CCR
BB-Fuse	**99**%	**Majority Voting**	96%
Best Sensor	97%	**Adaboost**	94%
CaRT	93%	**Borda Count Voting**	92%
C4.5	91%	**Condorcet Voting**	95%
Bayes Belief Integration	99%		

Table 3.9: Application II: Execution Times

Method	Training (s)	Testing (s)
Compute Confusion Matrices	3.29	-
BB-Fuse	2.40	1.92E-04
Best Sensor	0.55	3.53E-06
CaRT	0.32	1.90E-05
C4.5	82.92	1.34E-04
Bayes Belief Integration	-	1.58E-03
Majority Voting	-	2.12E-05
Adaboost	0.02	2.16E-04
Borda Count Voting	9.29	3.39E-05
Condorcet Voting	-	4.44E-03

to the fusion of different features and classifiers as well.

The BB-Fuse algorithm was validated on two datasets: the Opportunity Dataset and the User Identification from Walking Activity Dataset. In the Opportunity Data Set, time series data from 56 sensors were used to classify basic hand movements in a kitchen environment while in the User Identification Dataset the 3D acceleration data collected from the Android phones of several people were used to identify the users. In both applications, the performance of the BB Fuse algorithm was compared with the best sensor and several other existing methods; and it yielded better results. Besides, the BB-Fuse algorithm is computationally efficient because it sequentially tests only selected sensors and terminates upon reaching a terminal node.

The following areas are envisioned for future work:

- Use of classifier-independent information theoretic measure for selecting optimal sensor-class pairs in the decision tree construction.
- Expansion of the BB-Fuse algorithm to non-binary (or not one-vs-the-rest) decision trees.

CHAPTER 4

D*-FUSE: OPTIMAL SENSOR-CLASS PAIR SELECTION AND FUSION ALGORITHM FOR CLASSIFICATION

4.1 Introduction

Optimal sensor set selection and fusion for CCR improvement dilemmas are addressed in the first and second themes; however, with certain limitations. The first theme presented the Unsupervised Embedded algorithm, which selects the candidate list of sensors using the mRMR criterion then re-ranks the candidate list of sensors using the entropy. The Unsupervised Embedded algorithm is computationally efficient and leads to high CCR. However, it uses the mRMR algorithm which is criticized in literature for subtracting the redundancy term [11].

In the second theme, the BB-Fuse algorithm is presented. The BB-Fuse algorithm selects optimal sensor-class pairs using confusion matrices of individual sensors, then organizes them in the BB-Fuse Tree, which is a fusion structure that guarantees CCR improvement over the sensor that has the highest CCR. The BB-Fuse algorithm exploits the sensor-class pair information which is vital to design efficient sensor selection and fusion algorithms. This is because sensors vary in their accuracy isolating different classes. For example, a sensor might have high CCR; nevertheless, it has low accuracy isolating a specific class. It might also happen that a sensor has low CCR; however, its accuracy is better than other sensors isolating a specific class.

Exploiting sensor-class pair information, the BB-Fuse algorithm guarantees CCR improvement; however, the BB-Fuse algorithm requires as input the confusion matrices of individual sensors. The confusion matrices have two limitations: 1) they are expensive to compute and 2) they are pertinent to a specific classifier. The confusion matrices are expensive to compute because they are generated using the cross-validation algorithm, which is a computationally-expensive, iterative procedure. For each sensor, the cross-validation iteration splits the data into randomly selected training and testing data, then applies a specified classifier and updates the confusion matrix. The cross-validation algorithm is applied on each sensor using a specific classifier to produce sensor confusion matrices. Generation of sensor confusion matrices is evidently expensive in complex interconnected systems because the number of sensors is typically large. Besides that, if different classifiers are used, the sensor confusion matrices differ, and so do the optimal sensor-class pairs that are selected using the BB-Fuse algorithm. I.e., the BB-Fuse algorithm is classifier dependent.

The *Decomposed Sensor-Class pair Tree with maximum Admissible-Relevance based Fusion* (D*-Fuse) algorithm is presented in this theme to address the limitations of Unsupervised Embedded and BB-Fuse algorithms. Instead of using the mRMR, the D*-Fuse algorithm selects the optimal sensor-class pairs using a new information theoretic criterion that is called the Admissible Relevance (AR) criterion. The AR criterion exploits sensor-class pair information; i.e., the optimal sensor to isolate each class. Because the selected sensors carry information that isolates different classes, the AR criterion selects non-redundant sensor set. Therefore, it is not needed to subtract the redundancy term that is used by the mRMR.

Similar to the BB-Fuse algorithm, the D*-Fuse algorithm selects sensor-class pairs because sensors vary in the accuracy they can isolate different classes. The D*-Fuse algorithm then organizes the optimal pairs in a one-vs-the-rest tree which is called the D$*$-Fuse Tree, which is similar to the BB-Fuse Tree generated by the BB-Fuse algorithm. However, the D*-Fuse algorithm does not use confusion matrices. Instead, the D*-Fuse algorithm uses the AR criterion, which is not expensive to compute and does not pertain to a specific classifier. As a result, the D*-Fuse Tree is generalizable in the sense that any classifiers can be trained on its nodes after the sensor-class pair selection process (unlike the BB-Fuse Tree where we have to adhere to the classifiers that are used

to produce the sensor confusion matrices). This generalization is due to the use of the AR criterion which uses the classifier independent mutual information measure. The next subsection clarifies why mutual information can be used for sensor selection.

4.1.1 Basics of Information Theory

To understand the physical meaning of mutual information, the *entropy* has first to be introduced. Given a class set $\mathscr{C} = \{c_1, c_2, \ldots c_n\}$, define:

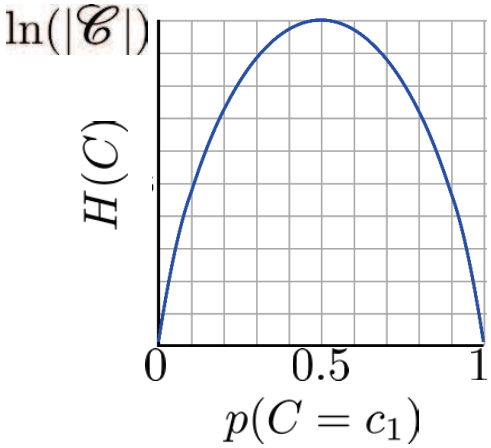
- C: A random variable whose sample space is the set of all training data and its outcome belongs to the class set $\mathcal{C} = \{c_1, \ldots c_n\}$ with probability mass function $p(C = c_i) = p(c_i)$.

The entropy is mathematically defined as follows:

$$H(C) \triangleq \sum_{i=1}^{n} p(c_i) \ln \frac{1}{p(c_i)} \tag{4.1}$$

The entropy can be used as a measure of uncertainty [62]. To explain, consider the case for which there are only two classes (i.e., binary classification problem and the number of classes $n = 2$). The plot of the binary entropy vs $p(c_1)$ (where a similar plot would be obtained for $H(C)$ vs $p(c_2) = 1 - p(c_1)$) is shown in Fig. **??**. As seen in the figure, the entropy is maximized when the classes are equi-probable (i.e., $p(c_1) = p(c_2) = 0.5$) [119]. This would be the most uncertain case where the classification error is maximum if the classification decision is based on the probability distribution of the classes only. On the other hand, the entropy drops down as one of the classes has more probability than the other one in which case a classification decision based on distribution will make less error.

The entropy can be represented by the circle in Fig. 4.1(a), where the classification decision may lay anywhere within the circle. Suppose now that we obtained data for the sensor $s \in \mathcal{S}$ and this data is used to make a classification decision. The classification uncertainty now drops as there is more information about the classes obtained using the sensor data. Mathematically, this can be



represented by the *conditional entropy* $H(C|S)$

$$H(C|S) = -\sum_{i=1}^{n}\sum_{j=1}^{|\Gamma|} p(c_i, \alpha_j) \ln p(c_i|\alpha_j) \tag{4.2}$$

where S is a random variable that represents the sensor data for the the sensor $s \in \mathcal{S}$ with alphabet, say $\Gamma = \{\alpha_1, \alpha_2, \ldots \alpha_{|\Gamma|}\}$. Figure 4.1(b) shows the entropy of C and the entropy of S. The conditional entropy is the area that is marked by the red line in the figure.

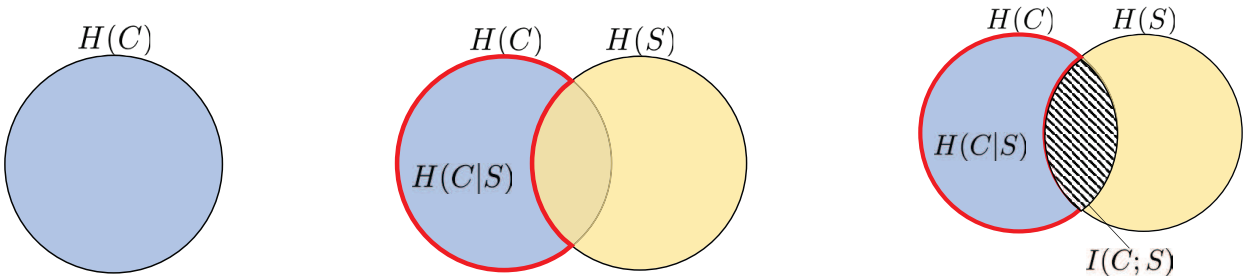
The mutual information is defined in terms of entropy as:

$$I(C;S) = H(C) - H(C|S) \tag{4.3}$$

which is shown graphically in Fig. 4.1(c). Being defined as such, the mutual information can be seen as the "reduction of uncertainty" (i.e., the amount of information learned) in classification after learning the data of the sensor s. Hence, it is intuitive to select the sensors that maximize mutual information as a group. This is further justified by the Fano's Inequality as discussed in Appendix A.3, which states that the classification error is lower bounded by

$$P_e \geq \frac{H(C|S) - 1}{\log|\mathcal{C}|} = \frac{-I(C;S) + H(C) - 1}{\log|\mathcal{C}|}. \tag{4.4}$$

where P_e is the lower bound of classification error. So, the lower bound of classification error P_e



(a) $H(C)$ (b) $H(C|S)$ (c) $I(C;S)$

Figure 4.1: Entropy $H(C)$, Conditional Entropy $H(C|S)$ and Mutual Information $I(C;S)$.

is minimized when the mutual information $I(C;S)$ is maximized.

4.2 Literature Review

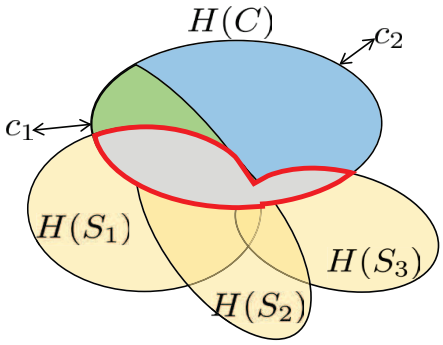
Two of the most important mutual information based sensor selection criteria are the Maximum Dependency (MD) and Maximum Relevance (MR) criteria. The MD criterion selects the subset of sensors that maximizes the mutual information as a group. Although accurate, the MD criterion is difficult to estimate especially when the number of sensors is large. Therefore, the MR criterion is used in literature as estimate for the MD criterion; however, the MR criterion may select redundant sensors as explained in this section.

While the literature review of Chapters 2 and 3 summarized various sensor selection and fusion algorithms in literature, the literature review in this chapter focuses primarily on the mutual information based sensor selection algorithms that are mentioned above.

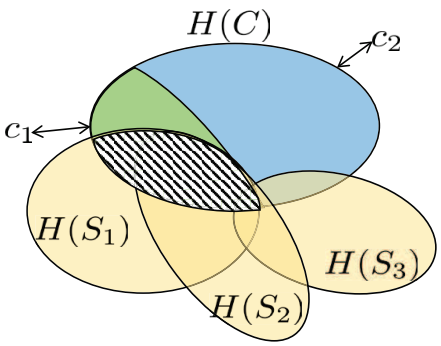
4.2.1 MD Criterion

The MD criterion [10] is a direct result of the Fano's Inequality. Given classification problem with class set $\mathscr{C} = \{c_1, c_2, \ldots c_n\}$ and sensor set $\mathcal{S} = \{s_1, s_2, \ldots s_m\}$, the maximum dependency criterion selects the subset of sensors $\mathcal{S}^*$ that jointly maximize the mutual information. Explicitly, the MD criterion can be expressed as:

$$\mathcal{S}^* = \arg\max_{\mathcal{S}' \subseteq \mathcal{S}} I(C; S') \tag{4.5}$$



(a) MD Criterion



(b) MR Criterion

Figure 4.2: Venn Diagrams for Maximum Dependency (MD) and Maximum Relevance (MR) criteria

where S' is the random variable that represents the subset $\mathcal{S}' \subseteq \mathcal{S}$ and its sample space is the joint events of sensor readings in $\mathcal{S}'$. For visual illustration, consider the following example.

Example 4.2.1 *Consider the classification problem in the Venn diagram in Fig. 4.2(a) with class set $\mathscr{C} = \{c_1, c_2\}$ and sensor set $\mathcal{S} = \{s_1, s_2, s_3\}$. The entropy of the class set is represented by the oval $H(C)$, which can be partitioned into two parts (one for each class as indicated). The sensor entropies are also illustrated in the figure by the ovals $H(S_1)$, $H(S_2)$ and $H(S_3)$. The objective in this example is to select the optimal sensor set of $m^* = 2$ sensors. There are three possible choices; namely, $\{s_1, s_2\}$, $\{s_1, s_3\}$ and $\{s_2, s_3\}$. Which subset should be selected?*

As indicated by Eq. (4.5), the MD criterion selects the subset of sensors that jointly maximizes the mutual information which is indicated by the intersection areas between C and the sensors (see Fig. 4.1(c)). Among the three possible subsets $\{s_1, s_2\}$, $\{s_1, s_3\}$ and $\{s_2, s_3\}$, the MD criterion will select $\mathcal{S}^* = \{s_1, s_3\}$ whose MD cost is indicated by the gray area in Fig. 4.2(a).

The MD criterion lays on strong mathematical basis (i.e., the Fano's Inequality explained in Appendix A.3). However, it is difficult to be correctly estimated because it requires estimating joint probabilities of several random variables. To see this, consider the cost function $I(C; S)$ that Eq. 4.5 maximizes, which is explicitly expressed below

$$I(C;S) \triangleq \sum_{c\in\mathscr{C}} \sum_{x_1\in\Gamma_1} \sum_{x_2\in\Gamma_2} \cdots \sum_{x_m\in\Gamma_m} p(x_1, x_2, \ldots x_m, c) \ln \frac{p(x_1, x_2, \ldots x_m, c)}{p(x_1, x_2, \ldots x_m)p(c)} \tag{4.6}$$

where Γ_j is the alphabet of sensor $s_j \; \forall \; j = 1, 2, \ldots m$ and S is a random variable whose alphabet is the Cartesian product $\Gamma_1 \times \Gamma_2 \ldots \Gamma_m$. To Estimate $I(C; S)$, all the probabilities of the joint events in Eq. (4.6) have to be estimated from sensor data. Unfortunately, estimating joint probabilities can be inaccurate even in moderate size classification problems because of insufficient statistics [10]. For example, suppose the size of training data is 10^6, the number of classes $n = 5$, and the number of sensors $m = 10$ where the alphabet size of each sensor s_j alphabet is $|\Gamma_j| = |\Gamma| = 10 \; \forall \; j = 1, 2 \ldots m$. To estimate $I(C; S)$, we need to estimate the joint probabilities of the events $(x_1, x_2, \ldots x_m, c) \; \forall \; x_j \in \Gamma_j, \; j = 1, 2, \ldots m$ and $c \in \mathscr{C}$. The number of these joint events is $n \times |\Gamma|^{10} = 5 \times 10^{10}$ which is larger than the size of the training data (10^6 in this example). Therefore, there is not sufficient statistics to estimate the joint probabilities $p(x_1, x_2, \ldots x_m, c) \forall \; x_j, c$. Subsequently, the MD criterion cannot be used for sensor selection in this classification problem.

