N° d'ordre : 41405

THESE

présentée et soutenue publiquement le 18 Avril 2014 en vue
d'obtenir le grade de

DOCTEUR

en

Spécialité : Automatique et Informatique Industrielle

par

Antoine Chammas

**DOCTORAT DELIVRE CONJOINTEMENT
PAR MINES DOUAI ET L'UNIVERSITE DE LILLE 1**

Titre de la thèse :

Drift Detection and Characterization for Supervision, Diagnosis and Prognosis of Dynamical Systems

Président	*Denis HAMAD, Professeur à l'Université du Littoral Côte d'Opale*
Rapporteur	*Nacer K. M'SIRDI, Professeur à Polytech Marseille*
Rapporteur	*Vicenç PUIG CAYUELA, Professeur à l'Universitat Politècnica de Catalunya (ESAII), Espagne.*
Examinateur	*Raphael Gouriveau – Professeur à l'Ecole Nationale Supérieure de Mécanique et des Microtechniques de Besançon*
Directeur de thèse	*Stéphane Lecoeuche, Professeur à l'Ecole des Mines de Douai*
Encadrant	*Eric Duviella, Maître-Assistant à l'Ecole des Mines de Douai*

Laboratoire(s) d'accueil : Unité de Recherche en Informatique Automatique, Mines Douai

Ecole Doctorale SPI 072 (Lille I, Lille III, Artois, ULCO, UVHC, Centrale Lille)

© 2014 Tous droits réservés.

Thèse de Antoine Chammas, Lille 1, 2014

© 2014 Tous droits réservés. doc.univ-lille1.fr

Thesis Acknowledgement

This thesis has benefited greatly from the support of many people, some of whom I would sincerely like to thank here.

To begin with, I would like to express my sincere gratitude to my thesis director Pr. **Stéphane Lecoeuche** for his patience, motivation, and immense knowledge. I would like to thank also my supervisor Dr. **Eric Duviella,** for his continuous support of my PhD. His guidance helped me in all the time of research and writing of this thesis. It was very pleasant to work with him.

Among those who played a major role in the work of this thesis, I would like to thank also Pr. **Moamar Sayed-Mouchaweh**, for transmitting his passion for research, his scientific help and guidance throughout my work.

I would like to thank my **friends** for their unequalled support. These last years with them were more than perfect. They made Douai look like Paris ☺. I also would like to thank all my **lab mates** for the nice time and discussions we used to have in the lab.

Last but not the least, a special appreciation and thanks to **my family, my parents** Dr. **Georges** and **Marcelle** Chammas**,** my sister **Aline** for encouraging me in all of my pursuits and inspiring me to follow my dream. Your prayers sustained me.

Finally, but first in my heart, a special thank you to my fiancée, **Cynthia Chidiac**. Words cannot describe how lucky I am to have her in my life. She was always there cheering me up and stood by me through the good and bad times. I always knew that you believed in me and wanted the best for me.

© 2014 Tous droits réservés.

Thèse de Antoine Chammas, Lille 1, 2014

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Contents

© 2014 Tous droits réservés. doc.univ-lille1.fr

© 2014 Tous droits réservés. doc.univ-lille1.fr

© 2014 Tous droits réservés. doc.univ-lille1.fr

© 2014 Tous droits réservés. doc.univ-lille1.fr

List of Figures

© 2014 Tous droits réservés.
doc.univ-lille1.fr

© 2014 Tous droits réservés. doc.univ-lille1.fr

© 2014 Tous droits réservés.
doc.univ-lille1.fr

Thèse de Antoine Chammas, Lille 1, 2014

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Thèse de Antoine Chammas, Lille 1, 2014

Introduction

The effectiveness of maintenance of industrial systems is an important economic issue for their commercial exploitation. A maintenance action consists in replacing or repairing system devices that are no longer able to perform their function. The main difficulties and inefficiencies reside in the choice of maintenance actions. In fact, maintenance operations are costly for two main reasons.

First, they often require a shutdown of the system. In this case, throughout the maintenance phase, the system is not operational. Thus, the longer the maintenance actions require, the more costly they are, due to the unavailability of the system. Therefore the maintenance phase should ideally be reduced to operations to replace (or repair), without fumbling, the equipments that have actually failed. The decision of a maintenance action is very complex and should be based on monitoring and intelligent analysis of the state of the system. Fault diagnosis is necessary to determine as precisely as possible the equipment that must be repaired. The more accurate the diagnosis, the more effective the maintenance actions are.

The second reason why maintenance can be costly is in emergency situations in which the safety and/or the performance of the functioning of the system are concerned. In fact, when equipment suddenly breaks down and the system can no longer perform its function, maintenance actions should be automatically done to keep the system working. These unexpected actions are naturally more expensive because they were not anticipated. To minimize the occurrence of this kind of situation, preventive maintenance can be considered. The failure of equipment can be anticipated and corrected before generating excessive damage that could cause an unexpected system shutdown.

One recent trend among the preventive maintenance strategies is the Condition-Based Maintenance (CBM). The strong point about CBM is that it incorporates a prognosis module. In fact, when correctly connected to a diagnosis module, a prognosis module can provide a valuable solution to minimize abnormal and unexpected situations. The prognosis model gives an estimate of the Remaining Useful Life (RUL) of a physical asset. This helps improving the planning for maintenance actions.

Thus the prognosis has become more and more attractive and dealing with it has become a relatively high research trend. Prognosis has a sense when the failure modes initiating faults correspond to drifts, which are slowly evolving faults (also called incipient faults). In addition, prognosis is related to a monitoring and a diagnosis module. The monitoring and the diagnosis modules are responsible

© 2014 Tous droits réservés. doc.univ-lille1.fr

for the detection and the isolation of a fault. This information is needed in order to know which is the failure mode that initiated the fault. Also, a signal indicating the degradation state is obviously needed. Only then, by using all these information, a prognosis result can be achieved.

In the work presented in this thesis, an architecture for supervision, diagnosis and prognosis is developed. The supervision consists in the monitoring phase of the controlled industrial system. The diagnosis and the prognosis consist in apprehending drifts and estimating a RUL. The architecture is based on the analysis and the exploitation of the data measured from a system. Thus, it is a data-driven approach, making it more suitable for complex systems, which cannot be described by a mathematical model. The architecture to be built is based on the study of drifts from a classification/clustering point of view.

Drifts are unexpected events that affect a system, leading it from a normal operating mode, to a failure operating mode, through a degraded operating mode. They are intrinsic changes in the property of the system, which make it evolve and change its dynamics. Thus, by studying the data that are generated by a drifting system, it can be considered that these data are generated by a non-stationary environment. The non-stationarity is a direct consequence of the process drift. Thus, in order to achieve having prognosis results, a thorough study of non-stationary environments should be made.
The research work that is presented in this thesis connects the study of prognosis modeling to the one of non-stationary environments. In order to clearly define the aim of the work of this thesis, let us define research challenges, when it comes to systems subject to drifts and on which a prognosis model should be made:

1. A drift affecting a process must be detected as fast as possible. The properties of the alarm system are its fast detection time and its low false alarm rate. It is equivalent to detecting anomalies or deviations from a normal behavior.

2. How to assess the health of the process?
   This question includes diagnosis challenges. A drift is initiated by a specific failure mode. If a drift has occurred, the question is if we are able to indicate which failure mode is behind it. In addition, if this failure mode is found, how far is the current operating mode is from reaching it?

3. Given a drift scenario towards an identified failure mode, and given the actual drift dynamics, how much time left before this failure mode is reached? How is the prognosis made?
   The prognosis model must give an estimate of the RUL. In addition to giving a RUL estimate, it is necessary to provide also a certain confidence interval, which is needed for a better maintenance planning. A big challenge concerning prognosis modeling is the need for a reactive model that always takes into consideration current operating conditions. This means that the prognosis model that needs to be defined is an online prognosis model. The RUL estimation should be based on the actual dynamics of the drift. Thus, the knowledge of these dynamics should be updated in an online manner.

© 2014 Tous droits réservés. doc.univ-lille1.fr

4. Another big challenge, which is gaining more and more importance recently, concerns unknown operating modes. So the question is: what happens if a drift is actually happening but the failure mode behind this drift is not identified? The fault in this case is not isolated. In many real applications, historical data do not cover all failure modes. There are possibly few unknown failure modes that must be detected and labeled as fast as possible. This challenge is thus related with unknown failure modes.

