POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOTTORATO DI RICERCA IN INGEGNERIA DELL'INFORMAZIONE

A COGNITIVE FAULT DETECTION AND DIAGNOSIS SYSTEM FOR SENSOR NETWORKS

Doctoral Dissertation of:
Francesco Trovò

Supervisor:
Prof. Manuel Roveri
Co-supervisor:
Prof. Cesare Alippi

Tutor:
Prof. Francesco Amigoni

The Chair of the Doctoral Program:
Prof. Carlo Ettore Fiorini

2014 – Cycle XXVII

Abstract

COGNITIVE fault detection and diagnosis systems represent a novel class of systems, which are able to detect and diagnose faults by characterizing the functional relationships existing among datastreams and by learning the nominal conditions and the fault dictionary during the operational life directly from incoming data. These abilities make the use of these systems particularly suitable for the field of sensor networks, where no a priori information is generally available about the monitored process and the possibly occurring faults. Moreover, these sensor networks generally work in harsh environmental conditions, thus the occurrence of faults, degradation effects and ageing effects in their units and sensors are likely to happen. It is of paramount importance to detect and diagnose such anomalous working conditions of sensor networks.

This dissertation introduces a novel *Cognitive Fault Detection and Diagnosis System* for sensor networks, able to characterize the nominal state of the process by relying on a fault-free dataset, detect and diagnose faults as soon as they appear and learn in an on-line manner the set of possible faults. To the best of our knowledge, we propose the first complete *Cognitive Fault Detection and Diagnosis System* where the cognitive approach is applied to all its phases: detection, isolation and identification. The proposed system is based on a theoretically grounded statistical framework, able to characterize the functional relationships existing among the acquired datastreams by relying on their modeling in the space of the parameters. As a basis for the detection and diagnosis phases, the proposed system is able to learn a dependency graph, which models the functional

relationships existing among the acquired data. The detection phase is performed by analysing the variation of these functional relationships, through a Hidden Markov Model statistical modeling of the nominal state in the parameter space of linear discrete time dynamic systems approximating models. By considering a logic partition of the learned dependency graph, the proposed system is able to isolate the possible occurring faults, through the analysis of the statistical behaviour of multiple relationships. The identification phase is performed by means of a novel evolving clustering-labeling algorithm specifically designed for this task, which is also capable of learning the fault dictionary in an on-line manner.

The proposed *Cognitive Fault Detection and Diagnosis System* has been validated in a wide experimental campaign on both synthetic data and two real-world challenging and valuable applications: an environmental monitoring application for rock collapse and landslide forecasting system and a water network distribution monitoring system. The experimental results, compared with state of the art methods in the field, provided evidence for the better detection and diagnostic abilities of the proposed *Cognitive Fault Detection and Diagnosis System*.

Sommario

I sistemi cognitivi per la rilevamento e la diagnosi di guasti sono una nuova classe di sistemi, che rilevano e diagnosticano guasti grazie alla caratterizzazione delle relazioni funzionali esistenti tra le stream di dati: essi apprendono le condizioni nominali e il dizionario dei guasti durante la loro vita operativa direttamente dai dati raccolti. Queste caratteristiche rendono l'utilizzo di questi sistemi particolarmente adatto allo scenario delle reti di sensori, dove solitamente non sono disponibili informazioni a priori relative al processo monitorato e ai guasti che possono presentarsi. Inoltre, le reti di sensori operano in condizioni ambientali sfavorevoli, il che implica che guasti, effetti dovuti all'usura e effetti di invecchiamento si presentino di frequente nelle loro unità o nei sensori.

Questa tesi di dottorato introduce un nuovo *Sistema di Rilevamento e di Diagnosi di Guasti Cognitivo* per reti di sensori, capace di caratterizzare lo stato nominale del processo basandosi su di un set di dati privo di guasti, di rilevare e diagnosticare guasti non appena essi si presentano e di apprendere on-line l'insieme dei possibili guasti. Proponiamo il primo *Sistema di Rilevamento e di Diagnosi di Guasti Cognitivo* in cui l'approccio cognitivo è stato utilizzato in tutte le sue fasi: rilevamento, isolamento ed identificazione. Il sistema proposto è basato su un solido approccio statistico, che caratterizza le relazioni funzionali esistenti tra le stream di dati grazie alla loro modellizzazione nello spazio dei parametri dei sistemi dinamici lineari a tempo discreto che le approssimano. Alla base delle fasi di rilevamento e diagnosi, il sistema proposto apprende un grafo delle dipendenze, che modellizza le relazioni funzionali esistenti tra le stream di dati. La fase di

rilevamento dei guasti si basa sull'analisi della variazione delle relazioni funzionali considerate nella fase precedente, che viene rilevata grazie alla modellizzazione statistica dello stato nominale fornita dagli Hidden Markov Model. Grazie alla partizione logica del grafo delle dipendenze ed all'analisi statistica del comportamento di tutte le relazioni funzionali considerate, il sistema proposto riesce inoltre ad isolare il guasto. La fase di identificazione del guasto viene svolta da un innovativo algoritmo di clustering evolutivo, il quale è in grado di apprendere on-line il dizionario dei guasti.

Il *Sistema di Rilevamento e di Diagnosi di Guasti Cognitivo* proposto è stato validato in un'estesa campagna sperimentale, che ha considerato sia dati sintetici, sia dati generati da due applicazioni particolarmente interessanti: un sistema di monitoraggio ambientale per la predizione di frane ed un sistema di monitoraggio di una rete di distribuzione idrica. I risultati sperimentali ottenuti dal *Sistema di Rilevamento e di Diagnosi di Guasti Cognitivo* proposto sono stati confrontati con quelli dei metodi nello stato dell'arte nel campo considerato e hanno dimostrato di ottenere migliori performance in termini di rilevamento e di diagnosi dei guasti.

Contents

CHAPTER 1

Introduction

Sensor networks represent an important and valuable technological solution to monitor and acquire data from an environment, a critical infrastructure or a cyber-physical system. These data are generally used as input for an application, which is able to take a decision or react to the change in the inspected system. Examples of these applications based on sensor networks, are those inspecting an environmental phenomenon (e.g., a river, a rock wall or a coral reef) protecting critical infrastructures or those monitoring the behaviour of a water distribution network.

Sensor networks are composed by a set of sensing units, each of which is composed by a set of sensors, a processing board and a data transfer apparatus. Each unit is deployed in a different location in the area influenced by the inspected phenomenon and contains multiple sensors, each of which gathers meaningful information about it. The measurements provided by the sensors are homogeneous or heterogeneous, in the sense that they are measuring the same physical quantity or different but related physical quantities. The processing boards are responsible for the processing of the acquired data and management of the units (e.g., they manage the sensors sampling synchronization and/or analog to digital conversion of the received signals). Finally, the data transferring apparatus sends acquired

data to a central processing station, where the data are stored and elaborated. During the operational life of this infrastructure, the sensor network continuously inspects the system status, by sending measurements to the central processing station, where the application runs. In fact, based on measurements coming from the sensor network, applications can be designed to take decisions (e.g., in the case a deviation from the usual working conditions is registered, an alarm is raised) and contextually react to the change (e.g., for critical infrastructures, alert the population about the expected threat).

In this scenario, a model for the inspected phenomenon is usually unknown and the assumption of process stationarity may not hold, even though these assumptions would generally improve the decision abilities of the aforementioned applications. Thus, information coming from the sensor network is critical to monitor the underlying process, to check the status of the system and react according to its behaviour. For instance, even if the modeling of the processes affecting a rock wall is generally not feasible, a constant monitoring action of cracks width and rock inclination measurements may predict if a collapse is likely to happen.

Sensor networks usually work in harsh real-working conditions, which may induce permanent or transient faults, thermal drifts or ageing affects affecting both the embedded electronic boards which manage the sensors and the sensors themselves. In fact, both the electro-mechanical components and the embedded electronics of the sensors are affected by physical degradation (due to e.g., humidity, dust, chemicals and electromagnetic radiations), which may induce a gradual deviation of the measured value from the real one. Moreover, problems may arise on the processing boards, which may convert data coming from sensors (e.g., from analog to digital) in a incorrect way. Finally, a correct behaviour of the communication apparatus is crucial for the collection of the measurements, since a degradation of the information carried by the transferred data may occur during the transmission phase, both if the channels are physical, e.g., if the cables are not properly maintained, or if they are wireless, e.g., if an unexpected perturbation influences the data transmission frequencies.

It is of paramount importance to promptly detect and diagnose faults occurring in a specific unit, since they could affect the application layer, which operates based on the assumption that information provided by the sensor network is not corrupted. If the application does not take into account the possibility of a fault (i.e., it is not robust to faults), it may take an incorrect decision, e.g., not alert the population when a threat is present or vice versa. Moreover, the corruption of a single unit could spread through-

out the entire network (domino or cascade effect), which could eventually compromise the effectiveness of the monitoring action, even in the part of the network which is working properly.

Moreover, in the sensor network scenario the unexpected deviation from nominal conditions may be caused either by a fault or by a change in the inspected process. It is crucial to distinguish between these two situations: in the former case, direct maintenance should be performed, while the latter one requires a reaction to an environmental change, specifically chosen based on the considered application scenario. Thus, the objective of this dissertation is twofold: at first, to be able to understand when the sensor network is changing and, after that, to distinguish between the change in the environment and a fault affecting a sensing unit.

1.1 Detection and Diagnosis of Faults in Sensor Networks

Fault Detection and Diagnosis Systems (FDDSs) are systems specifically designed to detect and diagnose faults possibly occurring in complex systems. The general scheme for a FDDS is presented in Figure 1.1, where it takes into account the data coming from the sensor network to alert the application level if the data are deviating from nominal conditions. More specifically, the tasks of a FDDS are:

- detection of the fault, i.e., promptly understand whether a deviation from the nominal state has occurred;
- diagnosis of the fault to characterize the detected fault. Its main subtasks are:
  - isolation, i.e., determine which unit is providing faulty measurements;
  - identification, i.e., capture the main characteristics (e.g., type, intensity) associated to the fault;

FDDSs have been widely studied in the past decades [27, 37, 45, 81], and successfully applied to real world applications [46]. They revealed to be particularly effective in several application domains, provided that generally a priori information about the system or the possible faults is (at least partially) available. To be effective, most of the FDDSs require to have a priori information about the system or of the possible faults, e.g., the availability of the fault-free nominal state or the "fault dictionary", containing the possibly occurring faults characterizations. Moreover, the knowledge

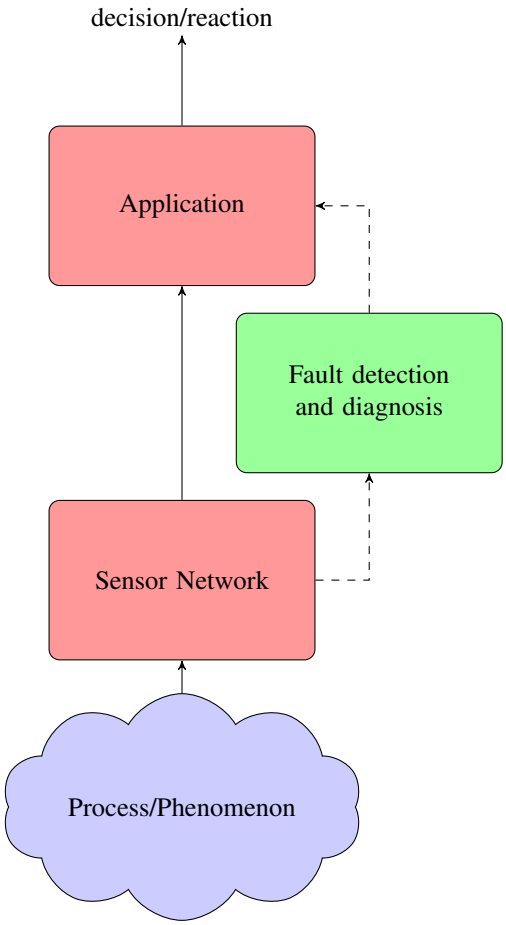


Figure 1.1: *General scheme of a monitoring system: it is able to inspect a process or a phenomenon with a sensor network. Data are gathered by sensors and transferred to an application which is able to take decision/reaction based on those data. A fault detection and diagnosis system is critical to avoid that malfunctioning of the sensor network might induce an incorrect behaviour of the application.*

of the noise level (or the assumption of its absence) allows to have mathematical models of dynamic processes by theoretical/physical modeling or by relying on system identification techniques. The knowledge of the fault dictionary and the noise level is hardly available in the sensor networks scenario, thus most of the traditional methods are not directly applicable in real world sensor network applications.

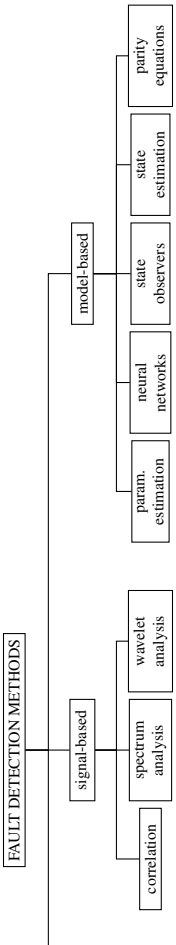


Figure 1.2: *Fault detection taxonomy*

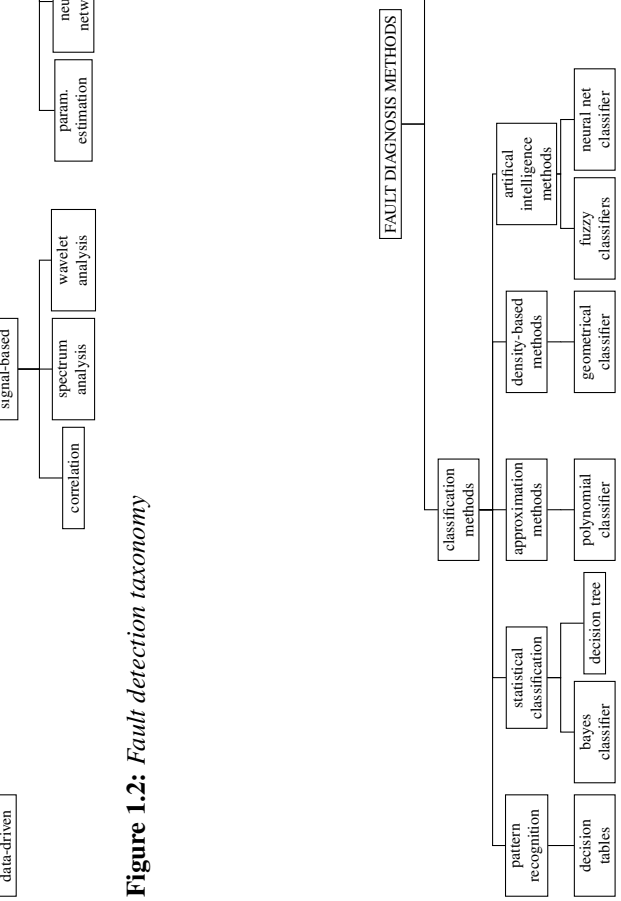


Figure 1.3: *Fault diagnosis taxonomy*

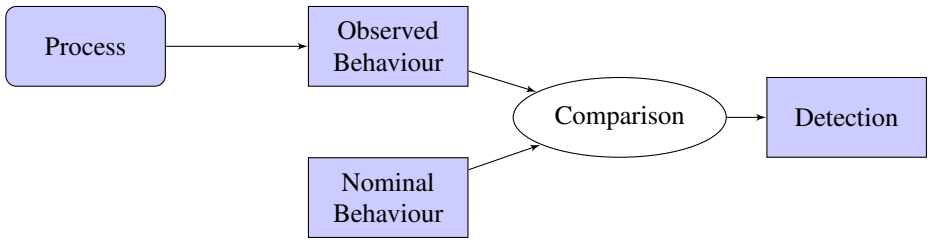


Figure 1.4: *Generic fault detection scheme: the* Process *is characterized through its* Nominal Behaviour. *The* Observed Behaviour *is compared with the nominal one and, if a discrepancy is assessed, the system detects a change*

A taxonomy of the methods present in the literature of fault detection and diagnosis systems, inspired by the one presented in [45], is provided in Figures 1.2 and 1.3, respectively.

1.1.1 Fault Detection

A general scheme for fault detection is depicted in Figure 1.4. This scheme is based on the knowledge of the nominal behaviour of the actual system, which is compared against the observed process behaviour. Usually, the comparison is performed by inspecting features that reflect the discrepancy between the two aforementioned behaviours. Different kinds of features can be encompassed. If we only have information about the characteristic value or behaviour of the chosen features in nominal conditions, *limit or trend checking* is the most simple and frequently used method for change detection. The measured values or trends (i.e., first derivatives) are monitored and checked to verify whether they exceed certain lower and upper thresholds. If we consider a known distribution as the model for features, it is possible to check for some meaningful statistics by estimating the moments of the distribution (e.g., mean, variance, kurtosis). In this case, Change Detection Tests (CDTs) [21] can be considered to check for changes in the distribution. This could be performed in an off-line way, through statistical tests, or by relying on sequential techniques, such as control charts, which are able to verify whether a change in the data distribution has happened in an on-line manner. Most of the off-line and on-line tests present in literature require that the distribution of the data is fixed or known. Finally, more complex methods for accounting for uncertainty in the system are the ones which are based on the use of adaptive or fuzzy thresholds.

The features considered in the change detection methods have different

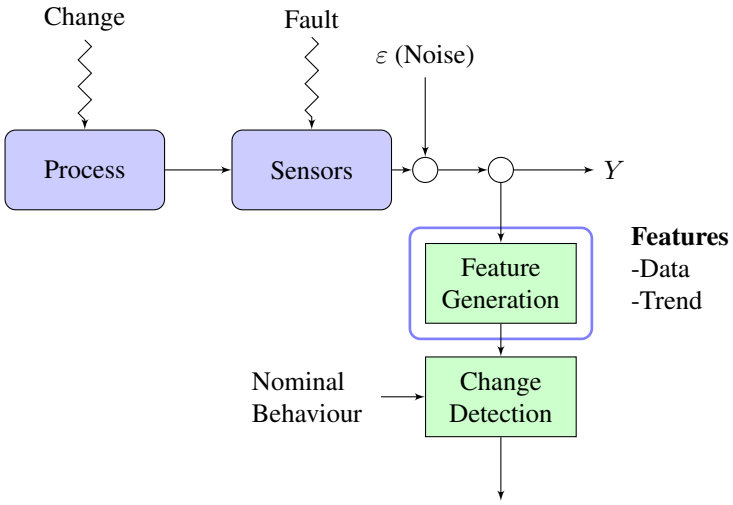


Figure 1.5: *Data-driven scheme for fault detection in the sensor network scenario.*

nature, which are based on the different models we may consider for the signal generated from the process: the approaches are data-driven, signal-based and model-based, as described in Figure 1.2. They differ in the modeling approach used for the signal considered for fault detection. While the data-driven approach does not assume any model for the incoming data, the signal-based one makes use of the characteristics of the signal to provide a detection result. Finally, the model-based approach relies on the modeling of the process generating the data to detect changes. All the aforementioned methods rely on one of the change detection techniques previously described to assess if a discrepancy between the expected and observed behavior is present.

The first class of modeling approaches for fault detection is the data-driven one, which is based on the direct analysis of the provided measurements. This approach does not require the assumption of a specific model for the data generation procedure. Its generic scheme is presented in Figure 1.5. The features which are monitored for fault detection purposes are the measured variables or trends present in the data.

When a signal model for the data generation process is provided, fault detection schemes as in Figure 1.6 can be adopted. If we consider the signal as a stochastic process, we can consider as features statistics like sample estimated moments (mean, variance or kurtosis) or autocorrelation. These statistics can be used in the change detection framework on the hypothesis that the signal distribution is known and fixed over time. Otherwise, if

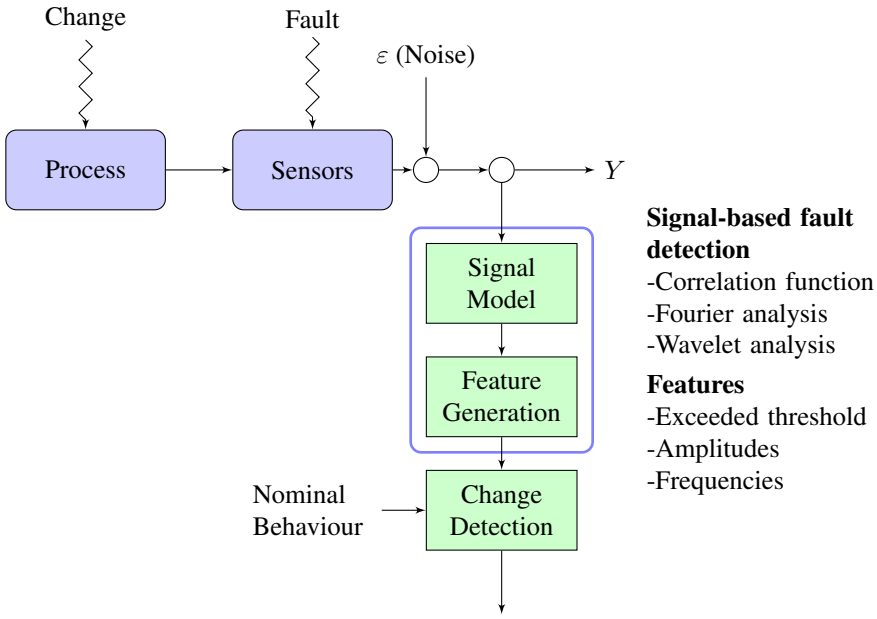


Figure 1.6: *Signal-based scheme for fault detection in the sensor network scenario.*

the signal presents periodic behaviours, techniques like bandpass filtering, Fourier analysis, correlation analysis and cepstrum analysis may be used to extract features. These techniques coupled with the aforementioned limit checking methods can be used to detect changes in the signal. If the stationarity assumption does not hold, short-time Fourier transform or wavelet transform could be performed to take into account local characteristics of the signal.

The third class of approaches to fault detection takes into account more than one stream of data at a time to model the relationship between input and output of a process, as depicted in Figure 1.7. Model-based methods exploit the relationships existing among a set of measured variables to extract information about possible changes caused by faults. These methods are based on different model identification techniques, both for linear and non-linear processes, like Least Square (LS) method, polynomial approximators, Artificial Neural Networks (ANNs) and semi-physical models. A widely known model-based method is the parity equations one: in this paradigm the model behaviour in nominal conditions is compared with the model during the operational life. This discrepancy, called residual, is checked for consistency to provide a detection. In the case the process is characterized by a linear continuous and discrete-time state-space dynamic

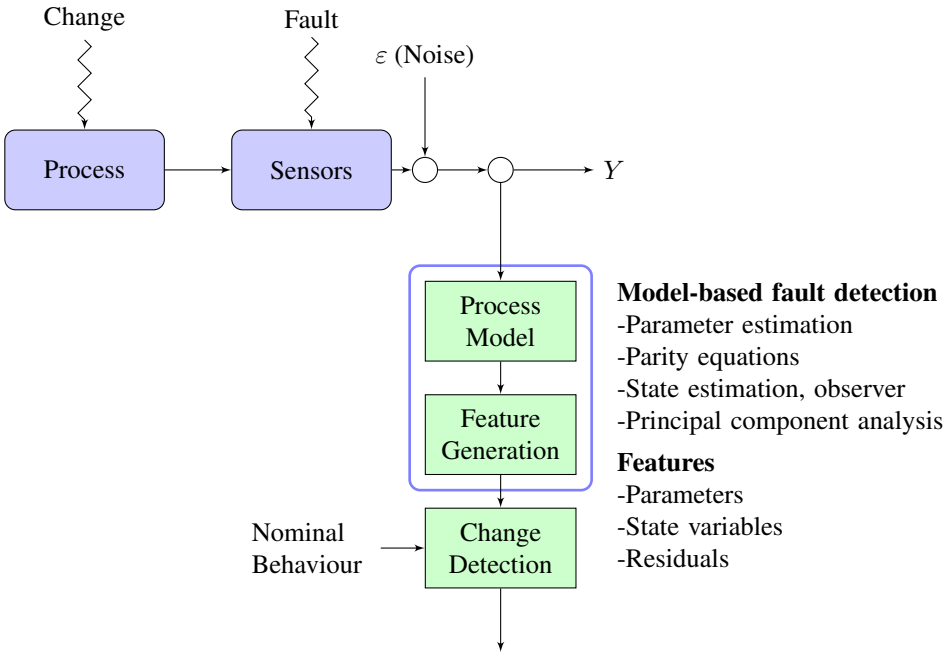


Figure 1.7: *Process-based scheme for fault detection in the sensor network scenario.*

system, state observer and state estimation, respectively, can be used for automatically providing a model for the system. Finally, a method which does not assume any model for the data, while analyzing all the streams of data is the Principal Component Analysis (PCA). This unsupervised method could be used in the fault detection field both as a dimensionality reduction method, coupled with one of the previously presented methods, or as a detection method itself, by using its reconstruction error as the feature.

One of the main drawbacks of the presented approaches is that they generally require a priori information on either the system in nominal conditions or the possible faults. In fact, the data-driven approach requires to set values for the threshold, which are generally very difficult to tune, since their proper values (the ones which would provide high detection abilities) depend on the faults we are expecting to face and on the noise level present in the data. The signal-based and model-based approaches require the knowledge of the signal or process, respectively, in nominal conditions, otherwise they might not be directly applicable to a specific problem.

1.1.2 Fault Diagnosis

The diagnosis task consists in the determination of the faults and all its characteristics, e.g., size and location. The taxonomy of the methods proposed in the literature is presented in Figure 1.3. In the field of fault diagnosis, classification methods are able to identify the most probable fault given a set of information coming from the sensor network, i.e., some specific features which are likely to characterize the occurred fault. This could be carried out with pattern classification methods, e.g., Bayesian classifiers, polynomial classifiers, K-Nearest Neighbours (KNN) classifiers, decision trees or ANNs. Another viable approach is the use of inference methods, which are able to express the qualitative knowledge in the form or if-then rules. Methods choosing this approach rely on the use of fault trees (to express the binary logic relationship between features and faults) or approximate reasoning (which extends fault trees to the case of continuous variables), implemented usually with neural or neuro-fuzzy methods.

Application scenarios taken into account in [45] assume to know the fault dictionary, i.e., the set of possible faults which may occur to a system. Once a fault is identified, its characterization can be retrieved from the fault dictionary, where also its location is provided, thus including the isolation phase in the identification one.

It is worth to mention that most of the diagnosis techniques presented above require information about the data generating process and/or the fault characteristics. Thus, their direct application to the sensor network scenario is far from being trivial, since, as pointed out before, we do not have information about the model of the system generating the data, or the fault dictionary.

1.2 Cognitive Fault Detection and Diagnosis Systems

In recent years, a novel and promising *cognitive* approach has been proposed to design FDDSs. This novel generation of CFDDS is able to automatically learn the nominal and the faulty states in an on-line manner, and is generally characterized by the ability to exploit temporal and spatial relationships present among the acquired data. A summary of the available techniques is presented in Table 1.1. More specifically, the bases of cognitive FDDSs stand on the fact that the sensor network is observing from multiple points of view the same physical phenomenon, hence it is possible to exploit redundancies present among the measurements. In fact, if we consider two different sensors in the same unit they may be affected

by a parasitic effect (e.g., dependence from temperature), thus providing temporal redundant data. Similarly, if we consider two sensors in different units providing homogeneous measurements we are able to infer if the phenomenon is affecting both the locations in the same way, i.e., if a spatial redundancy or causality is present. The learned relationships among sensor measurements are used to detect whether the data are deviating from the nominal behaviour and, possibly, diagnose the causes of this deviation.

Table 1.1: *Relevant approaches in Cognitive FDDS*

	Characteristics			Taxonomy	
Paper	Nominal State	Fault Dictionary	Application	Detection	Diagnosis
[36]	Given	Learned	Dynamic systems	Parity Equations	Neural Networks
[30]	Given (bounded error)	Learned	Dynamic systems	State estimator	N.a.
[80]	Given (bounded error)	Learned	Dynamic systems	State estimator	Adaptive filtering
[88]	Learned	Given	Chemical process	Neural Networks	Neural Networks Classifiers
[43]	Learned	Given	Transformer	N.a.	Neural Networks Classifiers
[86]	Given	On-line	Electric Motors	N.a.	Neural Networks Classifiers
[90]	Learned	Given	Stirred tank reactor	N.a.	Neuro-Fuzzy logic
[61]	Learned	Given	Ball bearings	N.a.	Neuro-Fuzzy logic
[77]	Learned	Given	Induction Motors	N.a.	Neuro-Fuzzy logic
[44]	Learned	Given	Transformers	N.a.	Fuzzy logic
[64]	Learned	Given	Transformers	N.a.	Neuro-Fuzzy logic
[57]	Learned	Given	Marine propulsion engine	N.a.	Neuro-Fuzzy logic
[49]	Learned	Given	Gearboxes	Neural Networks	N.a.
[3]	Learned	Given	Circuit transmission	N.a.	AI Methods
[26]	Learned	Learned	Dynamic systems	Model-Based	Inference method
[8]	Learned	Learned	Dynamic systems	Data-driven	Inference method

Cognitive approaches generally rely on Machine Learning (ML) techniques to configure the nominal state and characterize the faulty ones without requiring any a-priori information about the fault signature or of the fault time profile. The ancestors of this approach can be found in works published starting from the nineties. Most of these papers focus on specific applications or provide algorithms which confine the cognitive approach to a specific phase. The idea behind cognitive FDDSs is suggested in [45], while most of the systematic research relying on this paradigm has been developed in the last 2-3 years.

Most of existing cognitive FDDSs apply the learning mechanism only to a single aspect of the system [30, 36, 80], thus they still require at least to know partial information about the analysed process. In more detail, [36] proposes a learning procedure for unanticipated fault accommodation, under the assumption that the process model is given. [30] presents a learning methodology for incipient failure detection based on nonlinear on-line approximators, which aims at both inspecting variations in the system due to faults and provides information about the detected faults in an on-line manner. [80] extends the approach presented in [30], by coupling the on-line approximators with a learning procedure for the accommodation of scheme parameters.

There is a large literature addressing the design of cognitive FDDSs for specific applications based on neural networks [43, 86, 88] and fuzzy logic [3, 44, 49, 57, 61, 64, 77, 90], with cognitive mechanisms mostly applied during the training phase of the FDDS. For instance, [88] presents a supervised method for fault classification, which relies on the learning of a Radial Basis Function (RBF) network for chemical process modeling and fault detection, while a second neural network is used for fault identification (classification) task. [43] describes a FDDS specifically designed for fault identification in power transformers based on genetic algorithms for training a neural network. In [86], the authors propose an intelligent FDDS for electric motors based on Adaptive Resonance Theory and Kohonen Neural Network (ART-KNN): new faults can be included in the dictionary thanks to the design of a case-based reasoning system, where faults are reported by experts and automatically included in the system. [90] suggests a fault identification technique based on the joint use of fuzzy logic and feedforward neural networks, where the fuzzyfication logic is mainly based on a priori knowledge of the considered application, i.e., a stirred tank reactor. [61] proposes a classification method for faults occurring in ball bearing, which considers a wavelet analysis on the accelerometer measured signals, whose results are feed to an Adaptive Network-based Fuzzy Inference

System (ANFIS) able to adapt to newly received signals. [77] presents an approach for classification of faults affecting induction motors which relies on a combination of unsupervised and supervised learning to cluster and classify faulty patterns for the value of the current. [44] extends the usual dissolved gas analysis for transformer with an evolving programming fuzzy-based approach for identifying incipient faults, presenting a genetic approach to train the fuzzy rules used for classification. On the same application, in [64] a system based on subtractive clustering, as a basis for fuzzy rules creation and optimization, is proposed. In [57], a fault identification system is proposed for the specific field of propeller-shaft marine propulsion engine: a neuro-fuzzy system trained with genetic algorithms allows to distinguish among a predefined set of faults. In [49], a fault detection algorithm for inspecting the frequency spectrum of vibration signals coming from gearboxes is presented. This method uses function fuzzy c-means algorithm to cluster spectra and detect if a fault is likely to arise. Finally, the problem of fault identification in circuit transmission is solved in [3] by using fuzzy adaptive neural network, which are able to automatically learn the fuzzy rules associated with a specified set of faults.

We would like to remark that, the solutions designed for specific applications we mentioned above require supervised information on the faults, i.e., to know the fault dictionary. As mentioned before, the cognitive aspect of these methods is confined in the training phase, thus the characterization of the nominal state is automatic, but information about the fault signature is needed. Differently, [45] suggests the use of an unsupervised "clustering-labeling" method to automatically assign observations either to the nominal or the faulty classes, thus considering the possibility to improve the fault dictionary during the FDDS operational life. Unfortunately, no technical details about the implementation of the solution are given.

Two works [8, 26], fully relying on the concept of cognitive, have been developed in parallel with the work described in this dissertation. In [26] they propose a method for detection and identification of faults for generic unknown dynamic discrete-time systems, by relying on an approach based on the analysis of the provided datastream in the functional space. The requirement for training such FDDSs is a dataset composed entirely by fault-free instances of the dynamic system inspected by the sensor network. In [8] the fault detection problem is entailed by the use of Hidden Markov Models (HMMs), which are able to model in a statistical way the nominal state of the system. Thanks to this characterization, they develop a HMM-CDT able to detect faults by inspecting the discrepancy of the faulty data from the nominal model learned by the HMM. More details about this

method will be provided in the sequel of this dissertation.

1.3 Original Contribution

In this dissertation we propose a new CFDDS meant to operate on sensor networks. The proposed system is able to characterize the nominal conditions of the system, by relying on fault-free data coming from the sensor network. At first, the proposed system learns the dependency graph existing among datastreams, to select only relevant functional relationships. The analysed relationships are able to capture the temporal and spatial redundancies present in the data. Based on them, the proposed CFDDS is able to perform fault detection, isolation and identification, without requiring a priori information on either the inspected process or the faulty states.

To model the relationships constituting the causal dependency graph of the sensor network we rely on the concept of Granger causality, which allows to consider only those relationships providing meaningful information for fault detection and diagnosis. After learning the network dependency structure, fault detection and diagnosis is carried out in the space of estimated parameter vectors of Linear Time Invariant (LTI) models approximating the functional relationships included in the dependency graph. Deviations from the learned nominal concept are detected by means of a decrease of the loglikelihood provided by the HMM modeling, which is here considered to characterize the statistical pattern of the parameter vectors estimated on fault-free data (nominal state). Following the detection phase, an isolation mechanism based on the logic partition of the dependency graph is able to distinguish between faults (whose location is also inferred) and change in the environment. Finally, in the case a fault has occurred, an identification procedure is executed to characterize and discriminate among different faults. The identification phase is performed by means of a newly developed evolving clustering-labeling technique in the space of the parameter vectors. This framework was specifically designed for fault diagnosis purposes and is able to learn the fault dictionary in an on-line manner.

The innovative aspects of this cognitive framework for fault detection and diagnosis are:

- the design of a CFDDS completely based on the cognitive approach, relying on computational intelligence techniques, which is able to cover all the phases of fault detection and diagnosis in the sensor network scenario;

- the development of a set of integrated techniques, coming from statistics and machine learning fields, which rely on a theoretically sound framework developed by the system identification field;
- the ability to characterize the temporal and spatial relationship existing among data with the dependency graph, learned with the use of a statistical framework;
- the ability to characterize the nominal state of the system inspected by the sensor network through learning mechanisms based on the cognitive approach;
- the ability to learn the fault dictionary during the operational life of the system, without requiring a priori information about the possible faults.

To validate the abilities of the proposed method we considered two real world applications of environmental monitoring systems: the rock landslide and collapse forecasting system of the Rialba Towers in Northern Italy and the Barcelona Water Network Distribution System (BWNDS). We took into account these two relevant applications since, despite the lack of information about the data-generating process and their complexity, the cognitive approach is able to effectively address the problem of fault detection and diagnosis.

1.4 Dissertation Structure

The dissertation is structured as follows: in Chapter 2 the mathematical framework used in the sequel is presented, including the model of the considered data-generating processes, the fault models and their influences on the original process, as well as the purposes of the proposed framework. In Chapter 3, a general overview of the proposed system is presented, with particular emphasis on the model approximation techniques adopted and on the theoretical results they rely on. Chapters 4 to 7 detail the graph learning, the detection, isolation and identification phases adopted by the proposed CFDDS, respectively. Chapter 8 provides a brief overview of the implementation of the developed techniques, while Chapter 9 describes the wide experimental campaign conducted to validate the performance of the proposed framework. Finally, in Chapter 10, some concluding remarks are drawn, as well as future perspectives in the field.

CHAPTER 2

Problem Formulation

In this chapter, we provide a framework to formally describe data coming from a sensor network inspecting a process, whose model is unknown because of the lack of information or because its modeling is too complex. Towards this goal, in Section 2.1 we describe the model of the data generating process we consider. A specification of the considered faults affecting the sensor network is provided in Section 2.2. Subsequently, in Section 2.3, we present the model of the sensor network during the faulty state. Then, some examples of faults are presented in Section 2.4. Finally, the formulation of the problem of fault detection and diagnosis in the presented modeling framework is detailed in Section 2.5.

2.1 Sensor Network Measurements

We consider a time-invariant dynamic system $\mathcal{P}$ whose model description is unavailable. A sensor network acquiring data from $\mathcal{P}$ is composed by a set of $n \in \mathbb{N}$ sensors:

$$\mathcal{S} = \{s_1, \ldots, s_n\}, \tag{2.1}$$

where each sensor s_i acquires scalar measurements coming from $\mathcal{P}$. Selection of the most appropriate deployment positions for the sensors is outside

the scope of this dissertation. The interested reader can refer to [33, 52, 56, 71] for a comprehensive investigation of the placement problem. The assumption of considering sensors providing scalar measurements may be overcome in a straightforward way. In fact, for a generic sensor $s_{i'}$ providing d-variate measurements included in a network $\mathcal{S}'$, we can replace the original sensor network with an equivalent one:

$$\mathcal{S}' \leftarrow \mathcal{S}' \setminus \{s_{i'}\} \cup \{s'_1, \ldots, s'_d\}, \tag{2.2}$$

where s'_i are sensors providing as scalar measurement the i-th dimension of the measurement provided by $s_{i'}$. This transformation allows us to consider in the sequel only scalar measurements coming from sensors.