4.2.2 MR Criterion

Because the MD criterion is difficult to compute, the *Maximum Relevance* (MR) criterion [10] is used as an estimate of the MD criterion. Instead of selecting the subset of sensors that maximizes the mutual information as a group, it selects the set of sensors that maximize the average mutual information between the sensors and class labels.

For mathematical explanation, define the random variables $S_j \; \forall \; j = 1, 2, ...m$ as follows:

- S_j: A random variable whose sample space is the sensor readings for sensor s_j and its outcomes belong to the set $\Gamma_j = \{\alpha_1, \alpha_2, \ldots \alpha_{|\Gamma_j|}\}$.

Then the Maximum Relevance criterion is defined as below:

$$\begin{aligned} \mathcal{S}^* &= \arg \max_{\mathcal{S}' \subseteq \mathcal{S}, |\mathcal{S}'| = m^*} J_{MR}(C, S'); \\ J_{MR}(C, S') &= \frac{1}{m^*} \sum_{s_j \in \mathcal{S}'} I(C; S_j) \end{aligned} \tag{4.7}$$

Despite the MD criterion, the MR criterion does not require estimation of joint mutual infor-

mation. The MR criterion can be easily solved by calculating the terms $I(C;S_j)\ \forall\ j=1,2,\ldots m$, then ranking the sensors and selecting the top m^* sensors that will also maximize the average.

Nonetheless, the MR criterion has the limitation that it may select redundant sensors [10]. To see this, consider the following example.

Example 4.2.2 *Consider the same sensor selection problem of Example 4.2.1, with class set* $\mathscr{C}=\{c_1,c_2\}$ *and sensor set* $\mathcal{S}=\{s_1,s_2,s_3\}$ *where the objective is to select the optimal subset of* $m^*=2$ *sensors.*

The MR criterion selects the subset of sensors that maximizes the average mutual information. Explicitly, average mutual information quantities are:

$$\frac{1}{2}[I(C;S_1)+I(C;S_2)],\quad \frac{1}{2}[I(C;S_1)+I(C;S_3)],\ \textit{and}\ \frac{1}{2}[I(C;S_2)+I(C;S_3)].$$

By visual inspection, it can be seen that the optimal subset that maximizes the average mutual information is $\mathcal{S}^*=\{s_1,s_2\}$ *as indicated in Fig. 4.2(b). Unfortunately,* s_1 *and* s_2 *in this case provide redundant classification information as they both intersect with the area of* c_1*, but none of the selected sensors provide any information about class* c_2*. In other words, the MR criterion has selected "redundant sensors".*

Because MR criterion may select redundant sensors as seen in Example 4.2.2, the *minimum Redundancy Maximum Relevance* (mRMR) [10] criterion has been proposed in literature as a better approximation of the MD criterion. To account for redundancy, the mRMR criterion subtracts a redundancy term from the MR criterion. Although the mRMR algorithm has been popular in various applications, it has been criticized in literature for subtracting the redundancy term [120].

In this theme, the novel Admissible Relevance (AR) sensor-class pair selection criterion is developed that accounts for sensor redundancy without subtracting the redundancy term. Instead of subtracting the redundancy term, the AR criterion selects sensor-class pairs. By sensor-class pair selection (instead of sensor only selection), the AR criterion assures that the selected sensors carry non-redundant information. This explained in details in the following section.

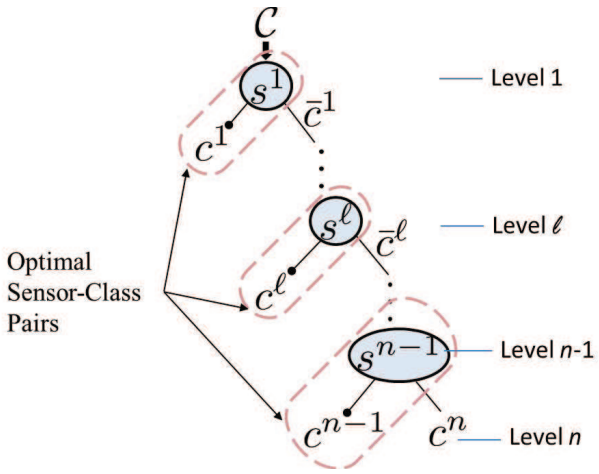


Figure 4.3: D*-Fuse Tree $\mathcal{T}_D$

4.3 D*-Fuse Algorithm

In this section, the D*-Fuse algorithm is described. The objective of the D*-Fuse algorithm is formally stated; after that, the training and testing phases to achieve the objective are presented.

4.3.1 Objectives

Given a classification problem with a set of n classes $\mathscr{C} = \{c_1, c_2, \ldots c_n\}$ and m sensors $\mathcal{S} = \{s_1, s_2, \ldots s_m\}$, the D*-Fuse algorithm aims to achieve two objectives:

- **Sensor-Class Pair Selection.** Which is selecting the optimal sensor-class pairs $(s^\ell, c^\ell)\ \forall\ \ell = 1, 2, \ldots n-1$. Each selected pair contains the class c^ℓ that can be best isolated from the rest of the classes and the sensor that is best used for it s^ℓ.
- **Sensor Fusion.** Which is to obtain a unified enhanced decision. This is done by organizing the selected sensor-class pairs in special fusion tree that is called the D*-Fuse Tree.

The structure of the D*-Fuse Tree (say $\mathcal{T}_D$) is similar to the BB-Fuse Tree that is described in Section 3.4.2 with one major difference: the D*-Fuse Tree is generalizable in the sense that any classifier can be trained on its internal nodes, whereas the BB-Fuse Tree is not. The D*-Fuse Tree $\mathcal{T}_D$ consists of n levels denoted by $\ell = 1, \ldots n$, as shown in Fig. 4.3. Level 1 consists of the root node but level n consists of the two bottom most leaf nodes with exit classes c^{n-1} and c^n. Levels

$2, 3 \ldots (n-1)$ consists of a *terminal* and a *non-terminal* node. Terminal nodes contain the classes that are isolated at previous levels (i.e., c^ℓ), and non-terminal nodes contain sensors $s^\ell \in \mathcal{S}$ which make decisions on the underlying binary classifications. The input to every non-terminal node is the rest of the remaining classes. To be explicit, the input to level 1 is $\bar{c}^0 \triangleq \mathscr{C}$ and the input to levels $2, 3, \ldots n-1$ are $\bar{c}^{\ell-1} \triangleq \bar{c}^{\ell-2} \setminus c^{\ell-1}$. Figure 4.3 shows the structure of the D*-Fuse Tree consisting of n leaf nodes and $n-1$ non-terminal nodes.

The objectives listed above are accomplished in two phases; namely, training and testing phases. In the training phase, a novel information theoretic criterion (called the AR criterion) for sensor-class pair selection is derived, then an algorithm is developed for D*-Fuse Tree construction. The testing phase uses the D*-Fuse Tree to obtain a unified enhanced decision.

4.3.2 D*-Fuse Training Phase

The D*-Fuse algorithm uses a novel sensor-class pair selection criterion that is called the AR criterion. There are two motivations for sensor-class pair selection instead of sensor only selection:

- Heterogeneous sensors may vary in the quality they isolate different classes from the rest. Therefore, the best overall sensor might not be the best sensor to isolate a specific class. Besides that, a sensor might have a low accuracy for the overall classification problem with n classes; however, it is superior in isolating one of the classes from the rest.

- Optimal sensor-class pair selection (rather than traditional sensor only selection) suppresses the undesired redundancy in the Maximum Relevance (MR) criterion. By sensor-class pair selection, the AR criterion ensures that selected sensors carry information about different classes; i.e., non-redundant classification information. This is further explained by the following example.

Example 4.3.1 *Again, consider the same problem in Example 4.2.1 with class set* $\mathscr{C} = \{c_1, c_2\}$ *and sensor set* $\mathcal{S} = \{s_1, s_2, s_3\}$ *with the objective of selecting the optimal sensor set* $\mathcal{S}^* \subseteq \mathcal{S}$ *of two sensors; i.e.,* $m^* = 2$*. Instead of selecting the sensor only set* $\mathcal{S}^*$*, optimal sensor-class pairs* (s^ℓ, c^ℓ) *are selected where optimal sensor set* $\mathcal{S}^* = \{s : (s, x) \in \{s^\ell, c^\ell\} \forall \ell = 1, 2, \ldots n\}$.

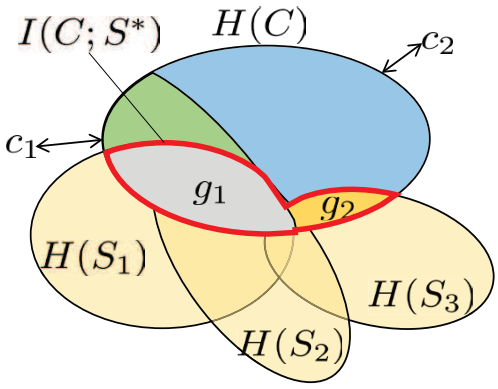


Figure 4.4: Admissible Relevance (AR) Venn Diagram

As seen in Fig. 4.4, the sensor-class pair that maximizes the intersection with $H(C)$ is (s_1, c_1) because of the area that is marked g_1. After (s_1, c_1) is selected, there are three sensor-class pair options to select for class c_2; namely, (s_1, c_2), (s_2, c_2) and (s_3, c_2). The optimal one of these is (s_3, c_2) because of the area marked g_2 in the figure.

Note that in one hand sensor s_2 (which provides good classification information) is not selected because it provides classification information that is "redundant" of s_1 ; i.e., information that pertains to class c_1. On the other hand, sensor s_3 (which provides less classification information than that s_2 provides) is selected because it carries "non-redundant" information of sensor s_1. So, the AR criterion selects non-redundant sensors by sensor-class pair selection.

Example 4.3.1 shows that selecting sensor-class pairs (instead of only sensor selection) intuitively leads to selection of non-redundant sensors. The challenge is now to find a mathematical expression for sensor-class pair selection (i.e., to mathematically express areas g_1 and g_2). We will begin with the term $I(C; S_j)$ and decompose it into a summation of terms such that

$$I(C; S_j) = Q(C_1, S_j) + Q(C_2, S_j) + \ldots Q(C_n, S_j) \tag{4.8}$$

where C_i is a random variable that models the isolation of class c_i from the rest of the classes.

Explicitly,

$$C_i = \begin{cases} 1 & \text{if } C = c_i \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}$$

The advantage of the decomposition is that it allows for sensor-class pair selection. Before decomposition, $I(C; S_j)$ can be maximized to find the best sensor for classification for the whole n classes. After decomposition; however, each term $Q(C_i, S_j)$ (aka *AR criterion*) in Eq. (4.8) can be individually maximized $\forall\ c_i \in \mathscr{C}, s_j \in \mathcal{S}$ to find the optimal sensor-class pairs $(s^\ell, c^\ell)\ \forall\ \ell = 1, 2, \ldots n$; i.e., the best sensor to isolate each class.

Optimization Problem

Taking the average of Eq. (4.8) over $j = 1, 2 \ldots m$ results in an expression that is equivalent to the MR criterion, which is called the *Decomposed MR* criterion. The Decomposed MR criterion is defined below with explicit expression of the term $Q(C_i, S_j)\ \forall\ i, j$. The equivalence of Decomposed MR and MR criteria is formally stated in Theorem 4.3.1.

Definition 4.3.1 (Decomposed MR Criterion) *The Decomposed MR criterion selects the m^* optimal sensor set $\mathcal{S}^*$ using the criterion:*

$$\mathcal{S}^* = \arg\max_{\mathcal{S}' \subseteq \mathcal{S}} J_{DMR} \tag{4.10}$$

where

$$J_{DMR} = \frac{1}{m^*} \sum_{s_j \in \mathcal{S}'} \sum_{c_i \in \mathscr{C}} Q(C_i, S_j)$$
$$Q(C_i, S_j) = I(C_i; S_j) - \frac{n-1}{n} I(V; S_j), \tag{4.11}$$

C_i is as defined in Eq. (4.9), and

- *V: a random variable whose sample space is the training data and its output belongs to the*

set $\mathscr{V} = \{\bar{c}_1, \bar{c}_2, \ldots \bar{c}_n\}$ *with probability mass function*

$$p(V = \bar{c}_i) = q_i = \frac{1 - p(c_i)}{\sum_{i=1}^{n} 1 - p(c_i)} = \frac{1 - p(c_i)}{n - 1} \; \forall \; i = 1, 2, \ldots n.$$

Theorem 4.3.1 *The MR and the Decomposed MR criteria are equal; i.e.,*

$$J_{MR} = J_{DMR}. \tag{4.12}$$

Proof: Consider the mutual information:

$$\begin{aligned} I(C_i; S_j) &= H(S_j) - H(S_j|C_i) \\ &= H(S_j) - p_i H(S_j|C_i = 1) - (1 - p_i) H(S_j|C_i = 0) \\ &= H(S_j) - p_i H(S_j|C_i = 1) - (n - 1) q_i H(S_j|C_i = 0) \end{aligned} \tag{4.13}$$

Now, lets sum Eq. 4.13 over $i = 1, 2, \ldots n$

$$\begin{aligned} \sum_{i=1}^{n} I(C_i; S_j) &= nH(S_j) - \overbrace{\sum_{i=1}^{n} \{p_i H(S_j|C_i = 1)\}}^{H(S_j|C_i)} - (n - 1) \sum_{i=1}^{n} \{q_i H(S_j|C_i = 0)\} \\ &= \overbrace{H(S_j) - H(S_j|C)}^{I(C;S_j)} + (n - 1) \overbrace{\left[H(S_j) - \sum_{i=1}^{n} q_i H(S_j|C_i = 0)\right]}^{I(V;S_j)} \end{aligned} \tag{4.14}$$

Therefore,

$$\begin{aligned} I(C; S_j) &= \sum_{i=1}^{n} \{I(C_i; S_j)\} - (n - 1) I(V; S_j) \\ &= \sum_{i=1}^{n} \{I(C_i; S_j) - \frac{n - 1}{n} I(V; S_j)\}. \end{aligned} \tag{4.15}$$

Hence,

$$Q(C_i, S_j) = I(C_i; S_j) - \frac{n - 1}{n} I(V; S_j) \tag{4.16}$$

and taking the average of both sides of Eq. (4.15) over $s_j \in \mathcal{S}'$ (where $\mathcal{S}' \subseteq \mathcal{S}$ and $|\mathcal{S}'| = m^*$) results in:

$$J_{MR} = \frac{1}{m^*} \sum_{s_j \in \mathcal{S}'} I(C; S_j) = \frac{1}{m^*} \sum_{s_j \in \mathcal{S}'} \sum_{c_i \in \mathscr{C}} Q(C_i, S_j) = J_{DMR}$$

□

Now that the MR criterion is decomposed, each term $Q(C_i, S_j)$, which is called the *Admissible Relevance* (AR) criterion, can be individually maximized to find the optimal sensor-class pairs.

Definition 4.3.2 (Admissible Relevance Criterion) *The Admissible Relevance (AR) criterion is a sensor-class pair selection criterion that selects pairs* $(s^\ell, c^\ell)\ \forall\ \ell = 1, 2, \ldots n$ *such that*

$$(s^\ell, c^\ell) = \arg \max_{c_i \in \mathscr{C}, s_j \in \mathcal{S}} Q(C_i, S_j) \tag{4.17}$$

Solution Process

The D*-Fuse uses the AR criterion to construct D*-Fuse Tree, which uses the best sensor-class pair to isolate a class at each level. The D*-Fuse relies on the fact that heterogeneous sensors may vary in the quality they can discern different classes; besides, sensor-class pair selection avoids selecting redundant sensors as shown in Example 4.3.1. Therefore, the D*-Fuse algorithm performs optimal sensor-class pair selection using the AR criterion instead of only sensor selection. Besides that, the D*-Fuse algorithm shrinks the class space at each iteration by removing the classes that are selected in previous iterations, which improves classification accuracy.