These research challenges are very interesting to be treated since little attention has been given to systems operating with drifts (more attention was attributed to the study of systems subject to abrupt faults). In this work, these challenges are going to be tackled. A global architecture combining several tools that answer each of these challenges is developed. This architecture is also adaptable to the general structure of the condition-based maintenance strategy.

The work developed in this thesis aims to develop this global architecture. The manuscript follows a logical chain of ideas that will lead, to the proposition of this global architecture. The manuscript is organized as follows:

1. **Chapter 1**: this chapter introduces the condition-based maintenance strategy and highlights its importance. Then, a thorough study of prognosis techniques is given. The chapter's last section shows also metrics for evaluation of a prognosis model. It is concluded by the need to develop new prognosis models that are online, allowing better reactivity to changing operating conditions. This need is related to the need of handling non-stationary environments.
2. **Chapter 2**: this chapter is concerned about the study of drifts. The definition, types and nature of drifts are detailed. Then, a review of algorithms that handle drifts is given. A synthesis of this review has allowed elaborating a classification scheme, which gives a sort of an identity card to any algorithm that handles drift. The study presented in this chapter concluded in finding an adequate algorithm, AuDyC, to be used for our purposes. It also showed the need to develop modules for drift detection, characterization and prognosis, that will be later used to construct the global architecture.
3. **Chapter 3**: this chapter begins by formally relating the problem of process drift to the one of concept drift. The algorithm AuDyC is then used and different modules, answering different challenge questions, are developed and incorporated to AuDyC. These modules are the drift detection, drift characterization and prognosis modules. The drift detection module is the one to alert the system of the presence of a change in the current functioning. The drift characterization module is responsible for giving meaningful indicators that help to isolate the failure mode behind the drift, and to provide an assessment of the health state of the system. The prognosis module, being related to these two former modules, provides an estimate of the Remaining Useful Life and associates a confidence interval to it.
4. **Chapter 4**: this chapter combines all the modules developped in chapter 3 with the aim of generating a global architecture. Thus, this chapter comes as a final contribution to all the research questions defined above. The global architecture combines the modules of drift

© 2014 Tous droits réservés. doc.univ-lille1.fr

detection, characterization and prognosis together. Its aim is to provide diagnosis and prognosis results, once it is plugged into a real working system. The chapter contains also two case studies which are used to validate the developed architecture.

Finally, the thesis is concluded with a brief description of the developed architecture. Later on, a section inside the conclusion is dedicated to the discussion and perspective points related to the work of the thesis.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Chapter 1

Prognosis: a crucial step for condition-based maintenance

1.1 Introduction

In the context of global economic competitiveness, a major issue is to optimize the overall performance of companies, particularly in key sectors (military, aeronautical, energy, etc.). To do this, there are new strategies designed to increase the availability of the dynamical systems considered (production process, machinery, etc.). The main objective is to minimize costs and ensure the safety of persons, goods and equipments. To ensure this goal, supervision and condition monitoring of systems including prognosis functions are new emerging technologies that are more and more implemented on real modern systems. These prognosis functions provide predictions of the Remaining Useful Life (RUL) before failure occurrence.

To ensure efficiency, these predictions need to be associated with performance evaluation indicators in order to trust them. Predictions and performance indicators could be the entrance of a decision process, leading to maintenance decisions. In addition, before being able to achieve prognosis results, a lot of steps should be overtaken. These steps include the treatement of data, the condition monitoring and the diagnosis. All these different modules have been grouped together to form a global architecture. This architecture defines a global strategy scheme for maintenance. It is called the Condition-Based Maintenance (CBM) and it will be studied in this chapter.

In this chapter, the focus is made on the prognosis module. The aim of this chapter is to place the prognosis module inside a global CBM architecture. The importance of prognosis for CBM is shown through a thorough study of prognosis methodologies. Lastly, metrics to evaluate prognosis results are shown.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Thèse de Antoine Chammas, Lille 1, 2014

1.2 Maintenance strategies - Condition Based Maintenance

The concept of global performance of a system was introduced in (Senechal, 2004) as the joint procurement of relevance (adequacy of resources and objectives), efficiency (adequacy of resources and results) and effectiveness (adequacy of results and objectives), assessed in terms of values on the entire life cycle of a system. In general, the global performance of a system is required from an economic point of view, by optimizing the operating cost and insuring the security of the personnel and the goods. The optimization of the operating cost is done on the overall life cycle of the systems.

Maintenance activities are a set of actions required or undertaken to conserve as nearly, and as long as possible the original condition of a system, and to maximize its global performance. Maintenance strategies consist in combining different tools and techniques in order to reach these aims. These strategies have evolved through time and are always improving thanks to the increasing technologies. In this section, we will introduce this time evolution in order to show the latest challenges in the context of maintenance technologies.

1.2.1 Evolution of maintenance strategies

In Fig. (1.1), the evolution of maintenance strategies is shown through the time period covering the last few decades. Earliest maintenance techniques are called **reactive**. They are breakdown or run-to-failure maintenance in which actions were taken after the breakdown of the machine. Reactive maintenance involves large amount of downtime and high cost of replacement and repair.

To solve these issues, **proactive** maintenance has been developed. Preventive maintenance is a form of proactive maintenance in which inspection is done systematically or on regular time intervals. These intervals are optimized in order to reduce maintenance costs. It is a static maintenance since no real-time Condition Monitoring (CM) data is required on the system. However, it can be argued that few supervision data can be demanded as for working hours, number of operating cycles, etc. The decision of the inspection times is predefined according to experience and an offline estimation of lifetime.

Preventive maintenance is effectively better than run-to-failure maintenance but still has few disadvantages:

- Perform unnecessary maintenance actions in some cases. This also means costly procedures that can be avoided.
- Do not prevent catastrophic failures.

Another form of proactive maintenance, the predictive maintenance, has been introduced to tackle these problems. Few years ago, for preventive maintenance, a unified architecture, the Condition-Based Maintenance (CBM) has been proposed (Lebold and Thurston, 2001). The CBM architecture offers the capability to mitigate many of the problems cited above. It does so by defining a structure gathering several modules. The CBM architecture is presented in the next subsection.