We would like to point out that the sensors may observe both heterogeneous or homogeneous physical quantities, e.g., sensors may provide temperature in different locations of the network (homogeneous measurements) or temperature and wind speed in the same one (heterogeneous measurements). All the sensors which are deployed in the same position belongs to a unit u_j of the network or, more formally the network sensors can be partitioned by a set of $m \in \mathbb{N}$ units:

$$\mathcal{U} = \{u_1, \ldots, u_m\}, \tag{2.3}$$

where each unit u_i is a set of sensors $u_i \subseteq \mathcal{S}$ and $\mathcal{S} = \bigcup_i u_i, u_i \cap u_j = \emptyset, i \neq j$.

In the sequel we consider, for the sake of simplicity, sensor networks where each unit is composed of a single sensor ($m = n$ or $u_i = \{s_i\}\ \forall i$) and where each sensor provides scalar measurements. Extensions to the CFDDS proposed in the following chapters are trivial.

We consider a sensor network that, through a suitable synchronization algorithm[1], is able to provide at each time instant $t \in \mathbb{N}$ the measurements column vector:

$$\boldsymbol{X}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix} \in \mathbb{R}^n, \tag{2.4}$$

where $x_i(t) \in \mathbb{R}$ is the scalar measurement acquired at time t by sensor s_i.

By considering the entire monitoring period of the sensor network we can define the datastream acquired by sensor s_i as:

$$\boldsymbol{x}_i = \{x_i(t)\}_{t \in \mathbb{N}} \tag{2.5}$$

[1]The problem of measurements synchronization is out of the scope of this dissertation. The interested reader can refer to [12].

and the multivariate vector containing datastreams $\boldsymbol{x}_i$ as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix}. \tag{2.6}$$

As final remark we would like to point out that in this dissertation we do not take into account continuous time dynamic processes, nor state-space formulations. In fact, for many applications of particular interest in the sensor networks field (e.g., environmental monitoring) we are interested in the analysis of the discrete-time input-output relationships existing among datastreams [60].

2.2 Fault Modeling

In general, the behavior of the system under faulty conditions is not known a priori, however, fault modeling is crucial in the design and analysis of fault diagnosis schemes, since it allows to fully characterize a posteriori the possible deviations occurring to the system. This section presents the overall fault modeling framework that will be used during this dissertation.

Definition 2.2.1. *A fault $F(\cdot)$ is defined as an unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable/usual/standard conditions. Formally, a fault $F : \mathbb{N} \to \mathbb{R}$ is represented as:*

$$F(t) = \sum_{\nu=1}^{\nu^*} \mathcal{B}_\nu(t; t_i^{(\nu)}, t_f^{(\nu)}) \phi_\nu(t), \tag{2.7}$$

where:

- *ν^* is the number of time intervals where fault is present;*
- *$\mathcal{B}_\nu(t; t_i^{(\nu)}, t_f^{(\nu)}) \in [0, 1]$ denotes the time profile of the fault that occurs at time instant $t_i^{(\nu)}$ and disappears at time instant $t_f^{(\nu)}$, with $t_f^{(\nu)} < t_i^{(\nu+1)}$;*
- *$t_i^{(\nu)} \in \mathbb{N}$ and $t_f^{(\nu)} \in \mathbb{N}$ are the occurrence and disappearance time of the fault in the ν-th time interval, respectively, with $t_i^{(\nu)} < t_f^{(\nu)}$;*
- *$\phi_\nu(t) \in \mathbb{R}$ is the fault signature within the ν-th time interval at time instant t.*

The ν-th time profile $\mathcal{B}_\nu : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to [0,1]$ is described by:

$$\mathcal{B}_\nu(t; t_i, t_f) = \beta_i^{(\nu)}(t - t_i^{(\nu)}) - \beta_f^{(\nu)}(t - t_f^{(\nu)}) \tag{2.8}$$

where $\beta_i : \mathbb{Z} \to [0,1]$ is the evolution mode of the fault occurrence from time $t_i^{(\nu)}$ and $\beta_f : \mathbb{Z} \to [0,1]$ is the evolution mode of disappearance from time $t_f^{(\nu)}$,

$$\beta_i(t) = \beta_f(t) = 0 \ \ \forall t < 0 \tag{2.9}$$

and

$$\beta_i^{(\nu)}(t - t_i^{(\nu)}) \geq \beta_f^{(\nu)}(t - t_f^{(\nu)}) \quad \forall t \in \mathbb{N}. \tag{2.10}$$

According to the *time duration*, a fault F can be characterized as [37, 45]:

- intermittent ($\nu^* > 1$)

$$F(t) = \sum_{\nu=1}^{\nu^*} \mathcal{B}_\nu \left(t; t_i^{(\nu)}, t_f^{(\nu)}\right) \phi_\nu \left(t\right), \tag{2.11}$$

- transient ($\nu^* = 1$, $\exists\, \bar{t} \leq \infty$ s.t. $\beta_f(t) = 1 \ \ \forall t > \bar{t}$):

$$F(t) = \mathcal{B}\left(t; t_i, t_f\right) \phi(t), \tag{2.12}$$

- permanent ($\nu^* = 1$, $\beta_f(t) = 0 \ \ \forall t \in \mathbb{N}$):

$$F(t) = \mathcal{B}\left(t; t_i, t_f\right) \phi(t), \tag{2.13}$$

The time profile $\mathcal{B}_\nu$ (and thus the corresponding fault F) can be characterized based on the *evolution mode* as:

- abrupt:

$$\beta_k^{(\nu)}(t) = \begin{cases} 0 & t \leq 0 \\ 1 & t > 0 \end{cases} \tag{2.14}$$

- incipient:

$$\beta_k^{(\nu)}(t) = \begin{cases} 0 & t \leq 0 \\ 1 - e^{-r_k t} & t > 0 \end{cases} \tag{2.15}$$

  where $k \in \{i, f\}$ and r_i and r_f are the evolution rate of occurrence and disappearance, respectively.

The fault signature $\phi(t)$ is usually a generic time varying scalar function. Here for modeling purposes, we specify some common faults:

- constant:
$$\phi(t) = \bar{\phi}, \;\; \bar{\phi} \in \mathbb{R}; \tag{2.16}$$

- drift (linear):
$$\phi(t) = Rt, \;\; R \in \mathbb{R} \tag{2.17}$$

- precision degradation:
$$\phi(t) \sim \mathcal{D}(t), \tag{2.18}$$
where $\mathcal{D}(t)$ is a distribution of a stochastic process, such that $\mathbb{E}[\mathcal{D}(t)] = 0$.

A graphical representation of the fault models is shown in Figure 2.1. In the case of intermittent faults, only the abrupt evolution mode of occurrence and disappearance is depicted. In the general case, for each time interval of fault presence, the evolution mode may be either abrupt or incipient.

In this dissertation, the assumption that data are continuously recorded, without any hole, is considered. Otherwise, while modeling faults we should take into account the possibility that for some time period $[t_i; t_f]$ data are not coming from the sensor network, which is usually addressed as *missing data fault*. If it is the case and the period is relatively short, it is possible to adopt reconstruction techniques as described in [13, 14, 69]. If the fault period is too long such techniques are not viable anymore, since their reconstruction error may explode with the progressing of time.

2.3 Modeling the Faulty System

Once the model for a fault F is specified, we need to determine how it affects the measurements coming from the sensor network $\mathcal{S}$. A monitoring system subject to a fault F is presented in Figure 2.2. Under healthy conditions, the output of the sensor network at time t is the measurement column vector $\boldsymbol{X}(t)$. The faults are usually represented either by external signals or as parameter deviations and are induced in an additive and/or multiplicative way. The output of the sensor network at time t under faulty conditions $\boldsymbol{X}_F(t)$ is described by the following equations:

$$\boldsymbol{X}_F(t) = [I_n + \Gamma(t)]\, \boldsymbol{X}(t) + \Phi(t) \tag{2.19}$$

where I_n is the identity matrix of order n, $\Gamma(t) \in \mathbb{R}^{n \times n}$ is the multiplicative matrix at time t and $\Phi(t) \in \mathbb{R}^n$ is the additive column vector at time t, modeling the effects of faults on the system. $\Gamma(t)$ is a diagonal matrix such that each diagonal element $[\Gamma(t)]_{ii} = \gamma_{ii}(t)$ is a fault $\gamma_{ii}(t) = F_i(t)$ with

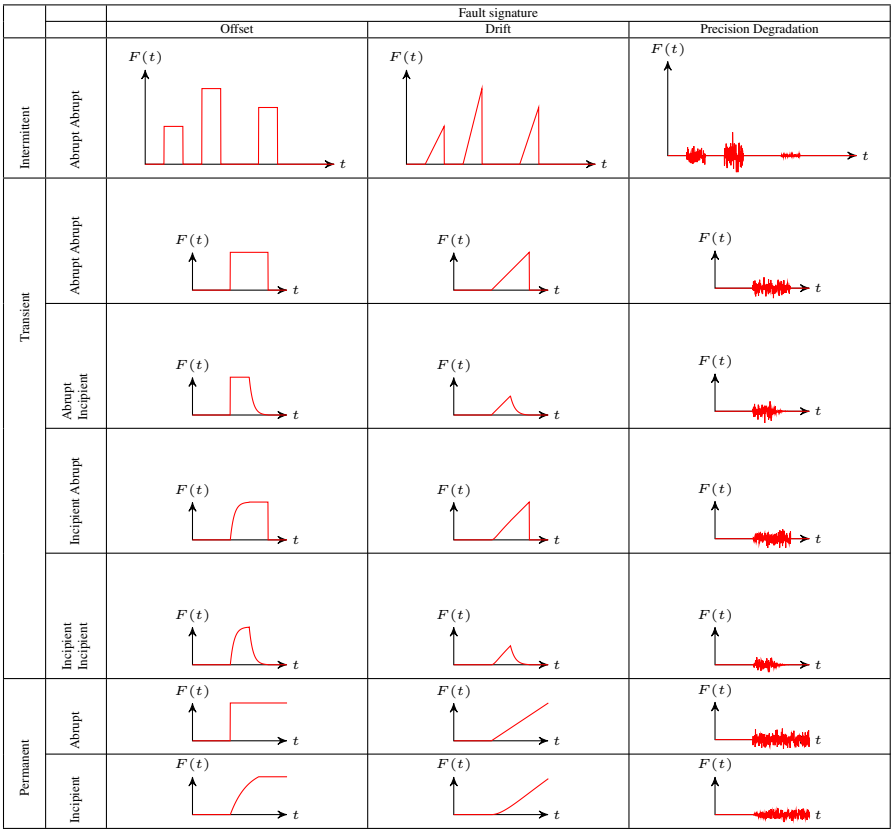


Figure 2.1: *Specification of the faults considered in this dissertation. For the sake of concision we depicted only abrupt intermittent faults.*

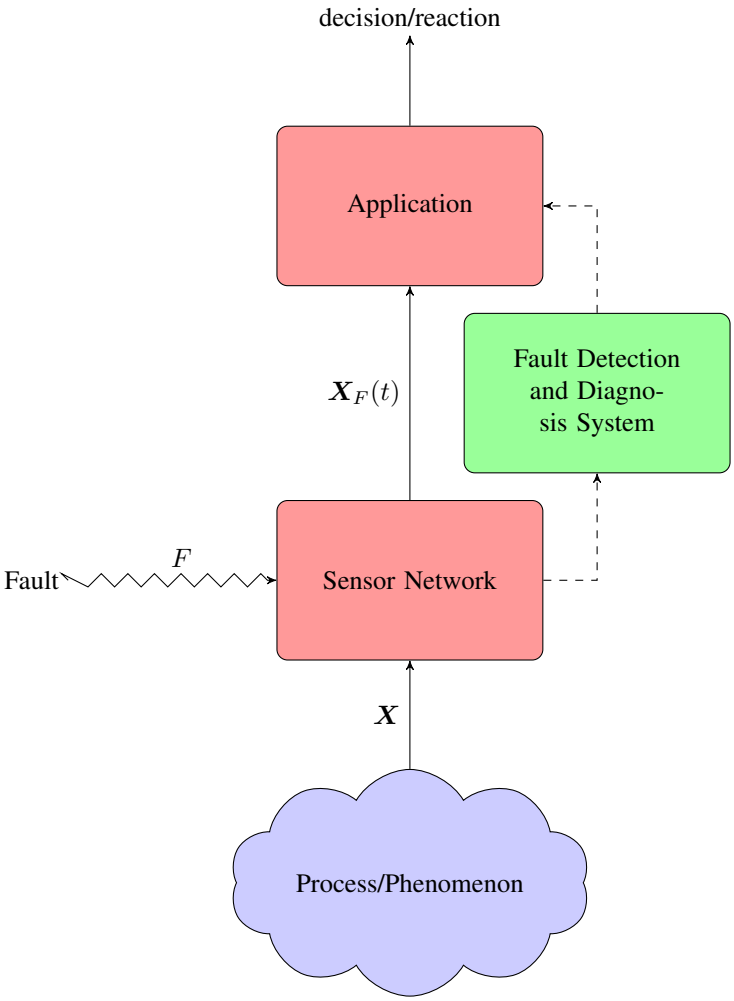


Figure 2.2: *Fault monitoring system: when a fault F occurs, the measurements provided by the sensor network $\boldsymbol{X}_F$ are not describing the behaviour of the underlying system/phenomenon, given by $\boldsymbol{X}$, i.e., $\boldsymbol{X}_F \neq \boldsymbol{X}$. FDDSs are required to detect if this deviation from nominal condition occurs and to gather information about its causes.*

specific time profile and signature. The same applies to all the elements in $\Phi(t)$, i.e., $[\Phi(t)]_j = \phi_j(t) = F_j(t)$.

On the basis of the above definition we can define two different classes of faults affecting a system: additive and multiplicative.

Definition 2.3.1. *An additive fault at time t affecting a sensor network providing $\boldsymbol{X}(t)$ is defined by $\Gamma(t) = 0$ (null matrix of order n) and $\phi(t) \neq 0$, i.e.,*

$$\boldsymbol{X}_F(t) = \boldsymbol{X}(t) + \Phi(t). \tag{2.20}$$

Definition 2.3.2. *A multiplicative fault at time t affecting a sensor network providing $\boldsymbol{X}(t)$ is defined by $\Gamma(t) \neq 0$ and $\phi(t) = 0$, i.e.,*

$$\boldsymbol{X}_F(t) = [I_n + \Gamma(t)]\, \boldsymbol{X}(t). \tag{2.21}$$

There exist also some kind of faults which cannot be categorized into the two definitions presented above. For instance, a commonly occurring fault is the stuck-at, which results in the freezing of the measurement of a sensor on a single value over time, can be modeled as $\Gamma(t) = -I_n$ and $\Phi(t) = \bar{C}, \bar{C} \in \mathbb{R}^n$, i.e., resulting in:

$$\boldsymbol{X}_F(t) = \bar{C}. \tag{2.22}$$

2.4 Examples of Faults

In the sequel, we present some examples of faults affecting a system: each fault is coupled with a real one occurring in a sensor network designed to forecast rock landslides and collapses deployed at the Towers of Rialba, in Northern Italy.

Additive Permanent Fault Here we want to model an additive permanent fault which occurs in a sensor network composed of $n = 3$ univariate sensors $\mathcal{S} = \{s_1, s_2, s_3\}$. The fault is influencing only the data coming from the third sensor s_3, occurs at time t_i with an abrupt profile and has an unknown profile $\mu(t)$. This fault models the introduction of a dynamic signal in the nominal behaviour of the sensor, which could be due to an external cause. By following the modeling we presented in Section 2.2 we have:

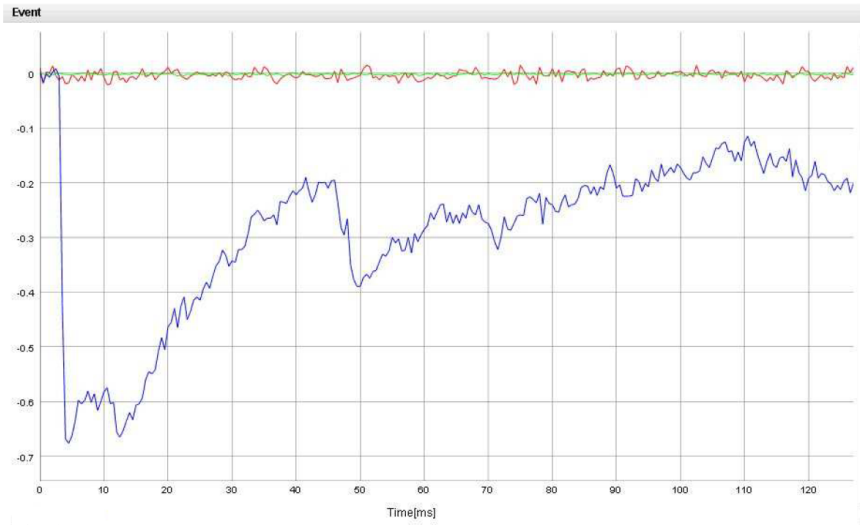


Figure 2.3: *Fault affecting the mounting of units on the rock face; only one of the signals is affected.*

$$\text{Multiplicative} \qquad \Gamma(t) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

$$\text{Additive} \qquad \Phi(t) = \begin{pmatrix} 0 \\ 0 \\ F(t) \end{pmatrix};$$

$$\text{Fault} \qquad F(t) = B(t; t_i, t_f)\phi(t);$$

$$\text{Time signature} \qquad B(t; t_i, t_f) = \beta_i(t - t_i) - 0 = \begin{cases} 0, t \leq 0 \\ 1, t > t_i \end{cases};$$

$$\text{Profile} \qquad \phi(t) = \mu(t).$$

By observing the data acquired by an accelerometer, shown in Figure 2.3, one may observe that the fault occurring at time $t_i = 7$ has the time profile and signature described above. The anomalous behaviour of the signal in the figure is classified as a fault since it is strange that an acceleration coming from the inside affects only one of the axes whereas, apart from very particular situations, all axes should be affected. The rationale

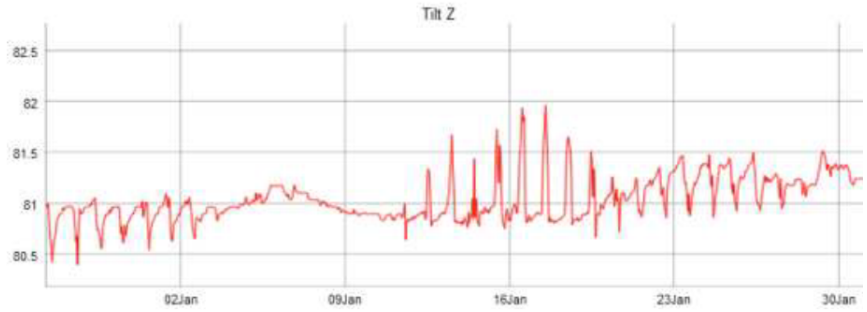


Figure 2.4: *Thermal drift affecting a MEMS tiltmeter. The reading should be constant but the thermal parasitic effects affect the measurements.*

behind the generation of this accelerometer signature is currently unknown, however, we give the following most likely interpretation: the fault appears to be associated with a movement of the nogs and screws system fixing the monitoring unit to the rock. The movement is probably induced by the freezing of the water present in the inner part of the drilled hole which, in turn, pushes the unit outside.

Thermal Drift Here, we want to model a fault affecting a single sensor s_i, which is making the signal drift with fixed slope $R \in \mathbb{R}^+$. The occurrence, at time t_i, of the fault is abrupt and it permanently affects the signal.

$$\begin{aligned}
&\text{Multiplicative} && \gamma_{ii}(t) = 0; \\
&\text{Additive} && \phi_{ii}(t) = F(t); \\
&\text{Fault} && F(t) = B(t; t_i, t_f)\phi(t); \\
&\text{Time signature} && B(t; t_i, t_f) = \beta_i(t - t_i) - 0 = \begin{cases} 0, t \leq t_i \\ 1, t > t_i \end{cases}; \\
&\text{Profile} && \phi(t) = R\,(t - t_i).
\end{aligned}$$

The sensor signal shown in Figure 2.4 is a monoaxial stream coming from a 3-axis Micro Electro-Mechanical Systems (MEMS) tiltmeter. We can see that the fault modeled above is able to represent the parasitic effects induced by variations in temperature over the year (year seasonality) starting from $t_i = 0$. A similar phenomenon arises with deformometers and many other sensors which show a temperature influence over time which needs to be compensated to provide a correct value.

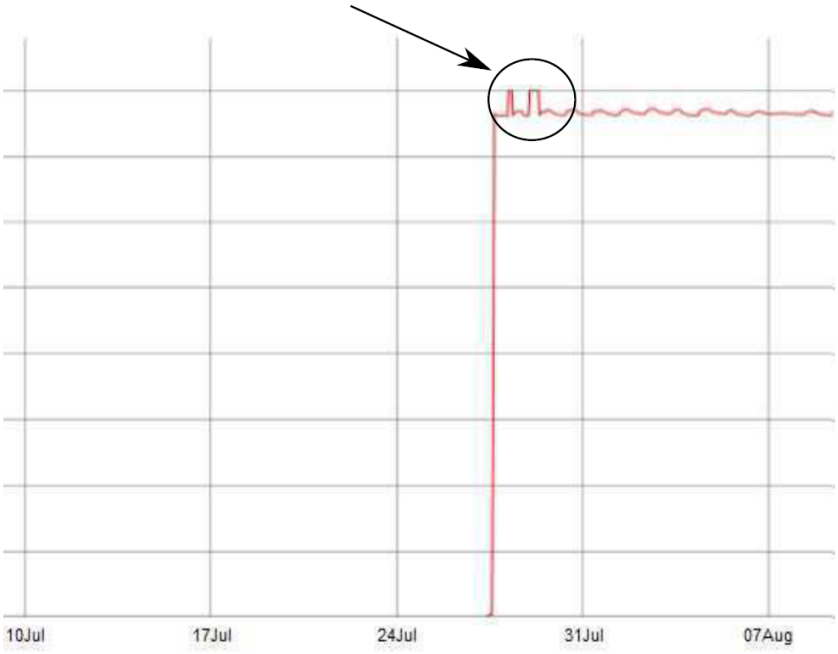


Figure 2.5: *A low-probability intermittent fault at software level*

Intermittent faults Finally we model an intermittent fault composed by $v^* = 2$ time intervals. Both the portions of the fault occur and disappear in an abrupt way and has a fixed magnitude $\theta \in \mathbb{R}$. Its modeling in the proposed framework is:

Multiplicative	$\gamma_{ii}(t) = 0;$
Additive	$\phi_i(t) = F(t);$
Fault	$F(t) = \sum_{\nu=1}^{2} B_\nu(t; t_i^{(\nu)}, t_f^{(\nu)}) \phi_\nu(t - t_i^{(\nu)});$
Time signature	$B_\nu(t; t_i^{(\nu)}, t_f^{(\nu)}) = \begin{cases} 1, & t_i^{(\nu)} \leq t \leq t_f^{(\nu)} \\ 0, & \text{otherwise} \end{cases};$
Profile	$\phi_1(t) = \phi_2(t) = \theta.$

This fault has a real counterpart in the one shown in Figure 2.5: the first time interval of the fault appears at $t_i^{(1)} = 8.00\ July\ 28$ and disappears at $t_f^{(1)} = 16.00\ July\ 28$ and the second one appears at $t_i^{(2)} = 0.00\ July\ 29$

and disappears at $t_f^{(2)} = 12.00\ July\ 29$. After an analysis of the system performed by an expert, it was assessed that the real fault was induced by a software bug which impacted on the data saved in the DataBase (DB).

2.5 Purposes of Fault Detection and Diagnosis

Once the model for the data coming from the sensor network $\mathcal{S}$ and for the possibly occurring faults F is defined, we want to define the task which a FDDS should perform in this scenario. Suppose that a given fault $\hat{F}(t)$, with specific time profile $\hat{B}(t; t_i, t_f)$ and signature $\hat{\phi}(t)$, occurs in the sensor network $\mathcal{S} = \{s_1, \ldots, s_n\}$. Clearly it will influence the data coming from the sensor network, which will be $\boldsymbol{X}_{\hat{F}}(t)$, with either additive $\phi_i(t)$ or multiplicative $\gamma_{ii}(t)$ component[2]. At first, we need to understand whether a change in the behaviour of $\mathcal{S}$ has occurred, usually by relying on the datastream $\boldsymbol{X}$ coming from the sensor network itself. The fault detection task may be formalized as follows:

Definition 2.5.1. *The fault detection task on the sensor network $\mathcal{S}$ for the faulty data $\hat{F}$ consists in identifying as soon as possible the occurrence of a fault. $t' \in \mathbb{N}$ is the time instant at which the fault $\hat{F}$ is detected.*

We have a positive detection if $t_i < t' < t_f$. The performance of a fault detection system may be evaluated on the basis of how long it takes to detect that a change has occurred, i.e., we want to minimize $t' - t_i$ (detection delay), as well as not to provide false positive detection ($t' < t_i$) or false negative ($t' \to \infty$).

Before trying to gather more information about the nature of the fault, we need to understand if the change detected in the previous phase is due to a change in the underlying system $\mathcal{P}$ or if it is due to a malfunction in one of the sensors. If a fault is detected, we want to isolate the sensor (or the unit of the network) where it has occurred. Thus the isolation task can be formalized as:

Definition 2.5.2. *The fault isolation task on the sensor network $\mathcal{S}$ for the fault $\hat{F}$ is meant to determine the sensor $s_{\hat{i}}$ s.t.:*

$$\exists \bar{t} \geq t_i \mid \gamma_{\hat{i}\hat{i}}(\bar{t}) \neq 0 \vee \phi_{\hat{i}}(\bar{t}) \neq 0, \tag{2.23}$$

where t_i is the time of appearance of the fault, i.e., the one which is influenced by the multiplicative or additive fault, respectively.

[2]For sake of concision we here consider a fault occurring to a single sensor s_i and during a single time interval $\nu^* = 1$

We here considered the formulation of the task in its more conservative formulation, since it may suffice to specify the location of the fault in terms of unit. In fact, if the fault is isolated in a sensor $s_{\hat{i}}$, there exists a single unit u_j s.t. $s_{\hat{i}} \in u_j$.

At last once we detected and isolated the fault, we would like to have more information about the characteristics it presents. Thus the identification task is defined as:

Definition 2.5.3. *The fault identification task on the sensor network $\mathcal{S}$ for the fault $\hat{F}$ is to determine the fault $F_h \in D$, where D is a predetermined fault dictionary, s.t. $\hat{F} = F_h$, where the equality is in term of time profile and signature, i.e., to be able to classify which fault has occurred choosing from a fixed set of faults.*

Clearly this task requires the knowledge or the definition of a fixed set of faults. In the considered CFDDS this is not the case, thus the identification task may be extended with the fault dictionary learning task:

Definition 2.5.4. *The fault identification task on the sensor network $\mathcal{S}$ for the fault $\hat{F}$ is to determine a set of faults $\hat{D}$ possibly occurring to the sensor network $\mathcal{S}$ and, after that, to determine the fault $F_h \in \hat{D}$ s.t. $\hat{F} = F_h$.*

CHAPTER 3

The Proposed Cognitive Fault Detection and Diagnosis System

In the sensor network scenario, the modeling of a process $\mathcal{P}$ is a hard task, since a priori information about nominal conditions or the faults signatures is generally not available. In this situation these models can be inferred from measurements $\boldsymbol{X}$ provided by the sensor network $\mathcal{S}$. By relying on these data, the proposed CFDDS is able to characterize the nominal conditions and learn the structure of the system inspected by the sensor network and to perform detection and diagnosis. In the approach we considered in this dissertation, the detection procedure is analogous with the one described before, while the diagnosis phase specifically addresses to the sensor network scenario. At first, the identification here refers to the ability to distinguish between two possible causes for the change of the nominal conditions:

- **fault in a sensor**: the measurements provided by a sensor are no longer reliable, due to a malfunction of the sensor itself or in its processing board or in the channel where data are transmitted. The process $\mathcal{P}$ inspected by the network does not deviate from its nominal state;

- **change in the inspected system**: the measurements coming from the sensor network do not follow the behaviour they had in nominal conditions. This change is due to the fact that the process $\mathcal{P}$ is now in an alternative state.

These causes trigger completely different countermeasures. For instance, in the environmental monitoring for critical infrastructure applications we presented before, when a fault in the sensor is detected, a maintenance action should be performed. Otherwise, in the case of a change in the process, which here corresponds to a change in the environment, an alarm should be raised, since it might mean that a threat might affect the critical infrastructure (i.e., a rock collapse has happened or is about to happen).

In the case a fault has been identified, further efforts have to be considered in the isolation phase. In fact, once the nature of the change has been discriminated, the information about the location, i.e., the sensor s_i, where the fault took place, should be inferred. Finally, once the location of the fault is correctly identified, a characterization of the fault should be given, e.g., to be able to identify recurring faults.

The proposed CFDDS is based on the analysis of the functional relationships existing among acquired sensor measurements. Thanks to a novel algorithm able to learn both the dependency structure present among the datastreams and the functional constraints present between couples of sensors, the proposed CFDDS is able to perform fault detection and diagnosis. In fact, the change in a functional relationship allow us to detect if a change (in the environment or caused by a fault) has happened, while the analysis of the dependency structure is used to isolate the fault in a specific sensor of the network. Finally, if the fault is detected and isolated, the system is able to build in an on-line manner the fault dictionary and identify the occurred fault by comparing it with the ones contained in the fault dictionary.

In relation to the detection topology described in Figure 1.2 the proposed CFDDS could be categorized as a data-driven approach, since the approximating models are estimated directly from data, without considering any signal- or process-based technique. By taking into account the diagnosis taxonomy presented in Figure 1.3, the isolation phase is more likely to be considered as an inference method, while the identification one can be inserted in the statistical classification category.

The overall architecture of the proposed CFDDS is presented in Section 3.1. After that, the definition of the dependency graph and the theoretical justification for the proposed methodology are provided in Section 3.2 and Section 3.3, respectively. Then, the approach of fault diagnosis in the parameter space, considered in the proposed CFDDS is detailed in Sec-

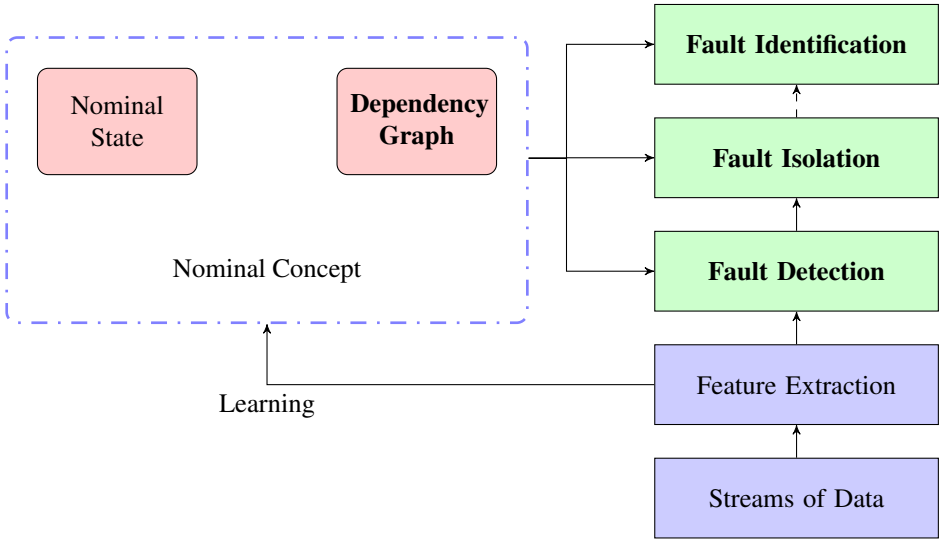


Figure 3.1: *General architecture of the proposed CFDDS: from a set of streams of data, features are extracted and a nominal concept is learned. After that, the CFDDS detects deviation from the nominal behaviour of the process and isolate the cause of the change. Only if the cause is identified to be a fault in a sensor, the identification phase is performed, to better characterize the occurred fault.*

tion 3.4. Finally, a brief description of the detection, isolation and identification phases of the proposed CFDDS is described in Section 3.5.

3.1 General Architecture

The general architecture of the proposed CFDDS is depicted in Figure 3.1, while the corresponding high level pseudoalgorithm is provided in Algorithm 1. The proposed system initially learns the dependency structure of the network, by means of a dependency graph, and the functional relationships $f_{(i,j)}$ between data coming from pairs of sensors s_i and s_j for each corresponding edge in the dependency graph, by relying on a fault-free training sequence. The dependency graph is learned directly from data, by relying on the causality concept, while the characterization of each functional relationship relies on the estimation of the parameters of an approximating LTI model and by considering them as features for detection, isolation and identification purposes. The estimated functional relationships and the dependency graph represent the *nominal concept* of the sensor network. Deviations from this nominal concept represent anomalous situations that must be detected and inspected by the proposed CFDDS. The training

Algorithm 1 CFDDS High Level Algorithm

```
1: Off-line phase
2: Dependency graph learning;
3: Nominal state characterization;
4: On-line phase
5: for each new data vector provided by the sensor network do
6:     Apply the detection phase;
7:     if discrepancy is detected then
8:         Apply the isolation phase;
9:         if a fault is isolated then
10:            Apply the identification phase;
11:        end if
12:    end if
13: end for
```

phase, which is performed in an off-line way, terminates once the nominal concept has been learned. During the operational life, all the functional relationships are inspected to detect variations with respect to the nominal concept (detection phase). When a change in the functional relationship $f_{(i,j)}$ is detected, by monitoring changes in the parameter vectors, this might be induced by a fault (either in sensor s_i or s_j), by a change in the environment in which the sensor network operates or by a false detection, e.g., induced by a model bias in the estimation of the relationship $f_{(i,j)}$. To discriminate among these three cases, the proposed CFDDS exploits a logic partition of the previously learned dependency graph. This last procedure allows to discriminate (identification phase) whether a change has happened in the environment or a fault has occurred in a sensor and, in this case, identify the location where it occurred (isolation phase). Finally, if a fault is detected and isolated, the proposed CFDDS provides its characterization (identification phase) by relying on a novel clustering algorithm in the parameter vector space, which is able to build in an on-line manner the fault dictionary.

3.2 Dependency Graph

As a basis for the proposed CFDDS, the learning of the dependency structure of the considered sensor network is needed. Thus, we considered each binary relationship $f_{(i,j)}$ between the datastream acquired by sensor s_i and sensor s_j, which introduces a constraint between the two datastreams. A dependency graph $\mathcal{G}$ models the relationships existing among datastreams x_is. More formally:

Definition 3.2.1. *Given a sensor network $\mathcal{S}$, a dependency graph $\mathcal{G}$ over $\mathcal{S}$ is defined as:*

$$\mathcal{G} = (V, E), \tag{3.1}$$

where $V = \{s_1, \ldots, s_n\}$ is the set of network sensors and E is a set of directed edges connecting sensors, such that the directed edge $e_{ij} = (s_j, s_i), e_{ij} \in E$ represents the relationship $f_{(i,j)}$ between datastreams $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, provided by s_i and s_j, respectively.

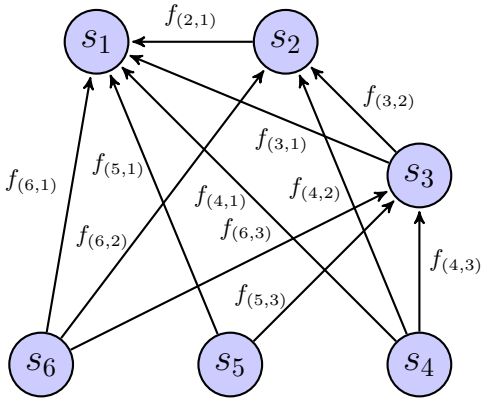


Figure 3.2: *Example of dependency graph for a network with $n = 6$ sensors. $V = \{s_1, \ldots, s_6\}$ and $E = \{e_{21}, e_{31}, e_{32}, e_{41}, e_{42}, e_{43}, e_{51}, e_{53}, e_{61}, e_{62}, e_{63}\}$. For each $e_{ij} \in E$ there is a non-trivial functional relationship $f_{(i,j)}$ between data measured by sensors s_i and s_j.*

An example of a dependency graph for a sensor network is shown in Figure 3.2. In the figure, it is possible to see that the sensor s_1 has functional relationships with all the other ones, thus the values assumed by $s_2, \ldots, s_6$ at a given time instant t influence the value acquired by sensor s_1 at time $t+1$. The same does not apply to sensor s_5, which is not influenced by any other sensor, but it influences s_1 and s_3, i.e., functional relationships $f_{(5,1)}$ and $f_{(5,3)}$ exist. Finally, for instance, the functional relationship $f_{(5,4)}$ is not included in $\mathcal{G}$, since it does not exist or is too weak. In fact, the variation of the value measured by the sensor s_5 does not influence the one in s_4 and vice versa.

We would like to point out that the dependency graph $\mathcal{G}$ does not coincide with the network topology, but might be influenced by it. In fact, it is more likely that two sensors near each other have a stronger functional dependency than far sensors. Nonetheless, depending on the specific application also distant sensors could be related as well, for instance the ocean temperature of water in the Mexican gulf influences the one on the coasts

of Northern Europe due to the Gulf Stream, thus a strong functional relationship exists, even though those locations are topologically distant.

3.3 Modeling Functional Relationships Between Pairs of Sensors

Given the dependency graph of the sensor network $\mathcal{G}$, a modeling technique to analyse each single relationship between a couple of sensors $f_{(i,j)}$ is needed. We emphasize that the theoretical results presented here are also valid for the case of a generic Multiple Input Single Output (MISO) functional relationship.