For initialization, define $c^0 \triangleq \emptyset$ and $\bar{c}^0 = \mathscr{C}$. Then, the D*-Fuse Tree is constructed in the training phase using an iterative procedure of $n - 1$ iterations denoted by $\ell = 1, 2, \ldots n - 1$. In each iteration ℓ, the following steps are performed:

1. The optimal pair (s^ℓ, c^ℓ) using the equation:

$$(s^\ell, c^\ell) = \arg \max_{c_i \in \bar{c}^{\ell-1}, s_j \in \mathcal{S}} Q(C_i, S_j)$$

2. The input class set $\bar{c}^\ell$ for the next iteration is obtained by removing the class that is already selected; i.e., $\bar{c}^\ell = \bar{c}^{\ell-1} \setminus c^\ell$.

Performing the iterations $\ell = 1, 2, \ldots n-1$ as described above outputs the D*-Fuse Tree that is shown in Fig. 4.3. This decision tree is used in the testing phase as described next.

4.3.3 D*-Fuse Testing Phase

Given unlabeled time series data collected only from the optimal sensors $s^\ell;\ \ell = 1, \ldots n-1$, the execution moves down the D*-Fuse Tree starting from the root node using the corresponding optimal sensor-class pairs. At level ℓ; to illustrate, sensor s^ℓ is utilized for the binary classification problem of isolating c^ℓ from $\bar{c}^\ell$ to obtain a decision $\hat{c}^\ell$. If $\hat{c}^\ell = c^\ell$, the algorithm terminates and outputs a decision $\hat{c} = c^\ell$; otherwise, the next non-terminal node is used. The above process is repeated until a terminal node is reached.

4.4 Results and Discussion

For numerical verification, the D*-Fuse algorithm is applied on three applications in this theme. The first application is a toy example that is generated using normal distributions for different sensor-class pairs for a proof of concept. In the second application, data that are collected using Kinect camera sensor are utilized to classify human gestures. The third application performs human activity recognition using 3D linear and angular acceleration data, which are collected using smart phones. Results and more details of the experiments are presented below.

4.4.1 Application I: Simulated Data

Simulated data are generated assuming Gaussian distribution $\mathcal{G}(\mu_{ij}, \sigma_{ij}^2)\ \forall\ i = 1, 2, 3, 4, \jmath = 1, 2, 3$ for 3 sensors $\mathscr{S} = \{s_1, s_2, s_3\}$ and 4 classes $\mathscr{C} = \{c_1, c_2, c_3, c_4\}$. The means μ_{ij} and variances σ_{ij}^2 for sensor-class pairs $\{(s_j, c_i)\ \forall\ c_i \in \mathscr{C}$ and $s_j \in \mathscr{S}\}$ are shown in Table 4.1. As explained later, the means and variances of sensor-class pairs are deliberately selected so that the expected D*-Fuse Tree is as shown in Fig. 4.5.

For each sensor s_j, a total of 400 time series data $x_t\ \forall\ t = 1, 2, \ldots 400$ are generated with labels y_t such that each class c_i has 100 time series data, where each time series data x_t is a real vector of

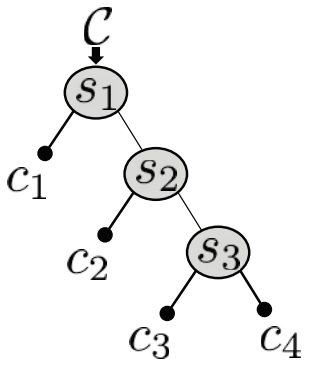


Figure 4.5: D*-Fuse Tree for Application I

Table 4.1: Sensor Data Distribution for Application I

	s_1	s_2	s_3
c_1	$\mathcal{G}(1.8, 9)$	$\mathcal{G}(3, 15)$	$\mathcal{G}(0.1, 0.5)$
c_2	$\mathcal{G}(2, 10)$	$\mathcal{G}(3, 15)$	$\mathcal{G}(0.1, 0.5)$
c_3	$\mathcal{G}(2, 10)$	$\mathcal{G}(3.17, 15.5)$	$\mathcal{G}(0.1, 0.5)$
c_4	$\mathcal{G}(2, 10)$	$\mathcal{G}(3.17, 15.5)$	$\mathcal{G}(0.12, 0.51)$

Table 4.2: Optimal Sensors and CCRs for Application I

(a) Optimal Sensors

Criterion	s_1^*	s_2^*	s_3^*
AR	s_1	s_2	s_3
mRMR	s_1	s_2	s_3
MR	s_1	s_2	s_3

(b) CCRs

	Sensor Selection Criterion		
Fusion Method	**AR**	**mRMR**	**MR**
D*-Fuse	**98%**	-	-
Majority Vote	58%	58%	58%
CaRT	97%	97%	97%
Adaboost	63%	63%	63%

size $10,000 \times 1$ with distribution $\mathcal{G}(\mu_{ij}, \sigma_{ij}^2)$, and $y_t \in \mathscr{C}$ is its label. The label y_t is determined as shown Table 4.1.

Next, the AR, mRMR and MR sensor selection criteria are applied. The AR criterion selected the optimal sensor-class pairs $(s^\ell, c^\ell) = (s_\ell, c_\ell)\ \forall\ \ell = 1, 2, 3$, respectively where the corresponding D*-Fuse Tree is shown in Fig. 4.5. Both of the mRMR and MR criteria resulted in the optimal sensor set $\mathcal{S}^* = \{s_1, s_2, s_3\}$, where s_1 was the first, s_2 was the second and s_3 was the last rank

sensor. Table 4.2(a) summarizes the optimal sensors selected these criteria.

After data is generated as described above, sample means an variances of the sensor $s_j \; \forall \; j = 1, 2, 3$ data are extracted and plotted in Fig. 4.6. Each column in Fig. 4.6 corresponds to a sensor and each row shows the updated feature space for Levels 1,2 and 3. As seen in the figure, the data is generated so that sensor s_1 can easily be used in Level 1 to isolate class c_1 from the rest of the classes $\bar{c}_1$. Subsequently, the root node is constructed that holds sensor s_1 with exit classes c_1 and $\bar{c}_1$, which indicates that sensor s_1 should be used to isolate class c_1 from the rest of the classes. At Level 2 next, class c_1 is removed as shown in the figure; therefore, class c_2 can now be easily isolated using sensor s_2 and the tree is expected to be updated as shown in Level 2 in Fig. 4.6. Similarly, class c_2 is removed in Level 3; so, class c_3 can now be isolated using sensor s_3. The simulated data is generated so that the D*-Fuse training phase is expected to result in the D*-Fuse Tree at the last row of Fig. 4.6 and so it does as shown in Fig. 4.5.

After feature extraction (i.e., sample mean and variance calculation), the k-NN classifier is

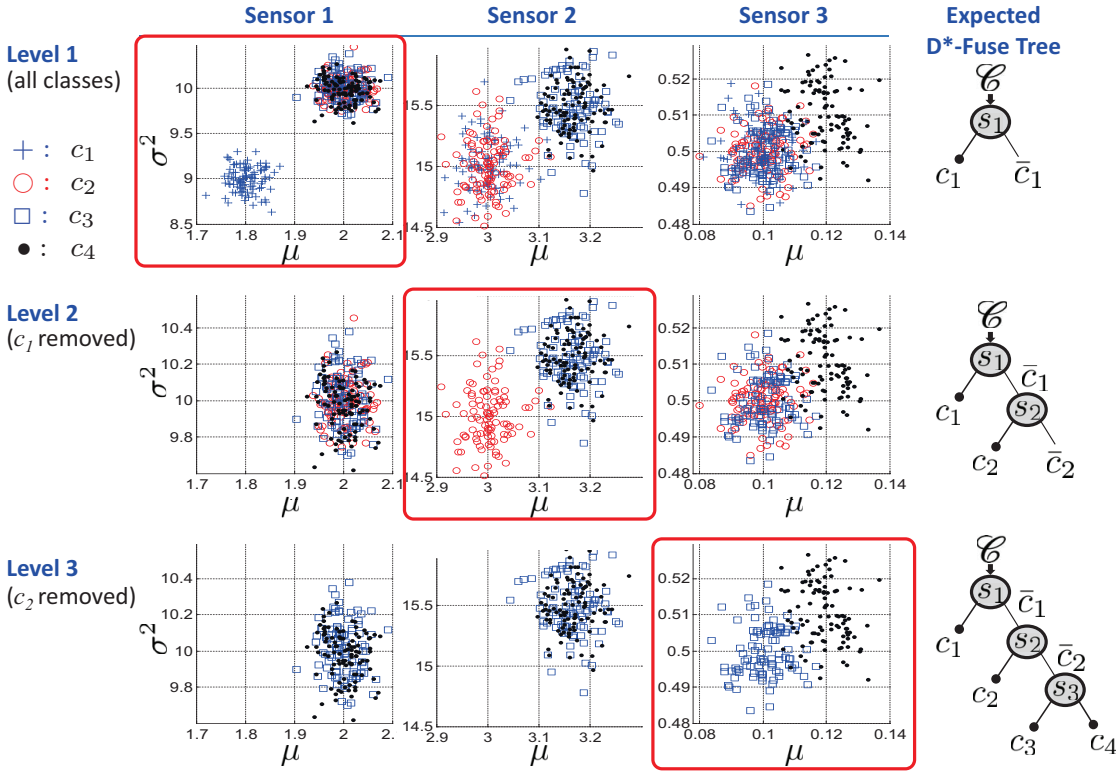


Figure 4.6: Sample Means and Variances for the Simulated Data

trained at each Level $\ell \; \forall \; \ell = 1, 2, 3$ to solve the corresponding binary classification problem of isolating class c^{ℓ} from $\bar{c}^{\ell}$. The D*-Fuse Tree is then evaluated using the 10-Fold cross-validation algorithm and the D*-Fuse Tree resulted in 98% as indicated in Table 4.2(b) with confusion matrix as shown in Table 4.3.

For comparison, Majority Vote, *Classification and Regression Tree* (CaRT), and *Adaptive Boosting* (Adaboost) fusion algorithms were applied on the optimal sensor sets that are obtained using AR, MR and mRMR and evaluated using the 10-Fold cross-validation algorithm as well. D*-Fuse algorithm (with 98% CCR) superseded other sensor-selection and fusion algorithms where the second best was the $CaRT$ algorithm with 97% CCR. Because Majority Vote and Adaboost algorithm does does not take advantage of sensor-class pair information, they achieved low CCRs of 58% and 63%. CCRs and confusion matrices are summarized in Tables 4.2(b) and 4.3.

The simulated data results indicate that an effective fusion algorithm must take advantage of sensor-class pair information. This is because some sensors might be good (or bad) classifying

Table 4.3: Confusion Matrices for Application I

Fusion Algorithm			**Sensor Selection Criterion**											
			Admissible Relevance Classifier Output				**mRMR Classifier Output**				**Maximum Relevance Classifier Output**			
			c_1	c_2	c_3	c_4	c_1	c_2	c_3	c_4	c_1	c_2	c_3	c_4
D*-Fuse	**Actual**	c_1	100	0	0	0	-	-	-	-	-	-	-	-
		c_2	0	98	2	0	-	-	-	-	-	-	-	-
		c_3	0	0	97	3	-	-	-	-	-	-	-	-
		c_4	0	1	2	97	-	-	-	-	-	-	-	-
Majority Vote	**Actual**	c_1	84	16	0	0	84	16	0	0	84	16	0	0
		c_2	46	42	11	1	46	42	11	1	46	42	11	1
		c_3	15	26	40	19	15	26	40	19	15	26	40	19
		c_4	6	18	10	66	6	18	10	66	6	18	10	66
CaRT	**Actual**	c_1	100	0	0	0	100	0	0	0	100	0	0	0
		c_2	0	99	1	0	0	99	1	0	0	99	1	0
		c_3	0	1	96	3	0	1	96	3	0	1	96	3
		c_4	0	3	6	91	0	3	6	91	0	3	6	91
Adaboost	**Actual**	c_1	78	15	6	1	78	15	6	1	78	15	6	1
		c_2	27	44	25	4	27	44	25	4	27	44	25	4
		c_3	11	17	51	21	11	17	51	21	11	17	51	21
		c_4	2	4	17	77	2	4	17	77	2	4	17	77

some classes but has low (or high) accuracy classifying other classes. For example, sensor s_1 (with feature space as shown in the first row of Fig. 4.6) is expected to have very low CCR; however, it can separate class c_1 very well.

This experiment is performed in a 32-bit MatLab running on a 3.10 GHz Intel(R) Core(TM) $i5-2400$ processor with 16 GB ram and Windows 7. AR criterion sensor-class pair selection execution time is 4.15s. D*-Fuse training execution time is 9.25s while D*-Fuse testing execution time is 3.00E-04s.

4.4.2 Application II: Gesture Phase Segmentation

Gesture phase segmentation is critical for the evolution of human-computer interaction applications. This classification task is interested in identifying 5 phases people go through when making gestures while speaking; namely [121]:

1. *Rest Phase*: the non-movement between gestures.
2. *Preparation Phase*: the transitory phase before performing the gesture stroke.
3. *Stroke Phase*: the main movement due to a gesture.
4. *Hold Phase*: a pause before or after performing a gesture.
5. *Retraction Phase*: the transitory phase after performing the gesture stroke.

The data is collected using Xbox Kinect sensor by video taping 3 subjects narrating three different comic stories. After that, the ground truth segmentation is manually established by an expert. Then, 3D positions of the 1) left palm $(x, y, z)^{lp}$, 2) left wrist $(x, y, z)^{lw}$, 3) right palm $(x, y, z)^{rp}$, 4) right wrist $(x, y, z)^{rw}$, 5) spine $(x, y, z)^{sp}$ and 6) head $(x, y, z)^{hd}$ of the subjects are tracked from the video frames which constructs the sensor time series data. These 3D locations constitute the sensor set $\mathcal{S} = \{s_1, s_2, \ldots s_{18}\}$ as explicitly listed in Table 4.4.

For each sensor $s_j \in \mathcal{S}$, 96 chunks of data are collected $\mathbf{x}_t \; \forall \; t = 1, 2, \ldots 96$, where each chunk is manually labeled by an expert as $y_t \in \mathscr{C}$ with sampling rate of 30Hz. Each chunk length is 100 samples (i.e., 3.3s), and the number of chunks for each class vary.

(a) Rest (b) Preparation (c) Stroke (d) Retraction

Figure 4.7: The 4 main Gesture Phases [122]. The Hold Phase is not shown and it is simply a pause before or after the Stroke Phase.

Table 4.4: Sensor Definition

Human Part	s	3D Location	Human Part	s	3D Location
Left Palm	s_1, s_2, s_3	$(x, y, z)^{lp}$	**Right Wrist**	s_{10}, s_{11}, s_{12}	$(x, y, z)^{rw}$
Left Wrist	s_4, s_5, s_6	$(x, y, z)^{lw}$	**Spine**	s_{13}, s_{14}, s_{15}	$(x, y, z)^{sp}$
Right Palm	s_7, s_8, s_9	$(x, y, z)^{rp}$	**Head**	s_{16}, s_{17}, s_{18}	$(x, y, z)^{hd}$

Sensor selection is applied after data generation applying the AR, mRMR and MR criteria on the data $\mathbf{x}_t \; \forall \; t = 1, 2, \ldots 96$ with labels y_t. The optimal sensor sets using different criteria are listed in Table 4.5(a). The AR algorithm selected 3 sensor-class pairs where the number of optimal sensors is automatically determined to be $m^* = 3$. The D*-Fuse training phase resulted in the D*-Fuse Tree that is shown in Fig. 4.8 (note that the sensor z^{hd} is selected twice at Level 1 and at Level 3, hence the number of optimal sensors is set to $m^* = 3$ instead of $n - 1 = 4$). For comparison, the mRMR and MR criteria are applied to select $m^* = 3$ sensors likewise. Interestingly, the AR, mRMR and MR criteria selected different sensor sets in this application as listed in Table 4.5(a).

After sensor selection, PCA is used to extract feature from optimal sensors only. Note that this reduces computational complexity because PCA is applied on $m^* = 3$ sensors instead of the whole sensor set with $m = 18$ sensors. This emphasizes the advantage of performing sensor selection before feature extraction (also before classifier training).