© 2014 Tous droits réservés. doc.univ-lille1.fr

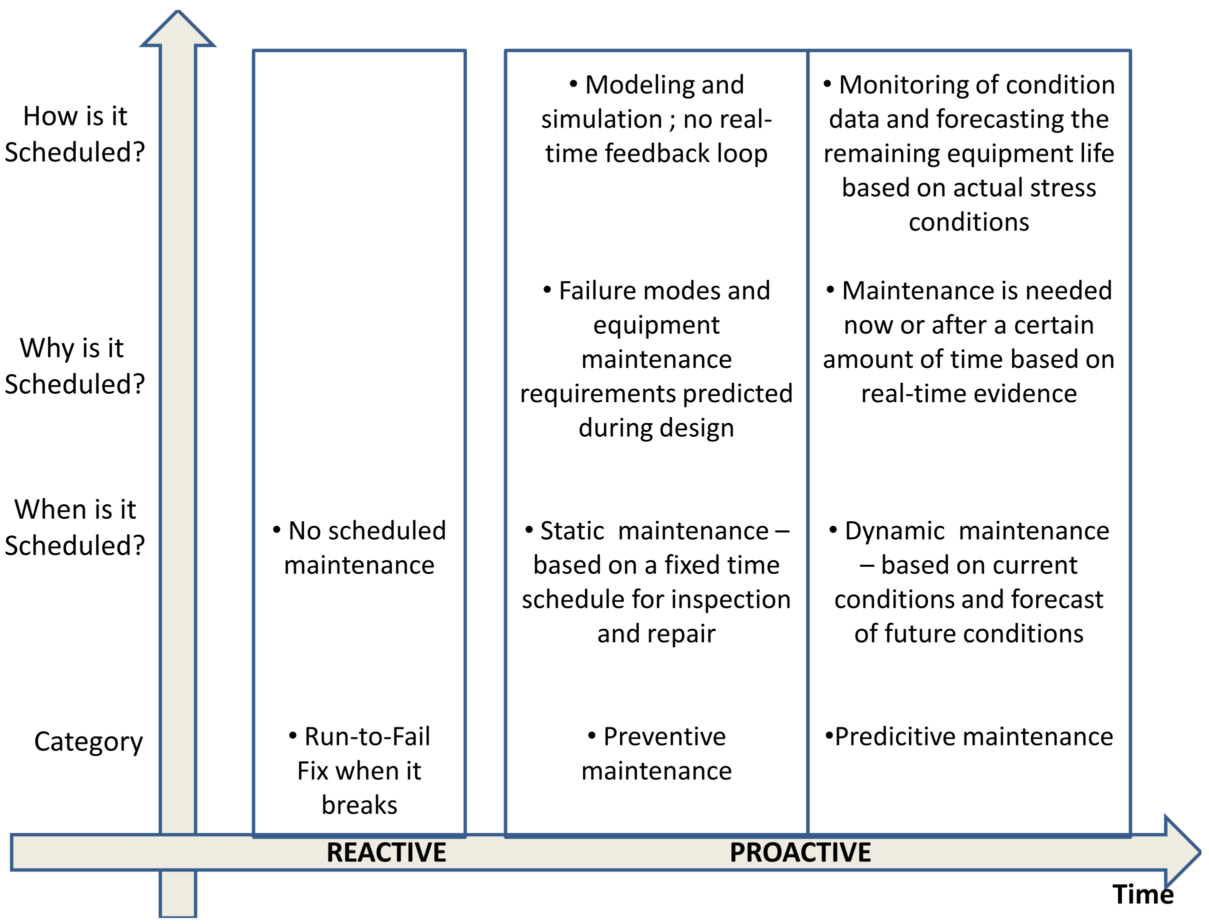

How is it Scheduled?
Why is it Scheduled?
When is it Scheduled?
Category
• No scheduled maintenance
• Run-to-Fail Fix when it breaks
• Modeling and simulation ; no real-time feedback loop
• Failure modes and equipment maintenance requirements predicted during design
• Static maintenance – based on a fixed time schedule for inspection and repair
• Preventive maintenance
• Monitoring of condition data and forecasting the remaining equipment life based on actual stress conditions
• Maintenance is needed now or after a certain amount of time based on real-time evidence
• Dynamic maintenance – based on current conditions and forecast of future conditions
•Predicitive maintenance
REACTIVE
PROACTIVE
Time

Figure 1.1: Evolution of maintenance strategies over time.

1.2.2 CBM architecture

As defined in the terminology NF-EN 13306 (2001), the CBM is:

"Preventive maintenance based on performance and/or parameter monitoring and the subsequent actions."

Fig. (1.2) shows the way how the CBM modules interact with each other forming a complete integrated system. At the center of this wheel structure lies the communications medium between the modules. It may be accomplished using popular communication and middleware technologies. Therefore, the different modules do not need to be executed in the same place but may be on a local or worldwide network. The CBM architecture design enables the integration of prognosis capability, thus allowing maximum flexibility (dynamic maintenance scheduling) of the system. Additional information about the CBM standard development and architecture may be found in the following references (Lebold and Thurston, 2001; Leobold et al., 2003; Lebold et al., 2002). The different module of the CBM architecture are explained in the next subsections. Afterwards, the prognosis module, which is the main subject of this chapter, will be studied.

© 2014 Tous droits réservés. doc.univ-lille1.fr

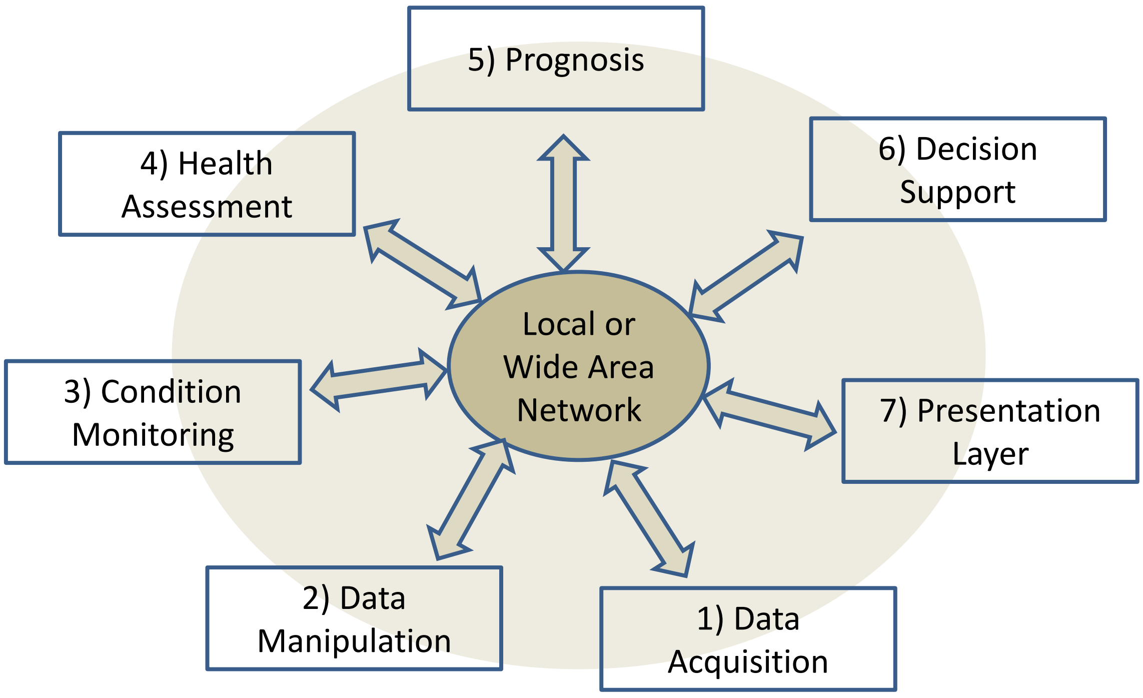


Figure 1.2: OSA-CBM architecture (Lebold and Thurston, 2001).

1.2.2.1 Data Acquisition

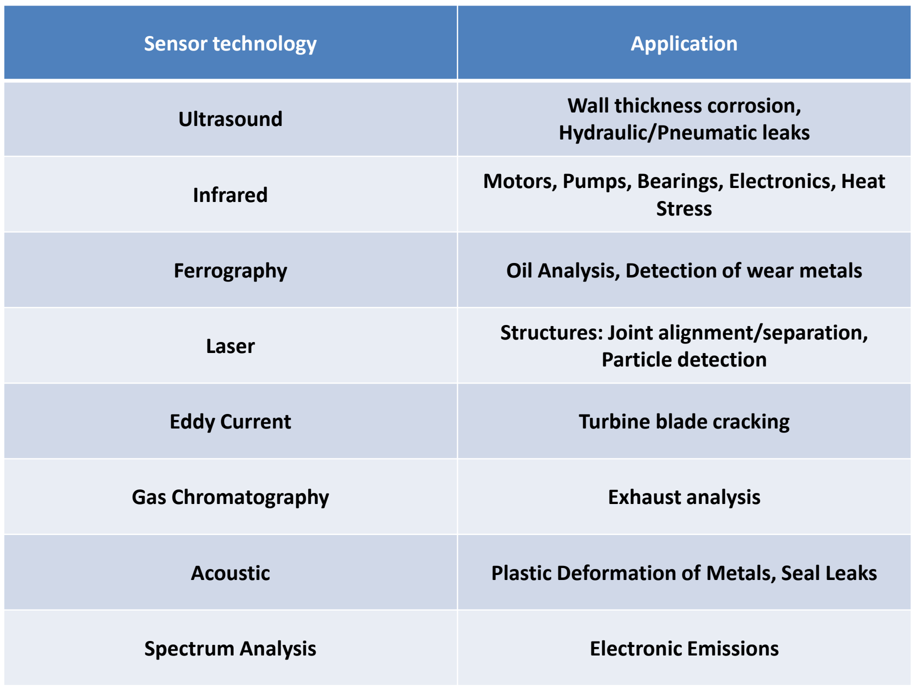
Data acquisition is an important step that consists in collecting useful data on a system. Without these data, it will be impossible to implement a CBM program. The data acquired from a system can have two types:

- **Event Data:** These data contain information on the actions that were made on a system and, if possible, the causes behind these actions. Examples of actions that are made on a system include maintenance actions, minor repairs, oil change, etc. Examples of causes can be breakdown, overhaul, etc.
- **Condition Monitoring (CM) Data:** CM data are raw versatile data coming from various sensors. The increasingly efficient sensor technologies help building highly informative sensor networks. Examples of such data are vibration data, acoustic data, oil analysis data, temperature, pressure, moisture, humidity, meteorological data, etc. These data are usually collected and saved in massive amounts. An example of a set of different sensors and their applications is given in Figure (1.3).