In the following, a functional relationship $f_{(i,j)}$ between the datastream provided by sensors s_i and s_j is approximated through a LTI predictive model belonging to a family $\mathcal{M}$ parametrised in $\theta \in \mathcal{D}_{\mathcal{M}}$, $\mathcal{D}_{\mathcal{M}} \subset \mathbb{R}^p$ being a compact C^1 manifold. MISO linear predictive models [60], Extreme Learning Machines (ELMs) [42], Reservoir Networks (RNs) [47, 74] are valuable instances for $\mathcal{M}$. A complete characterization of the properties needed for the considered space is provided in the sequel. In this dissertation we opt to present the modeling methodology as a generic linear one-step-ahead predictive model in the form:

$$\hat{x}_j(t|\theta) = \hat{f}_{(i,j)}\left(t, \theta, x_j(t-1), \ldots, x_j(t-\tau_j), x_i(t), \ldots, x_i(t-\tau_i+1)\right), \tag{3.2}$$

where $\hat{f}_{(i,j)} : \mathbb{N} \times \mathbb{R}^p \times \mathbb{R}^{\tau_j} \times \mathbb{R}^{\tau_i} \to \mathbb{R}$ is the approximating function in predictive form [59], e.g., Auto Regressive eXogenous (ARX), Auto Regressive Moving Average eXogenous (ARMAX) model, $t \in \mathbb{N}$ is the considered time instant, $x_i(t), x_j(t) \in \mathbb{R}$ are the model input and output at time t, respectively, and τ_i and τ_j are the orders of the input and output, respectively. Given a training sequence $\{(x_i(t), x_j(t))\}_{t=1}^{N}$ of length N and a quadratic loss function, we define the structural risk [59] to be:

$$W_N(\theta) = \frac{1}{N}\sum_{t=1}^{N} \mathbb{E}_{(\boldsymbol{x}_i, \boldsymbol{x}_j)}\left[\varepsilon^2(t,\theta)\right], \tag{3.3}$$

where $\mathbb{E}_{(\boldsymbol{x}_i, \boldsymbol{x}_j)}[\cdot]$ is the expected value over the space of the datastreams $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ of length N, and the empirical risk as:

$$V_N(\theta) = \frac{1}{N}\sum_{t=1}^{N} \varepsilon^2(t,\theta), \tag{3.4}$$

where $\varepsilon(t,\theta) = x_j(t) - \hat{x}_j(t|\theta)$ is the prediction error at time t. The optimal parameter $\theta^o \in \mathcal{D}_\mathcal{M}$ is defined as

$$\theta^o = \arg\min_{\theta\in\mathcal{D}_\mathcal{M}} \left[\lim_{N\to+\infty} W_N(\theta)\right]. \tag{3.5}$$

An estimate $\hat{\theta} \in \mathcal{D}_\mathcal{M}$ of θ^o can be obtained by minimizing the empirical risk:

$$\hat{\theta} = \arg\min_{\theta\in\mathcal{D}_\mathcal{M}} V_N(\theta). \tag{3.6}$$

It has been showed that the parameter vector $\hat{\theta}$ is characterized by a specific asymptotic distribution. More formally, the following theorem holds:

Theorem 3.3.1 (Ljung [58, 59])**.** *Consider a process $\mathcal{P}$, a model space $\mathcal{M}$ parametrized by $\theta \in \mathcal{D}(\mathcal{M})$, where $\mathcal{D}(\mathcal{M})$ is a compact C^1 differentiable manifold in $\mathbb{R}^p$, a function $l(\cdot,\cdot,\cdot) : \mathbb{N}\times\mathbb{R}^p\times\mathbb{R}\to\mathbb{R}^+$ and a structural risk function W_N. Under the hypotheses that:*

1. *$\exists C \in \mathbb{R}^+, \exists\lambda \in \mathbb{R}^+, \lambda \le 1, \exists\delta \in \mathbb{R}^+, \delta > 0$ s.t. $\forall t, \forall s$ s.t. $t \ge s, \exists x^0_{is}(t), \exists x^0_{js}(t)$ random vectors generated by $[x_i(0),\dots,x_i(t)]$ and $[x_j(0),\dots,x_j(t)]$ respectively, independent from $[x_i(0),\dots,x_i(s)]$ and $[x_j(0),\dots,x_j(s)]$ s.t.:*

$$\mathbb{E}_{(x_i,x_j)}\left[|x_j(t) - x^0_{js}(t)|^{4+\delta}\right] < C\lambda^{t-s} \tag{3.7}$$

$$\mathbb{E}_{(x_i,x_j)}\left[||x_i(t) - x^0_{is}(t)||^{4+\delta}\right] < C\lambda^{t-s} \tag{3.8}$$

   where $||\cdot||$ is an appropriately defined norm and $u^0_t(t) = 0$, $y^0_t(t) = 0$;

2. *$\exists C \in \mathbb{R}, \exists\lambda \in \mathbb{R}^+, \lambda \le 1, \forall\theta \in \mathcal{D}_\mathcal{M}$ (thus $\forall f \in \mathcal{M}$)*

$$\begin{aligned} &|f(\theta, x^t_{i1}, x^{t-1}_{j1}) - f(\theta, x^t_{i2}, x^{t-1}_{j2})| \\ &\le C\sum_{s=0}^{t}\lambda^{t-s}\left[||x_{i1}(s) - x_{i2}(s)|| + |x_{j1}(s) - x_{j2}(s)|\right], \end{aligned} \tag{3.9}$$

$$|f(\theta, 0^t, 0^{t-1})| \le C, \tag{3.10}$$

   where $x^t = (x(1),\dots,x(t)$ is a sequence drawn from the datastream distribution, $||\cdot||$ is a properly defined norm and $0^t = (0,\dots,0)$, i.e., a particular Lipschitz continuity condition [76] suited for the space we are considering;

3. *$\forall f(\theta,\cdot) \in \mathcal{M}(\theta)$, $f(\theta,\cdot)$ is three times differentiable w.r.t. $\theta \in \mathcal{D}_\mathcal{M}$ and these derivatives satisfy the conditions of Point 2;*

4. $\exists C \in \mathbb{R}^+, \forall \theta \in \mathcal{D}_{\mathcal{M}}, \forall t \in \mathbb{N}$ *s.t.:*

$$\left\| \frac{\partial^k}{\partial \theta^k} l(t, \theta, \varepsilon) \right\| \leq C||\varepsilon||^2 \quad k = 1, 2, 3 \tag{3.11}$$

$$\left\| \frac{\partial^k}{\partial \theta^k} \frac{\partial}{\partial \varepsilon} l(t, \theta, \varepsilon) \right\| \leq C||\varepsilon|| \quad k = 0, 1, 2 \tag{3.12}$$

$$\left\| \frac{\partial^k}{\partial \theta^k} \frac{\partial^2}{\partial \varepsilon^2} l(t, \theta, \varepsilon) \right\| \leq C \quad k = 0, 1 \tag{3.13}$$

$$\left\| \frac{\partial^3}{\partial \varepsilon^3} l(t, \theta, \varepsilon) \right\| \leq C \tag{3.14}$$

where $|| \cdot ||$ *is a properly defined matrix norm;*

5. $\exists \delta \in \mathbb{R}^+,\ \exists N_0 \in \mathbb{R}^+$ *s.t.* $\forall \theta \in \mathcal{D}_{\mathcal{M}},\ N \geq N_0$*:*

$$W_N''(\theta) > \delta I_p, \tag{3.15}$$

where W_N'' *is the Hessian matrix of the structural risk*

$$W_N(\theta) = \frac{1}{N} \sum_{t=1}^{N} \mathbb{E}_{(x_i, x_j)} \left[l(t, \theta, \varepsilon(t, \theta)) \right] \tag{3.16}$$

w.r.t. θ *and* I_p *is the identity matrix of order* p*;*

6. $\exists \delta \in \mathbb{R}^+$ *s.t.* $\Sigma_N > \delta I_p, U_N > \delta I_p$ *where* $\Sigma_N \in \mathbb{R}^{p \times p}$ *is defined as follows:*

$$\Sigma_N = \left[W_N''(\theta^o) \right]^{-1} U_N \left[W_N''(\theta^o) \right]^{-1}, \tag{3.17}$$

where $U_N \in \mathbb{R}^{p \times p}$*:*

$$U_N = N \mathbb{E}_{(x_i, x_j)} \left[V_N'(\theta^o) V_N'(\theta^o)^T \right] \tag{3.18}$$

where it is required that the matrix Σ_N *is semidefinite positive, in order to be a proper covariance matrix;*

then an estimated parameter vector $\hat{\theta}$ *estimated by minimizing the empirical risk:*

$$V_N(\theta) = \frac{1}{N} \sum_{t=1}^{N} l(t, \theta, \varepsilon(t, \theta)), \tag{3.19}$$

on a sequence $\{(x_i(t), x_j(t))\}_{t=1}^{N}$ *satisfies*

$$\lim_{N \to \infty} \hat{\theta} \to \theta^o \quad w.p.\ 1 \tag{3.20}$$

and

$$\lim_{N\to\infty} \sqrt{N}\Sigma_N^{-\frac{1}{2}}(\hat{\theta} - \theta^o) \sim \mathcal{N}(0, I_p). \tag{3.21}$$

The above result assures that, given a sufficiently large dataset, composed by N samples, the estimated parameter vector $\hat{\theta}$ follows a multivariate Gaussian distribution with mean θ^o and covariance matrix Σ_N. Interestingly, the results presented above contemplate the situation where $\mathcal{P} \notin \mathcal{M}$ i.e., a model bias $||\mathcal{M}(\theta^o) - \mathcal{P}|| \neq 0$ is present. This justifies the use of LTI models even when the dynamic system under investigation is non-linear. According to Equation (3.21), estimated parameters $\hat{\theta}$s follow a multivariate Gaussian distribution both approximating linear and nonlinear systems, provided that a sufficiently large dataset is available. We emphasize that, in what follows, we are not interested in providing a high approximation accuracy, since LTI models are not used for prediction purposes (where nonlinearities in the system might induce a high prediction error) but for fault diagnosis. Parameter vectors are the features to be used for fault diagnosis and, since a change in the probability density function of the parameter/features is associated with structural changes in the process generating the data (and non-linearity does not introduce structural changes), we can design a CFDDS that relies on an analysis in the parameter space.

Although the nonlinearity aspect is contemplated by the theory, we might experience numerical problems in correspondence with an ill conditioned Hessian W_N'', e.g., following highly correlated inputs. However, we must comment that if W_N'' degenerates in rank then, given the linearity assumption for the considered approximation model, we should simply remove the linear dependent variables. In the case we wish to keep them for the (small) innovation they provide, a Levenberg-Marquardt (LM) correction $W_N'' + \delta I_p$ (δ being a small positive scalar) should be introduced to grant a definite positive Hessian.

Extension to scenarios where the data generating process $\mathcal{P}$ is described by a finite set of non-overlapping processes $\{\mathcal{P}_1, \ldots, \mathcal{P}_\psi\}$ is immediate, as long as one knows the states of the system. Clearly this extension requires to have a priori information about the data generating process, for instance on the seasonality of the system inspected by the sensor network. Moreover, here we considered as model the MISO models, but an extension to Multiple Input Multiple Output (MIMO) and Single Input Multiple Output (SIMO) is trivial, e.g., by considering a set of the aforementioned functional relationships.

3.4 Feature in the Parameter Space

From the theoretical results delineated in Section 3.3 parameter vectors estimated from fault-free data are distributed according to a Gaussian distribution, provided that N is large enough (even though the system is nonlinear). According to Equation (3.21), the nominal state of a system can be described by parameter vectors estimated from data coming from the process, thus sampled from the same Gaussian distribution (each sampled point is a model).

Given the process $\mathcal{P}$ and the sensor network providing the multivariate datastream $\boldsymbol{X}$, let us define $Z_{N,t} \in \mathbb{R}^{n \times N}$ as a finite sequence from time $t - N + 1$ to t extracted from $\boldsymbol{X}$. More formally:

$$Z_{N,t} := [\boldsymbol{X}(t - N + 1), \ldots, \boldsymbol{X}(t)], \tag{3.22}$$

where $N \in \mathbb{N}$ is the length of the sequence extracted from the datastream and $t \geq N, t \in \mathbb{N}$ is the time instant the sequence of data considered ends. Moreover, if we want to consider only data coming from a specific couple of sensors (s_i, s_j) we define $Z^{(ij)}_{N,t} \in \mathbb{R}^{2 \times N}$ as:

$$Z^{(ij)}_{N,t} = \begin{bmatrix} x_i(t - N + 1) & \ldots & x_i(t) \\ x_j(t - N + 1) & \ldots & x_j(t) \end{bmatrix}, \tag{3.23}$$

i.e., the sequence of samples coming from sensors s_i and s_j, starting from time $t - N + 1$ and ending at time t.

For each couple of sensors s_i and s_j, the proposed CFDDS relies on an initial fault-free training sequence $Z^{(ij)}_{M,M}$, i.e., the acquired samples acquired from time $t = 1$ up to $t = M$, to characterize the nominal state of the process. For fault diagnosis purposes the training set $Z^{(ij)}_{M,M}$ is windowed into non-overlapping batches $Z^{(ij)}_{N,t}$ of length N, each of which is used to provide a parameter vector estimate $\hat{\theta}_q, q = \frac{t}{N}$. The outcome is the sequence of $L = \frac{M}{N}$[1], $L \in \mathbb{R}^+$ parameter vectors:

$$\Theta^{(ij)}_{L,M} = \left(\hat{\theta}^{(ij)}_1, \ldots, \hat{\theta}^{(ij)}_L\right), \tag{3.24}$$

where the parameter vector $\hat{\theta}^{(ij)}_q$ is estimated by minimizing the empirical risk on the dataset $Z^{(ij)}_{N,qN}$, as described in Section 3.3.

[1] Without loss of generality we assume M is a multiple of N

We also considered estimate parameter vectors computed on overlapping batches of the training set $Z^{(ij)}_{M,M}$. The obtained sequence is:

$$\Theta'^{(ij)}_{L,M} = \left(\hat{\theta}^{(ij)}_q\right)_{q=1}^{L} \tag{3.25}$$

where $L = M - N + 1$ and the parameter vector $\hat{\theta}^{(ij)}_q$ is estimated by minimizing the empirical risk on the dataset $Z^{(ij)}_{N,N+q-1}$. Without loss of generality, we assume a step of one sample between two overlapping batches; larger steps could be considered as well (e.g., to reduce the computational complexity of the following computations).

The twofold method for the estimation of the parameter vectors is due to the fact that the former one is able to provide a clear statistical characterization for the parameter vector sequence distribution, since they are i.i.d., which can be used during the identification phase, while the latter lacks of this characterization but provides a parameter vector for each sample coming from the sensor network, which is more suited for the detection phase, since it provides a more fine grained characterization of the process behaviour.

3.5 The Phases of the Proposed CFDDS

In the sequel, details about the phases considered in the proposed CFDDS are provided. They corresponds to the blocks with bold names in Figure 3.1 and will be treated in detail in Chapters 4 to 7.

3.5.1 Graph Learning

The proposed CFDDS makes use of the concept of Granger causality to infer the existing causal dependencies among datastreams [15]. Thanks to a statistical test performed directly on the training set $Z_{M,M}$, we are able to assess the causal dependencies between couples of datastreams. If the presence of the functional dependency $f_{(i,j)}$ is assessed, i.e., if the test provides enough statistical confidence for the considered causality, an edge is considered in the edge set E of the dependency graph $\mathcal{G} = (V, E)$. The graph $\mathcal{G}$ is not able to provide any diagnostic information per se, but it is used in the following phases: the detection and identification phases are performed only on functional relationships in E, while the isolation phase is based on the logic partition of the dependency graph. The dependency graph learning procedure is detailed in Chapter 4.

3.5.2 Detection

The detection phase is performed on all the binary functional relationships present in the dependency graph $\mathcal{G}$. Each functional relationship the nominal concept is characterized by the sequence of parameter vectors $\Theta'_{L,M}$, estimated on overlapping batches of data coming from a fault-free training sequence $Z_{M,M}$. The statistical behaviour of the parameter vectors sequence is modeled by means of an approach based on HMMs [72]. By relying on this characterization, it is possible to evaluate the statistical compatibility of a sequence of newly estimated parameter vectors $\Theta'_{k,t}$ at time t by computing the loglikelihoods. A detection happens when the loglikelihood of the HMM decreases below an automatically determined threshold. The detailed description of the proposed cognitive fault detection phase is provided in Chapter 5.

3.5.3 Isolation

The isolation phase is performed by relying on the learned dependency graph of the sensor network $\mathcal{G}$. Once a change has been detected, a cognitive level, which exploits a logical partition of the dependency graph $\mathcal{G}$, is activated. This mechanism is able to distinguish among a fault, a change in the environment and the presence of model bias [8, 12]. When a fault affects a sensor, the relationships that are connected to that sensor will perceive a change (by means of a loglikelihood decrease). When there is change in the environment, all the relationships in E are affected. Finally, when there is model bias in the considered relationship, none of the other relationships in E perceives a change. Furthermore, if the change is associated with a fault, the isolation phase is also able to provide information about its location, in terms of the affected unit. We would like to point out that, even during this phase, the characterization of the nominal concept of the inspected system relying on HMM learned during the detection phase is considered, but for isolation purposes. The isolation logic is formalized and described in Chapter 6.

3.5.4 Identification

Once the fault is detected and isolated, the proposed CFDDS algorithm is able to provide a characterization of the occurred fault [10, 11]. This phase is performed by means of a novel evolving clustering algorithm. During the training phase, a characterization of the nominal state of the system is performed by means of a clustering of the parameter vectors sequence $\Theta_{L,M}$, which are estimated on non-overlapping batches of data coming

from $Z_{M,M}$. The theoretical results presented above allow to specifically design statistics to check if newly estimated vectors belongs to the nominal state, according to the asymptotic distribution of the parameter vectors in the parameter space. As soon as the presence of a fault is assessed in the previous phases (detection and isolation) of the proposed CFDDS, it is identified by associating it to one of the clusters present in the fault dictionary. This dictionary, which is initially empty, is automatically populated during the operational life of the proposed CFDDS. A new cluster representing a fault is created only if enough confidence on the existence of a new fault is gathered (i.e., by performing a Kolmogorov-Smirnov (KS) multivariate test on the expected and empirical distributions). The clustering mechanism considered for the identification phase is described in Chapter 7.

CHAPTER 4

Dependency Graph Learning

Since the dependency graph is at the basis of the fault detection, isolation and identification phases of the proposed CFDDS, the ability to learn it correctly is fundamental to guarantee the fault detection and diagnosis performance. In particular, the ability to correctly learn the dependency graph is crucial, since it allows the proposed CFDDS to rely only on the relevant relationships (i.e., those relationships that really exist among streams of data). This allows the proposed CFDDS to increase its performance for two main reasons. At first, if we keep not relevant relationships in the dependency graph, false positives in detection might increase, since all relationships are inspected in parallel. Second, if a relationship providing information is not included in the dependency graph, we cannot use it in the isolation and identification phases to distinguish between the occurrence of a fault, a change in the environment or a model bias.

In this section, we describe the dependency graph learning phase of the proposed CFDDS, which is based on a statistical framework based on Granger causality. We recall that the dependency graph $\mathcal{G} = (V, E)$ is associated to the functional relationships existing among data and not to the topology of the network. The aim of the dependency graph learning phase, described in Figure 4.1, is to automatically select those relationships pre-

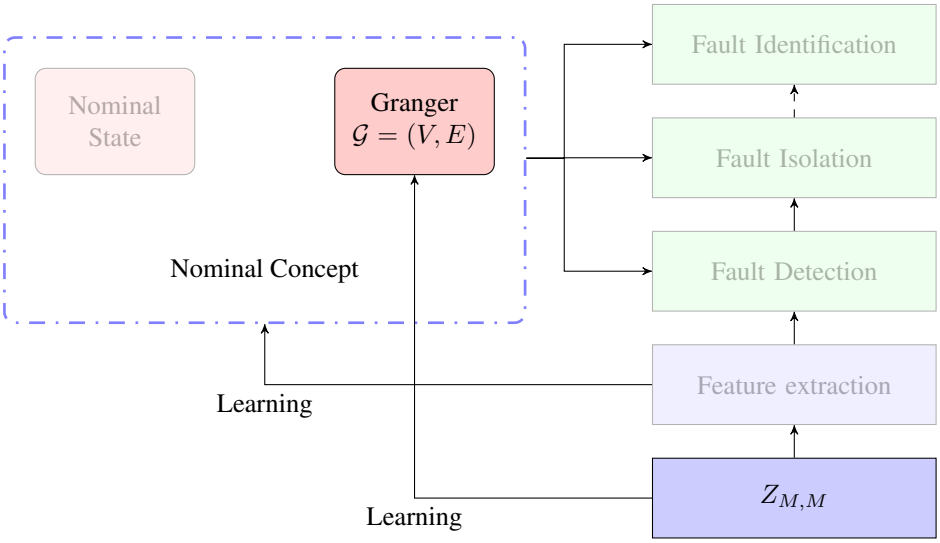


Figure 4.1: *Dependency graph learning phase of the proposed CFDDS: by relying on a fault-free dataset $Z_{M,M}$ and the concept of Granger causality, we select only those relationships which are relevant to the fault detection and diagnosis phases.*

senting causal dependency given a fault-free set of measurements $Z_{M,M}$ coming from the sensor network and a predefined level of confidence α_g regulating the probability of including not relevant relationships in E. In particular, in Section 4.1 we describe the technique for assessing the causal dependency of a single relationship, while in Section 4.2 we extend this learning mechanism to the entire dependency graph $\mathcal{G}$.

4.1 Modeling the Relationship Between a Couple of Datastreams

The level of dependency between datastreams x_i and x_j associated with relationship $f_{(i,j)}$ has been analysed also in other FDDSs. For instance, in [8] the figure of merit used to select functional relationships is the linear correlation index between two datastreams: when the peak of the crosscorrelation is above a suitably tuned threshold $\rho_{\min}$, the relationship is considered to be relevant and worth to be included in E. The main drawbacks of the aforementioned approach are the lack of a clear characterization of the user-defined threshold $\rho_{\min}$ (it is difficult to set it at design time), the capability to assess relationships only between couples of datastreams (a relationship involving more than two datastreams cannot be evaluated through cross-correlation) and the fact that results are heavily dependent on noise,

as pointed out in [72].

In the proposed CFDDS the method chosen to model the dependency between two datastreams relies on *causal dependency* as defined by Granger in [38]:

Definition 4.1.1 (Granger Causality)**.** *Consider a bivariate discrete-time stochastic vector (x_i, x_j). We say that x_i is Granger causing x_j if the presence of x_i allows for better prediction performance of x_j at time t.*

Interestingly, approaches based on the Granger causality has been successfully applied to different application fields like economic [4] and medical applications (e.g., functional Magnetic Resonance Imaging (fMRI) data [32] and ElectroEncephaloGraphy (EEG) analyses [20, 41]). To the best of out knowledge, the Granger causality have never been considered before for fault detection/diagnosis purposes in the existing literature.

Definition 4.1.1 is quite general and relies on the comparison between two different reconstruction abilities: with and without x_i as additional source of information. In the sensor network scenario, a more specific definition of Granger causality can be considered [19]:

Definition 4.1.2 (Multivariate Conditioned Granger Causality)**.** *Consider a multivariate discrete-time stochastic vector $\boldsymbol{X}$, we say that $x_i \in \boldsymbol{X}$ is Granger causing $x_j \in \boldsymbol{X}, j \neq i$, conditioned to $\boldsymbol{X} \setminus \{x_i, x_j\}$, if we are better able to predict x_j by using $\boldsymbol{X}$ instead of using $\boldsymbol{X} \setminus \{x_i\}$.*

Starting from the last definition, it is possible to derive a statistical test able to assess the multivariate conditioned Granger causality between two datastreams. To assess if x_i Granger causes x_j, we need to model the relationship existing among all datastreams provided by the sensor network as a linear Vector AutoRegressive (VAR) model [18, 39]. More specifically, following the Definition 4.1.2, we want to assess the influence of x_i in the prediction of x_j. To achieve this goal we consider two different versions of the VAR model: $\mathcal{M}_f^{(j)}$ (full model), i.e., the one that considers all the available datastreams $\boldsymbol{X}$ to predict x_j, and $\mathcal{M}_r^{(ij)}$ (reduced model), i.e., the one considering as predictors only $\boldsymbol{X} \setminus \{x_i\}$. More formally, the two predictive models assume forms:

$$\mathcal{M}_f^{(j)} : \; x_j(t) = \sum_{k=1}^{\tau} \sum_{h=1}^{n} a_{hjk} x_h(t-k), \tag{4.1}$$

$$\mathcal{M}_r^{(ij)} : \; x_j(t) = \sum_{k=1}^{\tau} \sum_{h=1, h\neq i}^{n} a'_{hjk} x_h(t-k), \tag{4.2}$$

where $a_{hjk} \in \mathbb{R}$ and $a'_{hjk} \in \mathbb{R}$ are the regression coefficients modeling the linear relationship between measurements coming from sensors s_h and s_j at the k-th time lag for model $\mathcal{M}_f^{(j)}$ and $\mathcal{M}_r^{(ij)}$, respectively, and $\tau \in \mathbb{N}$ is the order of the model[1].

It is possible to show that the problem of assessing the multivariate conditional Granger causality can be formulated as a loglikelihood ratio test between the full model $\mathcal{M}_f^{(j)}$ and the reduced model $\mathcal{M}_r^{(ij)}$ [50], where the null hypothesis is that x_i is not Granger causing x_j conditioned to $\boldsymbol{X} \setminus \{x_i, x_j\}$. More specifically, given a training sequence $Z_{M,M}$, we estimate parameters $\hat{a}_{hjk} \in \mathbb{R}$ and $\hat{a}'_{hjk} \in \mathbb{R}$ by mean of a LS procedure applied to $Z_{M,M}$. The loglikelihood ratio test is based on the computation of the test statistic:

$$F_{ij} = \frac{p_f - p_r}{M - p_f - 1} \frac{SSR_r^{(ij)} - SSR_f^{(j)}}{SSR_f^{(j)}} \sim F(p_f - p_r, M - p_f - 1) \quad (4.3)$$

where $p_f = n\tau$ is the number of parameters of the full model $\mathcal{M}_f^{(j)}$, $p_r = (n-1)\tau$ is the number of parameters of the reduced one $\mathcal{M}_r^{(ij)}$, $F(p_f - p_r, M - p_f - 1)$ is the Fisher distribution with $p_f - p_r$ and $M - p_f - 1$ degrees of freedom, $SSR_f^{(j)}$ and $SSR_r^{(ij)}$ are the sum of squared residual on $Z_{M,M}$ of the model $\mathcal{M}_f^{(j)}$ and $\mathcal{M}_r^{(ij)}$, respectively. In the test, the critical region of level α_g for the statistic is:

$$R_{ij}^{\alpha_g} = \{F \in \mathbb{R}^+ | F \geq F_{\alpha_g}(p_f - p_r, M - p_f - 1)\}, \quad (4.4)$$

where $F_{\alpha_g}(p_f - p_r, M - p_f - 1)$ is the quantile of order $1 - \alpha_g$ of the Fisher's distribution with $p_f - p_r$ and $M - p_f - 1$ degrees of freedom. When there is statistical evidence for rejecting the null hypothesis with confidence $1 - \alpha_g$ (i.e., $F_{ij} \in R_{ij}^{\alpha_g}$), we say that $\mathcal{F}_{x_i \to x_j | z}$, i.e., the datastream x_i Granger causes x_j conditioned to $z = \boldsymbol{X} \setminus \{x_i, x_j\}$, since the datastream x_i improves the prediction of x_j. Thus, α_g represents the probability to infer that the use of x_i does not improve the prediction of x_j when it actually does (type I error): the more one chooses low values for α_g, the more the chance that a relationship is included in the learned dependency graph when it does not exists in the real one decreases.

It is worth noting that, in principle, to evaluate the contribution in prediction abilities of datastream x_i to stream x_j, one could inspect the exogenous coefficients corresponding to x_i in the full model $\mathcal{M}_f^{(j)}$, i.e., a_{ijk} $\forall k \in$

[1] Albeit VAR models are linear models, it is possible to show that, under mild assumptions about the data-generating process (e.g., stationarity of the covariance), they are general enough to model several nonlinear multivariate time series [16].

$\{1, \ldots, \tau\}$. When these coefficients are statistically null (meaning that stream x_i does not provide improvement in the prediction of stream x_j), there is no causal dependency of datastreams x_j from x_i. On the contrary, when at least one of the coefficients differs from zero, a causal dependency relation between the two datastreams exists. Interestingly, it is shown in [50] that the problem of identifying a causal dependency between datastreams x_i and x_j, formulated in Equation (4.4), is equivalent to a hypothesis test which assesses if at least one of the coefficients $a_{ijk}, k \in \{1, \ldots, \tau\}$ is statistically different from zero, i.e.:

$$H_0 : a_{ijk} = 0 \; \forall k \quad vs. \quad H_1 \; \exists k \mid a_{ijk} \neq 0. \tag{4.5}$$

4.2 Creating the Granger-based Dependency Graph

Following the definition of Granger causality and the statistical test described above, it is possible to derive a statistical framework to learn the dependency graph for the whole sensor network. A detailed description of the proposed statistical framework is provided in Algorithm 2.

Algorithm 2 Granger-Based Dependency Graph Learning Algorithm

Input: faulty-free dataset $Z_{M,M}$, confidence level α_g
Output: dependency graph $\mathcal{G}$
Set an empty dependency graph $\mathcal{G} = (V, E)$, with $V = \{s_1, \ldots s_n\}$ and $E = \emptyset$
for $j \in \{1, \ldots, n\}$ **do**
  Estimate coefficients $\hat{a}_{ijk}$ of VAR model $\mathcal{M}_f^{(j)}$ in Equation (4.1) from data $Z_{M,M}$;
  for $i \in \{1, \ldots, n\}$ **do**
    if $i \neq j$ **then**
      Estimate coefficients $\hat{a}'_{ijk}$ of VAR model $\mathcal{M}_r^{(ij)}$ in Equation (4.2) from data $Z_{M,M}$;
      Compute F_{ij} as in Equation (4.3);
      if $F_{ij} \geq F_{\frac{\alpha_g}{n(n-1)}}(p_f - p_r, M - p_f - 1)$ **then**
        $E \leftarrow E \cup \{e_{ij}\}$;
      end if
    end if
  end for
end for
return $\mathcal{G}$;

At first, we consider a training set $Z_{M,M}$ of fault-free data coming from the sensor network and a user defined confidence level $\alpha_g \in (0, 1)$ for the loglikelihood ratio test. A graph $\mathcal{G} = (V, E)$ with $V = \{s_1, \ldots, s_n\}$ and $E = \emptyset$ is initially considered. By considering a couple of sensors s_i and s_j

and the dataset $Z_{M,M}$, we are able to compute the test statistics F_{ij}, as in Equation (4.3). When:

$$F_{ij} \geq F_{\frac{\alpha_g}{n(n-1)}}(p_f - p_r, M - p_f - 1), \tag{4.6}$$

we have statistical evidence (with confidence $\frac{\alpha_g}{n(n-1)}$) that x_i Granger causes x_j, conditioned on all the other streams of data, and we add $e_{ij} = (s_i, s_j)$ to the edge set E. We repeat the test for each couple of datastreams to build the dependency graph. Note that we used a confidence level $\frac{\alpha_g}{n(n-1)}$ to obtain an overall confidence level of α_g, thanks to the Bonferroni correction for multiple hypothesis [50]. The correction is here used since $n(n-1)$ hypothesis tests are performed at the same time and it assures to have an overall type I error of α_g.

We emphasize that both the performance and the complexity of the detection and isolation phases of the proposed CFDDS are highly influenced by the learned Granger-based dependency graph. In fact, the detection phase will be performed on each edge $e_{ij} \in E$, while the isolation phase relies on $\mathcal{G}$ to be able to distinguish among model bias, change in the environment and fault and, if this third option occurs, isolate it.

An example of the dependency graph learning phase is presented in Figure 4.2. To determine the dependency graph, a fault-free dataset $Z_{M,M}$ is required. Starting from an completely disconnected graph (in the upper right part of the figure), the Granger-based statistical test considered is able to determine which are the most relevant functional relationships existing among datastreams. The confidence on the overall procedure is given by setting α_g.

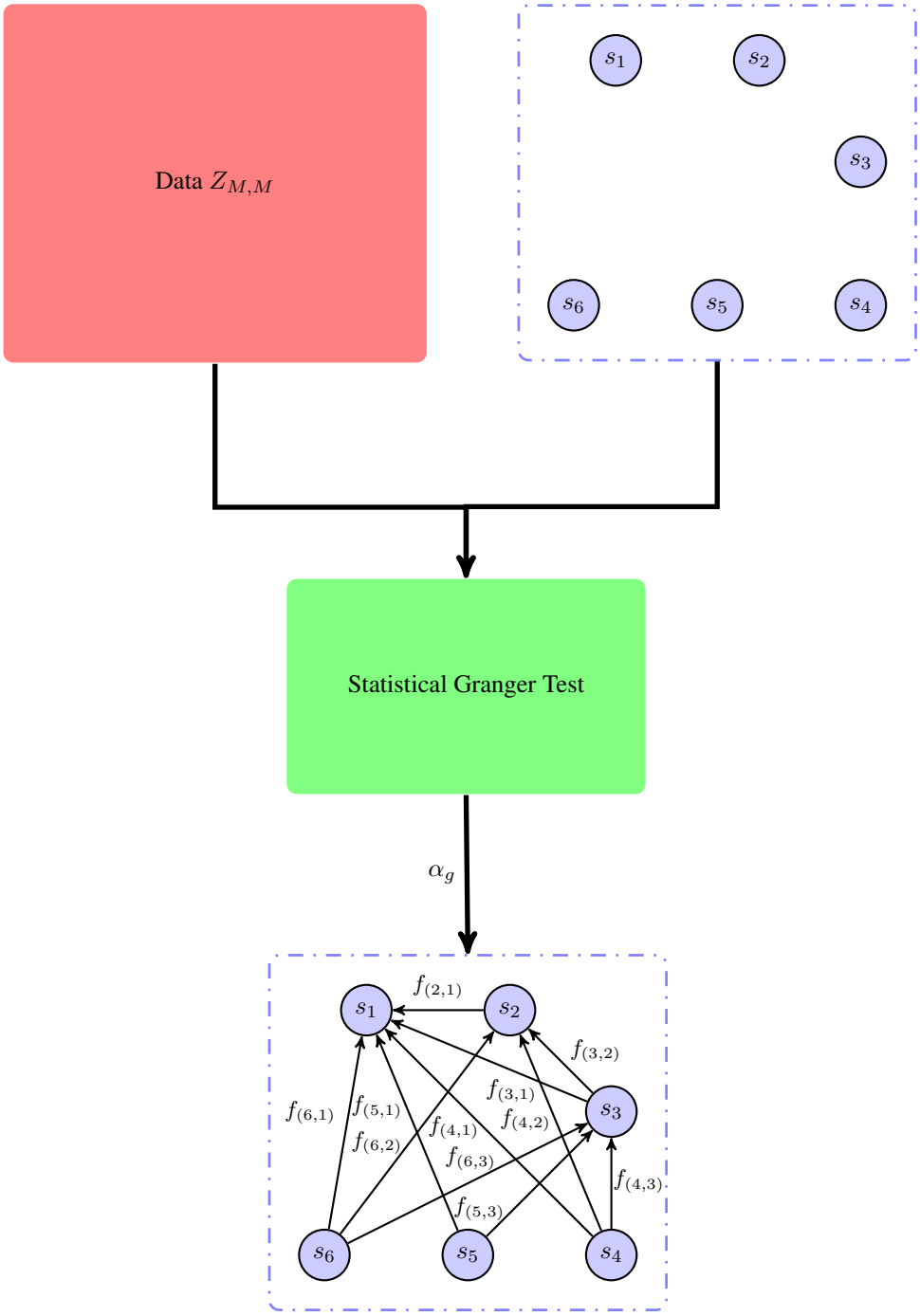


Figure 4.2: *Example of dependency graph learning phase*

CHAPTER 5

Fault Detection

In this chapter, we want to inspect the variation of the behaviour in the relationships $f_{(i,j)}$ such that $e_{ij} \in E$ to understand if a deviation from the nominal concept occurred (detection phase). We suppose to have learned a dependency graph $\mathcal{G} = (V, E)$, which describes the causal relationships existing among datastreams, as described in Chapter 4. In order to detect changes in the behaviour of the data coming from the sensor network, we consider all the binary relationships $f_{(i,j)}$ in parallel. Here we want to learn a characterization of each functional relationship in nominal conditions and be able to detect discrepancies from it.

In this chapter the detection phase of the proposed CFDDS is provided: at first, the general approach in the parameter space is delineated in Section 5.1, then the description of a Mahalanobis-based change detection method is presented in Section 5.2. A description of HMM-CDT, introduced in [7] and further developed in [72], is given in Section 5.3, while the proposed Ensemble approach to Hidden Markov Model-Change Detection Test (EHMM-CDT) is detailed in Section 5.4.

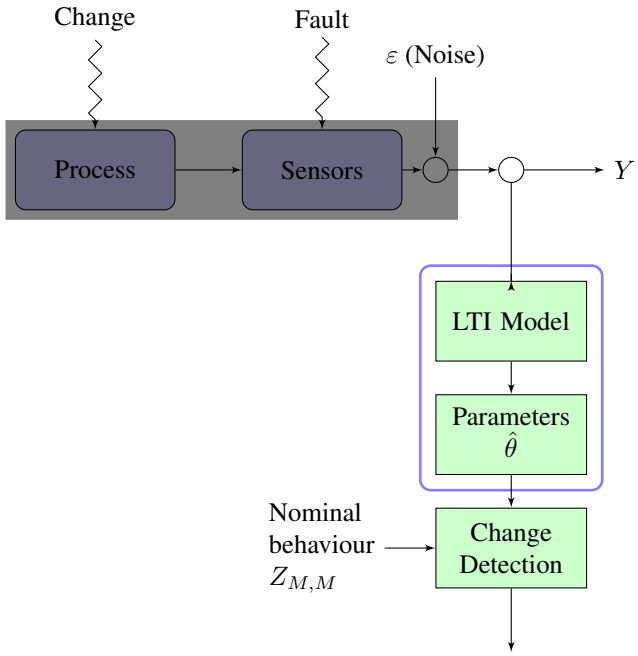


Figure 5.1: *Proposed scheme for fault detection: we do not have a priori information on the process, the sensors or the noise ε.*

5.1 Fault Detection in the Parameter Space

The detection phase of the proposed CFDDS is based on the analysis of the statistical behaviour the estimated parameter vectors presented in Chapter 2. Let us consider a generic functional relationship between a couple of streams $f_{(i,j)}$ s.t. $e_{ij} \in E$. By relying on a fault-free training dataset $Z^{(i,j)}_{M,M}$, it is possible to compute a parameter vectors sequence $\Theta^{(ij)}_{L,M}$ or $\Theta'^{(ij)}_{L,M}$, which characterizes the nominal conditions of the system. After that, we want to inspect newly estimated parameter vectors of the functional relationships $f_{(i,j)}$ selected during the graph learning phase and detect variations in their behaviour.