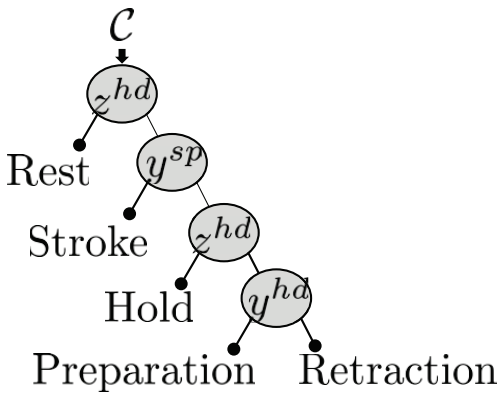


Figure 4.8: D*-Fuse Tree for Application II

Table 4.5: Optimal Sensors and CCRs for Application II

(a) Optimal Sensors

Criterion	s_1^*	s_2^*	s_3^*
AR	z^{hd}	y^{sp}	y^{hd}
mRMR	z^{hd}	x^{rp}	y^{hd}
MR	z^{hd}	z^{rp}	y^{sp}

(b) CCRs

	Sensor Selection Criterion		
Fusion Method	**AR**	**mRMR**	**MR**
D*-Fuse	**100%**	-	-
Majority Vote	100%	100%	100%
CaRT	84%	83%	82%
Adaboost	100%	99%	98%

To perform the PCA, each chunk $\mathbf{x}_t$ is first reshaped to a matrix $\mathbf{X}_t$ of size 5×20. Second, the matrix $\mathbf{X}$ is constructed by concatenating the matrices $\mathbf{X}_t \; \forall \; t = 1, 2, \ldots 96$ such that the size of $\mathbf{X}$ is 480×20 Using the matrix $\mathbf{X}$, the principal components are extracted as described in Appendix B.1, which results in the feature matrix $\mathbf{X}'$. The matrix $\mathbf{X}'$ has $96 \times 5 = 480$ rows, while the number of columns varies from a sensor to another and selected to satisfy the 95% energy criterion. The feature matrix of each sensor is the input to the k-NN algorithm that is described in Appendix B.3.

Next the k-NN classifier is trained for the corresponding binary classification problems at each level of the D*-Fuse Tree in Fig. 4.8. For comparison, majority vote, CaRT and Adaboost fusion methods are applied on the optimal sensor set selected by different criteria and the results are shown in Tables 4.5(b) and 4.6. The D*-Fuse Tree resulted in 100% CCR in this application and so does the majority vote fusion with the AR, MR and mRMR criteria. Note that the D*-Fuse testing

phase does not require the decisions of all of the optimal sensors; therefore, it is computationally less expensive than the majority vote fusion. The AR criterion lead to higher CCRs (84% and 100%) in comparison to mRMR and MR when used with CaRT and Adaboost. This is because the AR algorithm selected sensor-class pairs but the mRMR and the MR select optimal sensors only; hence, the AR algorithm extracts sensor-class pair information and lead to better results.

On a 32-bit MatLab running on a 3.10 GHz Intel(R) Core(TM) $i5-2400$ processor with 16 GB ram and Windows 7, the AR criterion sensor-class pair selection execution time is 0.86s. D*-Fuse training execution time is 0.41s while D*-Fuse testing execution time is $4.98e-02$s.

Table 4.6: Confusion Matrices for Application II

Fusion Algorithm		Sensor Selection Criterion														
		Admissible Relevance Classifier Output					mRMR Classifier Output					Maximum Relevance Classifier Output				
		c_1	c_2	c_3	c_4	c_5	c_1	c_2	c_3	c_4	c_5	c_1	c_2	c_3	c_4	c_5
D*-Fuse	c_1	50	0	0	0	0	-	-	-	-	-	-	-	-	-	-
	c_2	0	100	0	0	0	-	-	-	-	-	-	-	-	-	-
	c_3	0	0	145	0	0	-	-	-	-	-	-	-	-	-	-
	c_4	0	0	0	20	0	-	-	-	-	-	-	-	-	-	-
	c_5	0	0	0	0	165	-	-	-	-	-	-	-	-	-	-
Majority Vote	c_1	50	0	0	0	0	50	0	0	0	0	50	0	0	0	0
	c_2	0	100	0	0	0	0	100	0	0	0	0	100	0	0	0
	c_3	0	0	145	0	0	0	0	145	0	0	0	0	145	0	0
	c_4	0	0	0	20	0	0	0	0	20	0	0	0	0	20	0
	c_5	0	0	0	0	165	0	0	0	0	165	0	0	0	0	165
CaRT	c_1	38	1	8	1	2	34	4	5	0	7	35	0	8	1	6
	c_2	4	88	2	0	6	8	79	3	4	6	2	85	8	0	5
	c_3	6	8	123	0	8	4	3	134	0	4	6	7	113	4	15
	c_4	3	2	6	9	0	3	8	0	9	0	0	0	2	17	1
	c_5	0	6	16	0	143	5	8	8	3	141	5	6	11	1	142
Adaboost	c_1	50	0	0	0	0	50	0	0	0	0	50	0	0	0	0
	c_2	0	100	0	0	0	0	99	1	0	0	0	100	0	0	0
	c_3	0	0	145	0	0	0	0	144	1	0	0	0	142	0	3
	c_4	0	0	0	20	0	0	0	0	20	0	0	0	0	20	0
	c_5	0	0	0	0	165	0	1	0	0	164	0	0	6	0	159

4.4.3 Application III: Human Activity Recognition Dataset

Human Activity Recognition is an active research area with various applications including entertainment, assisted living, smart homes and others. Dataset used in this application is a human activity recognition dataset[123] with 7 classes; namely, c_1 walking, c_2 walking upstairs, c_3 walking downstairs, c_4 sitting, c_5 standing, c_6 lying and c_7 transition class. The transition class is is a) either stand-to-sit, b) sit-to-stand, c) sit-to-lie, d) lie-to-sit, e) stand-to-lie, or d) lie-to-stand transition. The 7 classes $\mathcal{C} = \{c_1, c_2, \ldots c_7\}$ in this application are listed in Table 4.7(a).

To perform the experiment, thirty volunteers were asked to perform several repetitions of the above activities and data are collected using smartphones mounted to the subjects' waists. "Sensing devices" include 3D linear and angular accelerations $\mathscr{Z} = \{z_1, z_2, \ldots z_6\}$ and the sampling rate is 50Hz. The sensing device set $\mathscr{Z}$ are listed in Table 4.7(b).

Table 4.7: Class and Sensor Set Definition for Application III

(a) Classes

Class	Description
c_1	Walking
c_2	Walking Upstairs
c_3	Walking Downstairs
c_4	Sitting
c_5	Standing
c_6	Lying
c_7	Transition

(b) Sensing Devices

z	Symbol	Definition
z_1	$\ddot{x}$	x-direction acceleration
z_2	$\ddot{y}$	y-direction acceleration
z_3	$\ddot{z}$	z-direction acceleration
z_4	$\ddot{\alpha}$	Angular acceleration around x-axis
z_5	$\ddot{\beta}$	Angular acceleration around y-axis
z_6	$\ddot{\gamma}$	Angular acceleration around z-axis

For each sensing device $z \in \mathscr{Z}$, a total of 2112 time series data $\mathbf{w}_t \; \forall \; t = 1, 2, \ldots 2112$ are collected and manually labeled as $y_t \in \mathscr{C}$ (where the number of time series data per class vary). Each time series data $\mathbf{w}_t$ is of length 300 samples (i.e., 6 seconds).

In this experiment, wavelet transform with 2^{nd} Gaussian mother wavelet (shown in Fig. 4.9) is first applied to the time series data $\mathbf{w}_t$ at 32 scales $8, 16, \ldots 256$. Wavelet transform shows frequency and time domain information rather than time only information that is indicated by the time series data, which could lead to better classification information. Depending on the data,

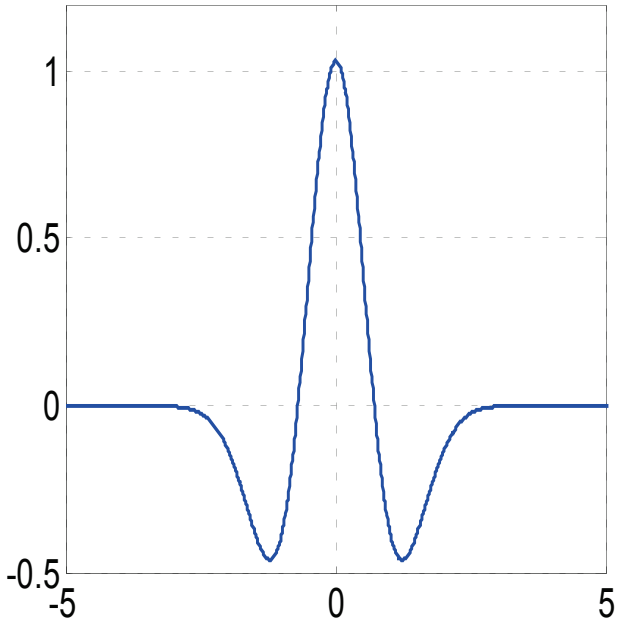


Figure 4.9: 2^{nd} Order Gaussian Mother Wavelet

wavelet scales that lead to high CCR might differ from a sensing device to another. Moreover, different scales might lead to better performance isolating a specific class. In other words, a specific scale might have low (or high) overall CCR but lead to superior (or poor) performance isolating a specific class. Therefore, this experiment selects the best "sensor" (i.e., sensing device + wavelet scale) for each class (i.e., selects optimal sensor-class pairs).

The sensor set is defined as $\mathcal{S} = \{s_1, s_2, \ldots s_m\}$ where $m = 6 \times 32 = 192$ and each sensor $s_j \; \forall \; j = 1, 2, \ldots m$ is a combination of a sensing device and a wavelet scale. After that, the AR criterion is applied for sensor selection, where the resultant D*-Fuse Tree is shown in Fig. 4.10 and the sensor set definition is in Table 4.8. Table 4.8 also shows the sensor set selected using the mRMR and MR criteria for comparison. In this experiment, the number of optimal sensors is automatically determined by the D*-Fuse algorithm to be $m^* = n - 1 = 6$ sensors. Interestingly, the AR, mRMR and MR criteria selected different sensor sets.

For feature extraction, PCA is applied on the wavelet transformed data. For a sensor $s \in \mathcal{S}$, denote the time series data by $\mathbf{w}_t \; \forall \; t = 1, 2, \ldots 2112$, where $\mathbf{w}_t$ is a vector of size 300×1. Denote the wavelet transformed data at a scale by $\tilde{\mathbf{w}}_t$, which is also a vector of size 300×1. To apply PCA, each $\tilde{\tilde{\approx}}_t$ is reshaped to a matrix $\tilde{\mathbf{W}}_t$ matrix of size 30×10, then a matrix $\mathbf{W}$ is formed by

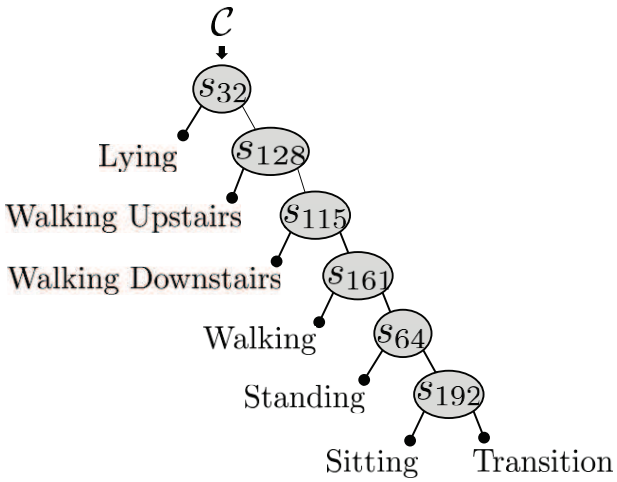


Figure 4.10: D*-Fuse Tree for Application III

Table 4.8: Optimal Sensor Sets for Application III

Rank	Admissible Relevance			mRMR			Maximum Relevance		
	s_j	SD	Scale	s_j	SD	Scale	s_j	SD	Scale
1	s_{32}	$\ddot{x}$	256	s_{128}	$\ddot{\alpha}$	256	s_{128}	$\ddot{\alpha}$	256
2	s_{128}	$\ddot{\alpha}$	256	s_{10}	$\ddot{x}$	80	s_{127}	$\ddot{\alpha}$	248
3	s_{115}	$\ddot{\alpha}$	152	s_{65}	$\ddot{z}$	8	s_{126}	$\ddot{\alpha}$	240
4	s_{161}	$\ddot{\gamma}$	8	s_{64}	$\ddot{y}$	256	s_{125}	$\ddot{\alpha}$	232
5	s_{64}	$\ddot{y}$	256	s_{131}	$\ddot{\beta}$	24	s_{124}	$\ddot{\alpha}$	224
6	s_{192}	$\ddot{\gamma}$	256	s_{160}	$\ddot{\beta}$	256	s_{123}	$\ddot{\alpha}$	216

SD: Sensing Device

concatenating matrices $\tilde{\mathbf{W}}_t \forall\ t = 1, 2, \ldots 2112$, where $\mathbf{W}$ has 2112×30 rows and 10 columns. After that, the transformation matrix is extracted from $\mathbf{W}$ by applying the KL algorithm that is described in Appendix B.1, which is results in the features $\mathbf{W}'$.

Finally, k-NN classifier is trained for classification at each level of the D*-Fuse Tree in Fig. 4.10. Evaluating the D*-Fuse Tree using 10-Fold cross-validation algorithm, the resultant CCR is 95%, where the confusion matrix is shown in Table 4.9.

For comparison, majority vote, CaRT and Adaboost fusion algorithms are applied on optimal sensor sets selected using the AR, mRMR and MR algorithms. The resultant confusion matrices are shown in Table 4.9 while the CCRs are summarized in table Table 4.10. The D*-Fuse results in

Table 4.9: Confusion Matrices for Application III

Fusion Algorithm		Sensor Selection Criterion																				
		Admissible Relevance							mRMR							Maximum Relevance						
		Classifier Output							Classifier Output							Classifier Output						
		c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_1	c_2	c_3	c_4	c_5	c_6	c_7
D*-Fuse	c_1	2047	0	0	15	68	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	c_2	11	1913	0	1	1	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	c_3	4	0	1532	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	c_4	1	0	0	1917	254	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	c_5	5	8	0	225	2156	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	c_6	0	0	0	0	0	2406	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	c_7	50	1	4	0	0	0	53	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Majority Vote	c_1	2098	4	2	0	25	1	0	2110	12	2	1	5	0	0	2050	0	0	5	65	10	0
	c_2	27	1899	0	0	0	0	0	140	1774	9	0	3	0	0	0	1916	0	0	6	4	0
	c_3	62	52	1422	0	0	0	0	224	68	1244	0	0	0	0	0	0	1532	0	0	2	2
	c_4	4	5	0	1926	108	129	0	3	0	1	2017	122	29	0	0	0	0	1641	188	343	0
	c_5	9	7	1	244	2101	32	0	11	1	0	155	2219	8	0	7	8	0	390	1766	223	0
	c_6	1	3	0	91	18	2293	0	0	0	1	34	22	2349	0	1	7	0	861	307	1230	0
	c_7	0	5	1	0	0	1	101	0	1	0	0	0	1	106	0	1	2	0	0	1	104
CaRT	c_1	1967	20	6	49	88	0	0	1984	20	6	40	80	0	0	1955	16	6	13	109	31	0
	c_2	17	1905	4	0	0	0	0	16	1905	4	0	1	0	0	6	1912	4	0	4	0	0
	c_3	12	0	1524	0	0	0	0	12	0	1524	0	0	0	0	13	0	1522	0	1	0	0
	c_4	46	0	0	1834	292	0	0	75	0	0	1845	252	0	0	58	0	0	1113	323	678	0
	c_5	137	24	0	267	1966	0	0	162	24	0	299	1909	0	0	194	24	0	416	1234	526	0
	c_6	0	0	0	0	0	2406	0	0	0	0	0	0	2406	0	138	18	17	722	521	990	0
	c_7	6	24	49	25	4	0	0	9	24	49	25	1	0	0	29	24	47	0	8	0	0
Adaboost	c_1	1988	0	0	12	111	19	0	1997	61	46	11	14	1	0	1988	0	0	12	111	19	0
	c_2	0	1909	0	0	8	9	0	176	1672	53	11	14	0	0	0	1909	0	0	8	9	0
	c_3	0	0	1526	0	1	5	4	225	135	1155	6	15	0	0	0	0	1526	0	1	5	4
	c_4	0	0	0	1490	238	444	0	3	0	2	1819	265	83	0	0	0	0	1490	238	444	0
	c_5	7	7	0	323	1782	275	0	10	2	1	257	2090	34	0	7	7	0	323	1782	275	0
	c_6	1	6	0	714	292	1393	0	0	0	0	21	11	2374	0	1	6	0	714	292	1393	0
	c_7	0	1	2	0	0	1	104	0	1	2	0	0	0	105	0	1	2	0	0	1	104

the highest CCR of 95%, while the highest CCR achieved by mRMR and MR are 93% and 81%. The MR algorithm in this experiment results in low CCR because it selected resultant sensors, which can be clearly seen in Table 4.8. To explain, all the sensors that MR selected use the sensing device $\ddot{\alpha}$ at consecutive scales $216, 224, \ldots 256$. Not only are the sensing devices redundant, but also consecutive scales are not expected to carry different information. The D*-Fuse algorithm lead to a higher CCR than mRMR+majority vote, mRMR+CaRT and mRMR+Adaboost because the D*-Fuse algorithm extracts sensor-class pair information.