1.2.2.2 Data Manipulation

This step is also known as **Data Processing**. It tackles a major challenge for a CBM program which is the extraction of useful information from massive raw databases. Data manipulation involves two major steps:

© 2014 Tous droits réservés. doc.univ-lille1.fr



Sensor technology	Application
Ultrasound	Wall thickness corrosion, Hydraulic/Pneumatic leaks
Infrared	Motors, Pumps, Bearings, Electronics, Heat Stress
Ferrography	Oil Analysis, Detection of wear metals
Laser	Structures: Joint alignment/separation, Particle detection
Eddy Current	Turbine blade cracking
Gas Chromatography	Exhaust analysis
Acoustic	Plastic Deformation of Metals, Seal Leaks
Spectrum Analysis	Electronic Emissions

Figure 1.3: Different sensors and their applications.

- The first is usually referred to as **pre-processing**. Its goal is to make the data easier to exploit, and to improve the signal-to-noise ratio. It includes filtering, amplification, denoising, removing of outliers, etc.
- The second step is feature extraction. It aims to extract meaningful features that are characteristic of the different operating modes of the system. This step is essential for building a discriminative model for diagnosis/prognosis purposes. For instance, for a good fault detection and isolation, these features must be sufficiently insensitive to noise, and highly discriminative of the fault class variations. A set of qualities can be attributed to 'good' features as:
  1. Computationally low cost.
  2. Mathematically definable and physically explainable.
  3. Characterized by large interclass mean distance and small interclass variance.
  4. Insensitive to extraneous variables.
  5. Uncorrelated with other features.

In general, data manipulation is done using tools that are used for analysis purposes. These tools can be in the *Time-domain*, *Frequency-domain* and *Time-Frequency domain*:
- Time-domain analysis:
  Traditional time domain analysis involves extracting statistical features directly from the data. These statistical features could be the mean, peak, peak-to-peak interval, standard deviation, crest factor, root-mean square, skewness, kurtosis, etc.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Another popular time-domain analysis is the Time-Synchronous-Average (TSA). It is essentially used with vibration signals with rotating equipment (Dalpiaz et al., 2000). Its idea is to use the mean or average of a raw signal over numerous revolutions. This enhances the signal's components that are supposed interesting. A brief review on TSA is given in (Dalpiaz et al., 2000). In (Miller, 1999), some drawbacks of TSA were pointed out.

Other advanced approaches for time-domain analysis are the time series regression modeling. Examples of the most popular models include Auto-Regressive (AR) models and Auto-Regressive with Moving-Average (ARMA) models. The features in this case are the parameters of these models (Jover and Hyotyniemi, 2004; Baillie and Mathew, 1996; Garga et al., 1997).

– Frequency-domain analysis: the frequency-domain analysis consists in transforming a signal from the time domain to its frequency domain. The main advantage of this transformation is to isolate frequency components of a signal that could be sensitive to certain process-specific inner mechanisms. The conventional transformation used is the Fast Fourier Transform (FFT). FFT allows us to obtain the spectrum of a signal that can be used for further analysis (for instance the power spectrum) (Schoen and Habetler, 1995; Almeida et al., 2002; Liu et al., 2004).

– Time-frequency analysis: a major advantage of the time-frequency analysis over the frequency analysis is its ability to handle non-stationary data. In fact, when fault occurs, its evolution can show non-stationarity in both time and frequency domains. Traditional time-frequency analysis represents the energy or the power of a signal in functions of two dimensions (time and frequency). This is called the time-frequency distributions and it is done STFT (Short Time Fourier Transform) (Andrade et al., 1999), Wigner-Ville distribution (Baydar and Ball, 2001), wavelet transform (Li et al., 2010). The latter has also an advantage of reducing noise in raw signals (Tonshoff et al., 2003).

1.2.2.3 Condition monitoring

The primary function of the condition monitoring is to compare data values against expected values or operational limits and output enumerated condition indicators (e.g. level low, level normal, level high, etc). These data values can be the features that were extracted from raw measurements, or condition indicators that are extracted from these features. In the former case, the CM data is referred to as 'Direct' (DCM) and in the latter, the CM data is referred to as 'Indirect' (ICM). In both cases, thresholds and alarms are set according to these parameters to indicate a change from normal behavior.

Different condition monitoring applications exist. The main most common categories are:

1. Vibration monitoring.
2. Thermography.

© 2014 Tous droits réservés. doc.univ-lille1.fr

3. Tribology.
4. Acoustic analysis.
5. Motor analysis techniques.
6. Process parameter monitoring.
7. Visual inspections.

For additional details about these application, the set of tools and techniques for condition monitoring can be found in (Mehala, 2010). The CM module is responsible for alarming the system about a fault occurrence or for a beginning of degradation in a component or a sub/system. If an alarm signal is given by the CM module, then the health assessment module is activated to investigate further about this alarm.

1.2.2.4 Health assessment

Health assessment is a critical step that includes fault diagnosis. Methods for fault diagnosis can be roughly divided into two major categories: **internal** approaches and **external** approaches. Internal approaches are also called model-based approach since they require the knowledge of the physical model of the system. These approaches are commonly referred to analytical modeling based approaches, physical modeling based approaches, observer based approaches, etc. Alternatively, external methods do not require this detailed physical knowledge. They are based on external measurements made on a system (Venkatasubramanian et al., 2003c,a,b). Examples of such approaches are parameter estimation techniques, state estimation techniques, and pattern recognition techniques (classification, clustering, regression).

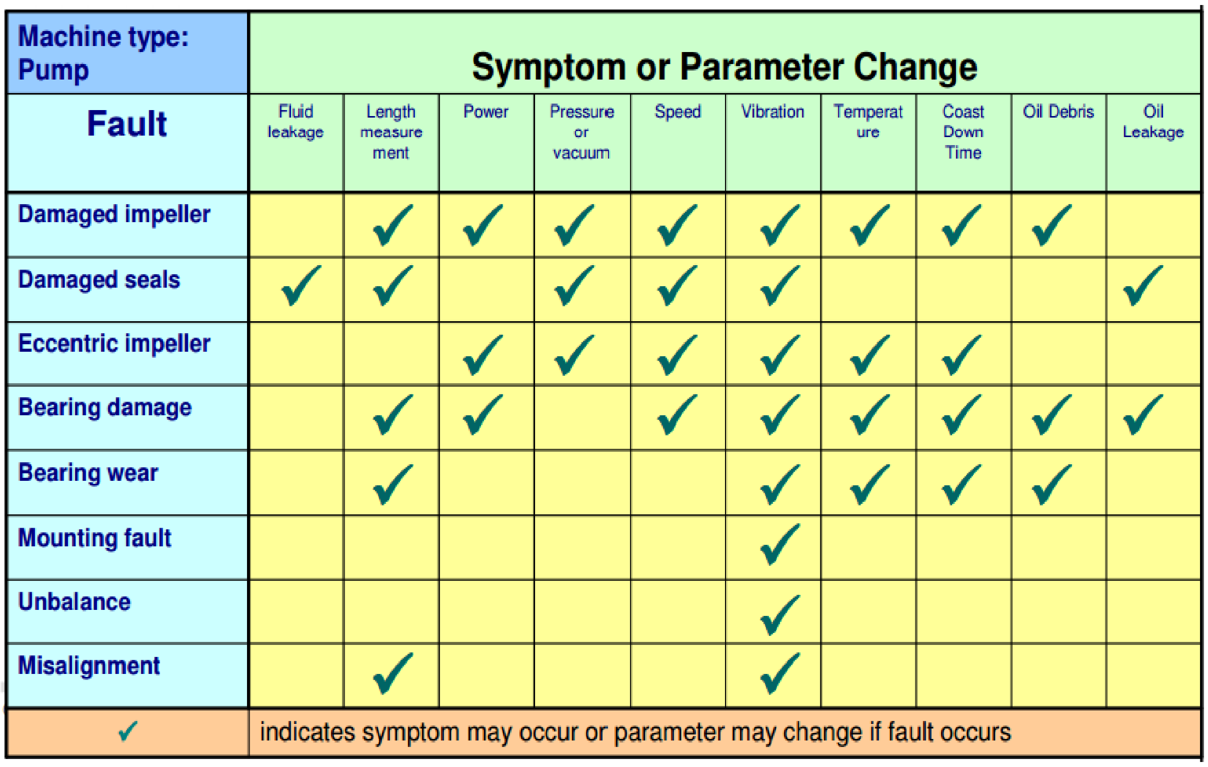
Fault detection is evaluated according to two criteria:
– Detection time: the challenge is to detect the fault as soon as it appears.
– False alarms: The detection should be accurate and not subject to false alarms. It is worth reminding that there are two types of false alarms: false positive alarm (not detecting an existing fault) and false negative (detecting a non-existing fault).