The general approach considered in this dissertation is presented in Figure 5.1. Since in the sensor network scenario the generating data process is unknown, as well as the noise characterization, we can not rely on traditional approaches (signal-based or model-based, as described in Section 1.1.1). The approach considered in the proposed CFDDS relies on the approximation the relationship existing among datastreams with a LTI model, which is able to capture the functional relationship $f_{(i,j)}$ deviation

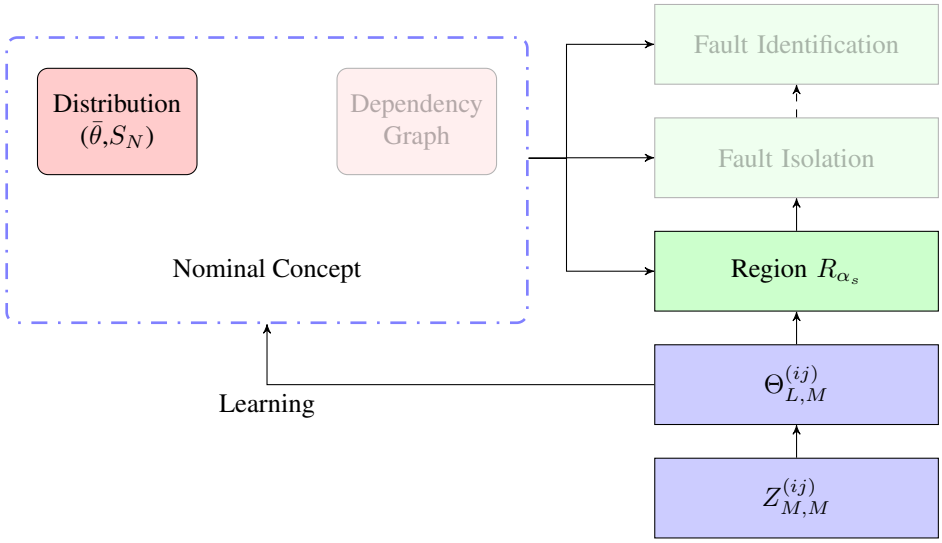


Figure 5.2: *Mahalanobis-based detection phase of the proposed CFDDS: by relying on a fault-free dataset $Z^{(ij)}_{M,M}$, the parameter vector sequence $\Theta^{(ij)}_{L,M}$ is extracted. They are used to learn the nominal state by means of a Gaussian distribution in the parameter space and a region R_{α_s} for the parameter vectors not belonging to the nominal state.*

from nominal behaviour. By considering as feature for fault detection the variations of the LTI estimated parameters $\hat{\theta}$s, we are able to detect changes in the behaviour of the data from the one we recorded in nominal conditions. The discrepancy from the nominal behaviour can be assessed only if a correct characterization of the nominal concept is performed beforehand: here we perform a characterization in the parameter space by relying on a fault-free dataset $Z_{M,M}$. Two different methods based on this paradigm are presented in the following: the former one exploits the asymptotic distribution of the parameters, while the latter one is able to characterize the sequence of the parameter vectors through a HMM modeling approach.

5.2 Mahalanobis-based Detection

Thanks to Equation (3.21), the parameter vectors in $\Theta^{(ij)}_{L,M}$ are distributed as a multivariate Gaussian distribution, which is fully characterized by its mean θ^o and covariance matrix Σ_N. Thus, in the parameter space it is possible to define a region where with probability $1-\alpha_s$ the parameter vectors estimated from data coming from nominal state can be drawn, by relying on the Mahalanobis distance [63]. This approach has been considered in the past, for instance, for quality control charts [50].

Algorithm 3 Mahalanobis-Based Change Detection Algorithm

Data: Training set $Z_{M,M}^{(ij)}$, Confidence level α_s;
Results: Detection time T;
Estimate the sequence $\Theta_{L,M}^{(ij)}$ from $Z_{M,M}^{(ij)}$;
Compute $\bar{\theta}$ and S_N;
while a new parameter vector $\hat{\theta}_t^{(ij)}$ is available at time t **do**
  if $\hat{\theta}_t \notin R_{\alpha_s}$ **then**
    return $T \leftarrow t$;
  end if
end while
return $T \leftarrow \emptyset$;

The general architecture of the detection phase based on this distance is presented in Figure 5.2 and an algorithm describing the procedure to apply the Mahalanobis-based detection method is provided in Algorithm 3. The mean $\bar{\theta} \in \mathbb{R}^p$ and covariance $S_N \in \mathbb{R}^{p \times p}$ can be estimated by using $\Theta_{L,M}^{(ij)}$ and constitute the definition of nominal cluster used in the sequel of the dissertation. More formally:

Definition 5.2.1. *Given a parameter vectors sequence $\Theta_{L,M}^{(ij)}$ estimated on a dataset $Z_{M,M}^{(i,j)}$ coming from a single process $\mathcal{P}$, a Gaussian cluster Υ, induced by the sequence $\Theta_{L,M}^{(ij)}$, is a tuple $(\bar{\theta}, S_N, \upsilon)$ ($\upsilon \in \mathbb{N}$), where:*

$$\bar{\theta} = \frac{1}{\upsilon} \sum_{\hat{\theta} \in \Theta_{L,M}^{(ij)}} \hat{\theta}, \tag{5.1}$$

$$S_N = \frac{1}{\upsilon - 1} \sum_{\hat{\theta} \in \Theta_{L,M}^{(ij)}} (\hat{\theta} - \bar{\theta})(\hat{\theta} - \bar{\theta})^T, \tag{5.2}$$

$$\upsilon = |\Theta_{L,M}^{(ij)}|, \tag{5.3}$$

and $|\cdot|$ is the cardinality operator.

A Gaussian cluster induces a topology on the parameter space, identified by the Mahalanobis distance [63], defined as:

Definition 5.2.2. *Given a parameter vector $\hat{\theta}_t^{(ij)} \in \mathbb{R}^p$ and a Gaussian cluster $\Upsilon = (\bar{\theta}, S_N, \upsilon)$, the* Mahalanobis distance $m(\hat{\theta}_t^{(ij)}, \Upsilon)$ *between the vector $\hat{\theta}_t^{(ij)}$ and the cluster Υ is defined as:*

$$m(\hat{\theta}_t^{(ij)}, \Upsilon) = (\bar{\theta} - \hat{\theta}_t^{(ij)})^T S_N^{-1} (\bar{\theta} - \hat{\theta}_t^{(ij)}). \tag{5.4}$$

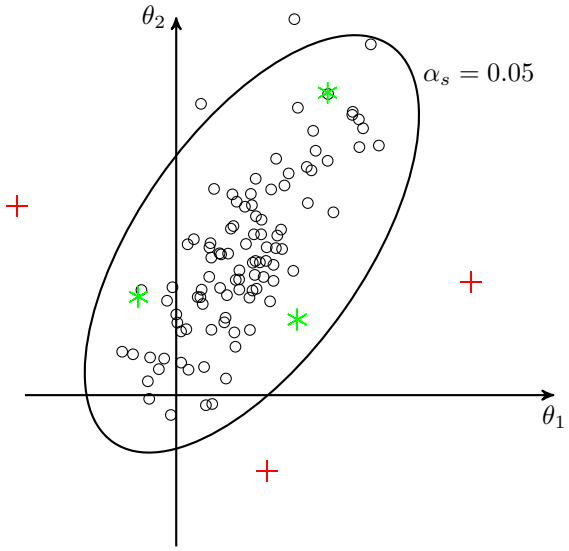


Figure 5.3: *Example of Mahalanobis-based detection in 2 dimension (θ_1, θ_2): the parameter vectors estimated on the training set (black circles) defines the nominal state; an area of probability $1 - \alpha_s = 0.95$ is defined by the ellipse. The detection phase is able to discriminate between points belonging to the nominal state region R_{α_s} (green asterisks) and those not belonging to it (red pluses), with confidence $1 - \alpha_s$.*

Since theoretical results presented in Section 3.3 allow us to assume the Gaussian distribution of the estimated parameter vectors, a neighbourhood centered in $\bar{\theta}$ can be induced by containing those $\hat{\theta}_t^{(ij)}$s belonging to the nominal state with probability $1 - \alpha_s$ [50], where α_s is a given confidence level. More specifically, the neighbourhood R_{α_s}, containing parameter vectors which belong to the nominal cluster, is defined as:

$$R_{\alpha_s} = \left\{ \hat{\theta} \mid \frac{\upsilon(\upsilon - p)}{p(\upsilon^2 - 1)} m(\hat{\theta}, \Upsilon) \leq F_{\alpha_s}(p, \upsilon - p) \right\}, \tag{5.5}$$

where $F_{\alpha_s}(p, \upsilon - p)$ is the quantile of order $1 - \alpha_s$ of the Fisher's distribution with parameters p and $\upsilon - p$.

For instance, in Figure 5.3, an example of the detection phase in a two dimensional parameter vector space $\theta = (\theta_1, \theta_2)$ is provided. The nominal state is characterized by the black circles, estimated on the training set $Z_{M,M}^{(i,j)}$: the knowledge of the statistical distribution allows us to define a region R_{α_s} where a newly estimated parameter vector should fall with probability $1 - \alpha_s = 0.95$ (inside the black ellipse). New parameter vectors are considered belonging to the nominal state if they fall inside the ellipse

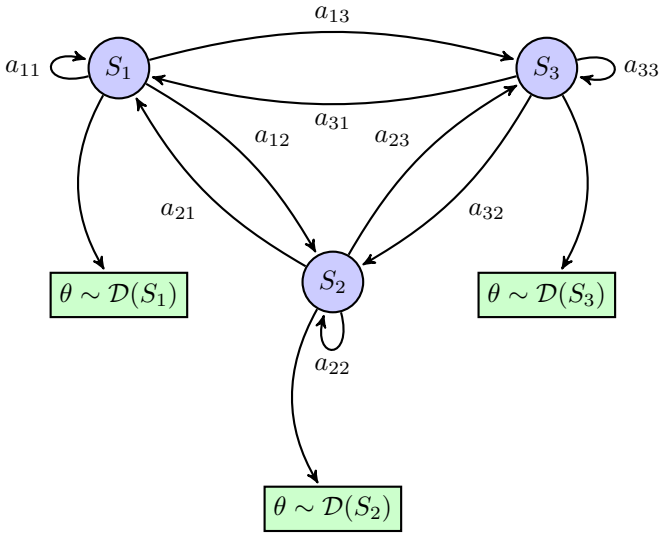


Figure 5.4: *Example of HMM: it has three hidden states S_1, S_2, S_3 with transition matrix $A = [a_{ij}]$ and each states S_i has emission θ, distributed as $\mathcal{D}(S_i)$.*

(green asterisks), while a change is detected when a parameter vector falls outside R_{α_s} (red pluses).

The proposed Mahalanobis-based detection is potentially a powerful and versatile method for detection, since it relies on a grounded theoretical base and requires only to set a confidence value α_s (which has a specific statistical meaning). Nonetheless, it is potentially a slow method, since the method operates on non overlapping windows and a detection could happen only every N time instants. In the following we rely on a different modeling of the nominal concept based on the estimation of parameter vectors on overlapping batches, i.e., by estimating $\Theta'^{(ij)}_{L,M}$ on $Z^{(i,j)}_{M,M}$ instead of $\Theta^{(ij)}_{L,M}$, to reduce the detection delay of this approach. Another drawback of the aforementioned method is that it provides a structural amount of false positive detections over time. In fact, the use of a statistical test implies to have the probability of α_s structural false positives (type I error) at each time instant. By repeating the test multiple time in an on-line manner, this probability increases, for instance, after k time instant the methods has the probability of a false positive detection of $1 - (1 - \alpha_s)^k$, whose limit for $k \to \infty$ is 1.

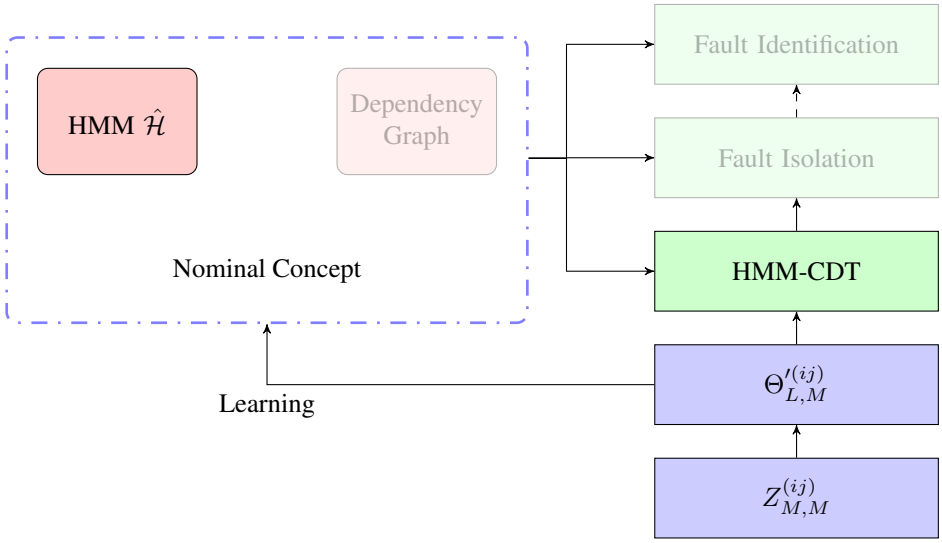


Figure 5.5: *HMM-CDT detection phase: by relying on a fault-free dataset $Z^{(ij)}_{M,M}$, the parameter vectors sequence $\Theta'^{(ij)}_{L,M}$ is extracted. It is used to learn the nominal state by means of a HMM, which is used for discrepancy detection in the proposed HMM-CDT.*

5.3 Hidden Markov Model Change Detection Test

The idea underlying the HMM-CDT is to analyse the statistical behaviour of the sequence of parameter vectors $\Theta'^{(ij)}_{L,t}$, estimated from $\boldsymbol{X}$, over time by means of a HMM modeling. A HMM is a statistical Markov model, in which the system being modeled is assumed to be a Markov process whose states are neither known, nor observed. An example of a HMM with three hidden states (S_1, S_2, S_3) is presented in Figure 5.4. While in traditional Markov chains, the state is observed and therefore the state transition probabilities a_{ij} are the only parameters to be estimated, in a HMM each state S_i is not directly visible, but only the output θ, emitted from one of the states with a given distribution $\mathcal{D}(S_i)$, is visible. Therefore, a sequence of outputs generated by a HMM provides information about the state sequence and can be used to learn the HMM parameters, i.e., number of states, transition matrix and emitting distributions. HMMs have been successfully applied to temporal pattern recognition problems such as speech recognition [70] and biology [55].

The HMM modeling approach applied to $\hat{\theta}^{(ij)}_t$s allow us to continuously inspect the relationship behaviour: when the statistical pattern of the estimated parameters does not follow what was learned during an initial train-

ing phase, a change in the relationship is detected. The general scheme of the HMM-based detection phase is presented in Figure 5.5. The use of a HMM ruled by a Gaussian Mixture Model (GMM) over a parameter vectors sequence $\Theta'^{(ij)}_{L,t}$ is the natural solution to model a functional relationship in the parameter space, since it is capable of modeling a Gaussian distribution and even take care of the possible seasonality or periodicity effects present in the data coming from a sensor network. More formally, a HMM [34] can be defined as a tuple $\mathcal{H} = (S, A, \pi)$, where:

- $S = \{S_1, \ldots, S_s\}, s \in \mathbb{N}$ is the indexed set of the states, each of which has an emission distribution $\mathcal{D}(S_i)$, defined by a GMM
- $A \in \mathbb{R}^{s \times s}, A = [a_{ij}], \sum_{i=1}^{s} a_{ij} = 1 \ \forall j \in \{1, \ldots, s\}$ is the transition matrix, i.e., a_{ij} is the transition probability from state i to state j;
- $\pi \in [0, 1]^s$, with $\sum_{i=1}^{s} \pi_i = 1$, is the initial distribution probability over S.

Algorithm 4 HMM-CDT Algorithm

Data: Training set $Z^{(ij)}_{M,M}$, Validation set $Z^{(ij)}_{O,M+O}$, C_D;
Results: Detection time T;
Estimate the sequence $\Theta'^{(ij)}_{L,M}$ from $Z^{(ij)}_{M,M}$;
Estimate HMM $\hat{\mathcal{H}}$ by using $\Theta'^{(ij)}_{L,M}$;
Estimate the sequence $\Theta'^{(ij)}_{O,O+M}$ from $Z^{(ij)}_{O,M+O}$;
for $h \in \{M + k, \ldots, O\}$ **do**
    Compute $l(\hat{\mathcal{H}}, \Theta'^{(ij)}_{k,h})$ as in Equation (5.6);
end for
Compute $\mathcal{T}$ as in Equation (5.7)
while a new sample $[x_i(t), x_j(t)]$ is available at time t **do**
    Estimate $\hat{\theta}^{(ij)}_t$ by using $Z^{(ij)}_{N,t}$;
    Built the sequence $\Theta'^{(ij)}_{k,t} = (\hat{\theta}^{(ij)}_{t-k+1}, \ldots, \hat{\theta}^{(ij)}_t)$;
    Compute $l(\hat{\mathcal{H}}, \Theta'^{(ij)}_{k,t})$ as in Equation (5.6);
    if $l(\hat{\mathcal{H}}, \Theta'^{(ij)}_{k,t}) \leq \mathcal{T}$ **then**
        return $T \leftarrow t$;
    end if
end while
return $T \leftarrow \emptyset$;

The HMM-CDT method is described in Algorithm 4. The HMM-CDT relies on an initial parameter vector sequence $\Theta'^{(ij)}_{L,M}$, estimated on overlapping batches of N data extracted from the fault-free training set $Z^{(i,j)}_{M,M}$,

where $L = M - N + 1$. The parameter sequence $\Theta'^{(ij)}_{L,M}$ is used to train a HMM $\hat{\mathcal{H}}$, aiming at capturing the statistical behaviour of the estimated parameter vectors sequence $\Theta'^{(ij)}_{L,M}$ in nominal conditions. Then, during the operational phase, the statistical affinity between the estimated parameter vectors $\hat{\theta}^{(ij)}_t$ and $\hat{\mathcal{H}}$ is assessed by looking at the HMM loglikelihood as follows:

Definition 5.3.1. *Given a parameter vector sequence* $\Theta'^{(ij)}_{k,t} = (\hat{\theta}^{(ij)}_{t-k+1}, \ldots, \hat{\theta}^{(ij)}_t)$ *and a HMM* $\hat{\mathcal{H}}$ *the* loglikelihood *is defined as:*

$$l(\hat{\mathcal{H}}, \Theta'^{(ij)}_{k,t}) = \log \Pr(\hat{\theta}^{(ij)}_{t-k+1} | \hat{\mathcal{H}}) + \sum_{h=t-k+1}^{t-1} \log \Pr(\hat{\theta}^{(ij)}_{h+1} | \hat{\mathcal{H}}, \hat{\theta}^{(ij)}_h). \quad (5.6)$$

where $\Pr(A|\hat{\mathcal{H}}, B)$ *is the probability associated to the parameter vector* A *given as model the HMM* $\hat{\mathcal{H}}$ *and the previously seen parameter vectors specified in* B*.*

The loglikelihood provides an estimate on how well the sequence $\Theta'^{(ij)}_{k,t}$ follows $\hat{\mathcal{H}}$, thus it can be used to detect changes in the relationship characterizing the nominal state of the system. In fact, if the relationship does not change over time, the value of the loglikelihood $l(\hat{\mathcal{H}}, \Theta'^{(ij)}_{k,t})$ is comparable with the one computed during the training phase. Otherwise, in case $l(\hat{\mathcal{H}}, \Theta'^{(ij)}_{k,t})$ falls below an automatically defined threshold $\mathcal{T}$, a change is detected, since $\Theta'^{(ij)}_{k,t}$ is no more compatible with the statistical model characterized by $\hat{\mathcal{H}}$. More specifically, by relying on a validation set $Z^{(ij)}_{O,M+O}$, we compute a threshold $\mathcal{T}$:

$$\mathcal{T} = \bar{l} - C_D \left[\bar{l} - \min_{h \in \{M+k, \ldots, O\}} l(\hat{\mathcal{H}}, \Theta'^{(ij)}_{k,h}) \right] \quad (5.7)$$

where

$$\bar{l} = \frac{\sum_{h=M+k}^{O} l(\hat{\mathcal{H}}, \Theta'^{(ij)}_{k,h})}{O} \quad (5.8)$$

and $C_D > 1$ is a user-defined coefficient factor, regulating the trade-off between the false positive detection rate and the promptness in the detection achieved by the HMM-CDT. High values for the parameter C_D corresponds to low false positive rate and an high delay in the time of detection, while low values provide a prompter detection but an higher amount of false positive detections.

During the operational life of the system, when the loglikelihood decreases below $\mathcal{T}$, a change in the statical behaviour of $\mathcal{P}$ is detected by the HMM-CDT. In more details, as soon as a new sample $[x_i(t), x_j(t)], t > M + O$ is available, the algorithm estimates a new parameter vector $\hat{\theta}_t^{(ij)}$ on $Z_{t,N}^{(ij)}$. The likelihood for the HMM characterizing the nominal state, i.e., $l(\mathcal{H}, \Theta_{k,h}^{\prime(ij)})$, is computed and is compared with the threshold $\mathcal{T}$ to assess if a change is detected. In case of a change, an alarm is raised and the detection time $T = t$ is returned, otherwise the algorithm keeps on monitoring data coming from the inspected process $\mathcal{P}$.

5.4 Ensemble Approach to HMM-CDT

One of the key aspects of the HMM-CDT presented above is the estimation of a HMM on $\Theta_{L,M}^{\prime(ij)}$, which aims at properly characterizing the nominal state. The state-of-the art training algorithm for HMMs is the Baum-Welch (BW) algorithm [22], which aims at finding the maximum likelihood estimates of the HMM parameters, given $\Theta_{L,M}^{\prime(ij)}$. Nonetheless, the BW algorithm does not provide any theoretical guarantee about the convergence to the global maximum of the likelihood function [31]. In fact, the BW algorithm requires a random initialization of the HMM parameters for the optimization procedure, which could lead the algorithm to get stuck to a local maximum. Thus, different estimated parameters for the HMM can be obtained by repeating the training phase with different randomly initialized parameters on the same training sequence $Z_{M,M}^{(i,j)}$. Hence, a viable solution to weaken the effect of the initialization procedure on the HMM-CDT is to repeat the use of BW algorithm and select the HMM guaranteeing the largest likelihood on a validation set. Unfortunately, as pointed out in [62], this solution may lead to overfit the training sequence $\Theta_{L,M}^{\prime(ij)}$ (and this is particularly evident for reduced training sets), leading to false positive detections (false alarms) during the operational life of the system.

In the framework of the proposed CFDDS, we extend the solution proposed in Section 5.3 by suggesting an Ensemble approach to Hidden Markov Model-Change Detection Test (EHMM-CDT) based on a set $\mathcal{E} = \{\mathcal{H}_1, \ldots, \mathcal{H}_Q\}$, $Q \in \mathbb{N}$ of HMMs, where Q is the ensemble cardinality. Interestingly, the possibility to combine different models to improve the generalization ability of a single model has been widely studied in the literature (mainly in the regression and classification scenarios) and effectively applied to different application fields [54, 68, 91]. Recently, ensembles of models have been also successfully considered in time series prediction [82, 89] and on-line

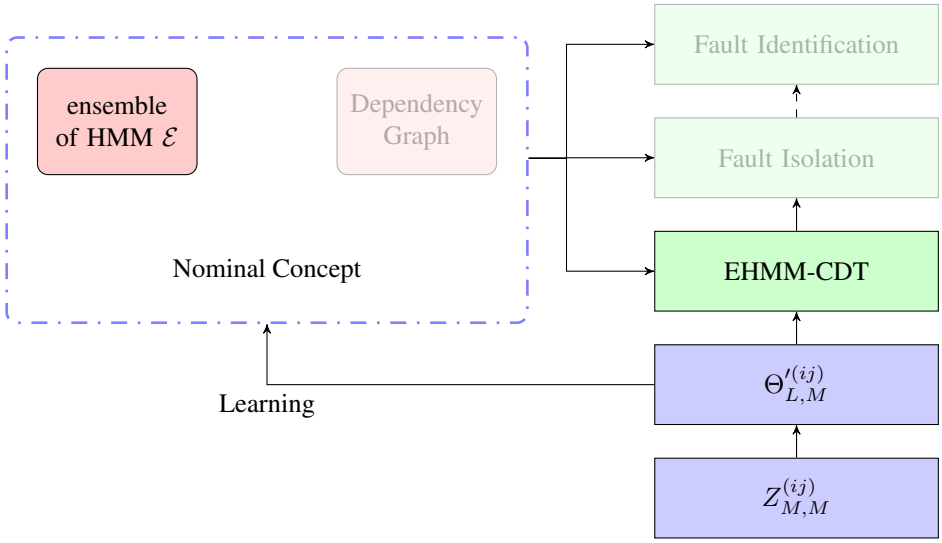


Figure 5.6: *Detection phase of the proposed CFDDS: by relying on a fault-free dataset $Z^{(ij)}_{M,M}$, the parameter vector sequence $\Theta'^{(ij)}_{L,M}$ is extracted. It is used to learn the nominal state by means of an ensemble of HMM, which are used for discrepancy detection in the proposed EHMM-CDT.*

missing data reconstruction [9]. Moreover, an ensemble of HMMs within a Bayesian framework for parameter estimation has been derived in [62], while an algorithm for training HMMs in the ensemble framework is presented in [29].

The general scheme of the proposed detection phase is described in Figure 5.6. The main point of the proposed solution is that the HMMs $\mathcal{H}_q \in \mathcal{E}$ are trained on the same training set $Z^{(ij)}_{M,M}$ with different initialization points of the BW algorithm. This ensemble approach allows to weaken the influence of the initial conditions of the HMM training algorithm, providing a better generalization ability and, consequently, better detection performance.

The proposed EHMM-CDT is detailed in Algorithm 5. To characterize the nominal state we rely on a training set $Z^{(ij)}_{M,M}$, which is assumed to be change-free. We estimate the parameter vectors sequence $\Theta'^{(ij)}_{L,M}$ on overlapping windows of N data $Z^{(ij)}_{N,t}$. We build the ensemble $\mathcal{E} = \{\mathcal{H}_1, \ldots, \mathcal{H}_Q\}$ by repeating e times the training of a HMM on $\Theta'^{(i,j)}_{L,M}$, with random initial conditions for the BW algorithm. Note that, since $Z^{(ij)}_{M,M}$ is assumed to be change-free, the ensemble $\mathcal{E}$ aims at modeling the statical behaviour of $\mathcal{P}$

Algorithm 5 EHMM-CDT algorithm

Data: Training set $Z_{M,M}^{(ij)}$, Validation set $Z_{O,M+O}^{(ij)}$, Aggregation method $\mathcal{A}$, C_D;
Results: Detection time T;
Estimate the sequence $\Theta_{L,M}^{\prime(i,j)}$ from $Z_{M,M}^{(ij)}$;
for $q \in \{1, \ldots, Q\}$ **do**
    Estimate HMM $\mathcal{H}_q$ using $\Theta_{L,M}^{\prime(i,j)}$ by considering a random initialization point;
end for
Built the ensemble $\mathcal{E} = \{\mathcal{H}_1, \ldots, \mathcal{H}_Q\}$;
Estimate the sequence $\Theta_{M,M+O}^{\prime(i,j)}$ from $Z_{O,M+O}^{(ij)}$;
for $h \in \{M + k, \ldots, O\}$ **do**
    for $q \in \{1, \ldots, Q\}$ **do**
        Compute $l(\mathcal{H}_q, \Theta_{k,h}^{\prime(i,j)})$ as in Equation (5.6);
    end for
    Compute $\mathcal{A}(h) = \mathcal{A}(l(\mathcal{H}_1, \Theta_{k,h}^{\prime(i,j)}), \ldots, l(\mathcal{H}_Q, \Theta_{k,h}^{\prime(i,j)}))$;
end for
Compute $\mathcal{T}$ as in Equation (5.11);
while a new couple $[x_i(t), x_j(t)]$ is available at time t **do**
    Estimate $\hat{\theta}_t^{(i,j)}$ by using $Z_{N,t}^{(ij)}$;
    Built the sequence $\Theta_{k,t}^{\prime(i,j)}$;
    for $q \in \{1, \ldots, Q\}$ **do**
        Compute $l(\mathcal{H}_q, \Theta_{k,t}^{\prime(i,j)})$ as in Equation (5.6);
    end for
    Compute $\mathcal{A}(t) = \mathcal{A}(l(\mathcal{H}_1, \Theta_{k,t}^{\prime(i,j)}), \ldots, l(\mathcal{H}_Q, \Theta_{k,t}^{\prime(i,j)}))$;
    if $\mathcal{A}(t) \leq \mathcal{T}$ **then**
        return $T \leftarrow t$;
    end if
end while
return $T \leftarrow \emptyset$;

in nominal conditions.

One of the key aspects of ensemble methods is the definition of the aggregation mechanism $\mathcal{A}$, that, in this case, specifies how to aggregate the loglikelihoods of the ensemble elements $\mathcal{H}_q \in \mathcal{E}$ to provide the ensemble output. In this dissertation, we considered two different aggregation mech-

anisms, i.e., $\mathcal{A} \in \{\mathcal{A}_{mean}, \mathcal{A}_{\min}\}$:

$$\mathcal{A}(t) = \mathcal{A}_{mean}(l(\mathcal{H}_1, \Theta_{k,t}^{\prime(i,j)}), \ldots, l(\mathcal{H}_Q, \Theta_{k,t}^{\prime(i,j)}))) = \sum_{q=1}^{Q} \frac{l(\mathcal{H}_q, \Theta_{k,t}^{\prime(i,j)})}{Q} \tag{5.9}$$

$$\mathcal{A}(t) = \mathcal{A}_{\min}(l(\mathcal{H}_1, \Theta_{k,t}^{\prime(i,j)}), \ldots, l(\mathcal{H}_Q, \Theta_{k,t}^{\prime(i,j)}))) = \min_{q \in \{1,\ldots,Q\}} l(\mathcal{H}_q, \Theta_{k,t}^{\prime(i,j)}), \tag{5.10}$$

where $\mathcal{A}_{mean}$ computes the average value of the loglikelihoods of the HMMs belonging to the ensemble $\mathcal{E}$, while $\mathcal{A}_{\min}$ takes into account their minimum. The last step of the EHMM-CDT training phase is the computation of the threshold $\mathcal{T}$ on a validation set $Z_{O,O+M}^{(ij)}$. Similarly to what presented in Section 5.3, $\mathcal{T}$ is computed as follows:

$$\mathcal{T} = \bar{l} - C_D \left[\bar{l} - \min_{h \in \{M+k,\ldots,O\}} \mathcal{A}(h) \right] \tag{5.11}$$

where $\bar{l} = \frac{\sum_{h=M+k}^{O} \mathcal{A}(h)}{O}$.

An example of the proposed approach with $Q = 5$ is proposed in Figure 5.7. At first the characterization of the process nominal conditions is obtained through the estimation of the ensemble of HMMs $\mathcal{E} = \{\mathcal{H}_1, \ldots, \mathcal{H}_5\}$, trained on the same couple of datastreams (x_i, x_j). Each HMM provides a value for the loglikelihood on newly samples coming from the streams (x_i, x_j). Loglikelihood values are used in an aggregation mechanism $\mathcal{A}$ to assess if a change has occurred.

While ensemble approaches are generally able to increase the generalization ability of a single model, they are characterized by an increased computational complexity, that in this case scales linearly with the number of HMMs Q. Two comments arise:

- the most time consuming part of the EHMM-CDT refers to the HMMs training (the loglikelihood computation is much lighter than training). Interestingly, the training phase is performed only once during the initial configuration of the proposed CFDDS, while during the operational life only the likelihoods are computed. In addition, in scenarios where networked embedded systems are operating, the training of HMMs could be performed in a centralized high-powerful unit, leaving only the computation of likelihoods directly at the low-power distributed units of the network;

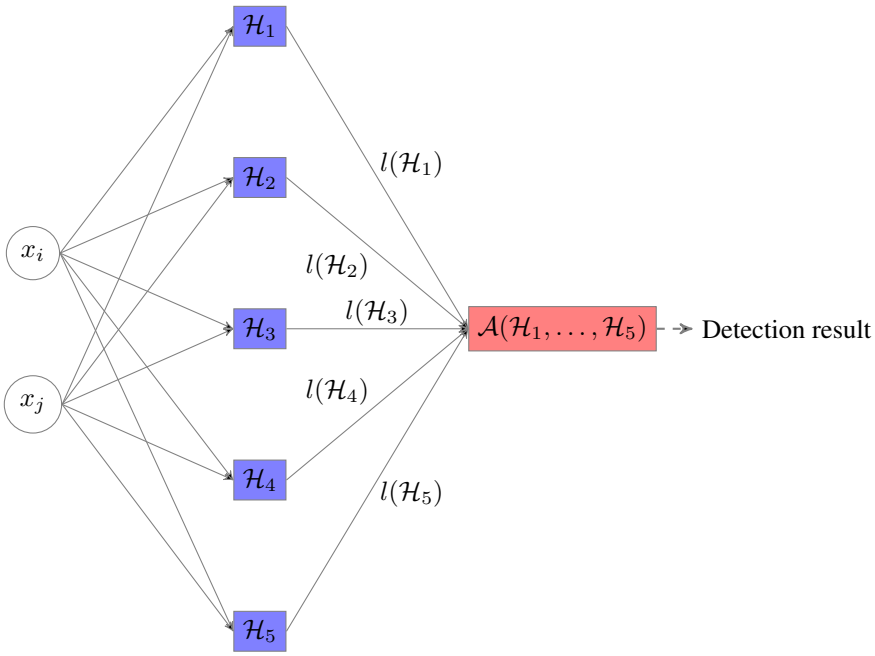


Figure 5.7: *Example of EHMM-CDT with $Q = 5$. The loglikelihood computed on each element of the ensemble of HMM $l(\mathcal{H}_q)$ provides information about the discrepancy of the process from nominal conditions. The detection decision occurs after the loglikelihoods are aggregated through a suitable aggregation mechanism $\mathcal{A}$.*

- in scenarios where the fault detection ability is a relevant activity, the increase in the complexity induced by the ensemble approach is well compensated if we are able to decrease false and missed alarms and detection delays.

As a final remark we would like to point out that the extension of the proposed EHMM-CDT to multiple functional relationships is trivial (the detection occurs when at least one change is detected in a functional relationship). Moreover, its extension to MIMO models is possible, but is not suggested here, since it does not allow to perform the isolation phase as proposed in Chapter 6.

CHAPTER 6

Fault Isolation

Once we are aware of the fact that the data provided by the sensor network are deviating from the nominal behaviour, by relying on the detection technique presented in Chapter 5, the following step is to infer if this discrepancy is due to a change of the environment in which the network is operating, to a fault in a specific sensor or to a model bias. To distinguish among these situations we have to consider both the dependency graph $\mathcal{G}$ and the HMM modeling used in the previous phases. The isolation phase of the proposed CFDDS, presented in Figure 6.1, is based on a logic partition of the dependency graph edge set and the evaluation of the loglikelihood provided by the ensemble $\mathcal{E}$ on each of the partitions. In Section 6.1, we describe the overall cognitive isolation logic, while in Section 6.2 we describe in detail the proposed isolation phase.