Table 4.10: CCRs for Various Sensor Selection and Fusion Algorithms for the Human Activity Recognition Dataset

	Sensor Selection Criterion		
Fusion Method	**AR**	**mRMR**	**MR**
D*-Fuse	**95%**	-	-
Majority Vote	93%	93%	81%
CaRT	92%	91%	69%
Adaboost	80%	88%	80%

On a 64-bit Windows 7 operating system, with a Intel(R) Core(TM) i5-2400, 3.10GHz processor, with 16GB RAM, the execution time for the AR sensor selection is 35s while the D*-Fuse training took 9.2s. The testing execution time of the D*-Fuse is $2.45e-4$s.

4.5 Conclusions

In this theme, the D*-Fuse algorithm is developed. The D*-Fuse algorithm uses the novel AR criterion for sensor-class pair selection. The AR criterion is a novel information theoretic based sensor-class pair selection. Sensor-class pair selection is important for efficient sensor fusion because sensors typically differ in the accuracy they isolate classes. It might occur that a sensor has high overall CCR; however, it has low accuracy isolating a specific class. In contrast, another sensor might have low overall CCR although it is accurate in isolating a specific class. Therefore, an efficient sensor fusion algorithm must extract sensor-class pair information where each pair contains the class that can be isolated with highest CCR and the associated sensor.

Moreover, sensor-class pair selection avoids selecting redundant sensors. By sensor-class pair selection (instead of sensor-only selection), it is assured that selected sensors carry information about different classes. Hence, the selected sensors carry non-redundant classification information as discussed in details in Section 4.3. Non-redundant sensors are expected to lead to higher CCR when fused properly.

The D*-Fuse algorithm uses the AR algorithm at each level of the D*-Fuse Tree to select optimal sensor-class pairs in the training phase. Once a senor-class pair is selected, the class set is reduced by removing the selected class. At the next level, the AR algorithm is used again to select the next optimal sensor-class pair using the updated class set. The algorithm continues till all the classes are selected, which occurs in $n-1$ iterations where n is the number of classes.

The D*-Fuse algorithm is applied in this chapter on three data sets. The first data is generated using Gaussian distributions with different means and variances for different sensor-class pairs. In the second application, data are collected using Kinect camera for the objective of gesture phase segmentation. The third application is a human activity recognition dataset where the data are collected using smartphones mounted to the waists of the human subjects. The D*-Fuse results are compared with majority vote, CaRT and Adaboost fusion algorithms (described in Appendix C) applied on sensors selected using the AR, mRMR and MR sensor selection criteria. The D*-Fuse algorithm shows improvement over other fusion and sensor selection algorithms with reasonable execution time as discussed in Section 4.4.

For future work, three areas of improvement are considered:

- Extension of the D*-Fuse algorithm so that it allows for multiple splits at each level instead of the one-vs-the-rest trees that allows only for binary splits at each level.
- Extension of the D*-Fuse algorithm so that it allows for selecting multiple sensors at each level to solve the corresponding classification problems.
- Extension of the D*-Fuse algorithm to multiple fault diagnosis problems where multiple faults of different components can co-exist at the same time in the system.

CHAPTER 5

CONCLUSIONS

The purpose of this thesis is sensor selection and fusion for pattern recognition in complex interconnected systems. There are typically a large number of sensors in complex interconnected systems and sensor selection and fusion is challenging. For example, some of these sensors may misleading classification information. Some other sensors might carry contradicting classification information. In addition, one of these sensors might carry incomplete information such that it has to be used with another sensor as a group to lead to accurate classification information. In other cases, a subset of sensors might carry redundant classification information to each other such that using them as a group does not improve classification accuracy. This thesis tackles these challenges by 1) sensor selection and 2) fusion for classification.

5.1 Unsupervised Embedded Algorithm

To solve the above challenges, three themes are developed. The first theme presents a novel sensor selection algorithm called the Unsupervised Embedded algorithm. The algorithm aims to select sensors in which the inter-class distances are maximized and the intra-class distances are minimized. This is accomplished in two steps:

1. **Candidate List Sensor Selection**. In this step, a filter algorithm (e.g., mRMR algorithm) is used to select the candidate list of sensors.

2. **Entropy Based Sensor Ranking**. Here, the sensors in the candidate list are re-ranked after that the top m^* sensors are picked for classification. Re-ranking is accomplished in two steps:

   a) The sensor data is clustered using a clustering algorithm (e.g., Kmeans algorithm).

   b) The clusters are evaluated using the entropy, then the average entropy for each sensor (over the clusters) is calculated. Sensors are then ranked based on the average entropy where sensors with low entropy are given a higher rank and sent to the top of the list.

In comparison to existing sensor selection techniques, the Unsupervised Embedded algorithm is computationally efficient, leads to high CCR and does not rely on a specific classifier. For application, the Unsupervised Embedded algorithm is utilized to design a heat exchanger fouling diagnosis in the ECS of an aircraft. The ECS is an air conditioning system that is used to regulate the cabin air temperature, pressure and humidity in an aircraft. The ECS is a complex interconnected system with variety of components and heterogeneous sensors. A major component in the ECS is the heat exchanger, which exchanges heat with outside ram air. The heat exchanger is prone to a phenomenon that is known as fouling. Fouling degrades the ECS efficiency and may occur frequently and/or unexpectedly. This motivates the design of an automated fouling diagnosis algorithm, which allows for Condition Based Maintenance (CBM).

The fouling diagnosis algorithm that is presented in Chapter 2 is of two phases: training and testing phases. In the training phase, the data are generated using an experimentally validated Simulink model that is provided by our industry partner, where data include uncertainties in fouling levels, ambient temperatures and occupant counts. The Simulink model outputs time series data of 109 sensors. Noting that the ambient temperature changes data dramatically from a day type to another, 5 day types are defined based on the ambient temperatures; namely, i) extremely cold, ii) cold, iii) medium, iv) hot and v) extremely hot day types. The fouling diagnosis algorithm is applied for each day type separately.

For each day type; after data generation, the Unsupervised Embedded algorithm is used for sensor selection. Then PCA and GMM algorithms are used for feature extraction and k-NN algorithm is used for classification, and the results are summarized in Tabels 2.5, 2.4 and 2.6. For future

work, two improvements are considered for heat exchanger fouling diagnosis:

- Including more types (e.g., humidity level and altitude) of uncertainties in the analysis. Models that generate data with these uncertainties are being developed by our industry partner.
- Applying the heat exchanger fouling diagnosis on actual flight data as soon as the data is available from our industry partner.

5.2 BB-Fuse Algorithm

The Unsupervised Embedded algorithm is a computationally efficient and accurate sensor selection algorithm; however, it does not lead to a classification decision. Next, the optimal sensors have to be fused to produce a decision with high CCR. For reasons that might be due to selected sensors or the fusion algorithms themselves, fusion might achieve a CCR that is lower than the the CCR of the best provided sensor. Moreover, sensors may vary in the ability they isolate specific classes due to the heterogeneity of sensors in complex interconnected systems. A sensor might have high overall CCR (may be the best provided sensor); however, it has low performance isolating a specific class. In contrast, a sensor might have low overall CCR (may be the worst sensor); however, it is superior isolating a specific class. Hence, the BB-Fuse algorithm extracts sensor-class pair information by selecting the optimal classes (i.e., they can be isolated from the rest with highest accuracies) and the optimal sensors that can be used with them (i.e., the best sensors that can be used to isolate the optimal classes from the rest).

In the second theme, the BB-Fuse algorithm is developed. The BB-Fuse is a sensor-class pair selection and fusion algorithm which is analytically guaranteed to achieve CCR improvement over the best sensor. To emphasize, the BB-Fuse algorithm has two main contributions:

1. **Sensor-Class Pair Selection**, which harvests the ability of sensors to isolate different classes.
2. **Guaranteed CCR Improvement**, which is accomplished by constructing a one-vs-the-rest decision tree (BB-Fuse Tree) that uses the optimal sensor-class pairs at each level.

The BB-Fuse algorithm training phase takes the confusion matrices of the individual sensors at the input, and use them to output the BB-Fuse Tree. The BB-Fuse Tree is a one-vs-the-rest tree

that isolates one class at each level; therefore, it has $n-1$ levels (or less if node merging can be applied as explained in Def. 3.3.5) where n is the number of classes.

The BB-Fuse Tree is constructed in $n-1$ iterations. Each iteration is of 3 steps. First, the optimal sensor-class pair is selected. This is accomplished in three steps:

i) constructing binary confusion matrices,

ii) calculating binary CCRs and sensitivities from each confusion matrix,

iii) selecting the candidate list of sensors with sensitivities that are higher than the best sensor sensitivity, and finally

iv) selecting the optimal sensor-class pairs that maximizes the binary CCRs from the candidate list of sensors.

After the optimal sensor-class pair is selected, the second step is to reduce the confusion matrices by removing the rows and columns that correspond to the selected class. After that, the third step is to reduce the class set by removing the optimal class, then the next iteration is similarly performed. The BB-Fuse algorithm organizes the optimal sensor-class pairs in a one-vs-the-rest tree that is called the BB-Fuse Tree. This tree can be applied in the testing phase by sequentially applying the sensors in the BB-Fuse Tree. The testing phase terminates if a terminal node is reached; i.e., an optimal sensor decision on the testing data is equal to the corresponding optimal class that it can best isolate. Theoretically, the BB-Fuse algorithm is guaranteed to achieve CCR improvement as stated in Theorem 3.5.3.

The BB-Fuse algorithm is applied on two data sets in the second theme for numerical validation. The first application is a human activity recognition data set where hand interaction with environment is classified into 10 different classes. The results are summarized in Tables 3.2 and 3.4. In the second application, the BB-Fuse is applied to identify and authenticate users (10 users) from walking activities and the results are summarized in Tabels 3.7 and 3.8. The performance of BB-Fuse algorithm is compared to the performance of various other fusion techniques.

For future work, the BB-Fuse algorithm can be generalized for multiple-split decision trees where each node splits to a meta-class that is composed of one or more classes.

5.3 D*-Fuse Algorithm

The third theme presents the D*-Fuse algorithm which is an improvement of both the first and second themes as they both have limitations. In the first theme, the Unsupervised Embedded algorithm uses the mRMR criterion as a filter to select the candidate list of sensors. The mRMR algorithm has been criticized in literature for subtracting the redundancy term [120].

The second theme presents the BB-Fuse algorithm, which performs sensor-class pair selection and fusion to guarantee CCR improvement. The BB-Fuse algorithm utilizes the confusion matrices, which have two major limitations:

- **Computation Complexity**. The confusion matrices are computed using the cross-validation algorithm. The cross-validation algorithm is an iterative technique where in each iteration a specific classifier is trained and tested. The repetition of training and testing of the classifier makes the cross-validation algorithm computationally expensive; especially when there is a large number of sensors or classes.

- **Classifier Dependence**. Confusion matrices are computed for a specific classifier. Therefore, a different BB-Fuse Tree (possibly with higher CCR) results if different classifiers are used.

The D*-Fuse algorithm improves Themes 1 and 2 by using information theory. First, a novel information theoretic sensor-class pair selection criterion (called the AR criterion) is introduced. The AR criterion perform sensor-class pair selection instead of sensor-only selection because sensors vary in the accuracy they isolate different classes. The AR criterion assures that the selected sensors provide different classification information by sensor-class pair selection; hence, avoiding the use of the redundancy term used by the mRMR criterion. Because the AR criterion is classifier independent, the selected sensor-class pairs are also classifier independent.

After sensor-class pair selection using the AR criterion, the D*-Fuse algorithm organizes the pairs in the D*-Fuse Tree. The D*-Fuse Tree is one-vs-the-rest decision tree that is similar to the BB-Fuse Tree; however, the classifiers are not specified in the D*-Fuse Tree. Any classifier can be trained at each level of the D*-Fuse Tree on the corresponding binary classification problems. In contrast, the BB-Fuse Tree can only be used with specified classifiers.

The D*-Fuse algorithm is applied in Theme 3 on three datasets. The first dataset is a simulated data that is generated using Gaussian distributions with means and variances that vary for different sensor-class pairs, and the results are presented in Tabels 4.3 and 4.2(b). In the second application gesture phases are identified from video tapes and the results are presented in Tables 4.6 and 4.5(b). The third application utilizes a human activity recognition dataset and its results are presented in Tables 4.9 and 4.10. The D*-Fuse performance is compared to majority vote, CaRT and Adaboost fusion algorithms applied on sensors selected by AR, mRMR and MR criteria.

For future work, the following improvements to the D*-Fuse algorithm are considered:

- Allowing multiple splits at each level instead of the one-vs-the-rest trees
- Allowing for multiple sensor selection at each level.
- Extending to multiple fault diagnosis problems.

APPENDIX A

MATTERS RELEVANT TO INFORMATION THEORY

A.1 Maximum Entropy Distribution

Consider the optimization problem in Eq. (2.10). Using the method of Lagrange multipliers, define:

$$\mathcal{J} = H(X) - (\lambda - 1)\left(\sum_{j=1}^{r} p^j - 1\right) \tag{A.1}$$

where $\lambda \geq 0$ is a real number. Taking the partial derivative with respect to p^j, $\forall j = 1, ...r$, and equating to zero, we get

$$\frac{\partial \mathcal{J}}{\partial p^j} = -\ln p^j - 1 - \lambda + 1 = 0 \tag{A.2}$$

Therefore we get:

$$p^{j*} = e^{-\lambda} \tag{A.3}$$

Summing Eq. (A.3) for $j = 1, ...r$, we get:

$$\lambda = \ln r \tag{A.4}$$

Using Eq. (A.4) into Eq. (A.3), we get the maximum entropy distribution as

$$p^{j*} = \frac{1}{r} \quad \forall j = 1 \dots r \tag{A.5}$$

which is the uniform distribution.