Upon detection, a hard task consists in isolating the faulty component by determining the failure mode behind the fault. This task depends mainly on the choice of the monitored symptoms. A table relating different fault symptoms to different failure modes is necessary. An example of such tables in the case of a pump is given in Fig. (1.4). These tables can be established by executing more formal symptom-failure analysis, as the FMECA (Failure mode, effect and criticality analysis) and the Dysfunctional analysis.

1.2.2.5 Prognosis

Prognosis derives from the greek words *pro* and *gnosis*: *pro* means anterior event and *gnosis* means to acquire knowledge. It is a key aspect in a CBM program because it allows the anticipation

© 2014 Tous droits réservés. doc.univ-lille1.fr



Machine type: Pump	Symptom or Parameter Change									
Fault	Fluid leakage	Length measure ment	Power	Pressure or vacuum	Speed	Vibration	Temperat ure	Coast Down Time	Oil Debris	Oil Leakage
Damaged impeller		✓	✓	✓	✓	✓	✓	✓	✓	
Damaged seals	✓	✓		✓	✓	✓				✓
Eccentric impeller			✓	✓	✓	✓	✓	✓		
Bearing damage		✓	✓		✓	✓	✓	✓	✓	✓
Bearing wear		✓				✓	✓	✓	✓	
Mounting fault						✓				
Unbalance						✓				
Misalignment		✓				✓				
✓	indicates symptom may occur or parameter may change if fault occurs									

Figure 1.4: Example on a pump of different failure modes associated to different symptoms.

of a failure before its occurrence. The output of a prognosis module is an estimation of the remaining time before failure, and, if possible, an associated confidence interval. This is needed for planning successful maintenance missions, reduce maintenance cost and down-time (Nima et al., 2009). The increasing importance of prognosis is reflected by the increasing number of review papers dedicated to the subject (Si et al., 2011; Jardine et al., 2006; Heng et al., 2009; Nima et al., 2009; Dragomir et al., 2009; Sikorska et al., 2011; Pandian and Ali, 2010).

This chapter is dedicated to the study of the prognosis process. In the coming sections, the prognosis process, the relationship between diagnosis and prognosis and the state-of-the-art of prognosis approaches are given.

1.2.2.6 Automatic decision reasoning

The main function of the decision support module is to provide recommendations that are related to maintenance action schedules as well as modification of the equipment configuration or mission profiles in order to accomplish mission objectives. The decision support module needs to take into account some criteria as resource constraints, maintenance history, current and future mission profiles, etc. These criteria can be found in (Aoufir and Bouami, 2003) and (Muller, 2005). Other works also offer modeling and evaluating maintenance strategies and tools (Zille, 2009; Ribot, 2009).

© 2014 Tous droits réservés. doc.univ-lille1.fr

1.2.2.7 Presentation layer

This module represents the human interface. Alerts, health assessments, prognosis assessments, and decision support recommendations are displayed at this layer with the ability to drill down to multiple layers of access depending on the information needs of the user. The actual decisions are taken at this level.

1.2.2.8 Research orientation

The CBM architecture has attracted the attention of the automatic control community since more than a decade. Thus, a certain scientific maturity has been reached in some of its modules. For instance, sensor technologies are improving each year and thus Condition Monitoring is becoming less and less expensive. Morover, fault diagnosis has also acquired a certain advancement in industrial applications, as it is well deployed now. However, as stated in (Sikorska et al., 2011), the prognosis has not been regularly adopted in industrial applications yet. Some of the reasons for that are:

1. High costs: prognosis systems require a highly technological software/hardware development, which a lot of companies still avoid.
2. Availability of failure data: as we will see in the rest of the chapter, a lot of prognosis models need a big amount of failures and degradation data in order to learn the degradation behavior. These data are not always easy to obtain in sufficient quantities.
3. Reactivity of prognosis models: a lot of prognosis models are built in an offline manner under certain operating conditions. These models lack reactivity to varying operating conditions. A more reactive prognosis model has the ability to acquire new degradation data and adapt its parameters accordingly.
4. Verification and validation (V&V): the V&V process consists in verifing if the prognosis algorithms meet the required specifications or conditions imposed at the start of a development phase. There are not usually enoug data to correctly validate them. The lack of measured data could be compensated by creating simulated data to validate the indicators of health of a system and the prognosis models built for it (Volov, 2014; Dzakowic, 2007).

Consequently, the need to develop more the prognosis module is obvious. A robust prognosis module, coupled to an automatic decision reasoning module, can help improve the CBM goals. The improvement on the prognosis module in this work concerns mainly the reactivity issue, the availability of failure data and the numerical simulation for verification and validation. The orientation of research is towards defining a prognosis algorithm that can adapt its parameters when new degradation data are available, increasing thus the reactivity of a model.

In the rest of the chapter, a state-of-the-art on the prognosis module is given. At the beginning, few prognosis definitions are proposed and one is retained in order to clarify the vocabulary that is always used. Then, a taxonomy of the prognosis techniques is presented. Later, a set of metrics used in the literature is shown. The chapter is ended with a more concluding research orientation, which will set the ground for the thesis work development presented in the third chapter.

© 2014 Tous droits réservés. doc.univ-lille1.fr

1.3 Prognosis: definition and positioning

In this section, the definition of prognosis is given. It shows different possible definitions for prognosis. It highlights the relationship between diagnosis and prognosis. The difference between optimistic and pessimistic RUL predictions is established. Finally, after detailing the basic steps before obtaining a RUL estimate, a classification of prognosis methodologies is given.

1.3.1 Prognosis definition

Several definitions for the prognosis have been proposed in the literature. Few definitions are retained and listed below:

– The norm ISO (13381-1, 2004) defined the prognosis as:
"*An estimation of time to failure and risk for one or more existing and future failure modes.*"

– In (Engle, 1986), the authors defined prognosis as:
"*The capability to provide early detecting of the precursor and/or incipient fault condition of a component, and to have the technology and means to manage and predict the progression of this fault condition to component failure.*"

– In (Hess et al., 2005), the authors defined prognosis as:
"*Predictive diagnosis, which includes determining the remaining useful life*"

From these definitions, along with others (Baruah and Chinnam, 2005; Heng et al., 2009), it can be seen that:

1. Prognosis is about predicting the remaining time for a gradual fault, which is specific to a failure mode, to pass from incipience to failure of the component. this remaining time is the Remaining Useful life, or RUL.
2. The prognosis should be performed at a component or at a subsystem level.

When it comes to provide a RUL, it is not only a mean value of the Estimated Time To Failure (ETTF) that should be provided. In fact, this value should also be associated to a confidence interval. The limits that are provided in this interval are called the optimistic and the pessimistic limits. The reason that this confidence interval should be provided is that the estimation of a RUL is affected by uncertainty. This uncertainty is attributed to the deterioration process, the measurement noise and errors and the ambiguity concerning the future operation of the machine. Thus, it is necessary for any model that provides a RUL to also take into consideration these uncertainties.

1.3.2 Prognosis positioning

There are several steps to be accomplished before being able to obtain a prognosis result, *i.e.* a RUL and a confidence interval.