6.1 Cognitive Fault Isolation

We here consider the dependency graph $\mathcal{G} = (V, E)$ learned during the dependency graph learning phase (Chapter 4) and the result of the detection provided by the detection phase (Chapter 5), i.e., detection of a change on a specific functional relationship $f_{(\bar{i},\bar{j})}$. From now on, we assume that the

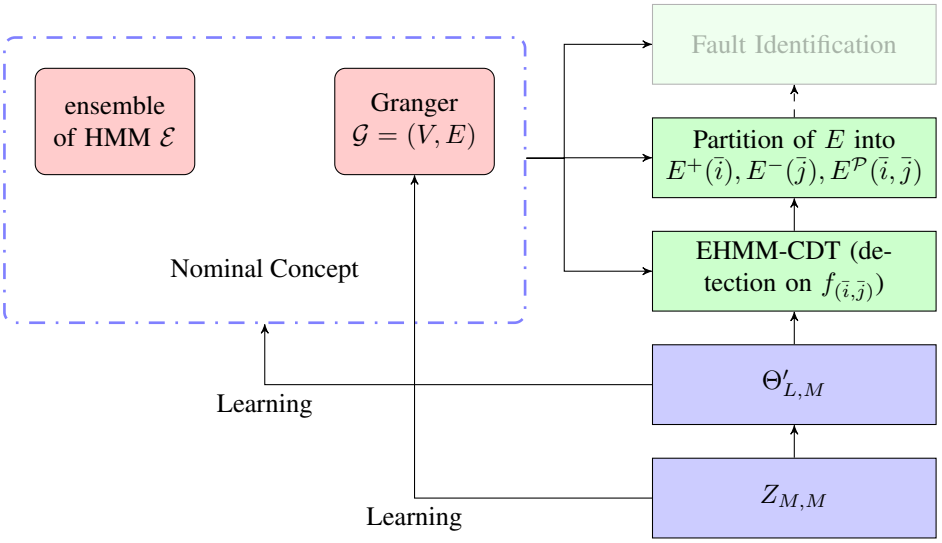


Figure 6.1: *Isolation phase of the proposed CFDDS: by relying on a fault-free dataset $Z_{M,M}$ it is possible to distinguish among model bias, change in the environment and fault. If a fault is identified, it is also isolated in the sensor where it occurred.*

network might be either affected by a change in the environment or a single fault. To perform the fault isolation task with the proposed CFDDS, we need to take into account all the edges contained in E at the same time, since a fault should influence several of the functional relationships corresponding to these edges. After a detection in $f_{(\bar{i},\bar{j})}$ has occurred, we need to partition the edges of the dependency graph in three parts: all those relationships related to sensor $s_{\bar{i}}$, all those related to $s_{\bar{j}}$ and those not having as input or output nor $s_{\bar{i}}$, neither $s_{\bar{j}}$. More formally:

Definition 6.1.1. *Given a dependency graph $\mathcal{G} = (V, E)$ and a detection in $f_{(\bar{i},\bar{j})}$:*

- $E^+(\bar{i}) = \{e_{ij} \in E \mid e_{ij} \neq e_{\bar{i}\bar{j}}, i = \bar{i} \vee j = \bar{i}\}$*, i.e., all the edges connected with $s_{\bar{i}}$;*
- $E^-(\bar{j}) = \{e_{ij} \in E \mid e_{ij} \neq e_{\bar{i}\bar{j}}, i = \bar{j} \vee j = \bar{j}\}$*, i.e., all the edges connected with $s_{\bar{j}}$;*
- $E^{\mathcal{P}}(\bar{i}, \bar{j}) = E \setminus (\{e_{\bar{i}\bar{j}}\} \cup E^+ \cup E^-)$*, i.e., all the other edges which are not connected neither to $s_{\bar{i}}$, nor to $s_{\bar{j}}$.*

With this partition we are able to diagnose the cause of the detection occurring on the relationship $f_{(\bar{i},\bar{j})}$. If $E^+(\bar{i})$ or $E^-(\bar{j})$ is also influenced

by the change, we may infer that a fault has happened on sensor $s_{\bar{i}}$ or $s_{\bar{j}}$, respectively. In the case $E^{\mathcal{P}}(\bar{i}, \bar{j})$ is also influenced by the change, we are facing a change in the environment inspected by the network, while if no other relationship is influenced by the detected change we infer that the cause is an incomplete characterization of the nominal state (model bias).

Finally, we define two properties of the fault affecting sensors included in the dependency graph $\mathcal{G}$. In fact, depending on its topology, a fault can be isolated on a specific subset of nodes. Formally:

Definition 6.1.2. *Given a dependency graph $\mathcal{G} = (V, E)$, a fault F occurring in the sensor $s_h \in V$ is* detectable *if and only if $\exists e_{ij} \in E$ s.t. $i = h \vee j = h$.*

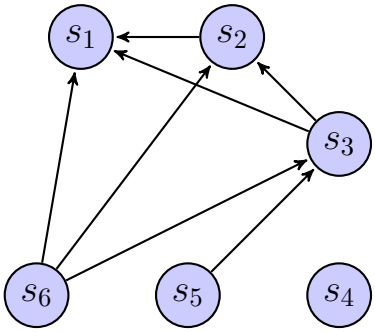
Definition 6.1.3. *Given a dependency graph $\mathcal{G} = (V, E)$, a fault F occurring in the sensor $s_h \in V$ is* isolable *if and only if $K = \{e_{ij} \in E$ s.t. $i = h \vee j = h\}$, $|K| > 1$.*

It is clear that isolability implies detectability, since isolability implies to have more than one edge connected to a specific sensor s_h, while detectability requires the existence of at least one of these edges.

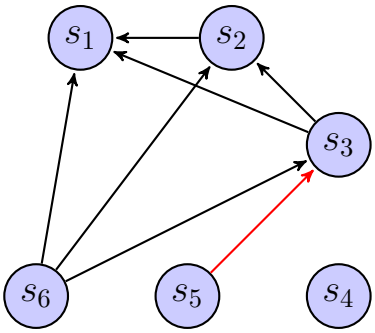
In Figure 6.2a, it is possible to see how faults in s_4 cannot be detected by the proposed CFDDS, since no relationship including the sensor is considered in the dependency graph. In the situation where a sensor is completely disconnected in the dependency graph, techniques considering a single node could be considered [21], in parallel with the proposed CFDDS. In this case (a fault is not detectable in the sensor), it is possible to analyze the datastream provided by the sensor for detection purposes with traditional CDTs methods and rely on the proposed isolation methodology to spot if a change in the environment has occurred.

The situation in Figure 6.2b may lead to a detection if a fault occurs in the sensor s_5, but this does not allow to isolate it with the proposed isolation logic. In this case an enhanced version of the dependency graph could be considered (Figure 6.2c), including, in this case, $f_{(5,2)}$ and $f_{(5,1)}$. This inclusion is based on the fact that if the two relationships $f_{(5,3)}$ and $f_{(3,2)}$ are considered, then also $f_{(5,2)}$ exists (maybe with less significance), and it could be used to spot faults in s_5 during the isolation phase. The same applies to $f_{(5,1)}$.

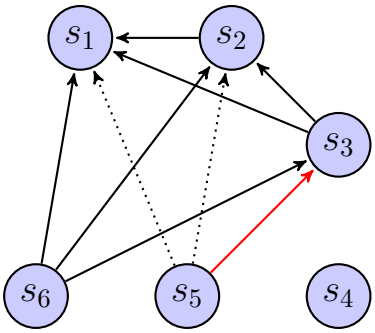
An example of the execution of the partitioning of the dependency graph $\mathcal{G}$, on the basis of a detection, is presented in Figure 6.3. A posteriori of a detection in the relationship $f_{(3,2)}$ (red solid edge), the isolation algorithm partitions the dependency graph into three sets:



(a) *Since there are no selected relationships including the sensor s_4, faults in this sensor cannot be detected with the proposed methodology (non-detectable)*

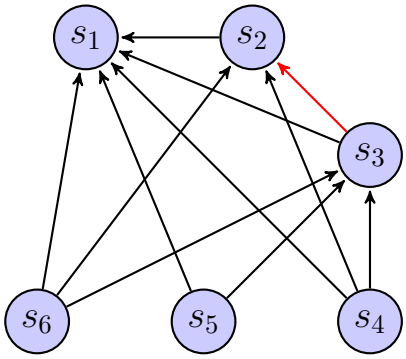


(b) *Since there is a single selected relationships including the sensor s_5, faults in this sensor can be detected (by the relationship $f_{(5,3)}$) cannot be isolated with the proposed methodology (non-isolable)*

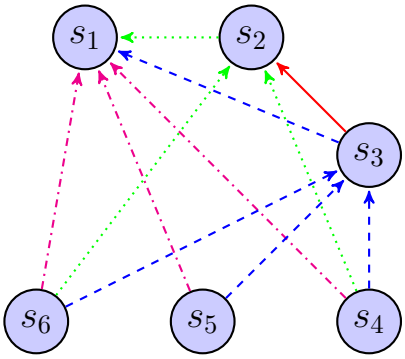


(c) *This enhanced version of the dependency graph allows to isolate faults also in s_5*

Figure 6.2: *Examples of dependency graphs with non detectable and non isolable sensors*



(a) *Situation after the detection phase: a change has been detected in the relationship* $f_{(3,2)}$



(b) *The graph is partitioned: all the relationships which includes the sensors* s_2 *and* s_3 *are analysed to identify if a fault has taken place (green dotted and blue dashed ones, respectively); all the other relationship are analysed to identify environmental changes (magenta dash-dotted ones)*

Figure 6.3: *Example of the proposed cognitive isolation*

- $E^{+}(3) = \{f_{(3,1)}, f_{(4,3)}, f_{(5,3)}, f_{(6,3)}\}$ (blue dashed edges);
- $E^{-}(2) = \{f_{(2,1)}, f_{(4,2)}, f_{(6,2)}\}$ (green dotted edges);
- $E^{\mathcal{P}}(3,2) = \{f_{(4,1)}, f_{(5,1)}, f_{(6,1)}\}$ (magenta dash-dotted edges);

The proposed method inspects these sets:

- if the process is affected by an environmental change, it is likely that all the relationships corresponding to edges in $E^{\mathcal{P}}(3,2)$ perceive the change;
- if a fault occurs, only those connected with the faulty sensor, i.e., $E^{+}(3)$ or $E^{-}(2)$, are influenced by it;
- if the detection was due to a model bias, only the original relationship $f_{(3,2)}$ is affected by the change.

6.2 Fault Isolation Phase

Following the methodology presented in [8] and by considering the above definitions we provide an algorithm able to isolate faults within the proposed CFDDS, described in Algorithm 6. The algorithm relies on a training set $Z_{M,M}$ and a validation set $Z_{O,M+O}$, similarly to the detection phase,

Algorithm 6 Cognitive Isolation Algorithm

Input: Training set $Z_{M;M}$, Validation set $Z_{O;M+O}$, C_I
Output: Isolation result Is, Fault location Loc
Learn $\mathcal{G} = (V, E)$ by using $Z_{M;M}$;
for all $e_{ij} \in E$ **do**
  Train $\mathcal{E}^{(ij)}$ by using $Z^{(ij)}_{M;M}$;
  Compute $\overline{l}^{(ij)}$ as in Equation (6.1)
  Compute $l^{(ij)}_{\min}$ as in Equation (6.2);
end for
if a detection occurs on $f_{(\bar{i},\bar{j})}$ **then**
  Partition E into $E^+(\bar{i})$, $E^-(\bar{j})$ and $E^{\mathcal{P}}((\bar{i},\bar{j}))$ as described in Section 6.1;
  if $|E^{\mathcal{P}}(\bar{i},\bar{j})| > 0 \wedge \frac{1}{|E^{\mathcal{P}}((\bar{i},\bar{j}))|} \sum_{e_{ij} \in E^{\mathcal{P}}((\bar{i},\bar{j}))} \frac{\mathcal{A}^{(ij)} - \overline{l}^{(ij)}}{\overline{l}^{(ij)} - l^{(ij)}_{\min}} \leq -C_I$ **then**
    return $Is =$ "$environmental\ change$" and $Loc = \emptyset$;
  end if
  if $|E^+(\bar{i})| > 0 \wedge \frac{1}{|E^+(\bar{i})|} \sum_{e_{ij} \in E^+(\bar{i})} \frac{\mathcal{A}^{(ij)} - \overline{l}^{(ij)}}{\overline{l}^{(ij)} - l^{(ij)}_{\min}} \leq -C_I$ **then**
    return $Is =$ "$fault$" and $Loc = s_{\bar{i}}$;
  end if
  if $|E^-(\bar{j})| > 0 \wedge \frac{1}{|E^-(\bar{j})|} \sum_{e_{ij} \in E^-(\bar{j})} \frac{\mathcal{A}^{(ij)} - \overline{l}^{(ij)}}{\overline{l}^{(ij)} - l^{(ij)}_{\min}} \leq -C_I$ **then**
    return $Is =$ "$fault$" and $Loc = s_{\bar{j}}$;
  end if
  return $Is =$ "$model\ bias$" and $Loc = \emptyset$;
end if

and an isolation coefficient C_I, which similarly to C_D regulates the trade-off between isolation promptness and false positives rate.

During the training phase, which can be performed in an off-line way, the algorithm learns the dependency graph $\mathcal{G} = (V, E)$ of the network based on $Z_{M,M}$, relying on the algorithm proposed in Chapter 4. For each edge $e_{ij} \in E$ an ensemble of HMM $\mathcal{E}_{ij}$ is trained, as depicted in Chapter 5, on the training data $Z^{(ij)}_{M,M}$. For each ensemble $\mathcal{E}_{ij}$, based on the validation set $Z^{(ij)}_{O,M+O}$ the average $\overline{l}^{(ij)}$ and minimum $l^{(ij)}_{\min}$ loglikelihoods are computed in the following way:

$$\overline{l}^{(ij)} = \frac{\sum_{h=M+k}^{O} \mathcal{A}^{(ij)}(h)}{O} \tag{6.1}$$

$$l^{(ij)}_{\min} = \min_{h \in \{M+k,\ldots,O\}} \mathcal{A}^{(ij)}(h) \tag{6.2}$$

$$\tag{6.3}$$

where aggregated loglikelihoods $\mathcal{A}^{(ij)}(h)$ are computed with the parameter vectors sequences $\Theta'^{(ij)}_{k,h}$ estimated on $Z^{(ij)}_{O,M+O}$. These quantities will be

used, as before, to evaluate the discrepancy of newly estimated parameter vectors from the ensemble of HMM $\mathcal{E}_{ij}$ characterizing the nominal state.

As soon as a detection occurs at time t on a specific functional relationship $f_{(\bar{i},\bar{j})}$, the algorithm partitions the dependency graph into $E^{+}(\bar{i}), E^{-}(\bar{j})$ and $E^{\mathcal{P}}(\bar{i},\bar{j})$ and computes aggregated loglikelihoods $\mathcal{A}^{(ij)}(t)$ for all $e_{ij} \in E$. For each $\hat{E} \in \{E^{+}(\bar{i}), E^{-}(\bar{j}), E^{\mathcal{P}}(\bar{i},\bar{j})\}$, if it is not empty, we evaluate the following expression:

$$\frac{1}{|\hat{E}|} \sum_{e_{ij} \in \hat{E}} \frac{\mathcal{A}^{(ij)} - \bar{l}^{(ij)}}{\bar{l}^{(ij)} - l^{(ij)}_{\min}} \leq -C_I \tag{6.4}$$

where $|\cdot|$ is the cardinality operator, $C_I < C_D$ and C_D is the coefficient factor defined in Section 5.3.

At first, we apply the expression in Equation (6.4) to the partition $E^{\mathcal{P}}(\bar{i},\bar{j})$, to evaluate if a change in the environment has occurred. Otherwise, with the similar procedure applied to $E^{+}(\bar{i})$ and $E^{-}(\bar{j})$, we try to isolate faults in $s_{\bar{i}}$ and $s_{\bar{j}}$, respectively. If none of the above options are assessed, we consider the detected change due to a model bias in the relationship $f_{(\bar{i},\bar{j})}$.

If an environmental change or a fault has been detected and isolated, the sensor network should be reconfigured, by training again the proposed CFDDS and by inspecting the damaged unit, respectively. Obviously, if the result of the proposed isolation algorithm is $Is = \text{“}modelbias''$, we need to estimate again the relationship which was affected by the change.

We would like to point out that the coefficient C_I regulates how much the loglikelihood has to decrease before we consider it meaningful: an high value for C_I will require a high loglikelihood drop to isolate a fault.

CHAPTER 7

Fault Identification

In this chapter we present the identification phase of the proposed CFDDS. On the basis of the techniques presented in the previous chapters, we suppose to have successfully detected a fault through the technique presented in Chapter 5 and isolated it in a sensor s_i through the technique presented in Chapter 6. After that, we would like to characterize the fault, for instance to understand if the fault already affected the system in the past or if it presents new characteristics. The general scheme for the proposed identification phase is presented in Figure 7.1. In this chapter, we consider a fault-free dataset $Z_{M,M}^{(ij)}$ coming from a single data generating process $\mathcal{P}$ and that faults affecting the system are of the abrupt type (see Section 2.2 on fault modeling for details).

In Section 7.1 we propose the overall procedure adopted by the proposed CFDDS for fault identification, which is based on a newly developed evolving clustering technique. In Section 7.2 we detail the procedure for the fault dictionary learning. Moreover, in Section 7.3 we discuss possible extensions of the proposed framework to the case of incipient faults.

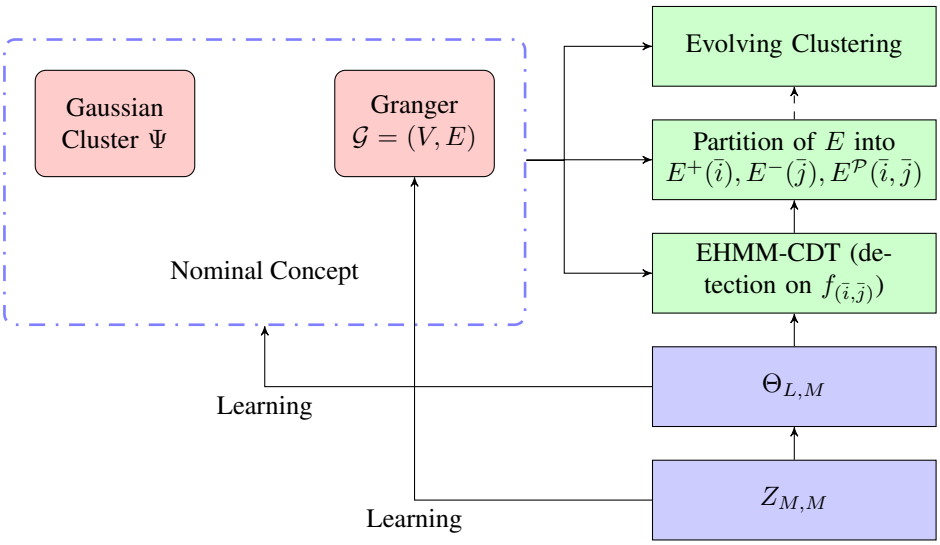


Figure 7.1: *Identification phase of the proposed CFDDS: by relying on a fault-free dataset $Z_{M,M}$ the system characterizes the nominal state by means of a Gaussian cluster Ψ. New faults are identified by means of an evolving clustering algorithm, able to learn the fault dictionary in an on-line manner*

7.1 Modeling the Nominal State

The identification problem is here addressed as a clustering problem in the parameter space. In fact, by relying on the fault-free training set, we are able to characterize only the nominal state, and we have to update the fault dictionary, (i.e., include a new entity into the fault dictionary when a new cluster ha to be created) once faults occur. This scenario is well fitted in the framework usually addressed by evolving clustering methods. In the literature, there are several clustering methods considering both offline [24, 35, 65, 84] and evolving [17, 79] solutions.

For instance, [35] suggests the Density-Based Spatial Clustering of Applications with Noise (DBSCAN), which is an algorithm relying on a density-based notion of clusters designed to discover clusters of arbitrary shape. This algorithm is appealing since it requires minimal information on the dataset to properly operate and considers the possibility of having outliers (which are desirable due to the Gaussian topology of the parameter space). [24] shows the Affinity Propagation (AP) that is an algorithm considering as input measures of similarity between pairs of data points, used as basis for a mechanism using real-valued messages exchanged between data points to select a set of exemplars and corresponding clusters. Both

the algorithms presented above are not developed in an on-line framework. Their use in an on-line scenario would require a huge computational effort, since as soon as a new parameter vector is available they would require to consider the whole dataset again. Differently, Evolving Cluster Method (ECM), described in [79], manages clusters with evolving strategies: it requires the setting of a parameter D_{thr}, which is strictly related to the number of clusters the algorithm will create during the operational life. None of the aforementioned methods is able to guarantee:

- a logic differentiation among clusters (i.e., those corresponding to nominal and fault states), inducing a different management on different sets of cluster;
- the use of a Gaussian topology, required by the parameter space topology;
- an evolving mechanism able to manage newly estimated parameter vectors.

Thus, we proposed a newly developed evolving clustering algorithm for the identification phase, which takes advantage of temporal and spatial dependencies of the estimated parameters. For more details on alternative approach to cognitive fault identification, other than clustering, the interested reader can refer to Section 1.2.

For the identification phase, the proposed CFDDS relies on an initial training phase to characterize the nominal state Ψ by exploiting a fault-free training sequence $Z_{M,M}^{(ij)}$. The dataset is windowed into non-overlapping batches of length N, each of which is used to provide a parameter vector estimate $\hat{\theta}_t^{(ij)}$. The outcome is the sequence $\Theta_{L,M}^{(ij)}$ ($L = M/N$).

The proposed algorithm is given in Algorithm 7. From results delineated in Chapter 3 parameter vectors in $\Theta_{L,M}^{(ij)}$ are distributed according to a Gaussian distribution, provided that N is large enough, even though the system is non-linear. Here we rely on a slightly different definition of cluster than definition 5.2.1. In this chapter we consider the following:

Definition 7.1.1. *Given a parameter vectors sequence* $\Theta_{L,M}^{(ij)} = \{\hat{\theta}_1^{(ij)}, \ldots, \hat{\theta}_L^{(ij)}\}$, *where* $\hat{\theta}_h^{(ij)}$ *is estimated on a dataset* $Z_{N,hN}^{(i,j)}$ *(non-overlapping batches), coming from a single process* $\mathcal{P}$*, a Gaussian cluster* Υ*, induced by the*

Algorithm 7 Evolving Clustering-Based Identification Algorithm

Data: Training set $Z_{M,M}^{(ij)}$; α_s, α_m, η_t;
Results: Nominal cluster Ψ, Fault dictionary Φ;
Compute mean $\bar{\theta}_\Psi$ and covariance matrix S_Ψ for the nominal state cluster
Set $n_\Psi = L$ and $t_\Psi = L$
Set $\Phi = \emptyset$ ($\phi = 0$) and $O = \emptyset$
while A new $\hat{\theta}_t^{(ij)}$ is available **do**
  if Equation (7.6) holds Ψ **then**
    Associate $\hat{\theta}_t^{(ij)}$ to Ψ
    if $|t_\Psi - t| \leq \eta_t$ **then**
      Update Ψ as in Equations (7.9) to (7.11)
    end if
    $t_{\Psi^*} \leftarrow t$
  else
    if $\phi > 0$ and Equation (7.6) holds for at least one $\Phi_r \in \Phi$ **then**
      Select Φ^* minimizing Equation (7.7)
      Associate $\hat{\theta}_t^{(ij)}$ to Φ^*
      if $|t_{\Phi^*} - t| \leq \eta_t$ **then**
        Update Φ^* as in Equations (7.9) to (7.11)
        for $\hat{\theta}_h^{(ij)} \in O$ **do**
          if Equation (7.6) holds for Φ^* **then**
            Remove $\hat{\theta}_h^{(ij)}$ from outlier set O
            Associate $\hat{\theta}_h^{(ij)}$ to Φ^*
            if $|t_{\Phi^*} - h| \leq \eta_t$ **then**
              Update Φ^* as in Equations (7.9) to (7.11)
            end if
          end if
        end for
        for $\Phi_r \in \Phi, \Phi \neq \Phi^*$ **do**
          if Equations (7.12) and (7.13) hold for Φ^*, Φ_r **then**
            Merge Φ^*, Φ_r as in Equations (7.14) to (7.17)
          end if
        end for
      end if
      $t_{\Phi^*} \leftarrow t$
    else
      Insert $\hat{\theta}_t$ in O
      Create $\bar{O}$ according to Algorithm 8
      if $\bar{O} \neq \emptyset$ **then**
        $\phi \leftarrow \phi + 1$
        Create Φ_ϕ using $\hat{\theta}_k^{(ij)} \in \bar{O}$
      end if
    end if
  end if
end while

sequence $\Theta_{L,M}^{(ij)}$*, is a tuple* $(\bar{\theta}_\Upsilon, S_\Upsilon, n_\Upsilon, t_\Upsilon)$*, where:*

$$\bar{\theta}_\Upsilon = \frac{1}{\upsilon} \sum_{\hat{\theta} \in \Theta_{L,M}^{(ij)}} \hat{\theta}, \tag{7.1}$$

$$S_\Upsilon = \frac{1}{\upsilon - 1} \sum_{\hat{\theta} \in \Theta_{L,M}^{(ij)}} (\hat{\theta} - \bar{\theta})(\hat{\theta} - \bar{\theta})^T, \tag{7.2}$$

$$n_\Upsilon = |\Theta_{L,M}^{(ij)}| = \upsilon, \tag{7.3}$$

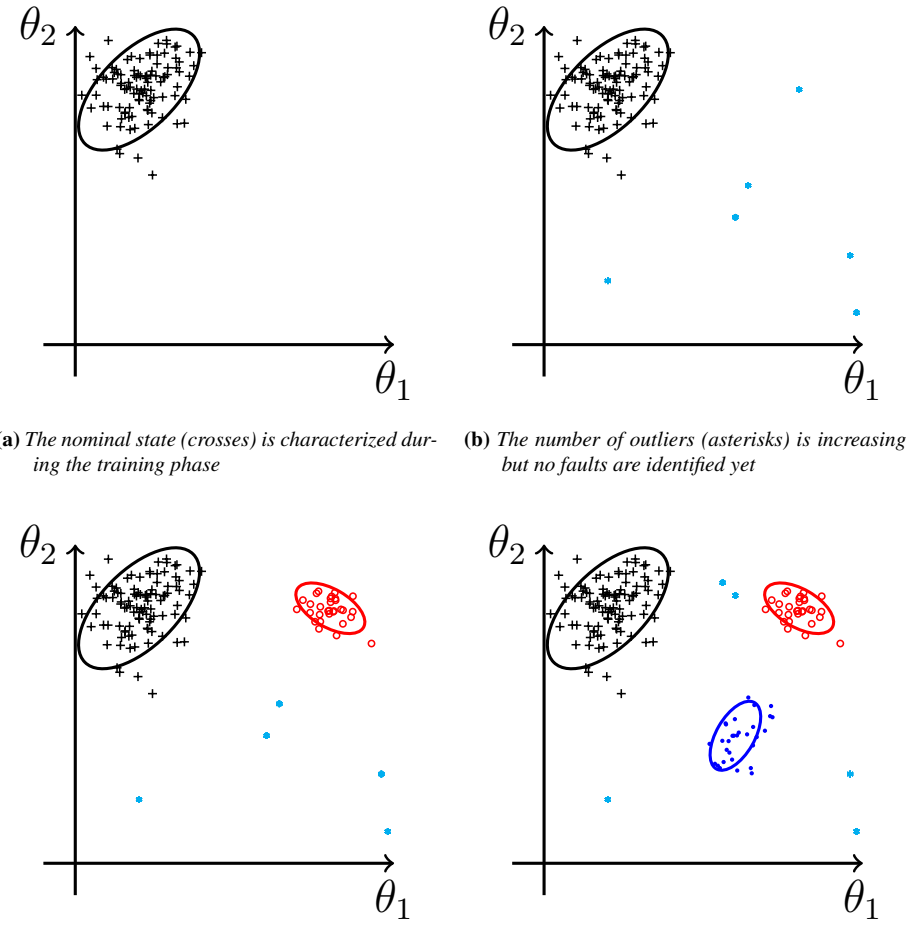
$$t_\Upsilon = L, \tag{7.4}$$

and $|\cdot|$ *is the cardinality operator.*

Thanks to Equation (3.21) the nominal state Ψ can be described as a Gaussian cluster composed by estimated parameter vectors, whose mean vector $\bar{\theta}_\Psi$ and covariance matrix S_Ψ can be estimated on $\Theta_{L,M}^{(ij)}$ (Line 3). For diagnosis purposes we assign to the nominal state the number n_Ψ of parameter vectors used to estimate $\bar{\theta}_\Psi$ and S_Ψ and the last time instant t_Ψ for which a $\hat{\theta}_t^{(ij)}$ was associated to the nominal state Ψ. At the end of the training phase $n_\Psi = L$ and $t_\Psi = L$ (Line 4).

During the operational life, the proposed CFDDS estimates parameter vectors from incoming non-overlapping N-sample data windows. The corresponding estimated parameter vector $\hat{\theta}_t^{(ij)}$ is then either associated to the nominal state Ψ or to a generic r-th faulty one Φ_r present in the fault dictionary $\Phi = \{\Phi_1, \ldots, \Phi_\phi\}$ (ϕ represents the current number of faulty classes in the fault dictionary). If the assignment cannot be granted according to a given confidence level, the estimated parameter vector $\hat{\theta}_t^{(ij)}$ is currently considered an outlier and moved to an outlier set O. Similarly, other "housekeeping" operations are executed on the existing structures (outlier and faulty sets), e.g., leading to the merge of two faulty states, whenever appropriate.

An example of the execution of the identification phase is provided in Figure 7.2. At the beginning, both the fault dictionary and the outlier set are empty (Figure 7.2a). They are populated during the operational life of the system, as data come in. The outlier set O is regularly inspected to determine whether a new faulty state $\Phi_{\phi+1}$ needs to be generated by relying on parameter vectors included in it (Figure 7.2b). If a parameter vector cannot be associated to either the nominal state or one of the faulty states according to the given confidence level, it is considered an outlier and moved to the outlier set O (e.g., see the asterisks near the ellipse in the upper-right side of Figure 7.2d).



(a) *The nominal state (crosses) is characterized during the training phase*

(b) *The number of outliers (asterisks) is increasing but no faults are identified yet*

(c) *As soon as enough confidence is gathered for the presence of a new faulty state, a new cluster is created (circles) and instances added to it*

(d) *When a different fault is identified (dots), it is added to the fault dictionary.*

Figure 7.2: *Example of the identification phase in the proposed CFDDS*

Here, we assume that a fault affecting the optimal parameter θ^o (defined as in Section 3.3) abruptly moves the process from a stationary state to a new stationary one (abrupt fault). A faulty state Φ_r is hence characterized by a mean vector $\bar{\theta}_{\Phi_r}$ and a covariance matrix S_{Φ_r}, which can be estimated from parameter vectors $\hat{\theta}_h^{(ij)} \in \Phi_r$. The proposed CFDDS stores the number n_{Φ_r} of vectors used to estimate $\bar{\theta}_{\Phi_r}$ and S_{Φ_r} and the latest time instant t_{Φ_r} where $\hat{\theta}_h^{(ij)}$ was associated to Φ_r.

The distance between an estimated parameter vector $\hat{\theta}_t^{(ij)}$ and a cluster Υ can be computed by means of the Mahalanobis distance (as defined in Section 5.2):

$$m(\hat{\theta}_t^{(ij)}, \Upsilon) = (\bar{\theta}_\Upsilon - \hat{\theta}_t^{(ij)})^T S_\Upsilon^{-1} (\bar{\theta}_\Upsilon - \hat{\theta}_t^{(ij)}), \tag{7.5}$$

where $\Upsilon \in \{\Psi, \Phi_1, \ldots, \Phi_\phi\}$. Since Ψ and $\Phi_r \in \Phi$ are Gaussian clusters, a neighbourhood centered in $\bar{\theta}_\Upsilon$ can be induced by containing those $\hat{\theta}_t^{(ij)}$s belonging to Υ with probability $1 - \alpha_s$ [50], where α_s is a given confidence level. More specifically, the spatial neighbourhood R_{α_s} of a cluster Υ is defines as:

$$R_{\alpha_s} = \left\{ \hat{\theta} \mid \frac{n_\Upsilon (n_\Upsilon - p)}{p(n_\Upsilon^2 - 1)} m(\hat{\theta}, \Upsilon) \leq F_{\alpha_s}(p, n_\Upsilon - p) \right\} \tag{7.6}$$

holds, where $F_{\alpha_s}(p, n_\Upsilon - p)$ is the quantile of order $1 - \alpha_s$ of the Fisher's distribution with parameters p and $n_\Upsilon - p$. Similarly, a neighbourhood is assigned to each cluster $\Upsilon \in \{\Psi, \Phi_1, \ldots, \Phi_\phi\}$ and constitutes the core of the fault identification phase of the proposed CFDDS (Lines 7 and 14). The proposed CFDDS also contemplates the case of $\hat{\theta}_t^{(ij)}$ satisfying Equation (7.6) for multiple clusters. In this case, $\hat{\theta}_t^{(ij)}$ is associated to the cluster Υ^* (either nominal or faulty, Line 15):

$$\Upsilon^* = \min_{\Upsilon \in \{\Psi, \Phi_1, \ldots, \Phi_\phi\}} \frac{n_\Upsilon (n_\Upsilon - p)}{p(n_\Upsilon^2 - 1)} m(\hat{\theta}_t^{(ij)}, \Upsilon). \tag{7.7}$$

In other words $\hat{\theta}_t^{(ij)}$ is assigned to the nearest cluster, provided that confidence $1 - \alpha_s$ is attained. Once Υ^* has been determined, we set $t_{\Upsilon^*} = t$ (Lines 12 and 34). If $\hat{\theta}_t^{(ij)}$ cannot be associated either to Ψ, or to $\{\Phi_1, \ldots, \Phi_\phi\}$, it is considered to be an outlier and inserted in O (Line 36).

The algorithm, after taking into account the "spatial" locality between parameter vectors, analyses the "temporal" one, by evaluating to which extent $\hat{\theta}_t^{(ij)}$ has been associated to Υ^* (Lines 9 and 17), i.e.,

$$|t_{\Upsilon^*} - t| \leq \eta_t, \tag{7.8}$$

where $\eta_t \in \mathbb{N}$ is a temporal threshold (when $\eta_t = 1$, the proposed CFDDS verifies if two consecutive time vectors $\hat{\theta}_t^{(ij)}$ and $\hat{\theta}_{t-1}^{(ij)}$ have been assigned to the same cluster). This operation is important since we expect models built over time to be temporally dependent.

If $\hat{\theta}_t^{(ij)}$ satisfies both the spatial (Equation (7.6)) and the temporal (Equation (7.8)) membership conditions for a specific cluster Υ^*, it is inserted in there and Υ^* statistics are updated as follows (Lines 10 and 18):

$$\bar{\theta}_{\Upsilon^*} \leftarrow \frac{n_{\Upsilon^*}}{n_{\Upsilon^*}+1}\bar{\theta}_{\Upsilon^*} + \frac{1}{n_{\Upsilon^*}+1}\hat{\theta}_t^{(ij)} \tag{7.9}$$

$$S_{\Upsilon^*} \leftarrow \frac{n_{\Upsilon^*}-1}{n_{\Upsilon^*}}S_{\Upsilon^*} + \frac{n_{\Upsilon^*}+1}{n_{\Upsilon^*}^2}(\hat{\theta}_t^{(ij)} - \bar{\theta}_{\Upsilon^*})(\hat{\theta}_t^{(ij)} - \bar{\theta}_{\Upsilon^*})^T \tag{7.10}$$

$$n_{\Upsilon^*} \leftarrow n_{\Upsilon^*} + 1. \tag{7.11}$$

The aforementioned procedure might update cluster Υ_r so that it partly overlaps with another one Υ_k. The proposed algorithm handles this situation with a cluster merging procedure (Lines 28 to 30). The union of clusters Υ_r and Υ_k is performed when the following two conditions are jointly satisfied,

$$\frac{n_{\Upsilon_r}(n_{\Upsilon_k}n_{\Upsilon_r} - n_{\Upsilon_k} - p + 1)}{(n_{\Upsilon_k}+1)(n_{\Upsilon_r}-1)p} m(\bar{\theta}_{\Upsilon_r}, \Upsilon_k) \leq F_{\frac{\alpha_m}{2}}(p, n_{\Upsilon_k}n_{\Upsilon_r} - n_{\Upsilon_k} - p + 1) \tag{7.12}$$

$$\frac{n_{\Upsilon_k}(n_{\Upsilon_r}n_{\Upsilon_k} - n_{\Upsilon_r} - p + 1)}{(n_{\Upsilon_r}+1)(n_{\Upsilon_k}-1)p} m(\bar{\theta}_{\Upsilon_k}, \Upsilon_r) \leq F_{\frac{\alpha_m}{2}}(p, n_{\Upsilon_r}n_{\Upsilon_k} - n_{\Upsilon_r} - p + 1), \tag{7.13}$$

i.e., if the cluster means $\bar{\theta}_{\Upsilon_r}$ and $\bar{\theta}_{\Upsilon_k}$ have probability greater than $1 - \alpha_m$ to belong to Υ_k and Υ_r, respectively. In Equations (7.12) and (7.13), $F_{\frac{\alpha_m}{2}}(p, n_{\Upsilon_k}n_{\Upsilon_r} - n_{\Upsilon_k} - p + 1)$ is the quantile of order $1 - \alpha_m/2$ of the Fisher's distribution quantile with parameters p and $n_{\Upsilon_k}n_{\Upsilon_r} - n_{\Upsilon_k} - p + 1$. Approximated results for the confidence α_m follow from the Bonferroni correction for multiple tests. If the above conditions are satisfied, the CFDDS merges the two clusters Υ_r and Υ_k to generate cluster Υ' defined as:

$$\bar{\theta}_{\Upsilon'} \leftarrow \frac{n_{\Upsilon_r}}{n_{\Upsilon_r} + n_{\Upsilon_k}}\bar{\theta}_{\Upsilon_r} + \frac{n_{\Upsilon_k}}{n_{\Upsilon_r} + n_{\Upsilon_k}}\bar{\theta}_{\Upsilon_k}; \tag{7.14}$$

$$S_{\Upsilon'} \leftarrow S_{\Upsilon_r} + S_{\Upsilon_k} + \frac{n_{\Upsilon_r}n_{\Upsilon_k}}{n_{\Upsilon_r} + n_{\Upsilon_k}}(\bar{\theta}_{\Upsilon_r} - \bar{\theta}_{\Upsilon_k})(\bar{\theta}_{\Upsilon_r} - \bar{\theta}_{\Upsilon_k})^T; \tag{7.15}$$

$$n_{\Upsilon'} \leftarrow n_{\Upsilon_r} + n_{\Upsilon_k}; \tag{7.16}$$

$$t_{\Upsilon'} \leftarrow \max\{t_{\Upsilon_r}, t_{\Upsilon_k}\}. \tag{7.17}$$

The exact computation of the update for the covariance matrix is performed as in [66].

After a cluster update or the merge of two clusters, the proposed CFDDS checks if parameter vectors in the outlier set O can now be associated either to the nominal state or to one of the faulty ones (Lines 19 to 24).

7.2 On-line Modeling the Fault Dictionary

We addressed so far the procedure allowing the insertion of the parameter vectors into the nominal and faulty clusters and the merging of two faulty clusters. The remaining $\hat{\theta}_t^{(ij)}$ are collected into the outlier set O, where further inspection is performed during the operational life of the system, to verify whether a new faulty state must be created or not.