A.2 Calculation of Mutual Information

This section describes the calculation of the mutual information quantities $I(S_i, C)$ and $I(S_i, S_j)$. Let the symbol alphabet be equal to $\Sigma = \{\alpha^1 \dots \alpha^r\}$ such that $\alpha^\omega \in \Sigma, \omega \in \{1, \dots r\}$. Let $\gamma \in \Gamma, |\Gamma| = 1024$ denote a particular parametric combination for a simulation run. Let $\boldsymbol{\sigma}_i(\gamma, \bullet) = [\sigma_i(\gamma, 1), \dots \sigma_i(\gamma, L)]$ denote the symbol sequence for sensor s_i and simulation run γ, where $\sigma_i(\gamma, t) \in \Sigma, \ \forall\ t = 1, \dots L$. Then, lets define a $|\Gamma| \times L$ matrix of all symbol sequences generated from the sensor s_i, $i = 1, \dots N$, under different simulation runs as:

$$\mathcal{G}_i = [\boldsymbol{\sigma}_i(\gamma, \bullet)]_{\gamma=1,\dots|\Gamma|} \tag{A.6}$$

Lets now associate a label $a_i(\gamma)$ to each row $\boldsymbol{\sigma}_i(\gamma, \bullet)$ of the matrix $\mathcal{G}_i$, such that $a_i(\gamma) = m$ if and only if $\boldsymbol{\sigma}_i(\gamma, \bullet)$ belongs to the class c_m, $m \in \{1 \dots M\}$. Now, lets define the following:

- $\mu_i(\alpha^\omega)$: the number of occurrences of the event when $\sigma_i(\gamma, t) = \alpha^\omega$ in the matrix $\mathcal{G}_i$.
- $\nu_i(\alpha^\omega, c_m)$: the number of occurrences of the joint event when $\sigma_i(\gamma, t) = \alpha^\omega$ and $a_i(\gamma) = m$ in the matrix $\mathcal{G}_i$.
- $\chi_{i,j}(\alpha^\omega, \alpha^\upsilon)$: the number of occurrences of the joint events $\sigma_i(\gamma, t) = \alpha^\omega$ and $\sigma_j(\gamma, t) = \alpha^\upsilon$.

Then the following entropies are calculated:

$$H(S_i) = -\sum_{\omega=1}^{r} \frac{\mu_i(\alpha^\omega)}{|\Gamma| \cdot L} \ln \frac{\mu_i(\alpha^\omega)}{|\Gamma| \cdot L} \tag{A.7}$$

$$H(S_i, C) = -\sum_{m=1}^{M} \sum_{\omega=1}^{r} \frac{\nu_i(\alpha^\omega, c_m)}{|\Gamma| \cdot L} \ln \frac{\nu_i(\alpha^\omega, c_m)}{|\Gamma| \cdot L} \tag{A.8}$$

$$H(S_i, S_j) = -\sum_{\omega=1}^{r} \sum_{v=1}^{r} \frac{\chi_{i,j}(\alpha^\omega, \alpha^v)}{|\Gamma|^2 \cdot L} \ln \frac{\chi_{i,j}(\alpha^\omega, \alpha^v)}{|\Gamma|^2 \cdot L} \tag{A.9}$$

The mutual information quantities required in Eq. (2.18) are then calculated using Eq. (2.8).

A.3 Fano's Inequality

Theorem A.3.1 (Fano's Inequality [119]) *Define random variables C and $\hat{C}$ whose sample space is the training sensor readings of sensor s (that is modeled by the random variable S). However, the output of C is the true class label and the output of $\hat{C}$ is the classifier decision. Let E be the classification error random variable such that*

$$E = \begin{cases} 1 & \textit{if } \hat{C} \neq C \\ 0 & \textit{otherwise.} \end{cases}$$

and define the probability of error $P_e = p(\hat{C} \neq C)$. Then,

$$H(E) + P_e \log |\mathscr{C}| \geq H(C|\hat{C}) \geq H(C|S) \tag{A.10}$$

This inequality can be weakened to

$$1 + P_e \log |\mathscr{C}| \geq H(C|S) \tag{A.11}$$

or

$$P_e \geq \frac{H(C|S) - 1}{\log |\mathscr{C}|} = \frac{-I(C;S) + H(C) - 1}{\log |\mathscr{C}|} \tag{A.12}$$

Proof:

$$H(E, C|\hat{C}) = H(C|\hat{C}) + H(E|C, \hat{C})$$

but the probability of making an error is zero given both C and $\hat{C}$; so, $H(E|C, \hat{C}) = 0$

However,

$$H(E, C|\hat{C}) = H(C|\hat{C}) + H(C|E, \hat{C}) \tag{A.13}$$

where it can be easily shown that $H(E|\hat{C}) \leq H(E)$. Now, $H(C|E, \hat{C})$ can be expressed as:

$$\begin{aligned} H(C|E, \hat{C}) &= P(E = 0)H(C|\hat{C}, E = 0) \\ &\quad + P(E = 1)H(C|\hat{C}, E = 1) \\ &\leq P_e \log |\mathscr{C}| \end{aligned} \tag{A.14}$$

Therefore,

$$H(E) + P_e \log |\mathscr{C}| \geq H(X|\hat{C})$$

where the $\log$ operator is of base 2 [119]. □

The Fano's Inequality shows that the lower bound of the probability of classification error is inversely proportional to the mutual information between the class lables and the sensor data. Therefore, various research papers proposed to maximize the mutual information for sensor selection [120].

APPENDIX B

MACHINE LEARNING ALGORITHMS

Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are used in this thesis for feature extraction. PCA is a feature extraction method that projects the data into the direction on which data change is maximum. LDA; on the other hand, is a feature extraction method that projects the data on the direction that different classes are the most separable.

The k-Nearest Neighbor (k-NN) algorithm is used in this thesis for classification. The k-NN algorithm classifies a testing point by performing majority voting on the nearest k neighbors, where k is a tuning parameter. A feature extraction method and a k-NN classifier is applied on each sensor; after that, the decisions are fused using a fusion method be it BB-Fuse or other fusion methods discussed in the thesis. This appendix explains the PCA, LDA and k-NN algorithms.

For mathematical notation, let $\mathcal{P}$ be the classification problem under consideration with class set $\mathscr{C} = \{c_1, c_2, \ldots c_n\}$ where the data is collected using the sensor set $\mathcal{S} = \{s_1, s_2, \ldots s_m\}$. Let the input training data be $X = \{(x_{t,j}, y_t) \; \forall \; t = 1, 2, \ldots T, j = 1, 2, \ldots m\}$, where $x_{t,j} \in \mathcal{R}$ is the training sample t of the sensor s_j and $y_t \in \mathscr{C}$ is the associated label. Furthermore, define the *data matrix* to be $\mathbf{X} \in \mathcal{R}^{T \times m}$ where $x_{t,j}$ is the element at the t^{th} row and j^{th} column $\forall \; t = 1, 2, \ldots T$ and $j = 1, 2, \ldots m$.

B.1 Principal Component Analysis (PCA)

The Principal Component Analysis (PCA) [124] is a data reduction method. The objective of PCA is to transform the data matrix $\mathbf{X}$ into a matrix $\mathbf{X}'$ of size $T \times m'$, where $m' < m$. The columns of $\mathbf{X}'$ hold the *Score Vectors* (also known as the *Principal Components*). This transformation is accomplished using the *Karhunen-Loéve* (KL) algorithm as summarized here. First, the $m \times m$ covariance matrix Q of $\mathbf{X}$ is computed and the corresponding eigenvalues are obtained and sorted in descending order. Second, the eigenvectors associated with the m' largest eigenvalues are generated and arranged into an $m \times m'$ transformation matrix $\mathbf{\Omega}$. Finally, the scores of $\mathbf{X}'$ are computed using the following linear transformation

$$\mathbf{X}' = \mathbf{X} \times \mathbf{\Omega} \tag{B.1}$$

In case $\mathbf{X}$ is one dimensional time series data vector (i.e., $\mathbf{X}$ is of size $T \times m$ and $m = 1$), the data vector $\mathbf{X}$ is first reshaped to a matrix $\tilde{\mathbf{X}}$, which is of size $m_1 \times m_2$ where $T = m_1 \times m_2$. After that the KL algorithm is applied to extract the Score Vectors $\mathbf{X}'$ as described above, where the scores $\mathbf{X}'$ is of size $T \times m'$ such that $m' \leq m_2$.

B.2 Linear Discriminant Analysis

Linear Discriminant Analysis(LDA) [125] is a feature extraction method that converts the data matrix $\mathbf{X}$ into features $\mathbf{X}'$, where $\mathbf{X}' \in \mathcal{R}^{T \times m'}$, $m' \leq m$ and the classes are separable. Let $\mathbf{\Omega} \in \mathcal{R}^{m \times m'}$ be a liner transformation matrix such that $\mathbf{X}' = \mathbf{X} \times \mathbf{\Omega}$. Let the columns of $\mathbf{\Omega}$ be $\omega_1, \omega_2, \ldots \omega_{m'}$, then the optimal transformation matrix $\mathbf{\Omega}^*$ can be found by solving the following optimization problem

$$\begin{aligned} \mathbf{\Omega}^* &= \arg \max_{\mathbf{\Omega} \in \mathcal{R}^{m \times m'}} \text{tr}\left(\frac{\mathbf{\Omega}^t \mathbf{Q}_b \mathbf{\Omega}}{\mathbf{\Omega}^t \mathbf{Q}_w \mathbf{\Omega}}\right) \\ \text{s.t. } ||\omega_i|| &= 1 \; \forall i = 1, 2, \ldots m' \end{aligned} \tag{B.2}$$

where $(\bullet)^t$ denotes the transpose operator, tr$(\bullet)$ denotes the trace operator and

$$\begin{aligned} \mathbf{Q}_b &= \sum_{i=1}^{n} p_i(\mu_i - \mu)(\mu_i - \mu)^t \text{ and} \\ \mathbf{Q}_w &= \sum_{i=1}^{n} p_i Q_i \end{aligned} \tag{B.3}$$

are respectively the between-class covariance and the within-class covariance. $p_i \; \forall i = 1, 2, \ldots n$ is the prior probability of the classes $c_i \in \mathscr{C}$. The vectors $\mu_i \in \mathcal{R}^{m \times 1} \; \forall \; i = 1, 2, \ldots n$ is calculated from the subset of data that belong to class c_i, which is $\mathbf{X}^i = \{\Theta : \Theta \in \mathbf{X} \; \& \; y_t = c_i \; \forall \; t = 1, 2, \ldots |\mathbf{X}^i|\}$, and $\mu = \sum_{i=1}^{n} p_i \mu_i \in \mathcal{R}^{m \times 1}$ is the sample mean vector of the columns $\mathbf{X}$. $\mathbf{Q}_i$ is the covariance the data subset $X_i \; \forall i = 1, 2, \ldots n$. The optimal solution $\mathbf{\Omega}^* = [\omega_1^*, \omega_2^*, \ldots \omega_{m'}^*]$ is composed of the eigenvectors that correspond to the largest eigenvalues of the symmetric matrix $\mathbf{Q}_w^{-1}\mathbf{Q}_b$.

Steps to find the transformation matrix $\mathbf{\Omega}^*$ are similar to the KL algorithm, except that the eigenvectors are calculated from the matrix $\mathbf{Q}_w^{-1}\mathbf{Q}_b$. The "reshaping trick" as described in the PCA section can also be used to extract LDA features from one dimensional time series data ($m = 1$).

B.3 k-Nearest Neighbor Classification Algorithm

Due to simplicity, efficiency and low complexity, the k-Nearest Neighbor (k-NN) [124] classification algorithm is applied in this thesis on the labeled feature set $\{(\mathbf{x}'_t, y_t) \forall t = 1, 2, \ldots T\}$, where the vector $\mathbf{x}'_t \in \mathcal{R}^{1 \times m'}$ is a row in the feature matrix $\mathbf{X}'$ obtained by PCA, GMM or LDA. The k-NN classifier is composed of two phases: training and testing. The purpose of the training phase is to specify the parameter k (where k represents the number of nearest neighbors to observe in the testing phase) which maximizes the CCR. In the testing phase, the unlabeled testing feature, say $\mathbf{x} \in \mathcal{R}^{1 \times m'}$, is classified by applying the majority rule in the k-neighborhood of the testing feature using the equation:

$$\hat{c} = \arg \max_{i=1 \ldots n} \frac{k_i}{k} \tag{B.4}$$

where $\hat{c}$ is the classifier output and k_i is the number of training features that belong to class $c_i \in \mathscr{C}$ among the k nearest neighbors of the testing feature $\mathbf{x}$ in the set $\{\mathbf{x}'_t \; \forall \; t = 1, 2, \ldots T\}$.

APPENDIX C

FUSION ALGORITHMS

In literature, several fusion algorithms for classification have been proposed and frequently used for improving the classification decisions. In this work, the BB-Fuse algorithm performance is compared with 7 fusion methods which are *Classification and Regression Tree* (CaRT), *C4.5*, *Bayes Belief Integration*, *Majority Voting*, *Adaptive Boosting* (Adaboost), Borda Count Voting and *Condorcet Count Voting*. These algorithms are explained in brevity in this appendix. Due to similarity, CaRT and C4.5 are explained together in the next subsection.

C.1 CaRT and C4.5

Both CaRT and C4.5 are *data level decision tree algorithms* and they construct the decision tree using the *Hunt's algorithm* [124]. The Hunt's algorithm recursively splits the input data at each node into two partitions till it reaches a leaf node where all the data are of the same class.

At a node v_ℓ, let $\mathbf{X}^\ell \subseteq \mathbf{X}$ be the input data. To construct the decision tree, the Hunt's algorithm performs two steps at the node v_ℓ:

1. Evaluate the input data X^ℓ to the node. If all the input data belong to one class, the algorithm terminates. Otherwise, step 2 is performed.

2. Find the *optimal test* that minimizes the impurity measure J, such that:

$$(s^{\ell}, \theta^{\ell}) = \arg \min_{s_j \in \mathcal{S}, \theta_j} J(s_j, \theta_j) \tag{C.1}$$

where $s_j \in \mathcal{S}$ is a sensor and θ_j is a threshold on sensor s_j, and (s_j, θ_j) comprises a "test". Based on the test θ_j, the input data $\mathbf{X}^{\ell}$ can be split into two partitions $\mathbf{X}^{\ell,L} = \{(x, y) : (x, y) \in \mathbf{X}^{\ell} \,\&\, x_j \leq \theta_j\}$ and $\mathbf{X}^{\ell,R} = X \setminus \mathbf{X}^{\ell,L}$. Once the optimal test is found, the node v_{ℓ} is established, then the same steps are repeated on its children. The Hunt's algorithm tends to grow large trees; therefore, an optional *pruning* step is performed. Pruning is the act of replacing sub-trees with a leaf node based on some criterion (e.g., classification error). Although both CaRT and C4.5 algorithms uses the Hunt's method, they differ in the impurity measure and pruning methods they use.

In particular, Classification and Regression Tree (CaRT) algorithm utilizes the Gini Index $J = J_{GI}$ for the impurity measure, which is defined as:

$$\begin{aligned} J_{GI} = \frac{|\mathbf{X}^{\ell,L}|}{T}(1 - \sum_{i=1}^{n}[p(c_i|\mathbf{X}^{\ell,L})]^2 \\ + \frac{|\mathbf{X}^{\ell,R}|}{T}(1 - \sum_{i=1}^{n}[p(c_i|\mathbf{X}^{\ell,R})]^2 \end{aligned} \tag{C.2}$$

where $p(c_i|\mathbf{X}^{\ell,R})$ and $p(c_i|\mathbf{X}^{\ell,R})$ are the probabilities that class c_i occurs in the sets $\mathbf{X}^{\ell,R}$ and $\mathbf{X}^{\ell,L}$ respectively.

However, C4.5 uses the *Split Information* J_{SI} for impurity measure [113], which is explicitly defined as:

$$J_{SI} = \frac{|X^{\ell,L}|}{T} H(C|X^{\ell,L}) + \frac{|X^{\ell,R}|}{T} H(C|X^{\ell,R}) \tag{C.3}$$

where $H(C|X^{\ell,L})$ and $H(C|X^{\ell,R})$ are the conditional entropy of the classes given the sets $X^{\ell,L}$ and $X^{\ell,R}$, respectively. The conditional entropy is defined as:

$$H(C|X) = -\sum_{i=1}^{n} p(c_i|X) \log p(c_i|X) \tag{C.4}$$

CaRT uses cross-validation for pruning while C4.5 algorithm prunes the trees using the *pessimistic pruning* algorithm [113]. In comparison to CaRT, C4.5 has several advantages (e.g., handling missing data). On the other hand, there is not theoretical evidence that one of the two algorithms is better than the other.

Both of CaRT and C4.5 algorithms are popular and can achieve high CCRs on some data sets. However, both of them are sensitive to noise and may construct very complex trees for very simple data sets. CaRT and C4.5 are not guaranteed to outperform the best sensor.