© 2014 Tous droits réservés. doc.univ-lille1.fr

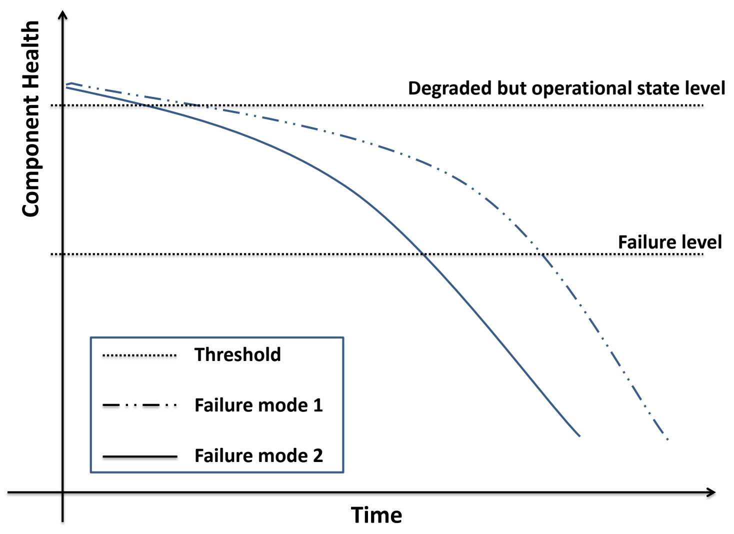


Figure 1.5: Examples of two different degradation curves of components initiated by two different failure modes.

Considering a component with degradation mechanisms represented in Fig. (1.5). Each one of these curves represents a degradation mechanism which is initiated by a different type of failure. For example, considering the rolling element bearing, different degradations are possible: cage crack, inner race spall, outer race spall. Each of these failures induces a degradation mechanism with different dynamics.

In order to compute a RUL on a component level, it is necessary to answer these basic questions:

1. Is there any degradation affecting a component?
2. What is the component that is starting to degrade?
3. What is the failure mode behind this damage initiation?
4. What is the severity of the damage? Where is it on the degradation curve?
5. How is the damage evolving? Is there a specific dynamical system that governs this evolution?
6. How does the environment affect the evolution of damage? Is it possible to take these environmental changes into consideration?
7. Once a RUL value is estimated, what is the confidence interval associated to it?

The first three questions can be related to the health assessment (diagnosis) module. Te three next questions are concerned with the severity of drift, and the dynamics of its evolution. The last question concerns the confidence that can be associated to the estimated value of RUL. This confirms the solid relationship that exists between the diagnosis and prognosis modules. This relationship is graphically illustrated in Fig. (1.6).

In the next section, a classification of prognosis approaches that are present in the literature is given.

© 2014 Tous droits réservés.

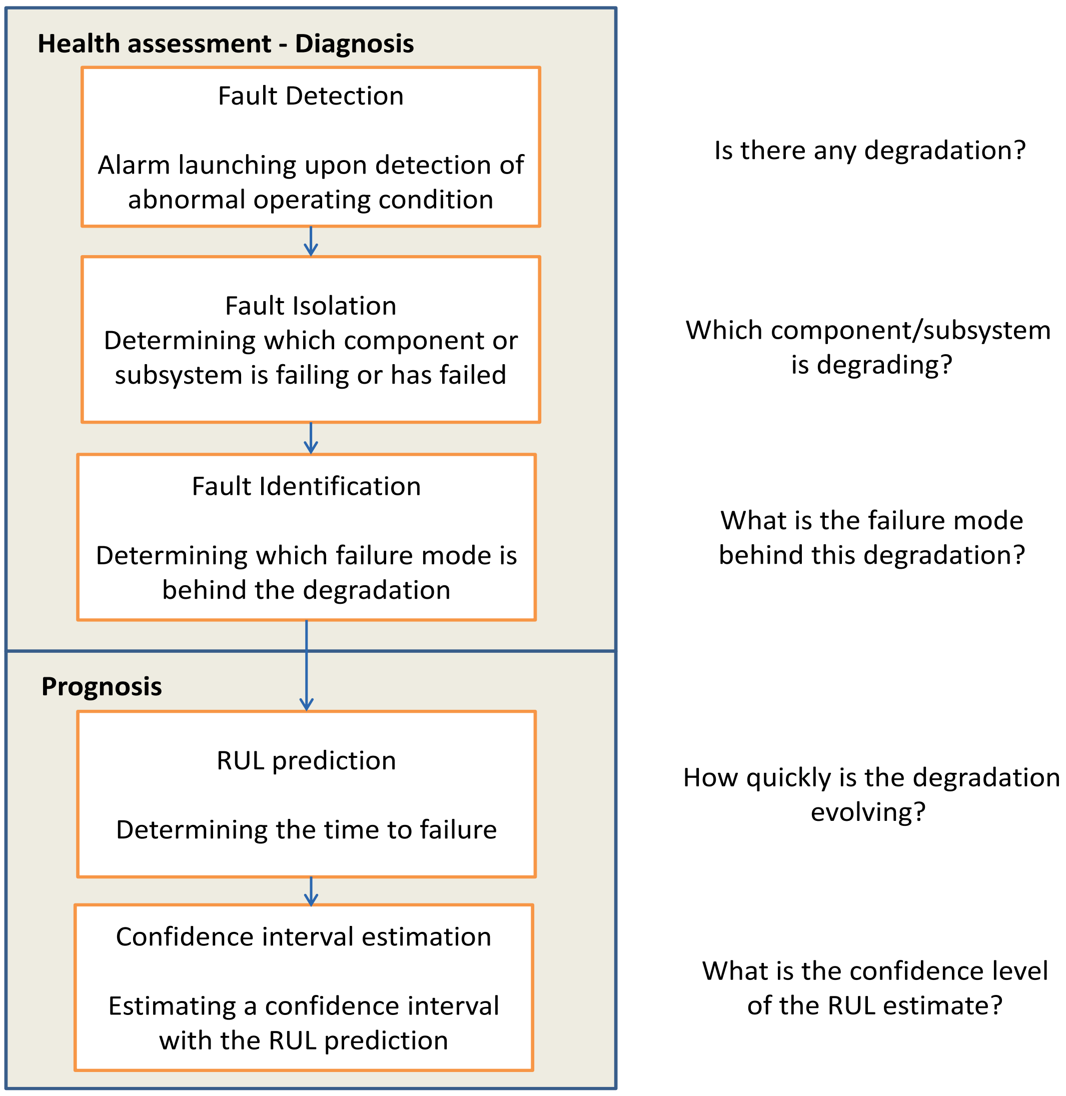

Health assessment - Diagnosis
Fault Detection
Alarm launching upon detection of abnormal operating condition
Fault Isolation
Determining which component or subsystem is failing or has failed
Fault Identification
Determining which failure mode is behind the degradation
Prognosis
RUL prediction
Determining the time to failure
Confidence interval estimation
Estimating a confidence interval with the RUL prediction
Is there any degradation?
Which component/subsystem is degrading?
What is the failure mode behind this degradation?
How quickly is the degradation evolving?
What is the confidence level of the RUL estimate?

Figure 1.6: Substantial relation between Diagnosis and Prognosis (Sikorska et al., 2011).

1.4 Prognosis: taxonomy of prognosis models

Several prognosis models have been developed in the literature. These models depend on the industrial process application, the data requirements and complexity. For this reason, some models are more appropriate in a situation than another. Thus, before being able to implement a prognosis model, let us remind the important requirements of these models.

Few questions must be answered before implementing a prognosis model. These questions are:

1. What is awaited from the prediction given by the prognosis model?
   Possible answers for this question is the precision or the accuracy of the forecasted RUL and the confidence interval it should provide.

© 2014 Tous droits réservés.

2. What are the data resources available to implement a prognosis model?
   The available resources can be the amount of data, its type (CM and/or Event data) and its quality. In addition, another possibly valuable resource is the expertise associated with these data. This includes expert personnel or skilled people in the domain. The latter can provide feedback on materials, statistical and mathematical expertise, etc. Finally, it is important to take into consideration the hardware and software resources available to implement a prognosis model.

3. Is the developed model compatible?
   Given a set of assumptions and specifications of design, the prognosis model must be compatible with the dynamics of degradation in real life. It should take into consideration the complexity of the system and the noise affecting the measurements.

Based on our review, the approaches for prognosis can be represented by a tree as in Fig. (1.7). The different methodologies inside this tree are explained below. It is important to note that categories of approaches are treated individually in this work. However, there exist hybrid methods that combine several approaches from different families in the same time. For more information on hybrid prognosis models, the reader can refer to (Celaya et al., 2011; Jouin et al., 2014; An et al., 2013).

1.4.1 Physics of failure based approaches

In these approaches, a physical model describing the degradation of the component is obtainable. Under this assumption, a comprehensive model of the degradation is given mathematically, *i.e.* equations describing how failure modes are related to the coupled systems are used. The big advantage of these types of models is that they incorporate physical and behavioral knowledge. Thus, they are very accurate compared to any other model.