With reference to Algorithm 8, a new cluster needs to be created depending on the outcome of the KS test (Line 4). The test compares the empirical Cumulative Distribution Function (CDF) of all the $\hat{\theta}_h^{(ij)}$s estimated by the proposed CFDDS during both the training and the operational phases and the CDF induced by considering the estimated nominal state Ψ and faulty states $\{\Phi_1, \ldots, \Phi_\phi\}$. If the distribution of the $\hat{\theta}_h^{(ij)}$s is no more coherent with the current set of clusters, a new cluster must be created and a new faulty class inserted in Φ. More in detail, the test is designed as:

$$H_0 : \hat{F} = F_\Gamma \text{ vs. } H_1 : \hat{F} \neq F_\Gamma \tag{7.18}$$

where $\hat{F}$ is the empirical CDF of all the $\hat{\theta}$s and F_Γ is the distribution induced by Gaussian clusters $\Gamma = \{\Psi, \Phi_1 \ldots \Phi_\phi\}$. The KS test statistics takes into account the maximum distance between the two CDFs

$$D^p = \max_{0 \leq \alpha \leq 1} |\hat{F}(B_\alpha) - F_\Gamma(B_\alpha)|, \tag{7.19}$$

where B_α is the region in the parameter space such that $F_\Gamma(B_\alpha) = \alpha$ (see [73] for further details). As stated in [73], D^p has the same distribution of the one-dimensional KS distribution, so, for the KS test, we can compare it with the asymptotic form of the KS distribution K [53, 78]. Given a confidence level α_c, the critical region of the KS test (i.e., for rejecting the null hypothesis H_0) is composed by all D^p:

$$D^p > K_{\alpha_c} \tag{7.20}$$

where K_{α_c} is the quantile of order $1-\alpha_c$ of the one-dimensional K distribution. The proposed statistical test suffers from the curse of dimensionality,

i.e., it needs an exponentially increasing number of samples to be effective as the parameter vectors dimension p increases. Therefore, if needed, we suggest to apply a dimensionality reduction method to the parameter vectors $\hat{\theta}_h^{(ij)} \in O$, e.g., based on PCA [50] or Random Projection (RP) method [23].

Once the KS test provides enough confidence to claim that a new cluster must be generated from the outlier set (i.e., hypothesis H_0 is rejected), suitable instances are removed from O and the new cluster is created. We assume the availability of a supervisor that is able to label new faulty clusters, e.g., by providing the type of encountered fault. This allows us for creating online the fault dictionary. On the contrary, when the hypothesis H_0 is not rejected, Algorithm 8 returns an empty set (Line 32).

It is worth noting that the Mahalanobis distance cannot be considered to measure the proximities between couples of parameter vectors in O, since the distribution of elements in the outlier set is unknown (i.e., we cannot assume that $\hat{\theta}_h^{(ij)}$s $\in O$ are Gaussian distributed as they are not). To address this issue we defined the spatial-temporal norm on $\hat{\theta}_h^{(ij)}, \hat{\theta}_k^{(ij)} \in O$, inspired by the metric suggested in [92]:

$$||\hat{\theta}_h^{(ij)} - \hat{\theta}_k^{(ij)}||_\lambda^2 = \lambda \frac{||\hat{\theta}_h^{(ij)} - \hat{\theta}_k^{(ij)}||^2}{2p} + (1-\lambda)\frac{|h-k|}{t} \tag{7.21}$$

where $||\cdot||$ is the euclidean norm, t is the last batch of data considered and $\lambda \in [0,1]$ is a penalty factor balancing the spatial locality and the temporal one. A normalization procedure is required so that both the spatial and temporal components of the norm are constrained to the $[0,1]$ interval. The proposed CFDDS adopts the online normalization procedure described in [48].

To select parameter vectors for the new cluster, we apply to the outlier set O the Mountain Method (MM) [83, 85, 87], which identifies the density center for the $\hat{\theta}_h^{(ij)}$s, $\hat{\theta}_h^{(ij)} \in O$ (Lines 7 to 12). Finally, this algorithm estimates the density as:

$$\Omega_{RMM}(c_k, \hat{\theta}_h^{(ij)}; r) = \exp\left(-\frac{||\hat{\theta}_h^{(ij)} - c_k||_\lambda^2}{2r^2}\right) \tag{7.22}$$

where $c_k \in \mathbb{R}^p$ is a center and r is an influence radius parameter. The algorithm iteratively approximates:

$$c^* = \max_c \sum_{\hat{\theta}_h^{(ij)} \in O} \Omega_{RMM}(c, \hat{\theta}_h^{(ij)}, r). \tag{7.23}$$

The potential function Ω_{RMM} is robust to outliers (see [87] for a formal proof) and, since it decreases slowly when $||\hat{\theta}_h^{(ij)} - c_k||_\lambda < r$ and fast if $||\hat{\theta}_h^{(ij)} - c_k||_\lambda > r$, it defines a neighbourhood around each class center c_k. For the purpose of the cluster creation a center will be initialized for each parameter vector $\hat{\theta}_h^{(ij)} \in O$. As described in [85], η_i of Algorithm 8 represents both a tolerance threshold for the convergence of the iterative procedure to identify the cluster center and the maximum error of the optimization procedure. As one might imagine the method is rather sensitive to r, which highly influences the clustering results. Here, we suggested three different heuristics to identify a suitable value for the radius r:

- power estimate using correlation [87];
- median distance criterion [83];
- maximum edge length of minimum spanning tree under the normal distribution hypothesis [67].

At the end of the mountain method each parameter vector is associated with a set O_s (Lines 14 to 15) and $\tilde{O}$, the set characterized by the largest cardinality, is selected as a new candidate cluster.

To identify the cluster shape of $\tilde{O}$ (we do not have a priori information about the covariance matrix of the novel cluster), a *Minimum Covariance Determinant* search method [40] is executed (Lines 16 to 29), i.e., a subset of elements $\bar{O} \subseteq \tilde{O}$ is selected s.t. the determinant of the parameter covariance is minimal. This method can be applied when the number of samples in $\tilde{O} \geq p$. When this condition is satisfied (Line 16), a new cluster is created: the mean and the covariance of the parameter vectors in $\bar{O}$ are computed, $n_{\Phi_{\phi+1}} = |\bar{O}|$, $t_{\Phi_{\phi+1}} = \max_{\hat{\theta}_h \in \bar{O}} h$ and the algorithm returns $\bar{O}$ (Line 27). Otherwise, when $\tilde{O} < p$, the algorithm returns the empty set $\emptyset$ (Line 29).

Note that the algorithm requires at least $n_\Upsilon = p + 1$ parameter vectors to create a cluster. More parameter vectors would allow a better characterization of the cluster itself since the variance of the estimation of the mean and the covariance matrix scales asymptotically as $\frac{1}{n_\Upsilon}$. Moreover, as time passes, more and more parameter vectors are inserted into the outlier set. To reduce as much as possible the creation of false faulty classes we should consider an oblivion coefficient on the parameter vectors in the outlier set or mechanisms to discard the oldest ones (e.g., by setting a maximum value on the cardinality of the outlier set and keeping the new ones). The algorithm can be easily modified to take into account this case.

Algorithm 8 Faulty Cluster Creation Algorithm

Data: Outlier set O; Clustering Γ; α_c, η_i;
Results: Faulty cluster $\bar{O}$;
Compute D^p according to Equation (7.20)
if $D^p > K_{\alpha_c}$ **then**
  Set $c_h = \hat{\theta}_h^{(ij)},\ \forall \hat{\theta}_h^{(ij)} \in O$ and $err \geq \eta_i$
  Compute r
  while $err \geq \eta_i$ **do**
    for k s.t, $\hat{\theta}_k^{(ij)} \in O$ **do**
      $\hat{c}_k \leftarrow c_k$
      $c_k \leftarrow \dfrac{\sum_{\hat{\theta}_h^{(ij)} \in O} \Omega_{RMM}(c_k, \hat{\theta}_h^{(ij)}; r)\hat{\theta}_h^{(ij)}}{\sum_{\hat{\theta}_h^{(ij)} \in O} \Omega_{RMM}(c_k, \hat{\theta}_h^{(ij)}; r)}$
    end for
    $err = \max_k ||\hat{c}_k - c_k||_\lambda$
  end while
  Associate all centers c_k, c_h s.t. $||c_k - c_h||_\lambda \leq 2\eta_i$ to a set, creating the sets $O_1, \ldots, O_S$
  Let $\tilde{O} = \arg\max_{s \in \{1,\ldots,S\}} |O_s|$, i.e., the set with the largest cardinality
  if $|\tilde{O}| \geq p$ **then**
    Choose randomly $\bar{h} = \frac{|\tilde{O}|+p+1}{2}$ elements in $\tilde{O}$ to define $\bar{O}$
    Set $S^* = \sum_{\hat{\theta}_h^{(ij)} \in \bar{O}} \frac{(\hat{\theta}_h^{(ij)} - c^*)(\hat{\theta}_h^{(ij)} - c^*)^T}{\bar{h}-1}$
    while $\bar{O} \neq \bar{O}'$ **do**
      $\bar{O}' \leftarrow \bar{O}$
      for $\hat{\theta}_h^{(ij)} \in \bar{O}$ **do**
        $d(\hat{\theta}_h^{(ij)}) = (\hat{\theta}_h^{(ij)} - c^*)(S^*)^{-1}(\hat{\theta}_h^{(ij)} - c^*)^T$
      end for
      $\bar{O} \leftarrow \arg\min_{O' \subseteq \tilde{O}, |O'| = \bar{h}} \sum_{\hat{\theta}_h^{(ij)} \in O'} d(\hat{\theta}_h^{(ij)})$
      $S^* \leftarrow \sum_{\hat{\theta}_h^{(ij)} \in \bar{O}} \frac{(\hat{\theta}_h^{(ij)} - c^*)(\hat{\theta}_h^{(ij)} - c^*)^T}{h-1}$
    end while
    return $\bar{O}$
  else
    return $\bar{O} = \emptyset$
  end if
else
  return $\bar{O} = \emptyset$
end if

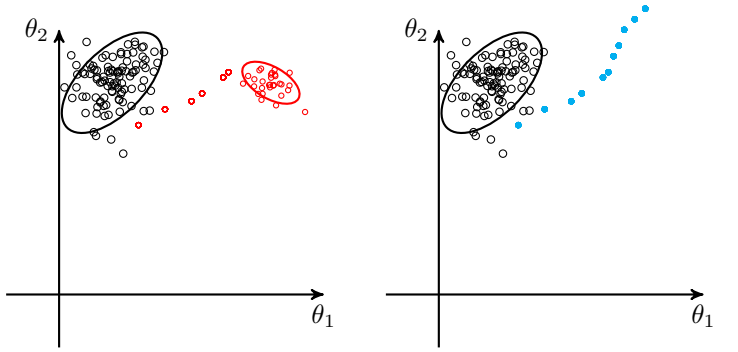
7.3 Dealing with Incipient Faults

One of the main assumption of the proposed identification phase is that faults abruptly moves the process from a stationary state to a new stationary one (abrupt faults). This is due to the fact that only stationary states induce a Gaussian topology in the parameter space (see Section 3.3 for more details), while there are no theoretical results for different kind of faults. Thus, if the assumption that only abrupt faults occur does not hold, situation different from those previously described may appear in the parameter space, where parameter vectors are not identically distributed anymore. Some examples, where the faults have incipient development are presented in Figure 7.3.

In particular, Figure 7.3a presents the situation where the fault has an incipient development and, after that, it stabilizes into a stationary state (incipient permanent fault). In this case, the identification phase of the proposed CFDDS is still able to deal with this situation, provided that a strong evidence for the creation of the cluster is provided. For instance, we could impose a stronger condition for the cluster creation procedure, like requiring to have at least $\bar{p} >> p+1$ to allow the formation of a new cluster.

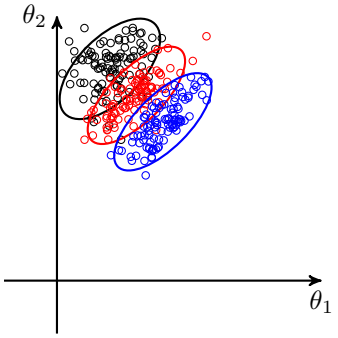
When the fault induces a set of continuously drifting parameter vectors, as in Figure 7.3b, the ability to characterize a fault in the parameter space seem to be not viable anymore. Moreover, in this case the analysis should be carried on the trajectory the parameter vectors are following, for instance by relying on joint information on its temporal and spatial evolution (speed and acceleration in the parameter space).

When the process slowly drifts in the parameter space from a stationary state to another, as in Figure 7.3c, it may happen that the nominal and the faulty clusters significantly overlap. If we are expecting such a phenomenon, a conservative solution is to fix the nominal state at the end of the training phase (i.e., we do not update its estimated mean $\bar{\theta}_\Psi$ and covariance matrix S_Ψ if a new parameter vector belonging to the nominal state is spotted). With this approach we avoid to include parameter vectors belonging to a fault into the nominal cluster.



(a) *The transition phase (for instance, given by an incipient fault) is fast and the fault stabilizes in a stationary state*

(b) *The fault is rapidly changing (for instance, a continuous drift) and the evolving algorithm is not able to characterize it. Its parameter vectors form a trajectory in the parameter space)*



(c) *Slowly drifting fault, which overlaps with the nominal state*

Figure 7.3: *Example of different incipient faults in the parameter vectors space*

CHAPTER 8

CFDDS Implementation

In this chapter, we report a brief overview of the main functions implementing the previously described phases of the proposed CFDDS, which are considered in Chapter 9 to validate the proposed framework. The code developed within this dissertation is available at [2].

Two external toolboxes were considered within the libraries developed: one for the estimation of the HMM and one for the multivariate granger causality toolbox [18]. The library implementing the proposed CFDDS is structured as follows:

- data generation procedures (`data_gen` directory)
- parameter vectors estimation (`model_estimation` directory)
- graph learning phase(`cogn_net` directory)
- detection and isolation phases (`cogn_hmm` directory)
- identification phase (`cogn_fds` directory)

Data Generation

`opts_generation`: generates the options for the input, output and fault generation procedures. It needs as input two structures describing the structure of the signal to be simulated and the fault signature.

`input_generation`: generates an input for a discrete time dynamic process. The function allows to generate data from a cosinusoidal function with additive noise, from a Gaussian random walk process or from a uniform distribution over $[-1, 1]$.

`signal_generation`: generates the output of a discrete time dynamic process, given the exogenous input and specifics about the dynamic model to take into account. It is possible to choose among a linear ARX model, a pendulum and a Van der Pol oscillator [51].

`fault_generation`: injects faults in a given signal. It allows to insert structured (with dynamic $\phi(t) = f(x, y)$) faults, as well as non structured ones ($\phi(t) = \bar{\phi}, \bar{\phi} \in \mathbb{R}$).

Parameter Estimation

`idarmax`: estimates the orders of a given model family (e.g., ARX), based on a given dataset. It allows to consider a specific range of orders and to choose the criterion used for the selection (e.g., Akaike Information Criterion (AIC), Mean Square Error (MSE) on a validation set).

`model_creation`: estimates model parameter vectors from an input-output dataset $\{(x(t), y(t))\}_{t=1}^{N}$ given the orders of the linear dynamic system to consider.

Graph Learning

`sel_rel`: learns the dependency graph from a fault-free dataset, by choosing among different methods (e,g,. correlation analysis and Granger-based statistical test).

`granger`: implements the Granger-based test for assessing the multivariate Granger causality in a multivariate dataset.

Detection Phase

`train_hmm_fds`: estimate the parameters of a HMM or an ensemble of HMMs relying on a given training dataset. It requires also to specify the binary relationships which should be considered, i.e., it requires to have a

dependency graph in order to decide which set of relationships to take into account.

`test_hmm_fds`: given a model trained with `train_hmm_fds` and a stream of data, it computes the loglikelihood for each trained HMM and for each relationship considered during training.

`plot_lls`: given a model tested with `test_hmm_fds` it is able to plot the computed loglikelihood for the data provided during the testing phase.

`hmm_fds_detection`: given a HMM model and an aggregation method, it is able to compute the detection result for a single functional relationship.

`detection_net`: aggregates the results provided by `hmm_fds_detection` and checks the consistency of the previously provided results.

Isolation Phase

`fds_isolation`: is able to perform the isolation phase of the proposed CFDDS, given the loglikelihoods provided by HMMs on different functional relationships.

Identification Phase `init_clu`: initializes an empty clustering structure, able to store both the cluster representing the nominal state of the system and the ones representing the faulty states.

`clust_train`: populates the nominal cluster and updates its statistics.

`clust_evol`: core of the evolving algorithm described in Algorithm 7. Given an estimated parameter vector, it is able to automatically decide to assign it to the nominal state, to a faulty one or to the outlier set, as well as to create new cluster, merge the old ones if necessary and update their statistics.

CHAPTER 9

Experimental Results

In this chapter, we analyse the performance of the proposed CFDDS on a synthetically generated dataset, a real benchmark coming from sensor networks designed to perform an environmental monitoring task and a simulated one based on data coming from a sensor network inspecting a water distribution system. We here considered each phase of the proposed CFDDS one at a time, to evaluate separately the advantages each phase is providing to the overall framework. The chapter is structured as follows: at first an overview of the considered datasets is presented in Section 9.1. After that the phases of the proposed CFDDS are tested on the aforementioned datasets in Sections 9.2 to 9.5. Finally, general remarks on the described experimental campaign are provided in Section 9.6.

9.1 The Considered Datasets

9.1.1 Application D1: Synthetic Datasets

Here, we consider datastreams coming from a synthetically generated sensor network with randomly generated relationships among datastreams. To generate these data, a directed acyclic graph, representing the dependency

graph of the existing functional relationships (with a fixed amount of nodes and a fixed number of edges) is generated. After that, for all those sensors without any incoming edge, we generate samples by relying on the following autoregressive process:

$$x_i(t) = \phi x_i(t-1) + \psi(t) \tag{9.1}$$

where $\phi = 0.4$, $\psi(t) \sim \mathcal{N}(0, 0.1)$, $\eta(t)$ is an uncorrelated white noise s.t. $\eta(t) \sim \mathcal{N}(0, \sigma^2)$ and σ is the standard deviation of the noise. Then, by considering a topological order of the graph[1], datastreams are generated either through linear $\mathcal{L}$ or sinusoidal $\mathcal{S}$ models as:

$$\mathcal{L}: \; x_j(t) = \theta_j^T \, \mathbf{z} + \eta(t); \tag{9.2}$$

$$\mathcal{S}: \; x_j(t) = \sin\left(\theta_j^T \, \mathbf{z}\right) + \eta(t), \tag{9.3}$$

where:

$$\begin{aligned} \mathbf{z} = \big[&x_j(t-1), x_j(t-2), x_{i_{1j}}(t-1), x_{i_{1j}}(t-2), \\ &\ldots, x_{i_{nj}}(t-1), x_{i_{nj}}(t-2)\big] \end{aligned} \tag{9.4}$$

$\mathbf{z} \in \mathbb{R}^{2(1+nj)}$ is the vector of the autoregressive $(x_j(t-1), x_j(t-2))$ and exogenous $(x_{i_{1j}}(t-1), \ldots, x_{i_{nj}}(t-2))$, $\{i_{1j}, \ldots i_{nj}\}$ components, is the set of the indexes of those sensors s^* s.t. exists an edge (s^*, s_j) in the graph, i.e., the indexes of the sensors s^* with an edge from s^* to s_j, and $\theta_j \in \mathbb{R}^{2(1+nj)}$ is a randomly generated vector.

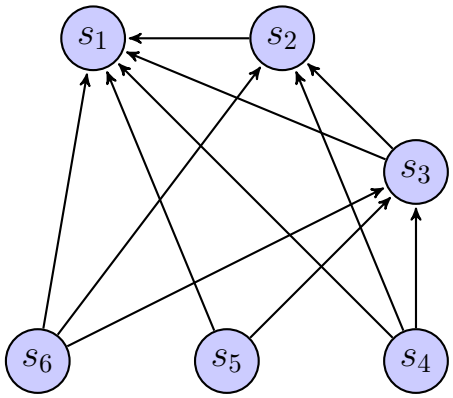


Figure 9.1: *Example of generation of the dependency graph for a network with $n = 6$ sensors.*

[1] Any of the possible topological orders provides the same results for what it concerns the generation of the datastreams.

For instance in the graph of Figure 9.1, we generate at first the datastreams x_6, x_5 or x_4 from Equation (9.1) (in any order). After that, by following the topologic order of the graph, we generate x_3 from either Equation (9.2) or Equation (9.3) by considering as input streams x_6, x_5 and x_4. By using the same model chosen to generate x_3, we generate x_2 (using as input x_6, x_4 and x_3) and x_1 (using as input all the other streams).

9.1.2 Application D2: Rialba Dataset

In this application data are gathered from a rock collapse and landslide forecasting system, [5, 6] deployed at the Towers of Rialba site, in Northern Italy. The sensor network deployment area is inspecting a limestone-conglomerate rock divided by a system of fractures. The towers are close to the main civil infrastructures connecting Lecco and Sondrio provinces (speedway, railway, gas, and electric distribution networks). The monitoring system consists of three sensing and processing units and a base station and has been active since July 7th, 2010. The base station relies on a Universal Mobile Telecommunications System (UMTS) modem for remote data and commands communication with the Intelligent Embedded System lab.

This dataset is available at [1]. The sampling period of the provided data is 5 minutes. We would like to point out that this scenario represents a particularly challenging application of the techniques developed within this dissertation, since it is hard to infer a precise analytical model of the observed phenomenon.

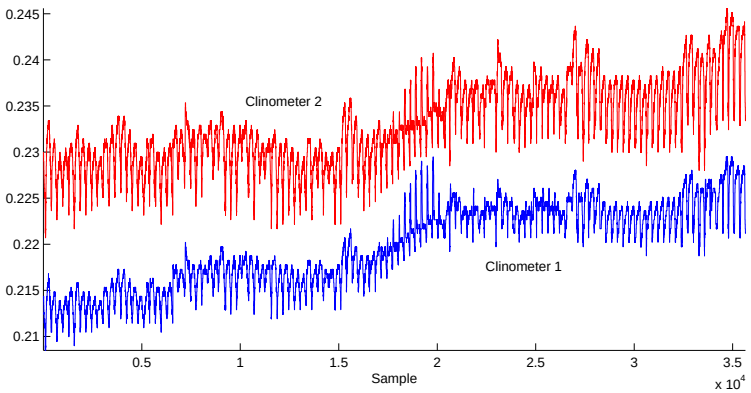


Figure 9.2: *The measurements acquired from two clinometers of the monitoring system deployed at the Towers of Rialba.*

An example of the data coming from this application are presented in

Figure 9.2, in particular here we show the measurements coming from two clinometers deployed in two different units of the sensor network.

9.1.3 Application D3: Barcelona Water Distribution Network System Dataset

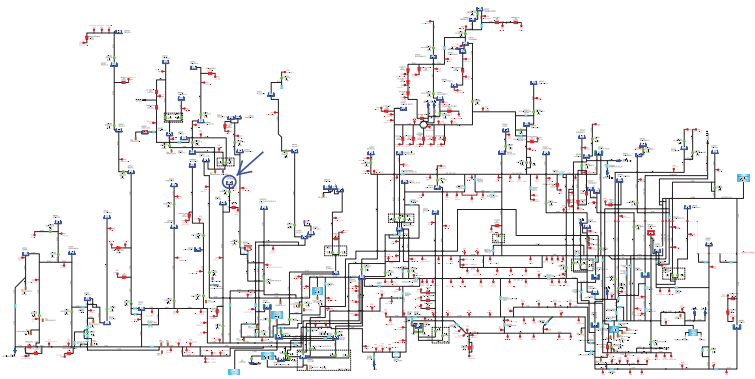


Figure 9.3: *Scheme for the BWNDS*

The second testbed refers to data generated from the BWNDS simulator [25], whose scheme is provided in Figure 9.3. By relying on a network of 17 tanks, 26 pumps, 35 valves, 9 external sources of the BWNDS, this simulator allows to artificially inject faults in a specific flow sensor of the network, by specifying the fault signature, the fault magnitude and the fault time-horizon. This simulator has been developed in `MATLAB/SIMULINK` environment, using a model calibrated and validated with real data providing a good degree of representativeness of the actual network behaviour.

9.2 Dependency Graph Learning

The aim of this section is to evaluate the performance of the proposed dependency graph learning phase using both data synthetically generated (Application D1) and coming from a real-world sensor network for rock collapse and landslide forecasting (Application D2). In particular, two different objectives have been considered and evaluated:

- the ability to correctly identify the meaningful relationships existing among the acquired datastreams;
- the fault detection performance of the HMM-CDT relying on the learned Granger-based dependency graph.

Figures of Merit With respect to the ability to identify the correct relationships in acquired datastreams, we consider the following figures of merit:

- Recall $R = \frac{|E_r \cap E|}{|E_r|}$: fraction of correctly selected functional relationships that are present in the learned dependency graph;
- Precision $P = \frac{|E_r \cap E|}{|E|}$: fraction of functional relationships in the learned dependency graphed that are correctly selected,

where E_r is the set of truly existing relationships among acquired data and $|\cdot|$ is the cardinality operator.

Differently, for the evaluation of the fault detection performance, we consider the following figures of merit:

- False Positive rate (FP): fraction of experiments in which the detection of the change happened before the change occurred;
- False Negative rate (FN): fraction of experiments where no change was detected in the relationship affected by the fault (or environmental change);
- Detection Delay (DD): number of samples necessary to detect a change, if no false positive detection occurred.

Comparison To compare the performance of the proposed dependency graph learning phase, we considered the correlation analysis adopted in [8]. More specifically we considered the maximum (w.r.t. different time lags) of the crosscorrelation between couples of datastream and we consider the relationship only if its value was higher than the chosen threshold $\rho, 0 < \rho < 1$. In particular, we considered three different values of the crosscorrelation threshold: *high* crosscorrelation (i.e., $\rho_{high} = 0.9$), *low* crosscorrelation ($\rho_{low} = 0.02$) and *best* crosscorrelation (ρ_{best} defined as the highest value of crosscorrelation that allows to consider all the truly existing relationships in the dependency graph). Obviously, the *best* crosscorrelation threshold cannot be a priori set (since it requires the knowledge of the true relationships present within the network) and, hence, it represents the correct value for the threshold. We verified a posteriori that the high and low values of the threshold have been properly set, since they actually were higher and lower than the value considered for the best one ρ_{best}, respectively.

Parameters Configuration The confidence on the Granger-based test is set to $\alpha_g = 0.05$. The parameters of the HMM-CDT have been assigned as follows: $N = 100$, $k = 10$, the best configuration for the HMMs was selected

by considering $s \in \{3, \ldots, 6\}$ hidden states and the number of GMMs is in $\{1, 2, 4, 8, 16, 32\}$. The orders of the Single Input Single Output (SISO) ARX models are chosen through a model selection procedure based on the mean square error on a validation set.

9.2.1 Application D1: Synthetic Dataset

Experimental Setting Here, we consider a sequence of 7145 samples generated from $\mathcal{S}$ or $\mathcal{L}$ with $n = 6$, $|E| = 6$ and the first $M = 4085$ used for the training and validation phase, while the remaining ones constitute the test set for the detection phase. In the experiments regarding the ability to correctly identify the meaningful relationships existing in acquired datastreams, we considered 20 equally spaced noise levels $\sigma \in [0.05\Delta_i, \ldots, \Delta_i]$, where $\Delta_i = \max_t x_i(t) - \min_t x_i(t)$. Differently, in the experiments on the fault detection performance, an additive constant abrupt fault:

A(2)	$\phi_i(t) = \begin{cases} 0.2\Delta_i & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$

has been injected at time instant $\bar{t} = 5309$ on each sensor *one at a time* (to model a sensor fault) and on all the sensors *at the same time* (to model a change in the environment[2]). Results are averaged over the different faults and repeated over 100 runs.

Results The experimental results on the ability to correctly identify the meaningful relationships are presented in Figures 9.4 to 9.7. As expected, the choice of the threshold value for the crosscorrelation analysis is critical. In fact, it is possible to see that the choice of a low correlation threshold ρ_{low} implies the selection of $|E| \approx 30$ relationships, leading to a recall $R \approx 1$. As expected, the drawback of this solution is the fact that the precision $P \approx 0.4$, thus the method is considering several non-relevant relationships. Conversely, the main drawback of the high correlation threshold ρ_{high} is the fact that relevant relationships might not be included into the dependency graph. More specifically, when $\sigma > 0.1$, the correlation analysis method does not select any relationship. As expected, by analysing Figure 9.8, we can see that the best correlation value ρ_{best} decreases as noise increases. Thus, one should rely on a priori information about the noise level to be able to set the proper threshold for the correlation analysis. Conversely, the

[2]Here, we consider the simplifying assumption that the injected change propagates instantly though the sensor network.

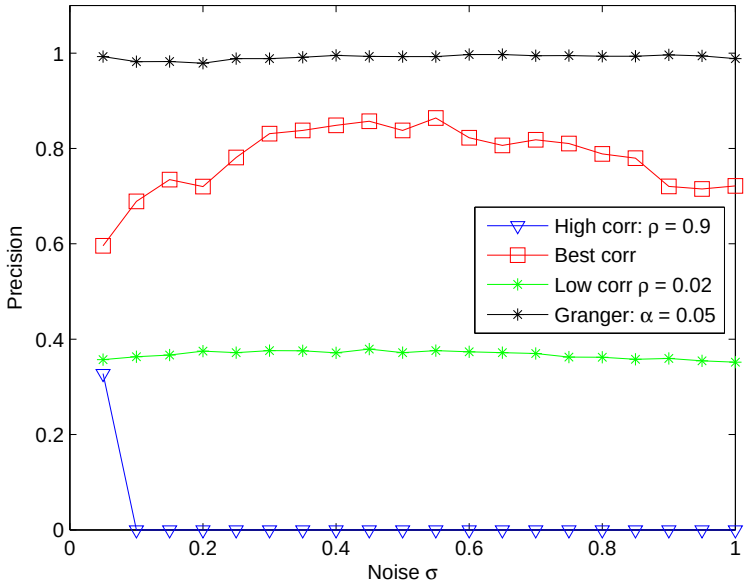


Figure 9.4: *Precision in the linear case*

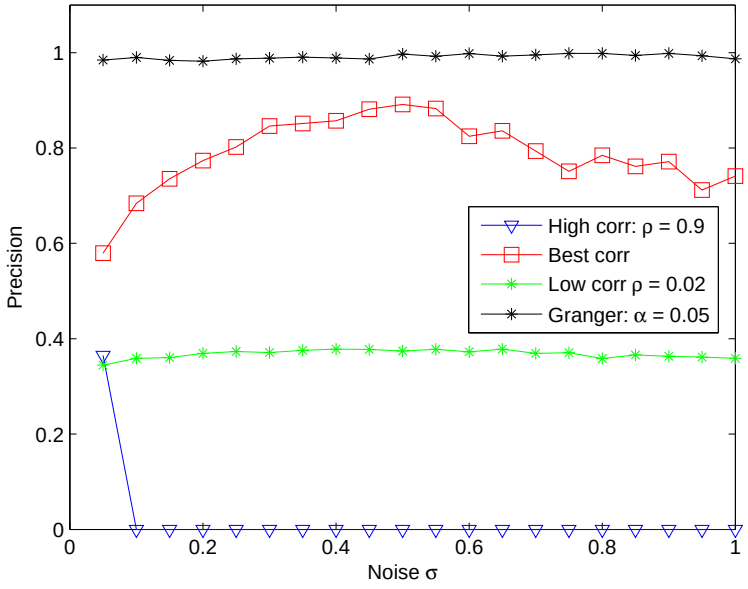


Figure 9.5: *Precision in the sinusoidal case*

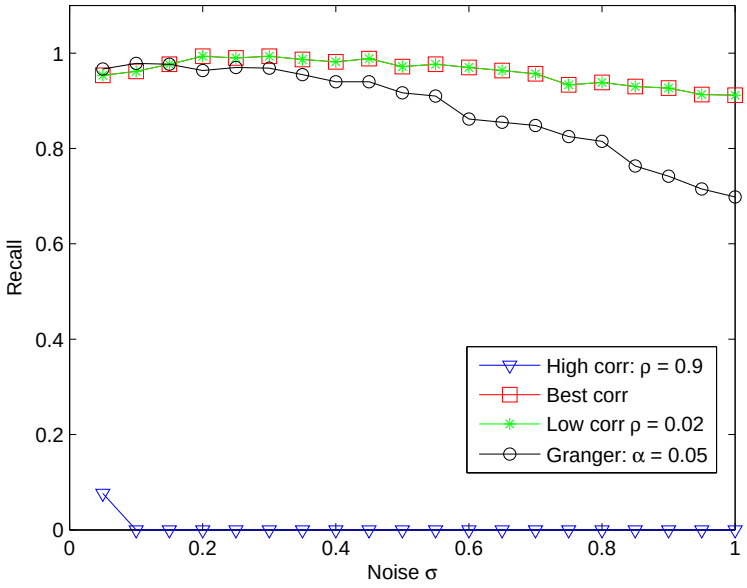


Figure 9.6: *Recall in the linear case*

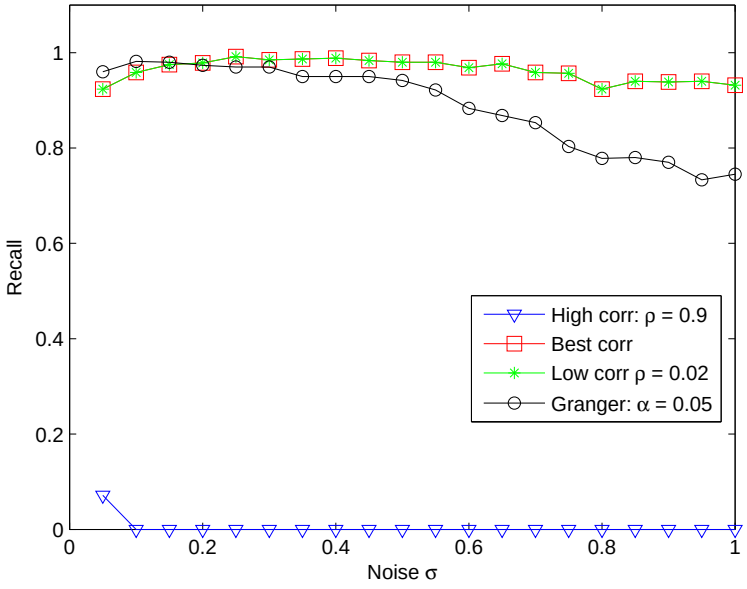


Figure 9.7: *Recall in the sinusoidal case*

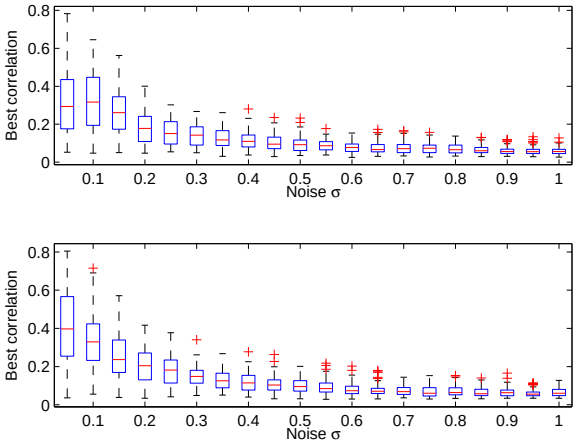


Figure 9.8: *Box-plot for the best correlation: the linear $\mathcal{L}$ (top) and sinusoidal $\mathcal{S}$ (bottom) cases*

proposed dependency graph learning phase based on Granger causality is able to maintain acceptable values for both precision and recall ($P \approx 1$, equal to the one obtained by choosing ρ_{low}, and $R > 0.7$) without any a priori information about the noise level.

Results about fault detection are shown in Table 9.1. We considered three different noise levels $\sigma \in \{0.05, 0.1, 0.3\}$ and three different values of the parameter C for the HMM-CDT, to allow a fair comparison among the proposed Granger-based framework (identified from now on with G), the crosscorrelation algorithm with ρ_{low} and the crosscorrelation algorithm with ρ_{best} (the one with ρ_{high} is not considered here). More specifically, C_{gran} is the smallest parameter that allows to have $FP = 0$ for the HMM-CDT based on the Granger-based dependency graph. Similarly C_{best} and C_{low} are the ones allowing to have $FP = 0$ for the HMM-CDT based on correlation analysis graph, by using ρ_{best} and ρ_{low}, respectively, as thresholds for the crosscorrelation. By looking at the experiments on the linear function $\mathcal{L}$ with $\sigma = 0.05$, we can see that, with C_{gran}, the Granger-based solution provides DD and FN similar to the ones obtained with ρ_{best} ($DD = 49.7$ and $FN = 0.113$ for G, $DD = 49.2$ and $FN = 0.079$ for ρ_{best}), and G provides a lower FP ($FP = 0$ for G and $FP = 0.41$ for ρ_{best}) without requiring any a priori information about the network or the noise level.