C.2 Bayes Belief Integration

Bayes Belief Integration is a *Bayesian Fusion* method. In this method, Bayes theorem is used to calculate a belief measure, Bel(c_i) $\forall\, i = 1, 2, \ldots n$ and the class that has the maximum belief wins. Let h_j be the classifier that is trained on the data of sensor s_j $\forall\, j = 1, 2, \ldots m$ in the training phase. Given a testing point in the testing phase, let the classification decisions obtained from the sensors be $\hat{c}^j \in \mathscr{C}$ $\forall\, j = 1, 2, \ldots m$. The belief measure Bel$(c_i)$ is defined by the equation:

$$\text{Bel}(c_i) = p(c_i) \frac{\prod_{j=1}^{m} p(\hat{c}^j \mid c_i)}{\prod_{j=1}^{m} p(\hat{c}^j)} \tag{C.5}$$

where $p(c_i)$ is the probability of class c_i, $p(\hat{c}^j \mid c_i)$ is the probability that the classifier h_j output is $\hat{c}^j \in \mathscr{C}$ given that the actual class of a testing point is c_i. Both $p(c_i)$ and $p(\hat{c}^j|c_i)$ can be calculated using the confusion matrices of the classifiers h_j $\forall\, j = 1, 2 \ldots m$. To be specific, let the confusion matrix for a specific classifier h_j be:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

where each element $a_{i,\ell}$ $\forall\, i = 1, 2, \ldots n, \ell = 1, 2, \ldots n$ represent the number of testing points that are classified as class c_i while they are actually of class c_ℓ. Then, the variables in Eq. C.5 can be

calculated as:

$$p(\hat{c}^j = c_\ell \mid c_i) = \frac{a_{i,\ell}}{\sum_{\ell=1}^{n} a_{i,\ell}} \tag{C.6}$$

$$p(\hat{c}^j = c_\ell) = \frac{\sum_{i=1}^{n} a_{i,\ell}}{\sum_{i=1}^{n}\sum_{\ell=1}^{n} a_{i,\ell}} \tag{C.7}$$

and the final decision is obtained using the equation:

$$\hat{c} = \arg \max_{i=1,2,\ldots n} \text{Bel}(c_i) \tag{C.8}$$

Despite its effectiveness, the Bayes Belief Integration method is computationally expensive because it requires the decisions of all the classifiers $h_j \; \forall \; j = 1, 2, \ldots m$ to make the final decision. There is not theoretical evidence that Bayes Belief Integration outperforms the best sensor in CCR.

C.3 Majority Voting

The voting methods are called so because of their similarity with the voting process in elections. In this context, the voters are the classifiers $h_j \; \forall \; j = 1, 2, \ldots m$ while the candidates are the classes in the set $\mathscr{C} = \{c_i \; \forall \; i = 1, 2, \ldots n\}$. Each classifier h_j casts a vote $\hat{c}^j \in \mathscr{C}$, and the objective is to fuse the decisions $\hat{c}^j \; \forall j = 1, 2, \ldots m$ to reach a final decision $\hat{c}$, which is more likely to be the correct class of the testing point under consideration.

The simplest form of voting algorithms is the Majority Voting method, where the class that has the maximum number of votes wins. The Majority Voting algorithm subdivides from the category of *Un-Weighted Voting* algorithms where all the classifiers' decisions $\hat{c}^j \; \forall \; j = 1, 2, \ldots m$ are assumed to have equal weights. Mathematically, the Majority Voting algorithm is explained in Eq. C.9.

$$\hat{c} = \arg \max_{i=1,2,\ldots n} p(\hat{c}^j = c_i) = \frac{\sum_{i=1}^{m} e(\hat{c}^j, c_i)}{m} \tag{C.9}$$

where $e(\hat{c}^j, c_i) = 1$ when $\hat{c}^j = c_i$ and 0 otherwise [92].

C.4 Adaptive Boosting

Adaptive Boosting (Adaboost) [105] is a popular iterative fusion method that combines sensors' decisions using optimal weights that minimizes the *pseudo-loss*. Adaboost is originally designed for binary classification problems (i.e., problems with 2 classes); however, several extensions to multi-class classification problems. The algorithm used in this thesis and explained here is called the Adaboost.M1 and is explained next.

The classifiers used with Adaboost in this thesis are modified so that they can accept distribution over the training samples in the training phase, but no distribution is used in the testing phase. Initializing the training data distribution $p_t = 1/T \; \forall \; t = 1, 2, \ldots T$, the following steps are applied in iteration ℓ of the Adaboost algorithm:

- **Step 1:**
  - Train a classifier h^ℓ using sensor data x_t^ℓ and distribution $p_t \; \forall \; t = 1, 2, \ldots T$
  - Obtain $\hat{y}_t^\ell = h^\ell(x_t^\ell, p_t)$
  - Compute the error $\epsilon^\ell = \sum_{t=1}^{T} p_t \cdot e(\hat{y}_t^\ell, y_t)$
    where $e(\hat{y}_t^\ell, y_t) = 1$ if $\hat{y}_t^\ell = y_t$ and 0 otherwise.
- **Step 2:** Set $\beta^\ell = \frac{\epsilon^\ell}{1-\epsilon^\ell}$
- **Step 3:** Update

$$p_t \leftarrow \frac{p_t}{N.F.} \cdot \begin{cases} \beta^\ell & \text{if } \hat{y}_t^\ell = y_t \\ 1 & \text{otherwise} \end{cases}$$

  where $N.F.$ is the appropriate normalization factor.

The iterations are repeated for a pre-specified number of iterations L. Finally, the training phase outputs are:

- The sensors and the trained classifiers $s^\ell, h^\ell \; \forall \; \ell = 1, 2, \ldots L$

- The associated parameters $\beta^\ell \; \forall \; \ell = 1, 2, \ldots L$

The final decision in the testing phase is obtained using *weighted majority vote*, where the weight for each classifier h^ℓ is $\log(1/\beta^\ell)$. Mathematically, the final decision $\hat{c}$ for any testing sample is obtained using the equation:

$$\hat{c} = \arg\max_{c_i \in \mathscr{C}} \sum_{\ell=1}^{L} \log \frac{1}{\beta^\ell} \cdot e(\hat{c}^\ell, c_i) \tag{C.10}$$

where $e(\hat{c}^\ell, c_i) = 1$ if $\hat{c}^\ell = c_i$ and 0 otherwise, and $\hat{c}^\ell$ is the decision obtained from classifier h^ℓ.

Adaboost is an effective and popular fusion algorithm; however, it is not guaranteed to perform better than the best provided sensor.

C.5 Borda Count Voting

Borda Count Voting method is a subcategory of *Ranked Voting* methods, where each classifier ranks the classes based on some preferences. In Borda Count Voting method training phase, each classifier $h_j \; \forall \; j = 1, 2, \ldots m$ ranks the classes based on classification accuracy; after that, a function (called the Borda Count) $\text{BC}(c_i, h_j)$ is defined to be the number of classes that ranks below the class $c_i \in \mathscr{C}$ by classifier h_j. To illustrate, let the ranked classes by a classifier h_0 to be $c_{(i)} \in \mathscr{C} \; \forall \; i = 1, 2, \ldots n$; then, $\text{BC}(c_{(1)}, h_0) = n - 1$, $\text{BC}(c_{(2)}, h_o) = n - 2$, ... and $\text{BC}(c_{(n)}, h_0) = 0$.

In the testing phase, each classifier casts a vote $\hat{c}^j \in \mathscr{C} \; \forall \; i = 1, 2, \ldots m$. Next, the Borda Count of each class $\hat{c}^j$ is calculated, and the class that achieves the maximum Borda Count wins. Mathematically,

$$\hat{c} = \arg \max_{i=1,2,\ldots n} \sum_{j=1}^{m} e(\hat{c}^j, c_i) \cdot \text{BC}(c_i, h_j) \tag{C.11}$$

where $e(\hat{c}^j, c_i) = 1$ if $\hat{c}^j = c_i$ and 0 otherwise. Despite its simplicity, Borda Count Voting method ignores the possible existence of several erratic classifiers that could reinforce the errors [101].

C.6 Condorcet Count Voting

An improvement over the Majority Voting method is the *Condorcet Count Voting* method. In this method, the classifiers $h_j \; \forall \; j = 1, 2, \ldots m$ vote for a pair of classes, say $(c_i, c_\ell) \in \mathscr{C} \times \mathscr{C}, \; \forall \; i = 1, 2, \ldots n, \ell = 1, 2, \ldots i-1$. The winning class for this pairwise election scores a point, then the class that scores the highest number of points wins. To be specific, denote the winner of the two classes in the pair (c_i, c_ℓ) by $\hat{c}^{i,\ell} \in \mathscr{C}$. Define the function $e(\hat{c}^{i,\ell}, c_r)$ so that $e(\hat{c}^{i,\ell}, c_r) = 1$ if $c_r = \hat{c}^{i,\ell}$ and 0 otherwise. Then the Condorcet Count is described by Eq. C.12.

$$\hat{c} = \arg \max_{c_r \; \forall \; r=1,2,\ldots n} \sum_{i=1}^{n} \sum_{\ell=1}^{i-1} e(\hat{c}^{i,\ell}, c_r) \tag{C.12}$$

where $\hat{c}$ is the final decision. Condorcet Count Voting method decreases the chances of winning of a minority candidate. However, it is compuatationally expensive because of the pairwise elections [92].

BIBLIOGRAPHY

[1] S. Kumar, J. Ghosh, and M. Crawford, "Hierarchical fusion of multiple classifiers for hyperspectral data analysis," *Pattern Analysis & Applications*, vol. 5, no. 2, pp. 210–220, 2002.

[2] **N. Najjar** and S. Gupta, "Better-than-the-Best Fusion algorithm with application in human activity recognition," in *Proceedings of SPIE Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2015*, SPIE, 2015, pp. 949805-1 – 949805-10.

[3] R. Marshall, "The use of Classification and Regression Trees in clinical epidemiology," *Journal of Clinical Epidemiology*, vol. 54, no. 6, pp. 603–609, 2001.

[4] A. Ross and A. Jain, "Information fusion in biometrics," *Pattern Recognition Letters*, vol. 24, no. 13, pp. 2115–2125, 2003.

[5] Y. Ding and A. Ross, "A comparison of imputation methods for handling missing scores in biometric fusion," *Pattern Recognition*, vol. 45, no. 3, pp. 919–933, 2012.

[6] S. Ribaric and I. Fratric, "A biometric identification system based on eigenpalm and eigenfinger features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1698–1709, 2005.

[7] N. Poh and J. Kittler, "A unified framework for biometric expert fusion incorporating quality measures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 3–18, 2012.

[8] V. Dasigi, R. Mann, and V. Protopopescu, "Information fusion for text classification —- an experimental comparison," *Pattern Recognition*, vol. 34, no. 12, pp. 2413–2425, 2001.

[9] A. Temko, D. Macho, and C. Nadeu, "Fuzzy integral based information fusion for classification of highly confusable non-speech sounds," *Pattern Recognition*, vol. 41, no. 5, pp. 1814–1823, 2008.

[10] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[11] C. Sakar, O. Kursun, and F. Gurgen, "A feature selection method based on kernel canonical correlation analysis and the minimum Redundancy–Maximum Relevance filter method," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3432–3437, 2012.

[12] T. Chow and D. Huang, "Estimating optimal feature subsets using efficient estimation of high-dimensional mutual information," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 213–224, 2005.

[13] R. Kohavi and G. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1–2, pp. 273–324, 1997.

[14] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, no. 3, pp. 131–156, 1997.

[15] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 537–550, 1994.

[16] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[17] M. Dash and H. Liu, "Consistency-based search in feature selection," *Artificial Intelligence*, vol. 151, no. 1–2, pp. 155–176, 2003.

[18] **N. Najjar**, S. Gupta, J. Hare, S. Kandil, and R. Walthall, "Optimal sensor selection and fusion for heat exchanger fouling diagnosis in aerospace systems," *IEEE Sensors Journal*, vol. 16, no. 12, pp. 4866–4881, 2016.

[19] J. Hare, S. Gupta, **N. Najjar**, P. D'Orlando, and R. Walthall, "System-level fault diagnosis with application to the environmental control system of an aircraft," in *Proceedings of SAE 2015 AeroTech Congress and Exhibition- Technical Paper 2015-01-2583*, SAE International, 2015.

[20] T. Pogiatzis, D. Wilson, and V. Vassiliadis, "Scheduling the cleaning actions for a fouled heat exchanger subject to ageing: MINLP formulation," *Computers & Chemical Engineering*, vol. 39, pp. 179–185, 2012.

[21] H. Han, B. Gu, T. Wang, and Z. Li, "Important sensors for chiller fault detection and diagnosis from the perspective of feature selection and machine learning," *International Journal of Refrigeration*, vol. 34, no. 2, pp. 586–599, 2011.

[22] S. Namburu, M. Azam, J. Luo, K. Choi, and K. Pattipati, "Data-driven modeling, fault diagnosis and optimal sensor selection for HVAC chillers," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 3, pp. 469–473, 2007.

[23] S. Jiang, R. Kumar, and H. Garcia, "Optimal sensor selection for discrete-event systems with partial observation," *IEEE Transactions on Automatic Control*, vol. 48, no. 3, pp. 369–381, 2003.

[24] V. Gupta, T. H. Chung, B. Hassibi, and R. Murray, "On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage," *Automatica*, vol. 42, no. 2, pp. 251–260, 2006.

[25] S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 451–462, 2009.

[26] A. Hero and D. Cochran, "Sensor management: Past, present, and future," *IEEE Sensors Journal*, vol. 11, no. 12, pp. 3064–3075, 2011.

[27] J. Xu, Y. Wang, and L. Xu, "PHM-oriented sensor optimization selection based on multiobjective model for aircraft engines," *IEEE Sensors Journal*, vol. 15, no. 9, pp. 4836–4844, 2015.

[28] X. Shen, S. Liu, and P. Varshney, "Sensor selection for nonlinear systems in large sensor networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 4, pp. 2664–2678, 2014.

[29] C. Lo, P. Chan, Y.-K. Wong, A. Rad, and K. Cheung, "Fuzzy-genetic algorithm for automatic fault detection in HVAC systems," *Applied Soft Computing*, vol. 7, no. 2, pp. 554–560, 2007.

[30] K. Choi, S. Namburu, M. Azam, J. Luo, K. Pattipati, and A. Patterson-Hine, "Fault diagnosis in HVAC chillers," *IEEE Instrumentation Measurement Magazine*, vol. 8, no. 3, pp. 24–32, 2005.

[31] Q. Zhou, S. Wang, and Z. Ma, "A model-based fault detection and diagnosis strategy for HVAC systems," *International Journal of Energy Research*, vol. 33, no. 10, pp. 903–918, 2009.

[32] S. Katipamula and M. Brambley, "Review article: Methods for fault detection, diagnostics, and prognostics for building systems - a review, part I," *HVAC & R Research*, vol. 11, no. 1, pp. 3–25, 2005.

[33] ——, "Review article: Methods for fault detection, diagnostics, and prognostics for building systems - a review, part II," *HVAC & R Research*, vol. 11, no. 2, pp. 169–187, 2005.

[34] R. Buswell and J. Wright, "Uncertainty in model-based condition monitoring," *Building Services Engineering Research and Technology*, vol. 25, no. 1, pp. 65–75, 2004.

[35] R. Kelso and J. Wright, "Application of fault detection and diagnosis techniques to automated functional testing," *ASHRAE Transactions*, vol. 111, no. 1, pp. 964–970, 2005.

[36] H. Yang, S. Cho, C.-S. Tae, and M. Zaheeruddin, "Sequential rule based algorithms for temperature sensor fault detection in Air Handling Units," *Energy Conversion and Management*, vol. 49, no. 8, pp. 2291–2306, 2008.

[37] S. Wang and F. Xiao, "AHU sensor fault diagnosis using Principal Component Analysis method," *Energy and Buildings*, vol. 36, no. 2, pp. 147–160, 2004.