Some of the well-known physics of failure models are:

– Fatigue crack growth (Lugtigheid et al., 2007).
– Load strength models (Todinov, 2005).
– Corrosion (Todinov, 2005).
– Crack life in metallic materials (Ray and Tangirala, 1996). In (Heng et al., 2009), an exhaustive list of deterioration models is given.

1.4.1.1 Example of physics of failure model: damage modeling of fatigue crack growth

Fatigue crack growth affects typical machinery components as bearings, gears, shafts, aircraft wings, etc (Vachtsevanos et al., 2006; Pugnoa et al., 2006). It is affected by various environmental factors as material properties, temperature, lubrication, etc. The notations for fatigue crack growth

© 2014 Tous droits réservés. doc.univ-lille1.fr

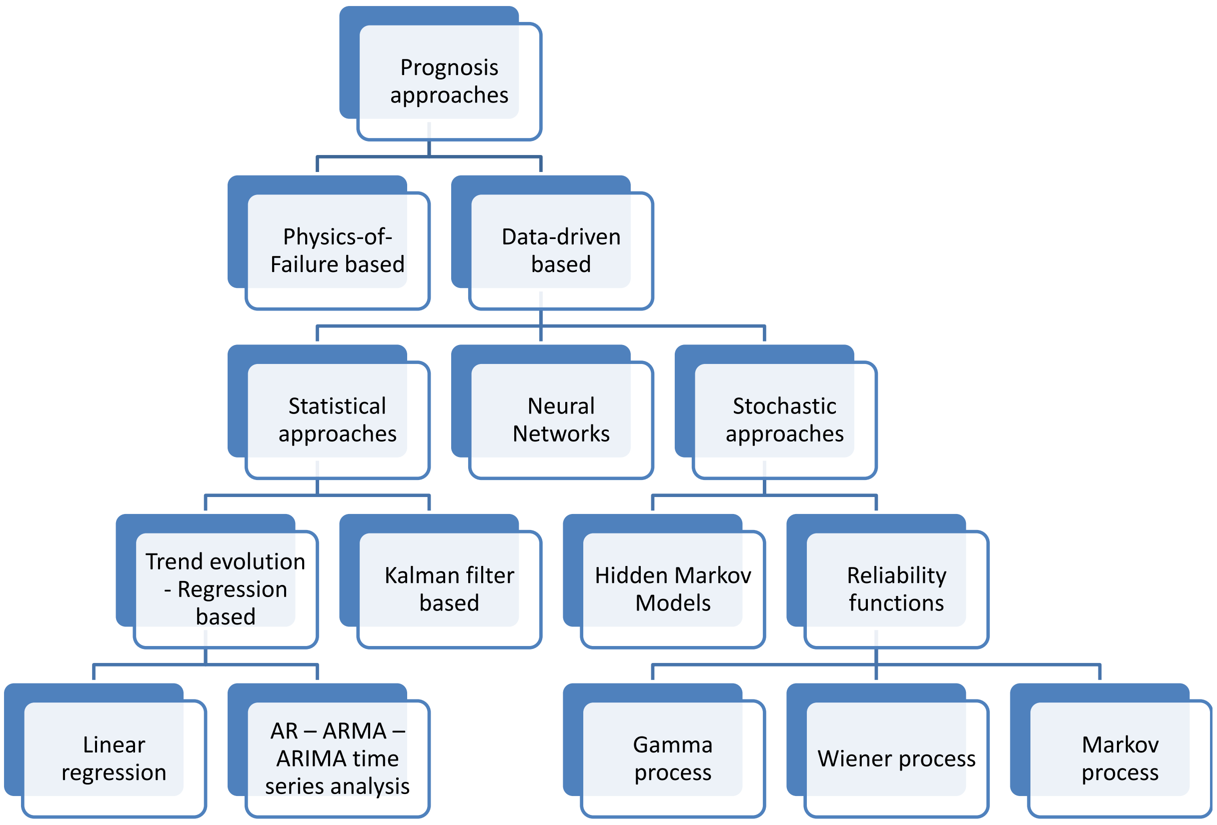


Figure 1.7: Classification of prognosis approaches.

© 2014 Tous droits réservés.

modeling are:

$$a = \text{instantaneous length of dominant crack,} \tag{1.1}$$

$$N = \text{running cycles,} \tag{1.2}$$

$$C_o, m = \text{material dependent constants,} \tag{1.3}$$

$$K = \text{Stress intensity factor,} \tag{1.4}$$

$$\Delta K = \text{range of stress intensity factor over one leading cycle.} \tag{1.5}$$

The stress intensity factor K is given by: $K = \sigma\sqrt{\pi a}\beta$, where σ is the applied stress, a is the crack length and β is a dimensionless factor which depends on crack length and component geometry. Based on the work in (Paris et al., 1961), it was shown that cyclic range in stress intensity factor ΔK controls the fatigue crack growth rate $\frac{da}{dN}$. Typically, the plot relating the logarithm of these two variables shows that the fatigue growth behavior of cracks has three different stages. These stages are graphically illustrated in Fig. (1.8).

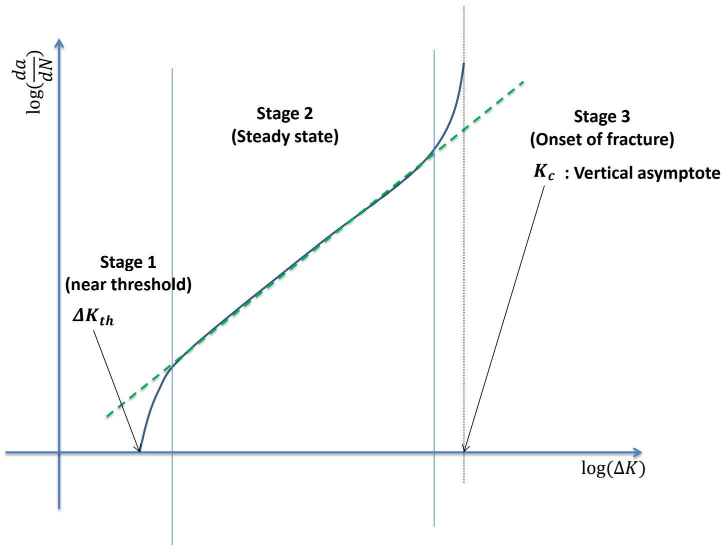


Figure 1.8: Typical fatigue growth behavior of cracks.

Stage 1 represents a lower asymptote for small values of ΔK ($\Delta K \leq \Delta K_{th}$) when there is no crack growth. Stage 3 represents an upper asymptote where K_{max} approaches fracture toughness K_c.

Concerning stage 2, Paris showed that this part of the fatigue crack growth has a linear behavior (in the logarithmic scale), such as there exists a model that describes it (green dashed line in Fig. (1.8)). This model is:

$$\frac{d\alpha}{dN} = C_o(\Delta K)^m, \tag{1.6}$$

Taking logarithms of both sides in Eq. (1.6), the equation becomes:

$$log(\frac{da}{dN}) = log(C_o) + m.log(\Delta K). \tag{1.7}$$

© 2014 Tous droits réservés.

Eq. (1.7) explains the linear behavior in stage 2 of Fig. (1.8).

In order to predict the fatigue life for a component, the Paris law is used. In fact, let N_f the number of running cycles until failure appears. Failure also corresponds to a maximal crack length, denoted a_f. Let a_0 be the initial length of crack. By integrating the Paris law, we have:

$$N_f = \int_0^{N_f} dN = \int_{a_0}^{a_f} \frac{da}{C_o(\Delta K)^m} \tag{1.8}$$

Numerical example: a relatively large sheet of steel is exposed to cyclic tensile stress of magnitudes 100MPa. Prior to testing, it has been determined that the length of the largest surface crack is 2mm. Estimate the fatigue life of this sheet if the critical crack length is 20mm and the values of m and C_o are 3 and 10^{-12}. Assume that β is independent of crack length and has a value of 1.