By comparing the proposed Granger-based framework with correlation

Table 9.1: *Detection results for Application D1*

		$\sigma = 0.05$			$\sigma = 0.1$			$\sigma = 0.3$		
		$C_{gran} = 4.5$			$C_{gran} = 4.9$			$C_{gran} = 4.8$		
		ρ_{best}	ρ_{low}	G	ρ_{best}	ρ_{low}	G	ρ_{best}	ρ_{low}	G
Linear function $\mathcal{L}$	$\|E\|$	10.1	16.3	**6.0**	8.3	16.0	**6.1**	7.3	15.9	**6.0**
	DD	49.2	65.0	**49.7**	87.4	93.6	**85.7**	130.3	131.9	**132.1**
	FN	0.079	0.044	**0.113**	0.126	0.064	**0.181**	0.307	0.104	**0.354**
	FP	0.410	0.650	**0.000**	0.280	0.610	**0.000**	0.120	0.620	**0.000**
		$C_{best} = 5.2$			$C_{best} = 5.2$			$C_{best} = 5.0$		
	$\|E\|$	**10.1**	16.3	6.0	**8.3**	16.0	6.1	**7.3**	15.9	6.0
	DD	**56.7**	62.0	56.9	**88.5**	103.2	92.1	**124.8**	127.6	133.0
	FN	**0.117**	0.061	0.129	**0.180**	0.080	0.206	**0.351**	0.129	0.359
	FP	**0.000**	0.390	0.030	**0.000**	0.460	0.020	**0.000**	0.530	0.020
		$C_{low} = 6.0$			$C_{low} = 6.3$			$C_{low} = 6.2$		
	$\|E\|$	10.1	**16.3**	6.0	8.3	**16.0**	6.1	7.3	**15.9**	6.0
	DD	63.1	**63.4**	62.3	109.6	**110.7**	112.3	149.0	**134.8**	156.1
	FN	0.139	**0.093**	0.154	0.217	**0.140**	0.247	0.434	**0.259**	0.450
	FP	0.000	**0.000**	0.020	0.000	**0.000**	0.000	0.000	**0.000**	0.000
		$\sigma = 0.05$			$\sigma = 0.1$			$\sigma = 0.3$		
		$C_{gran} = 5.1$			$C_{gran} = 5.1$			$C_{gran} = 4.6$		
		ρ_{best}	ρ_{low}	G	ρ_{best}	ρ_{low}	G	ρ_{best}	ρ_{low}	G
Sinusoidal function $\mathcal{S}$	$\|E\|$	10.0	16.1	**6.0**	9.0	16.0	**6.0**	7.4	15.8	**5.9**
	DD	47.5	51.5	**49.0**	84.3	84.5	**82.2**	112.0	127.0	**14.9**
	FN	0.081	0.043	**0.121**	0.111	0.047	**0.177**	0.256	0.076	**0.317**
	FP	0.350	0.610	**0.000**	0.360	0.660	**0.000**	0.200	0.670	**0.000**
		$C_{best} = 5.7$			$C_{best} = 5.7$			$C_{best} = 5.0$		
	$\|E\|$	**10.0**	16.1	6.0	**9.0**	16.0	6.0	**7.4**	15.8	5.9
	DD	**53.8**	60.2	55.2	**83.7**	86.5	90.1	**115.0**	127.8	123.3
	FN	**0.110**	0.057	0.136	**0.179**	0.076	0.200	**0.311**	0.114	0.346
	FP	**0.000**	0.360	0.030	**0.000**	0.410	0.030	**0.000**	0.500	0.000
		$C_{low} = 6.7(2.3)$			$C_{low} = 6.5(4.0)$			$C_{low} = 6.4(2.2)$		
	$\|E\|$	10.0	**16.1**	6.0	9.0	**16.0**	6.0	7.4	**15.8**	5.9
	DD	61.5	**61.3**	63.5	89.9	**91.7**	95.0	143.9	**129.5**	152.2
	FN	0.127	**0.103**	0.156	0.203	**0.131**	0.236	0.383	**0.239**	0.421
	FP	0.000	**0.000**	0.000	0.000	**0.000**	0.020	0.000	**0.000**	0.000

analysis with ρ_{low}, it presents higher values for detection delay and false positive rate ($DD = 65$ and $FP = 0.65$) and is able to maintain a lower false negative rate ($FN = 0.044$). Moreover, the amount of considered edges for G is $|E| = 6$, while both correlation analysis methods were proposing $|E| > 10$ edges.

Table 9.1 highlights in bold those results with threshold C s.t. $FP = 0$, to easily compare $|E|$, DD and FN for the considered methods. By considering the G performance with C_{gran} and those obtained with crosscorrelation analysis with ρ_{best} and C_{best}, we can observe that false negative rates for both methods are similar ($FN \approx 0.11$), while G is characterized by lower detection delay and uses a smaller set of relationships ($|E| = 6$ for G and $|E| = 10.1$ for ρ_{best}). At last, if we compare G with C_{gran} and ρ_{low} with C_{low}, we can see that the latter presents a slightly lower number of false negative detections, while G provides lower detection delay ($DD = 49.7$ for G and $DD = 63.4$ for ρ_{low}). This behaviour suggests that, if the only concern is to maintain low FN, one should consider all the available relationships at the expense of high FP. Results about the sinusoidal case $\mathcal{S}$, as well as those with different noise levels, are in line with those of the linear one $\mathcal{L}$ with $\sigma = 0.05$.

9.2.2 Application D2: Rialba Dataset

Experimental Setting In this set of experiments, we consider $n = 10$ sensors (i.e., 3 external temperature sensors, 3 in hole temperature sensors and 4 accelerometer sensors). A total of 1285 data are considered for each sensor and the first $M = 800$ samples are used for training. Abrupt additive constant faults:

A(5)	$\phi_i(t) = \begin{cases} 0.5\Delta_i & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$

has been injected at time instant $\bar{t} = 1042$ in each of the streams *one at a time* (to model fault in a sensor) and in all the sensors *at the same time* (to model a change in the environment).

Results Since we do not have information about the relevant relationships existing among acquired data, we neither can evaluate R and P, nor compute ρ_{best}. We report that, by considering the low crosscorrelation threshold ρ_{low}, 61 relationship are selected, while, with the high one ρ_{high}, 42 relationships are chosen. Differently, the proposed dependency graph learning

phase selects 19 relationships. Interestingly, 12 over 19 of those relationships are in common with the high crosscorrelation selection and 15 are in common with the low crosscorrelation one.

Table 9.2: *Detection results for Application D2: the numeric values are detection delays DD for different dependency graph learning algorithms, while FP and FN stands for false positive and false negative detections, respectively.*

	$C_{gran} = 1.271$			$C_{high} = C_{low} = 3.141$		
	DD			DD		
Faulty sensor	G	ρ_{high}	ρ_{low}	G	ρ_{high}	ρ_{low}
s_1	1	FP	FP	2	2	2
s_2	1	FP	FP	7	7	3
s_3	1	FP	FP	7	1	1
s_4	1	FP	FP	2	2	1
s_5	2	FP	FP	FN	8	8
s_6	2	FP	FP	6	3	3
s_7	1	FP	FP	2	2	1
s_8	2	FP	FP	55	55	9
s_9	2	FP	FP	5	5	5
s_{10}	1	FP	FP	3	1	1
All	1	FP	FP	1	1	1

Results for the detection phase are provided in Table 9.2. The HMM-CDT parameter C is here set to $C_{gran} = 1.271$ and $C_{high} = C_{low} = 3.141$, which are the smallest values s.t. $FP = 0$ by considering the Granger and correlation analysis approaches, respectively. The Granger-based method is able to maintain a DD generally better than the one provided by the correlation analysis. Is it also possible to see that in the case of fault located in sensor s_5 the Granger-based framework does not provide a detection (FN), supporting the idea that one should select a high number of relationships to maintain a low FN, at the expense of an increase in the FP.

9.3 Detection Phase

To evaluate the detection performance of the proposed proposed CFDDS based on the EHMM-CDT algorithm we considered the synthetic (Application D1) and a real data scenario (Application D2). Here the specific aim is the to test the proposed CFDDS to assess its detection promptness and accuracy, in terms of false positive, false negative rates and detection delays.

Figures of Merit To evaluate the detection ability of what proposed, the following figures of merit have been considered:

- False Positive rate (FP): fraction of the experiments where the method detected a change before it actually appears;
- False Negative rate (FN): fraction of the experiments where the method did not detected a change;
- Detection Delay (DD): the number of samples necessary to detect a change.

Comparison The detection phase of the proposed CFDDS is compared with the HMM-CDT, where a single HMM was considered (the one with largest loglikelihood on a validation set among those considered for the ensemble).

Parameters Configuration As regards the model family $\mathcal{M}$, we considered SISO LTI ARX models, whose orders are chosen through a model selection procedure, i.e., minimizing the mean square one-step-ahead prediction error on a validation set. The HMMs are configured by considering batches of $N = 100$ samples, parameter vector sequences of length $k = 10$ and exploring the space of HMMs with $s \in \{3, \ldots, 6\}$ hidden states and with $\{1, 2, 4, 8, 16, 32\}$ Gaussian distributions in each GMM. The cardinality of the ensemble was set to $Q = 30$. Finally, the parameter vectors estimation is performed with the LS method.

9.3.1 Application D1: Synthetic Dataset

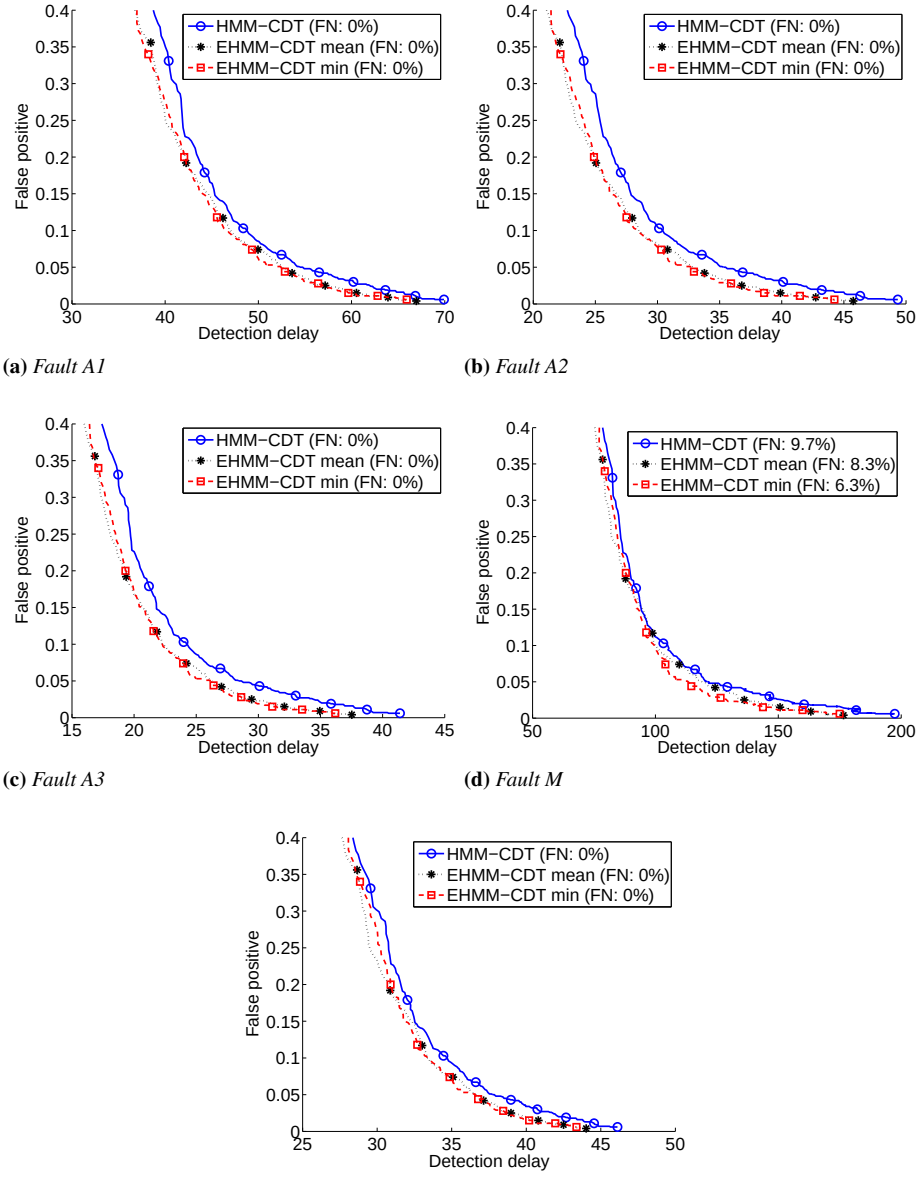
Experimental Setting The data for the synthetic application have been generated from $\mathcal{S}$ with $n = 2$ sensors, where $\theta_2 \in [0, 1]$[33], $\sigma \in \{0.01\Delta_2, 0.05\Delta_2\}$, where $\Delta_2 = \max_t x_2(t) - \min_t x_2(t)$. Each experiment lasts 6125 samples, where $M = 3268$ are used for training, $O = 817$ for validation and 2040 for testing. Results are averaged over 1000 independent runs.

To assess the detection abilities of the detection phase for the propose CFDDS, we injected five different kinds of abrupt permanent faults starting at time instant $\bar{t} = 5105$:

[3] In this set of experiments we considered only one exogenous input, i.e., the coefficient of $x_1(t-2)$ is always zero.

A(1)	$\phi_2(t) = \begin{cases} 0.1\Delta_2 & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
A(2)	$\phi_2(t) = \begin{cases} 0.2\Delta_2 & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
A(3)	$\phi_2(t) = \begin{cases} 0.3\Delta_2 & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
M(3)	$\psi_{22}(t) = \begin{cases} 0.3 & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
S	$\psi_{22}(t) = \begin{cases} -1 & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$ $\phi_{22}(t) = \begin{cases} x_2(\bar{t}) & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$

Results Figure 9.9 shows the experimental results in the case of the synthetic scenario (Application D1) and $\sigma = 0.05\Delta_2$. Curves are obtained by considering values of $C \in [1, 5.1]$, since 5.1 is the largest value of C for which $FN \leq 0.1$ for all the faulty scenarios. Interestingly, in all the considered scenarios the ensemble approach is able to improve the detection ability (in terms of both FP and DD) of the single HMM-CDT. These curves show that the performance of EHMM-CDT cannot be achieved by the HMM-CDT by simply tuning the parameter C. In addition, in the faulty scenario **M(3)** where FNs are present, the proposed EHMM-CDT behaves better than HMM-CDT even with respect to this figure of merit (see legend of Figure 9.9d).



(a) *Fault A1*

(b) *Fault A2*

(c) *Fault A3*

(d) *Fault M*

(e) *Fault S*

Figure 9.9: *Results for the detection phase on Application D1: the relationship between FP and DD is presented with* $\sigma = 0.05\Delta_2$*, where* $C \in [1, 5.1]$

Table 9.3: *Detection results for the single and ensemble HMM-CDT approaches from Application D1 (with different values for the noise levele σ) and ApplicationD2*

		HMM-CDT			EHMM-CDT					
					$\mathcal{A}_{mean}$			$\mathcal{A}_{\min}$		
		FN	FP	DD	FN	FP	DD	FN	FP	DD
$\sigma = 0.01\Delta_2$	**A(1)**	0.000	0.007	21.614	0.000	0.007	19.419	0.000	0.006	19.717
	A(2)	0.000	0.007	17.415	0.000	0.007	15.320	0.000	0.006	15.638
	A(3)	0.000	0.007	15.396	0.000	0.007	13.724	0.000	0.006	13.998
	M	0.000	0.007	37.057	0.000	0.007	35.355	0.000	0.006	35.794
	S	0.000	0.007	16.186	0.000	0.007	15.335	0.000	0.006	15.648
$\sigma = 0.05\Delta_2$	**A(1)**	0.000	0.007	67.939	0.000	0.009	63.745	0.000	0.009	64.708
	A(2)	0.000	0.007	47.465	0.000	0.009	42.625	0.000	0.009	43.281
	A(3)	0.000	0.007	39.676	0.000	0.009	34.734	0.000	0.009	35.006
	M(3)	0.077	0.007	185.989	0.057	0.009	161.401	0.055	0.009	166.641
	S	0.000	0.007	45.139	0.000	0.009	42.413	0.000	0.009	42.781
Rialba	**A(1)**	0.340	0.000	314.879	0.000	0.000	172.100	0.280	0.000	234.500
	A(2)	0.000	0.000	152.100	0.000	0.000	148.900	0.000	0.000	154.400
	A(3)	0.000	0.000	119.700	0.000	0.000	30.700	0.000	0.000	93.800
	M(3)	0.040	0.000	163.250	0.000	0.000	153.200	0.000	0.000	162.100
	S	0.000	0.000	81.500	0.000	0.000	82.300	0.000	0.000	89.600

Experimental results presented in Table 9.3 refer to FN, FP and DD with fixed values of C. To ease the comparison we set C in the EHMM-CDT and HMM-CDT as the lowest values guaranteeing $FN = 0$ and $FP \leq 0.01$ in Application D1 with $\sigma = 0.01$ (i.e., $C = 4.814$ for HMM-CDT, $C = 4.629$ for EHMM-CDT with $\mathcal{A}_{mean}$ and $C = 4.917$ for EHMM-CDT with $\mathcal{A}_{\min}$). Remarkably, both EHMM-CDT aggregations provide lower DD than HMM-CDT. Interestingly, the *mean* aggregation mechanism generally outperforms the *minimum* one, since the mean is less influenced than minimum by outliers, which generally induce false alarms. Nonetheless, by comparing these results with those provided in Figure 9.9, we may also comment that the *mean* aggregation does not provide better results for each value of C: hence the *minimum* is a viable solution too.

9.3.2 Application D2: Rialba Dataset

Experimental Setting We analysed two temperature datastreams (i.e., $n = 2$) composed by 5303 samples, where $M = 3000$ have been used for training, $O = 1000$ for validation and 1303 for testing, coming from two sensors deployed in different units. We injected faults:

A(1)	$\phi_2(t) = \begin{cases} 0.1\Delta_2 & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
A(2)	$\phi_2(t) = \begin{cases} 0.2\Delta_2 & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
A(3)	$\phi_2(t) = \begin{cases} 0.3\Delta_2 & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
M(3)	$\psi_{22}(t) = \begin{cases} 0.3 & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
S	$\psi_{22}(t) = \begin{cases} -1 & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$ $\phi_{22}(t) = \begin{cases} x_2(\bar{t}) & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$

at time instant $\bar{t} = 4695$. The presented results are obtained by averaging the results of 50 runs.

Results Experimental results are presented in Table 9.3. Interestingly, the results on the real-world datasets are in line with those of the synthetic one: the ensemble approach provides lower FN and DD for most of the considered scenarios.

9.4 Isolation Phase

In this section we consider the isolation phase of the proposed CFDDS, which is tested on a real data scenario (Application D3). We here want to inspect the ability of the isolation mechanism we proposed to correctly locate a fault in a pump/tank system, by also taking into account the delay, in term of time instant, needed for this task. Here, we considered also the detection phase to be able to compare the detection delays with the delays needed to isolate the fault. In Figure 9.10 the iOrioles subsystem

is presented: q_{in}, q_{out} and y are the incoming tank flow, consumer demand and tank level, respectively, and q_{in_m}, q_{out_m} and y_m are the corresponding measured values.

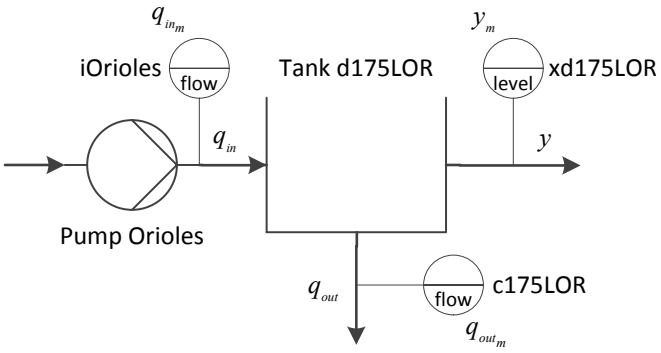


Figure 9.10: *iOrioles subsystem*

Figures of Merit The performance achieved in this fault detection stage is measured by the following figures of merit:

- False Positives (FP): percentage of test dataset faultless samples (i.e., non-affected by a certain fault) that are determined as faulty by the fault detection method;
- False Negatives (FN): percentage of test dataset faulty samples (i.e., affected by a certain fault) that are determined as faultless by the fault detection method;
- Detection delay (DD): number of samples needed by the fault diagnosis method to detect a certain fault.

Moreover fault isolation algorithms abilities are evaluated by the following figures of merit:

- Isolation Delay (ID): number of samples needed by the fault diagnosis method to isolate a certain fault;
- ISolation index (IS): percentage of samples that are properly isolated after a fault occurred.

Comparison We compare the isolation phase of the proposed CFDDS with the Physical/Temporal Parity Relations (PTPR) presented in [28], which is based on the physical modeling of the analysed system and by the temporal

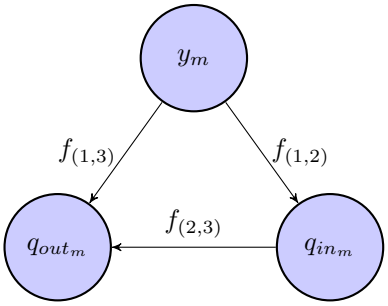


Figure 9.11: *Scheme of the iOrioles dependency graph*

analysis of the data coming from the available sensors. Clearly, if we would like to consider the PTPR method we need the model of the system, i.e., equations describing the dynamics of the system (whose parameters might be unknown). Here, we considered this method to compare the isolation performance of the proposed CFDDS, which is a model-free approach, with a method which requires strong assumptions about the system model.

Parameters Configuration The orders of ARX models have been chosen by means of a validation procedure, while the batch size and the log-likelihood window length have been set to $N = 96$ and $k = 10$, respectively. It should be mentioned that the HMM-CDT uses linear ARX models for the extraction of the estimated parameters $\hat{\theta}_i$s. The dependency graph is learned by considering all the binary relationships with crosscorrelation greater or equal to $\rho = 0.5$: the result is the graph presented in Figure 9.11.

9.4.1 Application D3: Barcelona Water Distribution Network System Dataset

Experimental Setting The dataset considered to implement this scenario lasts for 34 days (for a total of 816 samples), with a sampling period one hour and with a fault appearing at time instant $\bar{t} = 744$ in iOrioles pump sensor (q_{in_m}) or c175LOR demand sensor (q_{out_m}), respectively. Regarding the PTPR method initialization, the first thirteen days of data are used as training dataset to identify the model parameters, the next thirteen days are used as validation dataset to obtain the corresponding fault detection threshold and the remaining eight days of data is used as test dataset. Similarly, the proposed CFDDS has been trained on the first twenty-six days. More specifically, the first twenty-three days ($M = 552$) has been used to train the HMMs, while the remaining three days ($O = 72$) has been used for

validation [8].

To test and compare the performance of the proposed CFDDS isolation phase with the PTPR approach, the following fault scenarios have been defined:

AA(1,i)	$\phi_i(t) = \begin{cases} 0.1\Omega & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
AA(2.5,i)	$\phi_i(t) = \begin{cases} 0.25\Omega & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
AI(1,i)	$\phi_i(t) = \begin{cases} 0.1\Omega(1 - e^{-r_i(t-\bar{t})}) & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
AI(2.5,i)	$\phi_i(t) = \begin{cases} 0.25\Omega(1 - e^{-r_i(t-\bar{t})}) & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
DA(0.1,i)	$\phi_i(t) = \begin{cases} 0.01\Omega(t - \bar{t}) & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
DA(1,i)	$\phi_i(t) = \begin{cases} 0.1\Omega(t - \bar{t}) & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
DI(0.1,i)	$\phi_i(t) = \begin{cases} 0.01\Omega(t - \bar{t})(1 - e^{-r_i(t-\bar{t})}) & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
DI(1,i)	$\phi_i(t) = \begin{cases} 0.1\Omega(t - \bar{t})(1 - e^{-r_i(t-\bar{t})}) & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$

SA(i)	$\psi_{ii}(t) = \begin{cases} -1 & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$ $\phi_{ii}(t) = \begin{cases} x_i(\bar{t}) & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$
SI(i)	$\psi_{ii}(t) = \begin{cases} -\left(1 - e^{-r_i(t-\bar{t})}\right) & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$ $\phi_{ii}(t) = \begin{cases} x_i(\bar{t})\left(1 - e^{-r_i(t-\bar{t})}\right) & t > \bar{t} \\ 0 & \text{otherwise} \end{cases}$

where, the stream vector:

$$X(t) = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} q_{in_m} \\ q_{out_m} \\ y_m \end{pmatrix} \tag{9.5}$$

$r_i \in \mathbb{R}^+$ denotes is the constant describing the evolution rate of the fault and $\Omega = \max_t \frac{q_{in_m}(t)}{q_{out_m}(t)}$, is the maximum of flow/demand. Faults are injected either in the incoming tank flow sensor q_{in_m} or in the consumer demand sensor q_{out_m}.

Results

Table 9.5: *Detection and isolation results for the Application D3*

	Type of fault	PTPR method					CFDS method				
		Delay		FP	FN	IS	Delay		FP	FN	IS
		DD	ID				DD	ID			
q_{in_m}	AA(1,1)	2	4	0	7.34	22.22	4	4	7.44	94.52	5.48
	AA(2.5,1)	2	2	0	2.37	79.16	3	3	11.57	4.11	95.89
	AI(1,1)	12	23	0	3.04	4.16	35	35	12.40	82.19	17.81
	AI(2.5,1)	9	13	0	4.53	52.77	16	16	0.00	27.40	72.60
	DA(0.1,1)	9	13	0	2.71	73.61	8	8	13.22	10.96	89.04
	DA(1,1)	3	3	0	2.78	84.72	4	4	1.65	5.48	94.52
	DI(0.1,1)	12	23	0	3.54	59.72	18	18	9.92	24.66	75.34
	DI(1,1)	7	7	0	4.09	77.77	4	4	8.26	8.22	91.78
	SA(1)	7	12	0	7.22	8.33	17	17	10.74	68.49	31.51
	SI(1)	19	36	0	7.52	5.55	3	3	22.31	73.97	26.03
q_{out_m}	AA(1)	1	3	0	0.02	65.27	3	3	0.00	4.11	95.89
	AA(2.5,2)	1	3	0	0.02	65.27	1	1	0.00	1.37	98.63
	AI(1,2)	10	21	0	0.29	34.72	47	47	0.00	64.38	35.62
	AI(2.5,2)	7	11	0	0.13	58.33	65	65	0.00	89.04	10.96
	DA(0.1,2)	7	7	0	0.11	59.72	33	33	0.00	45.21	54.79
	DA(1,2)	2	4	0	0.04	63.88	4	4	0.00	5.48	94.52
	DI(0.1,2)	10	15	0	0.25	52.77	39	39	0.00	53.42	46.58
	DI(1,2)	5	7	0	0.11	59.72	8	8	38.02	10.96	89.04
	SA(2)	11	14	0	16.79	6.94	33	33	0.00	45.21	54.79
	SI(2)	59	-	0	92.84	0	58	58	4.13	79.45	20.55

In Table 9.5, fault detection and isolation results achieved by the considered methods are detailed. On the one hand, Table 9.5 shows generally better detection and isolation delays achieved by the PTPR method than those of the proposed CFDDS, for the incoming tank flow sensor q_{in_m} (e.g., fault **AI(1,1)**) and specially when considering the consumer demand sensor q_{out_m} (e.g., faults **AI(1,2)**, **AI(2.5,2)**, **DA(0.1,2)**, **DI(0.1,2)**, **SA(2)**). Moreover, generally better FP and FN rates are obtained with PTPR, with the only exception regarding stuck-at incipient fault in consumer demand sensor (fault **SI(2)**). Both methods achieve similar detection and isolation delay performance when considering abrupt faults (e.g., fault **AA(1,1)**, **AA(2.5,1)**, **DA(1,1)** for q_{in_m} and **AA(1,2)** for q_{out_m}). On the other hand, the proposed CFDDS grants better isolation rates in general (with the exception of faults **AA(1,1)** for the incoming tank flow sensor and **AI(2.5,2)**, **DA(0.1,2)**, **DI(0.1,2)** for the consumer demand sensor), suggesting to use it in order to confirm the isolation results provided by the PTPR. It is worth to recall that the proposed CFDDS provides the performance in Table 9.5 without assuming any a priori information about the system. In the case of faults with an increasing profile, the detection delays of the proposed CFDDS are worse than those obtained with the PTPR method. This is due to the fact that the HMM-CDT is more sensitive to abrupt changes in the parameter distribution.

Moreover, the proposed CFDDS is generally characterized by higher FP values. The reason of this behaviour is twofold: the nominal state approximation and the process time invariance. First, the nominal model is estimated during an initial training phase that, in principle, could led to inaccurate models, i.e., model bias due to an incorrect selection of the family of models, to the lack of enough data for training or to the fact that training data do not excite the whole dynamics of the process. This undesired model bias tends to induce FP detections in the testing phase. Second, the process under monitoring could be intrinsically time-varying, not following the Markov assumption. This could lead to FP detections induced by an estimated model which is not able to fully describe the true process.

9.5 Identification Phase

The aim of this section is to evaluate the effectiveness of the identification phase of the proposed CFDDS. As we have seen in Chapter 3, each state of the process (either nominal or faulty) is a cluster of parameter vectors: creation of the right number of clusters refers to the ability of the method

to correctly identify the number of states the process explores. Likewise, an accurate aggregation of parameter vectors coming from the same state refers to the ability of correctly characterizing the operational state. The identification phase of the considered CFDDS was applied to three different applications: a synthetic one (Application D1), a real-world application related to rock collapse forecasting (Application D2) and to simulated data coming from BWNDS (Application D3). On the aforementioned applications, faults are of the abrupt type as requested by the proposed CFDDS, as discussed in Section 7.3.

Figures of Merit To evaluate the performance of the suggested method, we consider the following figures of merit:

- n_c: the number of created clusters. It represents the number of states detected by the algorithm. When n_c is equal to the correct number of states the algorithm operates well;
- r: the percentage of experiments where the algorithm creates the correct number of clusters. Large values of r suggest that the fault diagnosis method is able to correctly characterize the number of process states;
- a: the accuracy in associating a parameter vector to the correct cluster. It represents the ability to correctly identify the state in which the process is operating;
- p_o: the percentage of outliers, i.e., the percentage of parameter vectors which cannot be associated to any state. Large values of p_o imply that the algorithm is not able to associate parameter vectors to any cluster.

Comparison The performance of the proposed CFDDS system has been compared with those of the DBSCAN [35], the AP [24] and the ECM [79]. In fact, to compare the performance of the proposed CFDDS, we consider algorithms designed to group unlabelled data, a task commonly addressed by clustering methods. We consider both off-line clustering algorithms, i.e., DBSCAN and AP, and an evolving one, i.e., ECM. DBSCAN and AP process the whole dataset and do not require a priori information about the number of clusters to be created, hence representing a relevant reference for the proposed CFDDS. On the contrary, ECM manages clusters with evolving strategies; the drawback here is that it requires parameter D_{thr}, which is strictly related to the number of clusters the algorithm will create

during the operational life (such information is obviously unknown in real applications).

Parameters Configuration We considered two different hierarchies of model family $\mathcal{M}$, since they satisfy the hypotheses required by the theoretical framework described in Section 3.3:

- the auto-regressive with exogenous input ARX linear model family. Here, the $p = \tau_j + \tau_i$ dimensional parameter vector $\theta \in \mathbb{R}^p$ is:

$$\theta = (\theta_1 \dots \theta_{\tau_j}\ \theta_{\tau_j+1} \dots \theta_{\tau_j+\tau_i}),$$

  where τ_j and τ_i are the autoregressive and exogenous orders, respectively;

- the RN [47] model defined as:

$$\begin{aligned} x(t) &= g\left(Wx(t-1) + W_{in}u(t)\right) \\ \hat{y}(t) &= \theta x(t) \end{aligned}$$

  where $\hat{y}(t) \in \mathbb{R}$ is the prediction value at time $t \in \mathbb{N}$, $u(t) \in \mathbb{R}^m$ is the input observation vector at time t, $x(t) \in \mathbb{R}^p$ is the internal state of the network at time t, $W \in \mathbb{R}^{p \times p}$ is the internal weight matrix and $W_{in} \in \mathbb{R}^{p \times m}$ is an input weight matrix, both randomly chosen. $g : \mathbb{R}^p \to \mathbb{R}^p$ is an activation function (e.g., $g_i(\cdot) = \tanh(\cdot), i \in \{1, \dots, p\}$) and $\theta \in \mathbb{R}^p$ is an output weight vector to be learned.

The considered structural risk is the squared error; the Bayesian Information Criterion (BIC) [75] was considered to identify model orders. In the following, batches of $N = 400$ not overlapping data are considered to estimate the parameters of the approximating models.

It is worth mentioning that the proposed CFDDS requires an initial training phase: it is trained on the training set and tested on a separate test set, while DBSCAN, AP and ECM are applied to the whole training and test set. In order to consider an unbiased dataset for the proposed CFDDS, we considered the performance of the different methods on the test set only. Since ECM and AP do not generate outliers, p_o is not provided for them.

Key parameters describing the identification phase of the proposed CFDDS are given in Table 9.6. In particular:

- the *spatial confidence* α_s has been set to 0.03. This parameter controls the rate of structural outliers generated by the proposed CFDDS. Large values of α_s would create more compact clusters and let the

Table 9.6: *Parameters of the proposed CFDDS*

Spatial confidence	$\alpha_s = 0.03$
Temporal threshold	$\eta_t = 1$
Merging confidence	$\alpha_m = 0.05$
KS-test confidence	$\alpha_c = 0.1$
Spatial-temporal penalization	$\lambda = 0.5$
Mountain method threshold	$\eta_i = 10^{-6}$

CFDDS be sensitive to new states, at the expense of a larger outlier set. On the contrary, small values of α_s would reduce the number of outliers at the expenses of a reduced sensitivity in identifying new states;

- the *temporal threshold* η_t has been set to 1, meaning that cluster statistics are updated when two consecutive parameter vectors are inserted into the same cluster. Larger values of η_t update the cluster statistics with less restrictive conditions. $\eta_t = 1$ represents a conservative choice for this parameter;

- the *merging confidence* α_m has been set set to 0.05. It represents the confidence of the hypothesis test designed to assess whether two clusters need to be merged or not;

- the *cluster creation confidence* α_c has been set to 0.1. It represents the confidence of the KS hypothesis test, meant to assess if a new cluster must be created by looking at the distribution of the parameter vectors in the outlier set.

Since the CFDDS creates clusters with as low as $p + 1$ parameter vectors (required by the minimum covariant determinant procedure), we set the DBSCAN parameter $minPts = p + 1$ to have a fair comparison. The parameter ε of DBSCAN has been set by using the heuristics described in [35]. The ECM parameter D_{thr} was set to 0.1, as suggested in [79].

9.5.1 Application D1: Synthetic Dataset

Experimental Setting Synthetic data are generated according to model $\mathcal{S}$, with $n = 2$ sensors, $\theta_2 \in [0, 1]^3$ and $\sigma = 10^{-2}$.

The length of each experiment is 60300 samples with the first $M = 24120$ ones used to train the proposed CFDDS. Faults affecting the system have been modeled as abrupt changes in the parameters θ_2. This models the situation where a fault affecting the system induces a change in the

dynamics of the relationship between input and output[4]. The first fault affects the system in sample interval $[24120, 36180]$, inducing an abrupt change which shifts the parameters from θ_2 to $\theta_\delta = (1 + \delta)\theta_2$, δ being a positive scalar controlling the intensity of the perturbation. Afterwards, the data-generating process returns to the nominal state. Then, another fault affects the system in sample interval $[48240, 60300]$, inducing a change in the parameters from θ_2 to $\theta_\delta = (1 - \delta)\theta_2$. As a consequence, the total number of states for this application is three (i.e., the nominal state and the two faulty ones). We considered different scenarios for this application by taking into account abrupt changes in the parameters with magnitude δ ranging from 0.01 to 0.3. For each scenario, we generated 200 independent experiments.

[4]It is possible to show that this type of faults can be modeled as abrupt faults depending on the input dynamic. We omit here this derivation for sake of concision.

Table 9.7: *Number of clusters created for the considered applications. Average values is given; standard deviation in parenthesis.*

		Fault	n_c			
			CFDDS	ECM	DBS	AP
Application D1	ARX	$\delta = 0.010$	1.8(0.8)	120.1(3.6)	1.0(0.1)	12.7(1.1)
		$\delta = 0.015$	3.3(0.5)	118.3(3.7)	1.0(0.2)	12.7(1.1)
		$\delta = 0.020$	3.2(0.4)	113.8(3.7)	1.1(0.4)	12.9(1.1)
		$\delta = 0.025$	3.2(0.5)	108.8(3.7)	1.8(0.7)	12.9(1.1)
		$\delta = 0.050$	3.2(0.4)	88.5(3.7)	3.0(0.2)	11.6(1.1)
		$\delta = 0.100$	3.2(0.5)	60.7(3.3)	3.0(0.0)	7.1(0.8)
		$\delta = 0.200$	3.2(0.4)	33.8(2.8)	3.0(0.0)	3.8(0.4)
		$\delta = 0.300$	3.1(0.3)	21.5(2.2)	3.0(0.0)	3.0(0.0)
	RN	$\delta = 0.010$	1.0(0.1)	89.4(20.6)	1.0(0.0)	11.2(2.0)
		$\delta = 0.015$	1.0(0.2)	90.2(21.2)	1.0(0.1)	11.0(2.0)
		$\delta = 0.020$	1.0(0.2)	88.0(22.5)	1.0(0.1)	11.2(2.1)
		$\delta = 0.025$	1.1(0.4)	90.6(22.0)	1.0(0.0)	11.1(1.9)
		$\delta = 0.050$	1.5(0.8)	88.8(22.1)	1.0(0.2)	10.9(2.0)
		$\delta = 0.100$	2.4(1.0)	84.9(18.9)	1.2(0.5)	10.1(2.2)
		$\delta = 0.200$	2.7(0.9)	68.7(22.0)	1.9(1.0)	8.1(2.9)
		$\delta = 0.300$	2.6(0.9)	52.1(23.3)	2.3(0.9)	6.6(3.1)
Application D2	ARX	$R1$	2	48	1	10
		$R2$	3	39	1	8
Application D3	ARX	$BW1$	2	38	1	4
		$BW2$	3	57	1	6
		$BW3$	4	47	2	4
		$BW4$	5	59	2	6

Results Results are presented in Table 9.7 and Table 9.8. In particular, Table 9.7 shows the number of created clusters n_c for the considered algorithms, model hierarchies and fault magnitudes. As expected, the ability to create the correct number of clusters increases with the magnitude of δ (a strong fault is easily identified). Interestingly, the proposed CFDDS based on ARX modeling is able to correctly create three clusters even with very low fault magnitudes (e.g., $\delta = 0.015$). ECM creates an excessive number of clusters making this algorithm not useful in this application. The reason of this behaviour resides in the incorrect setting of D_{thr} [35]. Unfortunately, as explained above, it is hard to set this parameter for fault diagnosis purposes, since it is related to the number of clusters to be created, which is obviously unknown a priori. Interestingly, both DBSCAN and AP with ARX are able to create the correct number of clusters, for large δ magnitudes, i.e., $\delta \geq 0.05$ and $\delta \geq 0.3$, respectively. Despite the evolving approach, the proposed CFDDS with *ARX* is more effective in creating the correct number of clusters once compared with non-evolving algorithms such as the DBSCAN and the AP even for small δs. The rationale behind this refers to the fact that the proposed CFDDS is able to simultaneously consider both temporal and spatial dependencies among parameter vectors.