[38] J. Pakanen and T. Sundquist, "Automation-assisted fault detection of an Air-Handling Unit; implementing the method in a real building," *Energy and Buildings*, vol. 35, no. 2, pp. 193–202, 2003.

[39] J. Qin and S. Wang, "A fault detection and diagnosis strategy of VAV air-conditioning systems for improved energy and control performances," *Energy and Buildings*, vol. 37, no. 10, pp. 1035–1048, 2005.

[40] T. Rossi and J. Braun, "A statistical, rule-based fault detection and diagnostic method for Vapor Compression Air Conditioners," *HVAC & R Research*, vol. 3, no. 1, pp. 19–37, 1997.

[41] Y. Zhao, S. Wang, and F. Xiao, "A statistical fault detection and diagnosis method for centrifugal chillers based on exponentially-weighted moving average control charts and Support Vector Regression," *Applied Thermal Engineering*, vol. 51, no. 1–2, pp. 560–572, 2013.

[42] **N. Najjar**, C. Sankavaram, J. Hare, S. Gupta, K. Pattipati, R. Walthal, and P. D'Orlando, "Health assessment of liquid cooling system in aircrafts: Data visualization, reduction, clustering and classification," *SAE International Journal of Aerospace*, vol. 5, no. 1, pp. 119–127, 2012.

[43] L. Shang and G. Liu, "Sensor and actuator fault detection and isolation for a high performance aircraft engine bleed air temperature control system," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 5, pp. 1260–1268, 2011.

[44] D. Gorinevsky, K. Dittmar, D. Mylaraswamy, and E. Nwadiogbu, "Model-based diagnostics for an aircraft auxiliary power unit," in *Proceedings of the 2002 International Conference on Control Applications*, vol. 1, 2002, 215–220 vol.1.

[45] R. Isermann, "Model-based fault-detection and diagnosis – status and applications," *Annual Reviews in Control*, vol. 29, no. 1, pp. 71–85, 2005.

[46] S. Lingfang, Z. Yingying, Z. Xinpeng, Y. Shanrang, and Q. Yukun, "Research on the fouling prediction of heat exchanger based on Support Vector Machine," in *International Conference on Intelligent Computation Technology and Automation (CICTA)*, vol. 1, 2008, pp. 240–244.

[47] S. Lingfang, Z. Yingying, and S. Rina, "Fouling prediction of heat exchanger based on genetic optimal SVM algorithm," in *Third International Conference on Genetic and Evolutionary Computing*, 2009, pp. 112–116.

[48] **N. Najjar**, J. Hare, P. D'Orlando, G. Leaper, K. Pattipati, A. Silva, S. Gupta, and R. Walthall, "Heat exchanger fouling diagnosis for an aircraft air-conditioning system," in *Proceedings of SAE 2013 AeroTech Congress and Exibition- Technical Paper 2013-01-2250*, SAE International, 2013.

[49] H. Kaneko, S. Inasawa, N. Morimoto, M. Nakamura, H. Inokuchi, Y. Yamaguchi, and K. Funatsu, "Statistical approach to constructing predictive models for thermal resistance based on operating conditions," *Industrial and Engineering Chemistry Research*, vol. 51, no. 29, pp. 9906–9912, 2012.

[50] L. Shang and G. Liu, "Heat exchanger fouling detection in a simulated aircraft engine bleed air temperature control system," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2010, pp. 774–778.

[51] C. Riverol and V. Napolitano, "Estimation of fouling in a plate heat exchanger through the application of neural networks," *Journal of Chemical Technology and Biotechnology*, vol. 80, no. 5, pp. 594–600, 2005.

[52] R. Garcia, "Improving heat exchanger supervision using neural networks and rule based techniques," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3012–3021, 2012.

[53] A. Adili, N. Hasni, C. Kerkeni, and S. Ben Nasrallah, "An inverse problem based on genetic algorithm to estimate thermophysical properties of fouling," *International Journal of Thermal Sciences*, vol. 49, no. 6, pp. 889–900, 2010.

[54] I. Perez-Grande and T. Leo, "Optimization of a commercial aircraft Environmental Control System," *Applied Thermal Engineering*, vol. 22, no. 17, pp. 1885–1904, 2002.

[55] H. Zhao, Y. Hou, Y. Zhu, L. Chen, and S. Chen, "Experimental study on the performance of an aircraft Environmental Control System," *Applied Thermal Engineering*, vol. 29, no. 16, pp. 3284–3288, 2009.

[56] J. Vargas and A. Bejan, "Thermodynamic optimization of finned crossflow heat exchangers for aircraft Environmental Control Systems," *International Journal of Heat and Fluid Flow*, vol. 22, no. 6, pp. 657–665, 2001.

[57] A. Alebrahim and A. Bejan, "Thermodynamic optimization of heat-transfer equipment configuration in an Environmental Control System," *International Journal of Energy Research*, vol. 25, no. 13, pp. 1127–1150, 2001.

[58] S. Wright, G. Andrews, and H. Sabir, "A review of heat exchanger fouling in the context of aircraft air-conditioning systems, and the potential for electrostatic filtering," *Applied Thermal Engineering*, vol. 29, no. 13, pp. 2596–2609, 2009.

[59] H. Peng and X. Ling, "Optimal design approach for the plate-fin heat exchangers using neural networks cooperated with genetic algorithms," *Applied Thermal Engineering*, vol. 28, no. 5–6, pp. 642–650, 2008.

[60] M. Mishra, P. Das, and S. Sarangi, "Second law based optimisation of crossflow plate-fin heat exchanger design using genetic algorithm," *Applied Thermal Engineering*, vol. 29, no. 14–15, pp. 2983–2989, 2009.

[61] T. Kuppan, *Heat exchanger design handbook*, 2nd ed. CRC PressI Llc, 2013.

[62] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[63] S. Gupta and A. Ray, "Statitical mechanics of complex systems for pattern identification," *Journal of Statistical Physics*, vol. 134, no. 2, pp. 337–364, 2009.

[64] D. Lind and M. Marcus, *An introduction to symbolic dynamics and coding*. Cambridge University Press, United Kingdom, 1995.

[65] E. T. Jaynes, "Information theory and statistical mechanics," *Physical Review*, vol. 106, no. 4, p. 620, 1957.

[66] D. Huang and T. Chow, "Effective feature selection scheme using mutual information," *Neurocomputing*, vol. 63, pp. 325–343, 2005.

[67] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained K-means clustering with background knowledge," in *ICML*, vol. 1, 2001, pp. 577–584.

[68] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[69] A. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[70] S. Mallat, *A wavelet tour of signal processing: The sparse way*, 3rd ed. Elsevier Science, 2008.

[71] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[72] J. Dai and J. Zhou, "Multifeature-based high-resolution palmprint recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 945–957, 2011.

[73] J. Nunez, X. Otazu, O. Fors, A. Prades, V. ˜Pala, and R. Arbiol, "Multiresolution-based image fusion with additive wavelet decomposition," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, pp. 1204–1211, 1999.

[74] Y. Barshalom, P. Willett, and X. Tian, *Tracking and data fusion: A handbook of algorithms*. YBS Publishing, 2011.

[75] D. Smith and S. Singh, "Approaches to multisensor data fusion in target tracking: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 12, pp. 1696–1710, 2006.

[76] A. Noulas, G. Englebienne, and B. Kröse, "Multimodal speaker diarization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 79–93, 2012.

[77] X. Zhou, D. Wang, F. Tian, C. Liu, and M. Nakagawa, "Handwritten Chinese/Japanese text recognition using semi-Markov conditional random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2413–2426, 2013.

[78] Y. Shi, Y. Gao, S. Liao, D. Zhang, Y. Gao, and D. Shen, "Semi-automatic segmentation of prostate in CT images via coupled feature representation and spatial-constrained transductive lasso," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2286–2303, 2015.

[79] X. Jin, S. Sarkar, A. Ray, S. Gupta, and T. Damarla, "Target detection and classification using seismic and pir sensors," *IEEE Sensors Journal*, vol. 12, no. 6, pp. 1709–1718, 2012.

[80] R. Singh, M. Vatsa, and A. Noore, "Hierarchical fusion of multi-spectral face images for improved recognition performance," *Information Fusion*, vol. 9, pp. 200–210, 2010.

[81] M. Demirkus, D. Precup, J. Clark, and T. Arbel, "Hierarchical spatio-temporal probabilistic graphical model with multiple feature fusion for estimating binary facial attribute classes in real-world face videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 6, pp. 1185–1203, 2016.

[82] J. Lu, X. Zhou, Y. Tan, Y. Shang, and J. Zhou, "Neighborhood repulsed metric learning for kinship verification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 2, pp. 331–345, 2014.

[83] X. Jin, S. Gupta, A. Ray, and T. Damarla, "Multimodal sensor fusion for personnel detection," in *Fusion*, 2011.

[84] T. Marcard, G. Pons-Moll, and B. Rosenhahn, "Human pose estimation from video and IMUs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–14, 2016.

[85] N. Neverova, C. Wolf, G. Taylor, and F. Nebout, "ModDrop: Adaptive multi-modal gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–14, 2015.

[86] H. Yin, S. Li, and L. Fang, "Simultaneous image fusion and super-resolution using sparse representation," *Information Fusion*, vol. 14, pp. 229–240, 2013.

[87] T. Stathaki, *Image fusion: Algorithms and applications*. Academic Press, 2011.

[88] Z. Liu, E. Blasch, Z. Xue, J. Zhao, R. Laganiere, and W. Wu, "Objective assessment of multiresolution image fusion algorithms for context enhancement in night vision: A comparative study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 94–109, 2012.

[89] K. Kiguchi, S. Kariya, K. Watanabe, K. Izumi, and T. Fukuda, "An exoskeletal robot for human elbow motion support—sensor fusion, adaptation, and control," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 31, no. 3, pp. 353–361, 2001.

[90] S. Majumder, S. Scheding, and H. F. Durrant-Whyte, "Multisensor data fusion for underwater navigation," *Robotics and Autonomous Systems*, vol. 35, no. 2, pp. 97–108, 2001.

[91] O. Basir and X. Yuan, "Engine fault diagnosis based on multi-sensor information fusion using Dempster – Shafer evidence theory," *Information Fusion*, vol. 8, no. 4, pp. 379–386, 2007.

[92] M. Erp, L. Vuurpijl, and L. Schomaker, "An overview and comparison of voting methods for pattern recognition," in *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, IEEE, 2002, pp. 195–200.

[93] W. Kim, J. Chai, and I. Kim, "Development of a majority vote decision module for self diagnostic monitoring system for an air-operated valve system," *Nuclear Engineering and Technology*, vol. 47, no. 5, pp. 624–632, 2015.

[94] C. Orrite, M. Rodriguez, F. Martinez, and M. Fairhurst, "Classifier ensemble generation for the majority vote rule," in *Progress in Pattern Recognition, Image Analysis and Applications*, vol. 5197, Springer, 2008, pp. 340–347.

[95] L. Kuncheva, C. Whitaker, C. Shipp, and R. Duin, "Limits on the majority vote accuracy in classifier fusion," *Pattern Analysis & Applications*, vol. 6, no. 1, pp. 22–31, 2003.

[96] A. Ahmed, F. Saeed, N. Salim, and A. Abdo, "Condorcet and Borda count fusion method for ligand-based virtual screening.," *Journal of Cheminformatics*, vol. 6, no. 1, p. 19, 2014.

[97] S. Wu, "The weighted condorcet fusion in information retrieval," *Information Processing & Management*, vol. 49, no. 1, pp. 108–122, 2013.

[98] T. Ho, J. Hull, and S. Srihari, "Decision combination in multiple classifier systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 66–75, 1994.

[99] C. Perez, L. Cament, and L. Castillo, "Methodological improvement on local Gabor face recognition based on feature selection and enhanced Borda count," *Pattern Recognition*, vol. 44, no. 4, pp. 951–963, 2011.

[100] G. Niu, S. Lee, B. Yang, and S. Lee, "Decision fusion system for fault diagnosis of elevator traction machine," *Journal of Mechanical Science and Technology*, vol. 22, no. 1, pp. 85–95, 2008.

[101] D. Ruta and B. Gabrys, "An overview of classifier fusion methods," *Computing and Information Systems*, vol. 7, no. 1, pp. 1–10, 2000.

[102] J. Qu, Z. Zhang, and T. Gong, "A novel intelligent method for mechanical fault diagnosis based on dual-tree complex wavelet packet transform and multiple classifier fusion," *Neurocomputing*, vol. 171, pp. 837–853, 2016.

[103] G. Kabir, G. Demissie, R. Sadiq, and S. Tesfamariam, "Integrating failure prediction models for water mains: Bayesian belief network based data fusion," *Knowledge-Based Systems*, vol. 85, pp. 159–169, 2015.

[104] M. Jin, R. Li, Z. Xu, and X. Zhao, "Reliable fault diagnosis method using ensemble fuzzy ARTMAP based on improved Bayesian belief method," *Neurocomputing*, vol. 133, pp. 309–316, 2014.

[105] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[106] L. Guo, P. Ge, M. Zhang, L. Li, and Y. Zhao, "Pedestrian detection for intelligent transportation systems combining AdaBoost algorithm and Support Vector Machine," *Expert Systems with Applications*, vol. 39, no. 4, pp. 4274–4286, 2012.

[107] W. Cheng and D. Jhan, "Triaxial accelerometer-based fall detection method using a self-constructing cascade-AdaBoost-SVM classifier," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 2, pp. 411–419, 2013.

[108] S. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[109] G. Death and K. Fabricius, "Classification and regression trees: A powerful yet simple technique for ecological data analysis," *Ecology*, vol. 81, pp. 3178–3192, 2000.

[110] J. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[111] D. Denison, B. Mallick, and A. Smith, "A Bayesian CaRT algorithm," *Biometrika*, vol. 85, no. 2, pp. 363–377, 1998.

[112] Y. Yasami and S. Mozaffari, "A novel unsupervised classification approach for network anomaly detection by K-means clustering and ID3 decision tree learning methods," *The Journal of Supercomputing*, vol. 53, no. 1, pp. 231–245, 2010.

[113] J. Quinlan, *C4.5: Programs for machine learning*. San Mateo, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[114] K. Polat and S. Gunes, "A novel hybrid intelligent method based on C4.5 decision tree classifier and one-against-all approach for multi-class classification problems," *Expert Systems with Applications*, vol. 36, no. 2, pp. 1587–1592, 2009.

[115] D. West, *Introduction to graph theory*. Upper Saddle River, NJ: Prentiss Hall, 1996.

[116] R. Chavarriaga, H. Sagha, A. Calatroni, S. Digumarti, G. Troster, J. Millan, and D. Roggen, "The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 2033–2042, 2013.

[117] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Forster, G. Troster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, and J. Millan, "Collecting complex activity datasets in highly rich networked sensor environments," in *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, IEEE, 2010, pp. 233–240.

[118] P. Casale, O. Pujol, and P. Radeva, "Personalization and user verification in wearable systems using biometric walking patterns," *Personal and Ubiquitous Computing*, vol. 16, no. 5, pp. 563–580, 2012.

[119] T. Cover and J. Thomas, *Elements of information theory*, 2nd ed. Wiley-Interscience, 2006.

[120] G. Brown, A. Pocock, M. Zhao, and M. Luján, "Conditional likelihood maximisation: A unifying framework for information theoretic feature selection," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 27–66, 2012.

[121] R. Madeo, C. Lima, and S. Peres, "Gesture unit segmentation using support vector machines: Segmenting gestures from rest positions," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ACM, 2013, pp. 46–52.

[122] R. Madeo, S. Peres, and C. Lima, "Gesture phase segmentation using support vector machines," *Expert Systems with Applications*, vol. 56, pp. 100–115, 2016.

[123] J. Reyes-Ortiz, L. Oneto, A. Sama, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754–767, 2016.

[124] P. Tan, V. Kumar, and M. Steinbach, *Introduction to data mining*. India: Pearson Education, 2006.

[125] H. Li, T. Jiang, and K. Zhang, "Efficient and robust feature extraction by maximum margin criterion," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 157–165, Jan. 2006.