RNs provide lower performance than ARX. The reason of this behaviour can be associated to the fact that the performance of RNs is highly influenced by the choice of the random network topology. In fact, training the network topology is entirely based on nominal state samples. This leads to a RN modeling the nominal state, but does not necessarily guarantee the ability to identify new states during the operational life. However, the ability to create the correct number of clusters increases with δ and the proposed CFDDS with RN is able to identify the correct number of faults with magnitude $\delta \geq 0.2$.

Table 9.8: *Accuracy results for the considered applications. Average value is given; standard deviation in parenthesis.*

		Fault	CFDDS			ECM		DBS			AP	
			r	a	p_o	r	a	r	a	p_o	r	a
Application D1	ARX	$\delta = 0.010$	12.0	51.7(9.3)	3.4(3.8)	0.0	N.a.	0.0	N.a.	N.a.	0.0	N.a.
		$\delta = 0.015$	72.5	84.5(9.6)	2.7(2.8)	0.0	N.a.	0.0	N.a.	N.a.	0.0	N.a.
		$\delta = 0.020$	82.0	95.4(3.5)	3.1(3.3)	0.0	N.a.	1.5	55.9(12.0)	6.7(2.9)	0.0	N.a.
		$\delta = 0.025$	83.5	96.5(2.8)	3.0(2.8)	0.0	N.a.	13.0	92.1(7.3)	5.6(2.4)	0.0	N.a.
		$\delta = 0.050$	82.0	96.2(3.4)	3.5(3.4)	0.0	N.a.	97.5	97.6(1.8)	2.4(1.8)	0.0	N.a.
		$\delta = 0.100$	81.5	96.7(3.1)	2.6(3.1)	0.0	N.a.	100.0	99.4(0.8)	0.6(0.8)	0.0	N.a.
		$\delta = 0.200$	86.5	96.5(2.9)	2.6(2.8)	0.0	N.a.	100.0	100.0(0.0)	0.0(0.2)	17.5	100.0(0.0)
		$\delta = 0.300$	88.0	96.8(2.8)	2.4(2.8)	0.0	N.a.	100.0	100.0(0.0)	0.0(0.0)	100.0	100.0(0.0)
	RN	$\delta = 0.010$	0.0	N.a.	N.a.	0.0	N.a.	0.0	N.a.	N.a.	0.0	N.a.
		$\delta = 0.015$	0.0	N.a.	N.a.	0.0	N.a.	0.0	N.a.	N.a.	0.0	N.a.
		$\delta = 0.020$	0.5	66.7(0.0)	1.1(0.0)	0.0	N.a.	0.0	N.a.	N.a.	0.0	N.a.
		$\delta = 0.025$	3.5	63.0(23.0)	3.0(3.0)	0.0	N.a.	0.0	N.a.	N.a.	0.0	N.a.
		$\delta = 0.050$	16.5	80.9(16.8)	3.7(3.4)	0.0	N.a.	1.5	97.8(1.1)	2.2(1.1)	0.0	N.a.
		$\delta = 0.100$	45.0	90.9(10.5)	3.9(4.4)	0.0	N.a.	6.5	97.9(2.6)	2.0(2.7)	0.0	N.a.
		$\delta = 0.200$	60.5	94.3(7.0)	3.4(4.6)	0.0	N.a.	38.5	98.5(3.6)	1.1(1.7)	4.0	100.0(0.0)
		$\delta = 0.300$	58.5	95.3(6.8)	3.0(3.5)	0.0	N.a.	63.5	99.3(1.5)	0.7(1.5)	25.5	100.0(0.0)
Application D2	ARX	$R1$	100	90.6	3.8	0	N.a.	0	N.a.	N.a.	0	N.a.
		$R2$	100	92.5	0.0	0	N.a.	0	N.a.	N.a.	0	N.a.
Application D3	ARX	$BW1$	100	95.5	0.0	0	N.a.	0	N.a.	N.a.	0	N.a.
		$BW2$	100	81.8	0.0	0	N.a.	0	N.a.	N.a.	0	N.a.
		$BW3$	100	81.5	0.0	0	N.a.	100	56.9	4.0	0	N.a.
		$BW4$	100	80.5	3.4	0	N.a.	0	N.a.	N.a.	0	N.a.

Then, in order to evaluate the ability to correctly identify the states where the process operates over time, we focus only on those experiments for which the number of created clusters is correct. Table 9.8 shows r, a and p_0 for ARX, RN and different fault magnitudes, when the number of clusters created by the algorithm is correct (i.e., $n_c = 3$ for Application D1).

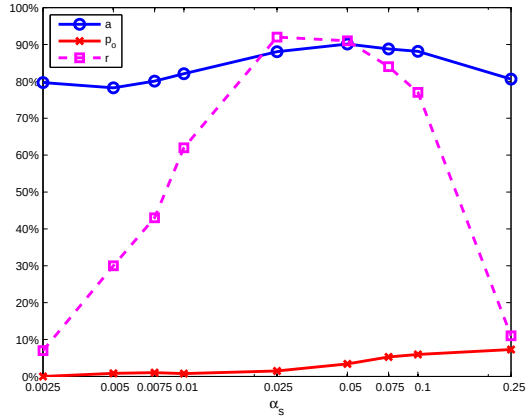
As expected, the proposed CFDDS improves its performance both in terms of percentage of experiments which identified the correct number of clusters r and in terms of the classification accuracy a as the magnitude of the fault increases. Interestingly, when $\delta < 0.015$ the CFDDS with ARX reduces its effectiveness in the clustering (i.e., $r = 12.0\%$ and $a = 51.7\%$) meaning that small fault magnitudes represent challenging situations for the proposed approach. In our opinion, this behaviour is due to the fact that the real parameter vectors corresponding to faulty states with magnitude $\delta \leq 0.01$ are included in the neighbourhood R_{α_s} of probability $1 - \alpha_s$ induced by the cluster corresponding to the nominal state, thus even the parameter vectors estimated from faulty data are associated by the proposed algorithm to the nominal state cluster.

Furthermore, the analysis of p_0 allows us to evaluate the effect of the choice of the proposed CFDDS parameters on performance. More specifically, the parameter α_s controls the percentage of structural outliers of the CFDDS and this is particularly evident by looking at the values of p_o in Table 9.8 which are in line with what expected from the theory. On the contrary, the percentage of outliers of DBSCAN decreases, when the fault magnitude increases. This is reasonable since the method does not contemplate a fixed percentage of structural outliers.

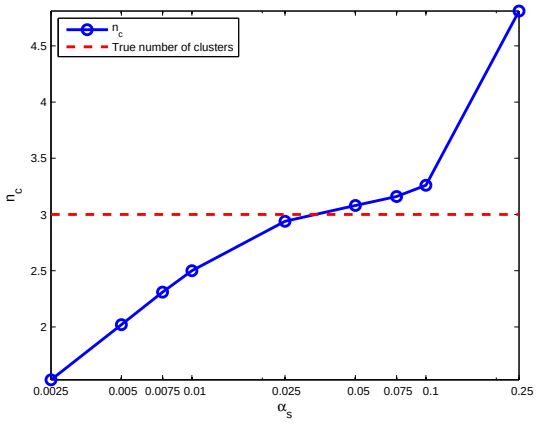
By inspecting accuracy a, we see that the CFDDS with ARX provides higher performance than the one with RN in the small perturbation case ($\delta \leq 0.025$). As the magnitude δ increases, there is no strong evidence for selecting a specific model family. Nevertheless, the standard deviation of the accuracy of ARX model is lower than the RN model one. This behaviour is in line with the difficulties in selecting the RN topology following the discussions given for Table 9.7.

As expected, non evolving clustering algorithms like DBSCAN or AP provide higher performance than the proposed CFDDS when $\delta = 0.3$. This is reasonable since these algorithms work in an off-line way, by analyzing the whole dataset at once. On the contrary, the proposed CFDDS provides better performance than DBSCAN and AP with small values of δ, making it suitable to manage subtle and not evident faults. Even in this case, the reason of this behaviour resides in the ability of the method to exploit temporal

dependencies among parameter vectors during the operational life (while not evolving algorithms do not exploit time dependencies in the clustering phase). ECM was never able to correctly identify the number of clusters, in line with comments following Table 9.7.



(a) *Average accuracy a, outlier percentage p_o and percentage of experiments where the algorithm creates the correct number of clusters r are reported for the experiments where $n_c = 3$.*



(b) *Average number of cluster n_c created with different values of α_s*

Figure 9.12: *Robustness analysis result for α_s*

We also performed a robustness analysis to evaluate the effects of variations of the main parameters of the proposed CFDDS, i.e., α_s, α_m, η_t and λ, on the considered figures of merit. In the considered scenario (Application D1), which is characterized by a stationary process affected by abrupt

changes, parameter α_s revealed to be the most sensitive one and its behaviour is deeply investigated in the sequel. Figures 9.12a and 9.12b show how the figures of merit a, p_0, r and n_c vary with α_s ranging in the interval $[2.5 \cdot 10^{-3}; 2.5 \cdot 10^{-1}]$. As expected, p_0 increases with α_s and this is quite obvious since we are creating clusters that are more and more compact. For the considered scenario, $\alpha_s = 0.025$ guarantees the highest value of r. Interestingly, lower values of α_s create a reduced number of clusters, while larger ones create an excessive number of clusters. This behaviour is evident by looking at the values of n_c in Figure 9.12b. The behaviour of the classification accuracy a is particularly interesting: small values of α_s create very large clusters, hence possibly misclassifying estimated parameter vectors that belong to a different state (e.g., a faulty one). On the contrary, large values of α_s create very small clusters, hence generating many outliers (and this is evident by looking at the behaviour of p_0 when α_s increases). Due to the fact that α_s is closely related to the confidence of including a parameter vector in a cluster (i.e., $1 - \alpha_s$), reasonable values for this parameters belong to the interval $0.01 \leq \alpha_s \leq 0.1$.

9.5.2 Application D2: Rialba Dataset

Experimental Setting The dataset here considered, consists in 35652 samples coming from two clinometers. In this application, the first $M = 14260$ samples have been used to train the proposed CFDDS. Two different faulty scenarios have been considered:

R1	An abrupt additive fault affecting the measurements of the clinometer, regarded as output, is injected in sample interval $[17468, 24956]$. The magnitude of the additive fault is -20% of the signal dynamics
R2	The first 24956 samples are equal to the **R1** case. Then, a degradation fault is injected in the same clinometer in sample interval $[28164, 35652]$. The degradation fault consists in an additive Gaussian noise with zero mean and standard deviation equal to 30% of the signal one

As a model hierarchy family we here considered the ARX one. Experimental results on this application are particularly interesting since data are coming from a real monitoring system.

Results In Table 9.8 the value of r is either 0 or 100, since here we are considering a single experiment. The proposed CFDDS accuracy in **R1**

and **R2**, presented in Table 9.8, are similar (i.e., 90.6% in **R1** and 92.5% in **R2**), showing that we are able to deal effectively with multiple faults.

By looking at Table 9.7, we see that the number of states of the process is correctly recognized by the proposed CFDDS in both scenarios whereas other methods are never able to create the correct number of clusters. These results are in line with the synthetic scenario (Application D1): AP and ECM are creating more clusters than necessary and DBSCAN is creating a single cluster.

9.5.3 Application D3: Barcelona Water Distribution Network System Dataset

Experimental Setting In this third scenario, we are considering a MISO ARX model for the estimation of the parameter vectors, where as output we considered the value of the flow in the `iOrioles` pump and as input the values of the flow in pumps near to it, i.e., `iStamClmCervello`, `iCesalpina1`, `iCesalpina2` and . Four different scenarios have been considered. The proposed CFDDS has been trained on the first $M = 8736$ samples (representing one year of observations in the BWNDS simulator); as a reference model we consider the ARX.

Four different faulty scenarios have been considered:

BW1	An abrupt additive fault affecting the measurements of the `iOrioles` pump is injected in sample interval $[9546, 17472]$. The magnitude of the additive fault is -20% of the signal dynamic (i.e., the range between the maximum and minimum value of the signal). The length of the dataset is 17472 samples
BW2	A sensor degradation fault is injected in sample interval $[18282, 26208]$. This fault consists in an additive Gaussian noise with zero mean and standard deviation equal to 30% of the signal one. The length of the dataset is 26208 samples. The first 17472 samples are equal to the **BW1** case
BW3	A stuck-at fault is injected in sample interval $[27018, 34944]$. The length of the dataset is 34944 samples. The first 26208 samples are equal to the **BW2** case
BW4	An abrupt additive fault affecting the measurements of the `iOrioles` pump is injected in sample interval $[35754, 43680]$. The magnitude of the additive fault is 20% of the range of the signal. The length of the dataset is 43680 samples. The first 34944 samples are equal to the **BW3** case

Results Results given in Table 9.7 (last four rows) are particularly interesting and show how the proposed CFDDS is able to correctly identify the number of clusters in all the four considered scenarios. All the other considered methods do not identify the correct number of clusters (with the exception of AP in the **BW3** scenario). In line with the synthetic application experiments, AP and DBSCAN usually detect a smaller (1-2) and larger (4-6) number of clusters than necessary, respectively, while ECM creates an excessive number of clusters, i.e., from 38 to 59. These results corroborate the ability of the proposed CFDDS method to correctly characterize the states explored by the process over time.

In Table 9.8 (last four rows) the value of r is either 0 or 100, since here we are considering a single experiment. CFDDS accuracy decreases from 95.5% in **BW1** to 81.8% in **BW2**, while there is no further significant reduction in accuracy in the other scenarios. In our opinion in scenario **BW2**, the injected degradation fault is particularly hard to be detect, since its effect on the estimated parameter vectors is not as evident as those induced by the other considered faults.

We emphasize that, to ease the comparison, **R1** and **R2** in the Rialba dataset (Application D2) correspond to **BW1** and **BW2** in the BWNDS one (Application D3), respectively. With respect to the BWNDS application, in the Rialba one we do not have a decrease in performance as the degradation fault appears, suggesting that, in this application, the effect of the degradation fault is more easy to be perceived by the proposed CFDDS.

9.6 General Remarks

The experimental campaign we presented above provides evidence for the abilities of the proposed CFDDS to perform the fault detection and diagnosis tasks (graph learning, detection, isolation and identification) on the considered sensor network scenarios. At first, it was shown that an effective learning of the dependency graph improves the performance in detecting faults or changes in the environment. Then, we showed that the use of an ensemble of HMM-CDTs improves the detection performance on a single HMM-CDT. The isolation phase of the proposed CFDDS is able to provided results which are most of the times comparable with those obtained with the PTPR approach. Interestingly, the proposed solution does not require any a priori information about the system, while the latter one requires the analytic model for the data generating process. Finally, results on the identification phase of the proposed CFDDS provided evidence for its ability to characterize the nominal state and to identify and learn new faulty states.

CHAPTER 10

Concluding Remarks

Sensor networks are well-known technological solutions to monitor complex systems or environmental phenomena. These sensing infrastructures work in harsh environmental conditions, which might induce degradation of the mechanical parts of the sensors and malfunctions in the embedded electronic boards. Hence, an effective fault detection and diagnosis activity is required. Since traditional methods for fault detection and diagnosis generally require the knowledge of a priori information about the data generating process or the faults, in the last few years a novel generation of fault detection and diagnosis systems has been developed. These systems, that are usually addressed as "cognitive", are able to characterize the nominal conditions of a system and learn the faults, by exploiting functional relationships present in the acquired data.

In this dissertation we propose a novel Cognitive Fault Detection and Diagnosis System (CFDDS), which operates in the sensor network scenario. This system is able to automatically characterize the nominal state of the process inspected by the network, by relying on the spatial and temporal redundancies provided by data gathered by the sensor network, and to detect the occurrence and characterize faults, as they appear. This method does not require a priori information about either the process in nominal condi-

tions, or the signature of the occurring faults (fault dictionary), but relies on a dataset composed of fault-free data that are used to characterize the nominal conditions of the data generating process.

The proposed CFDDS is able to learn the dependency graph corresponding to the functional dependencies existing among data by relying on a statistical test based on the Granger causality. This procedure does not constitute a diagnostic phase per se, but represents the basis of all the other detection and diagnosis phases.

The detection phase is carried out by means of a HMM modeling of the parameter vectors estimated on data coming from the nominal state of the functional relationships included in the previously learned dependency graph. A change in a functional relationship is detected by inspecting the loglikelihood provided by the HMM. An ensemble method for HMM-CDT has been proposed to further improve the detection performance.

Once the fault has been detected, the joint use of a partition of the learned dependency graph and the statistical modeling given by the HMM is used to discriminate between a change in the data generating process or a fault. In the case a fault occurred, the isolation phase is able also to identify the location where it appeared.

The proposed CFDDS is able to identify different faults by means of a novel evolving clustering algorithm, which is able to learn the characterization of the nominal state of the process in the parameter space, i.e., by creating a cluster of estimated parameter vectors. Newly estimated parameter vectors are assigned to either the nominal cluster or to one of the faulty ones. The method, starting from an empty fault dictionary, is also able to characterize new faulty states in an on-line manner.

The CFDDS performance has been tested on a synthetically generated nonlinear dataset and on two real testbed scenarios, i.e., data coming from a rock collapse and landslide forecasting system and from a system monitoring a water distribution network. The chosen real benchmarks are valuable examples of systems where the need for a flexible and reliable framework for fault detection and diagnosis is of paramount importance. In the former application the data generating process is not known, thus a priori information about the nominal state of the process is missing. Moreover, a direct inspection of the sensors is far from being trivial. In the latter system, the difficulty in obtaining a precise modeling of the whole network, as well as the difficulty in accessing the possibly faulty components (e.g., pipes and tanks) would benefit from a system which is able to detect and diagnose changes and faults by relying solely on the inspection of the data coming from sensors. By considering the performance of the different phases (i.e.,

dependency graph learning, detection, isolation and identification) of the proposed CFDDS one at a time we showed that it outperforms the state of the art methods for fault detection and diagnosis in the sensor network scenario, thus providing a complete and useful tool for real world scenarios.

10.1 Future Perspectives

Up to present times, the study of the field of fault detection and diagnosis in the sensor network scenario is far from being completed. For instance, in the last few years the problem of handling huge amount of information (big data) has been raising interest in the scientific and industrial fields. In the sensor network scenario we could consider two different situations where this problem is relevant: when the sensor network is characterized by a very high number of sensors and when it is not possible to store all the data gathered from the network over time (e.g., due to a very high sampling rate). In the former case, the study of appropriate techniques for handling dependency graphs with a high amount of nodes is required, for instance by considering portions of the network one at a time and then combining the decisions taken over each subset of the partition. Another viable solution is to apply dimensionality reduction techniques, where the goal is not to discard the redundant information provided by the network, but to use it for fault diagnosis purposes. In the latter case, we should implement algorithms which are able to process data as they come (one pass algorithms) and update the knowledge base incrementally and in an evolving manner. The proposed CFDDS, in its evolving clustering identification phase, is able to address these issues, but the learning of the HMM in the detection phase has not been designed to take into account this scenario.

In this dissertation we did not take into account the accommodation of occurred faults. It could be interesting to be able to deal with faults by maintaining an acceptable level of service provided by the application that bases its decision on the sensor network data, even if a sensor is no more reliable. For instance, one could consider systems which are able to either reconstruct the signal that a faulty sensor would have provided (virtual sensor) or to exclude the faulty sensor from the decision making algorithm. In the former case efficient and reliable mechanisms for the reconstruction of the signal should be considered, for instance by exploiting data coming from other sensors and from past observations of the faulty sensor itself. In the latter case, mechanisms which are able to reconfigure both the application and the fault detection and diagnosis system should be designed.

List of Figures

List of Tables

Bibliography

[1] isense project website: `http://www.i-sense.org/`, mar 2014.

[2] `matlab` cognitive fault diagnosis system toolbox and dataset: `http://home.deib.polimi.it/trovo/downloads`, jan 2015.

[3] Raj K Aggarwal, QY Xuan, Allan T Johns, Furong Li, and Allen Bennett. A novel approach to fault diagnosis in multicircuit transmission lines using fuzzy artmap neural networks. *Neural Networks, IEEE Transactions on*, 10(5):1214–1221, 1999.

[4] Richard A Ajayi, Joseph Friedman, and Seyed M Mehdian. On the relationship between stock returns and exchange rates: tests of granger causality. *Global Finance Journal*, 9(2):241–251, 1999.

[5] C. Alippi, R. Camplani, C. Galperti, A. Marullo, and M. Roveri. An hybrid wireless-wired monitoring system for real-time rock collapse forecasting. In *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, pages 224–231. IEEE, 2010.

[6] C. Alippi, R. Camplani, C. Galperti, A. Marullo, and M. Roveri. A high-frequency sampling monitoring system for environmental and structural applications. *ACM Transactions on Sensor Networks (TOSN)*, 9(4):41, 2013.

[7] C. Alippi, S. Ntalampiras, and M. Roveri. An hmm-based change detection method for intelligent embedded sensors. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–7. IEEE, 2012.

[8] C. Alippi, S. Ntalampiras, and M. Roveri. A cognitive fault diagnosis system for distributed sensor networks. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(8):1213 – 1226, 2013.

[9] C. Alippi, S. Ntalampiras, and M. Roveri. Model ensemble for an effective on-line reconstruction of missing data in sensor networks. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–6. IEEE, 2013.

[10] C. Alippi, M. Roveri, and F. Trovò. A "learning from models" cognitive fault diagnosis system. In *Artificial Neural Networks and Machine Learning–ICANN 2012*, pages 305–313. Springer, 2012.

[11] C. Alippi, M. Roveri, and F. Trovò. A self-building and cluster-based cognitive fault diagnosis system for sensor networks. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(6):1021 – 1032, 2014.

[12] Cesare Alippi. *Intelligence for Embedded Systems*. Springer, 2014.

[13] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri. On-line reconstruction of missing data in sensor/actuator networks by exploiting temporal and spatial redundancy. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012.

[14] Cesare Alippi, Stavros Ntalampiras, and Manuel Roveri. Model ensemble for an effective on-line reconstruction of missing data in sensor networks. In *IJCNN*, pages 1–6, 2013.

[15] Cesare Alippi, Manuel Roveri, and Francesco Trovò. Learning causal dependencies to detect and diagnose faults in sensor networks. In *Computational Intelligence, Intelligent Embedded Systems (IES), 2014 IEEE Symposium on*. IEEE, 2014 (to appear).

[16] T.W. Anderson. *The statistical analysis of time series*, volume 19. John Wiley & Sons, 2011.

[17] P.P. Angelov, P. Angelov, D.P. Filev, and N. Kasabov. *Evolving intelligent systems: methodology and applications*, volume 12. Wiley-IEEE Press, 2010.

[18] L. Barnett and A.K. Seth. The mvgc multivariate granger causality toolbox: A new approach to granger-causal inference. *Journal of neuroscience methods*, 223:50–68, 2014.

[19] A.B. Barrett, L. Barnett, and A.K. Seth. Multivariate granger causality and generalized variance. *Physical Review E*, 81(4):041907, 2010.

[20] Adam B Barrett, Michael Murphy, Marie-Aurélie Bruno, Quentin Noirhomme, Mélanie Boly, Steven Laureys, and Anil K Seth. Granger causality analysis of steady-state electroencephalographic signals during propofol-induced anaesthesia. *PloS one*, 7(1):e29072, 2012.

[21] Michèle Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs, 1993.

[22] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41:164–171, 1970.

[23] Avrim Blum. Random projection, margins, kernels, and feature-selection. In *Subspace, Latent Structure and Feature Selection*, pages 52–68. Springer, 2006.

[24] Frey Brendan J. and Dueck Delbert. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[25] E. Caini, V. Puig Cayuela, G. Cembrano Gennari, et al. Development of a simulation environment for water drinking networks: Application to the validation of a centralized mpc controller for the barcelona case study. Technical report, Universitat Politècnica de Catalunya, 2010.

[26] H. Chen, P. Tino, A. Rodan, and Xin Yao. Learning in the model space for cognitive fault diagnosis. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(1):124–136, Jan 2014.

[27] J.B. Comly, P.P. Bonissone, and M.E. Dausch. *Fuzzy logic for fault diagnosis*. GE Research & Development Center, 1990.

[28] M.A. Cuguero, J. Quevedo, C. Alippi, M. Roveri, V. Puig, D. Garcia, and F. Trovò. Fault diagnosis in drinking water transport networks: Physical/temporal parity relations vs. hmm-based cognitive method. *Journal of Hydroinformatics*, 2014 (submitted).

[29] R.I.A. Davis and B.C. Lovell. Comparing and evaluating hmm ensemble training algorithms using train and test and condition number criteria. *Formal Pattern Analysis & Applications*, 6(4):327–335, 2004.

[30] M.A. Demetriou and M.M. Polycarpou. Incipient fault diagnosis of dynamical systems using online approximators. *Automatic Control, IEEE Transactions on*, 43(11):1612–1617, 1998.

[31] A.P. Dempster et al. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal statistical Society*, 39(1):1–38, 1977.

[32] Gopikrishna Deshpande, Stephan LaConte, George Andrew James, Scott Peltier, and Xiaoping Hu. Multivariate granger causality analysis of fmri data. *Human brain mapping*, 30(4):1361–1373, 2009.

[33] Y. Ding, P. Kim, D. Ceglarek, and J. Jin. Optimal sensor distribution for variation diagnosis in multistation assembly processes. *Robotics and Automation, IEEE Transactions on*, 19(4):543–556, 2003.

[34] R.J. Elliott, L. Aggoun, and J.B. Moore. *Hidden Markov Models*. Springer, 1995.

[35] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of KDD*. KDD, 1996.

[36] J. Farrell, T. Berger, and BD Appleby. Using learning techniques to accommodate unanticipated faults. *Control Systems, IEEE*, 13(3):40–49, 1993.

[37] J. Gertler. *Fault detection and diagnosis in engineering systems*. CRC, 1998.

[38] C.W.J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, 27(3):424–438, July 1969.

[39] James Douglas Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[40] J. Hardin and D.M. Rocke. Outlier detection in the multiple cluster setting using the minimum covariance determinant estimator. *Computational Statistics & Data Analysis*, 44(4):625–638, 2004.

[41] Wolfram Hesse, Eva Möller, Matthias Arnold, and Bärbel Schack. The use of time-variant eeg granger causality for inspecting directed interdependencies of neural assemblies. *Journal of neuroscience methods*, 124(1):27–44, 2003.

[42] G.B. Huang, Q.Y. Zhu, and C.K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

[43] Yann-Chang Huang. Evolving neural nets for fault diagnosis of power transformers. *Power Delivery, IEEE Transactions on*, 18(3):843–848, 2003.

[44] Yann-Chang Huang, Hong-Tzer Yang, and Ching-Lien Huang. Developing a new transformer fault diagnosis system through evolutionary fuzzy logic. *Power Delivery, IEEE Transactions on*, 12(2):761–767, 1997.

[45] R. Isermann. *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Verlag, 2006.

[46] R. Isermann. *Fault-Diagnosis Applications: Model-Based Condition Monitoring: Actuators, drives, machinery, plants, sensors, and fault-tolerant systems*. Springer, 2011.

[47] H. Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach*. GMD-Forschungszentrum Informationstechnik, 2002.

[48] S. Jaiyen, C. Lursinsap, and S. Phimoltares. A very fast neural learning for classification using only new incoming datum. *Neural Networks, IEEE Transactions on*, 21(3):381–392, 2010.

[49] A. Joentgen, L. Mikenina, R. Weber, A. Zeugner, and H.J. Zimmermann. Automatic fault detection in gearboxes by dynamic fuzzy data analysis. *Fuzzy Sets and Systems*, 105(1):123–132, 1999.

[50] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[51] Daniel Kaplan and Leon Glass. *Understanding nonlinear dynamics*, volume 19. Springer, 1995.

[52] A. Khan and D. Ceglarek. Sensor optimization for fault diagnosis in multi-fixture assembly systems with distributed sensing. *TRANSACTIONS-AMERICAN SOCIETY OF MECHANICAL ENGINEERS JOURNAL OF MANUFACTURING SCIENCE AND ENGINEERING*, 122(1):215–226, 2000.

[53] A.N. Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *Giornale dell'Istituto Italiano degli Attuari*, 4(1):83–91, 1933.

[54] A. Krogh and P. Sollich. Statistical mechanics of ensemble learning. *Physical Review E*, 55(1):811, 1997.

[55] Anders Krogh, BjoÈrn Larsson, Gunnar Von Heijne, and Erik LL Sonnhammer. Predicting transmembrane protein topology with a hidden markov model: application to complete genomes. *Journal of molecular biology*, 305(3):567–580, 2001.

[56] M. Krysander and E. Frisk. Sensor placement for fault diagnosis. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(6):1398–1410, 2008.

[57] H.C. Kuo and H.K. Chang. A new symbiotic evolution-based fuzzy-neural approach to fault diagnosis of marine propulsion systems. *Engineering Applications of Artificial Intelligence*, 17(8):919–930, 2004.

[58] L. Ljung. Convergence analysis of parametric identification methods. *Automatic Control, IEEE Transactions on*, 23(5):770–783, 1978.

[59] L. Ljung and P.E. Caines. Asymptotic normality of prediction error estimators for approximate system models. In *Decision and Control including the 17th Symposium on Adaptive Processes, 1978 IEEE Conference on*, volume 17, pages 927–932. IEEE, 1978.

[60] Lennart Ljung. *System identification*. Wiley Online Library, 1999.

[61] Xinsheng Lou and Kenneth A Loparo. Bearing fault diagnosis based on wavelet transform and fuzzy inference. *Mechanical systems and signal processing*, 18(5):1077–1095, 2004.

[62] D.J.C. MacKay. Ensemble learning for hidden markov models. Technical report, Technical report, Cavendish Laboratory, University of Cambridge, 1997.

[63] P.C. Mahalanobis. On the generalized distance in statistics. In *Proceedings of the National Institute of Sciences of India*, volume 2, pages 49–55. New Delhi, 1936.

[64] R. Naresh, V. Sharma, and M. Vashisth. An integrated neural fuzzy approach for fault diagnosis of transformers. *Power Delivery, IEEE Transactions on*, 23(4):2017–2024, 2008.

[65] O. Nasraoui and C. Rojas. Robust clustering for tracking noisy evolving data streams. In *Proc. 2006 SIAM Conf. on Data Mining (SDM 2006)*, pages 80–99, 2006.

[66] Philippe Pébay. Formulas for robust, one-pass parallel computation of covariance and arbitrary-order statistical moments. Technical report, Sandia National Laboratories, 09 2008.

[67] M.D. Penrose. Extremes for the minimal spanning tree on normally distributed points. *Advances in Applied Probability*, 30(3):628–639, 1998.

[68] M.P. Perrone and L.N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, DTIC Document, 1992.

[69] J Quevedo, V Puig, G Cembrano, J Blanch, J Aguilar, D Saporta, G Benito, M Hedo, and A Molina. Validation and reconstruction of flow meter data in the barcelona water distribution network. *Control Engineering Practice*, 18(6):640–651, 2010.

[70] Lawrence Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[71] A. Rosich, R. Sarrate, V. Puig, and T. Escobet. Efficient optimal sensor placement for model-based fdi using an incremental algorithm. In *Decision and Control, 2007 46th IEEE Conference on*, pages 2590–2595. IEEE, 2007.

[72] Manuel Roveri and Francesco Trovò. An ensemble of hmms for cognitive fault detection in distributed sensor networks. In *Artificial Intelligence Applications and Innovations*. Springer, 2014 (to appear).

[73] R. Saunders and P. Laud. The mutltidimensional kolmogorov goodness-of-fit test. *Biometrika*, 67(1):237–237, 1980.

[74] B. Schrauwen, D. Verstraeten, and J. Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th European Symposium on Artificial Neural Networks*. Citeseer, 2007.

[75] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[76] M.Ó. Searcóid. *Metric spaces*. Springer Verlag, 2006.

[77] Manjeevan Seera, Chee Peng Lim, Dahaman Ishak, and Harapajan Singh. Fault detection and diagnosis of induction motors using motor current signature analysis and a hybrid fmm–cart model. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(1):97–108, 2012.

[78] N. Smirnov. Table for estimating the goodness of fit of empirical distributions. *The annals of mathematical statistics*, 19(2):279–281, 1948.

[79] Q. Song and N. Kasabov. Ecm-a novel on-line, evolving clustering method and its applications. In *Proceedings of the fifth biannual conference on artificial neural networks and expert systems (ANNES2001)*, pages 87–92. Citeseer, 2001.

[80] A.B. Trunov and M.M. Polycarpou. Automated fault diagnosis in nonlinear multivariable systems using a learning methodology. *Neural Networks, IEEE Transactions on*, 11(1):91–101, 2000.

[81] V. Venkatasubramanian, R. Rengaswamy, S.N. Kavuri, and K. Yin. A review of process fault detection and diagnosis:: Part iii: Process history based methods. *Computers & chemical engineering*, 27(3):327–346, 2003.

[82] J. Wichard and M. Ogorzalek. Time series prediction with ensemble models. In *Proceedings of International Joint Conference on Neural Networks (IJCNN 2004)*, pages 1625–1629. Citeseer, 2004.

[83] C.W. Wu, J.L. Chen, and J.H. Wang. Self-organizing mountain method for clustering. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 4, pages 2434–2438. IEEE, 2001.

[84] Rui Xu and Don Wunsch. *Clustering*, volume 10. Wiley-IEEE Press, 2008.

[85] R.R. Yager and D.P. Filev. Approximate clustering via the mountain method. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(8):1279–1284, 1994.

[86] B.S. Yang, T. Han, and Y.S. Kim. Integration of art-kohonen neural network and case-based reasoning for intelligent fault diagnosis. *Expert Systems with Applications*, 26(3):387–395, 2004.

[87] M.S. Yang and K.L. Wu. A similarity-based robust clustering method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(4):434–448, 2004.

[88] DL Yu, JB Gomm, and D Williams. Sensor fault diagnosis in a chemical process via rbf neural networks. *Control Engineering Practice*, 7(1):49–55, 1999.

[89] G.P. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[90] J Zhang and AJ Morris. On-line process fault diagnosis using fuzzy neural networks. *Intelligent systems engineering*, 3(1):37–47, 1994.

[91] Z.H. Zhou. *Ensemble methods: foundations and algorithms*. CRC Press, 2012.

[92] I. Žliobaitė. Combining similarity in time and space for training set formation under concept drift. *Intelligent Data Analysis*, 15(4):589–611, 2011.

Glossary

AIC Akaike Information Criterion. 94

ANFIS Adaptive Network-based Fuzzy Inference System. 14

ANN Artificial Neural Network. 9, 10

AP Affinity Propagation. 80, 121, 122, 126, 128, 131, 132

ARMAX Auto Regressive Moving Average eXogenous. 38

ART-KNN Adaptive Resonance Theory and Kohonen Neural Network. 14

ARX Auto Regressive eXogenous. 38, 94, 101, 109, 116, 122, 126, 128, 130, 131

BIC Bayesian Information Criterion. 122

BW Baum-Welch. 64, 65

BWNDS Barcelona Water Network Distribution System. 17, 100, 120, 131, 132

CDF Cumulative Distribution Function. 87

CDT Change Detection Test. 7, 73

CFDDS Cognitive Fault Detection and Diagnosis System. V, VI, 11, 16, 17, 20, 31, 33, 34, 35, 34, 35, 36, 40, 42, 43, 44, 45, 44, 47, 49, 52, 55, 56, 57, 64, 67, 71, 73, 75, 77, 79, 81, 83, 85, 86, 87, 88, 91, 93, 94, 95, 97, 108, 109, 114, 115, 116, 117, 120, 121, 122, 123, 126, 128, 129, 130, 131, 132, 133, 135, 136, 137, 139, 140, 141

DB DataBase. 29

DBSCAN Density-Based Spatial Clustering of Applications with Noise. 80, 121, 122, 123, 126, 128, 131, 132

ECM Evolving Cluster Method. 80, 121, 122, 123, 126, 128, 131, 132

EEG ElectroEncephaloGraphy. 49

EHMM-CDT Ensemble approach to Hidden Markov Model-Change Detection Test. 55, 64, 65, 67, 69, 71, 79, 108, 110, 113, 140

Glossary

ELM Extreme Learning Machine. 38

FDDS Fault Detection and Diagnosis System. 3, 11, 13, 14, 15, 23, 30, 48, 141

fMRI functional Magnetic Resonance Imaging. 49

GMM Gaussian Mixture Model. 61, 62, 101, 109

HMM Hidden Markov Model. 15, 16, 43, 44, 56, 60, 61, 62, 63, 64, 65, 67, 71, 76, 93, 94, 95, 101, 109, 116, 136, 137, 139

HMM-CDT Hidden Markov Model-Change Detection Test. VI, 15, 55, 60, 62, 63, 64, 65, 67, 69, 100, 101, 105, 108, 109, 110, 112, 113, 116, 120, 133, 136, 139, 141

KNN K-Nearest Neighbours. 10

KS Kolmogorov-Smirnov. 44, 87, 88, 123

LM Levenberg-Marquardt. 41

LS Least Square. 9, 50, 109

LTI Linear Time Invariant. 16, 35, 38, 40, 56, 109

MEMS Micro Electro-Mechanical Systems. 28

MIMO Multiple Input Multiple Output. 41, 69

MISO Multiple Input Single Output. 38, 41, 131

ML Machine Learning. 14

MM Mountain Method. 88

MSE Mean Square Error. 94

PCA Principal Component Analysis. 9, 87

PTPR Physical/Temporal Parity Relations. 115, 116, 117, 120, 133

RBF Radial Basis Function. 14

RN Reservoir Network. 38, 122, 126, 128

RP Random Projection. 87

SIMO Single Input Multiple Output. 41

SISO Single Input Single Output. 101, 109

UMTS Universal Mobile Telecommunications System. 99

VAR Vector AutoRegressive. 49