Solution:

$$\begin{aligned}
N_f &= \frac{1}{C_o(\Delta K)^m} \int_{a_0}^{a_f} \frac{da}{a^{\frac{m}{2}}} \\
&= \frac{1}{C_o(\pi)^{\frac{m}{2}}(\Delta\sigma)^m\beta^m} \int_{a_0}^{a_f} \frac{da}{a^{\frac{m}{2}}} \\
&= \frac{1}{C_o(\pi)^{\frac{3}{2}}(\Delta\sigma)^3\beta^3} \int_{a_0}^{a_f} \frac{da}{a^{\frac{3}{2}}} \\
&= \frac{-2}{C_o(\pi)^{\frac{3}{2}}(\Delta\sigma)^3\beta^3} [a^{\frac{-1}{2}}]_{a_0}^{a_f} \\
&= \frac{2}{C_o(\pi)^{\frac{3}{2}}(\Delta\sigma)^3\beta^3} (\frac{1}{\sqrt{a_0}} - \frac{1}{\sqrt{a_f}}) \\
&= \frac{2}{(10^{-12})(\pi)^{\frac{3}{2}}(100)^3 1^3} (\frac{1}{\sqrt{0.002}} - \frac{1}{\sqrt{0.02}}) \\
&= 5.5 \times 10^6 \text{ cycles.}
\end{aligned} \tag{1.9}$$

1.4.1.2 Successful and doubtful points of physics of failure prognosis models

The successful points and the doubtful points of physics of failure models are:
Successful points:

- Provide accurate RUL predictions.
- Provide confidence interval.

Doubtful points:

- A high-level knowledge of the physical behavior of the degradation is required.
- The accuracy of these models depends on the quality and the precision of the 'material constants'. Since the models are developed by means of experimental designs, the operating conditions of the physical asset must be similar to those in the experiments. If these conditions

© 2014 Tous droits réservés.

are not the same, the results (RUL prediction) will be less accurate.

1.4.2 Artificial Neural Networks

A prognosis model based on an ANN computes an estimation of the RUL by using the network that models the degradation behavior. Thus, the learning of the model is done by using observation data without requiring a physical understanding of the failure process. Generally speaking, a neural network has the ability to model complex non-linear dynamics and could cope with a big number of data. Direct estimation of the RUL is done by predicting the next values of a temporal data series until it reaches a final 'failure' value.

The first step to undertake while using an ANN is *training*. During this phase, the weights of all the neural architecture are automatically adjusted as to provide the desired output from a given set of inputs. The criterion for good training is the generalization error, or error between desired outputs and obtained outputs. Once the performance is optimized, the weights are fixed and used with unseen data.

The challenges encountered when building a prognosis model based on an ANN are:

- Having a large data set including degradation mechanisms.
- Initial choice of the parameters of the network: number of layers, number of neurons in each layer, and choice of the neuron's activation function.
- Defining a confidence level on the final outputs.

In (Huanga et al., 2007), the traditional Multi-Layer Perceptron was used for RUL prediction for bearing damage faults. It is an example of application of a static neural network. An example of dynamic neural network is presented in (Herzoga et al., 2009) where a regression neural network is proposed. The residual life was modeled for both laboratory samples and real operating pumps. In (Wanga et al., 2004), it was reported that neuro-fuzzy systems provide higher accuracy of RUL estimate than non-fuzzy time delayed neural network. Another type of neural networks that can be used for RUL estimation is the Dynamic Wavelet Neural Network (DWNN). WNN combines the theory of wavelets and neural networks into one. A WNN generally consists of a feed-forward neural network, with one hidden layer, whose activation functions are drawn from an orthonormal wavelet family. An example of application of neural networks for RUL prediction is given below with the use of a DWNN.

Finally, in some works, the neural network was not used to directly estimate the RUL directly, but used to estimate parameters of a known degradation function (derived from physics of failure techniques) (Jaw, 1999; Romeu, 2001). For instance, in (Jaw, 1999), the tip clearance of a turbine, which is related to the blade life, is modeled. In the following subsection, an example of application of neural networks for RUL prediction is given in the case of an industrial chiller.

© 2014 Tous droits réservés. doc.univ-lille1.fr

1.4.2.1 Example of application of neural networks for RUL prediction: bearing crack size prediction and RUL for an industrial chiller

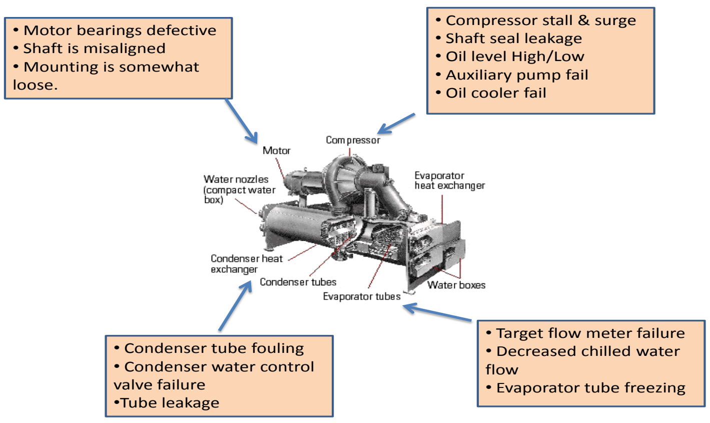

• Motor bearings defective
• Shaft is misaligned
• Mounting is somewhat loose.
• Compressor stall & surge
• Shaft seal leakage
• Oil level High/Low
• Auxiliary pump fail
• Oil cooler fail
Motor
Compressor
Water nozzles (compact water box)
Evaporator heat exchanger
Condenser heat exchanger
Condenser tubes
Evaporator tubes
Water boxes
• Condenser tube fouling
• Condenser water control valve failure
•Tube leakage
• Target flow meter failure
• Decreased chilled water flow
• Evaporator tube freezing

Figure 1.9: Industrial chiller and different failure modes affecting its different components (Vachtsevanos et al., 2006).

Industrial chillers are processes that are found in a lot of applications. They are used to support electronics, communications, etc. They are, by design, an assembly of several component including pumps, motors, compressors, etc. This example is taken from (Vachtsevanos et al., 2006).

A rolling-element bearing fault will be used to serve as an example for RUL prediction using DWNN. For this reason, triaxial vibration signals were collected via accelerometers. A first set of vibration signals with no cracked bearing was saved for a period of operating time. Then, the crack size of a bearing was increased (until failure) and another set of vibration signals was collected under these operating conditions. Time segments of vibrations signals from a good bearing and a defective one are shown in Fig. (1.11,(a)).

The raw vibration data were windowed, with each window containing several time points. The features that were extracted from these signals are the maximum values of the vibration signals and the peak value of the PSD (Power Spectral Density) signal, on each time window. In total, six features are obtained since each signal contains three components corresponding to three axes ($MaxS_x$, $MaxS_y$, $MaxS_z$ and $MaxPSD_x$, $MaxPSD_y$, $MaxPSD_z$). In Fig. (1.11,(c)), the signals $MaxPSD_x$, $MaxPSD_y$ and $MaxPSD_z$ are shown in order to have an idea on their evolution during time.

A DWNN was used to learn the crack growth behavior. The inputs are the six signals $MaxS_x$, $MaxS_y$, $MaxS_y$, $MaxPSD_x$, $MaxPSD_y$ and $MaxPSD_z$. The outputs are the crack growth signals, represented by the crack width and the crack depth (*see* Fig. (1.10) below).

The DWNN is first trained with the fault data up to the one-hundreth time window and then it is used to predict the evolution of the crack depth and width until the final bearing failure. A failure

© 2014 Tous droits réservés.

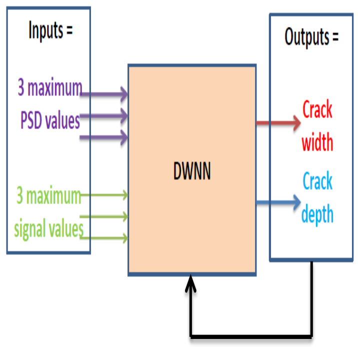


Figure 1.10: Dynamic wavelet neural network for RUL prediction in the case of an industrial chiller.

threshold must thus be defined for both the crack depth and the crack width. In Fig. (1.11,(b)), it is shown that the predicted values match the true values. In the example, the thresholds were set to $crack\_width = 2000\mu m$ and $crack\_depth = 1000\mu m$. In Fig. (1.11,(d)), it is seen that the crack reaches this hazard condition at approximately the one-hundred and seventy-fourth time window, and the crack depth threshold is reached first.

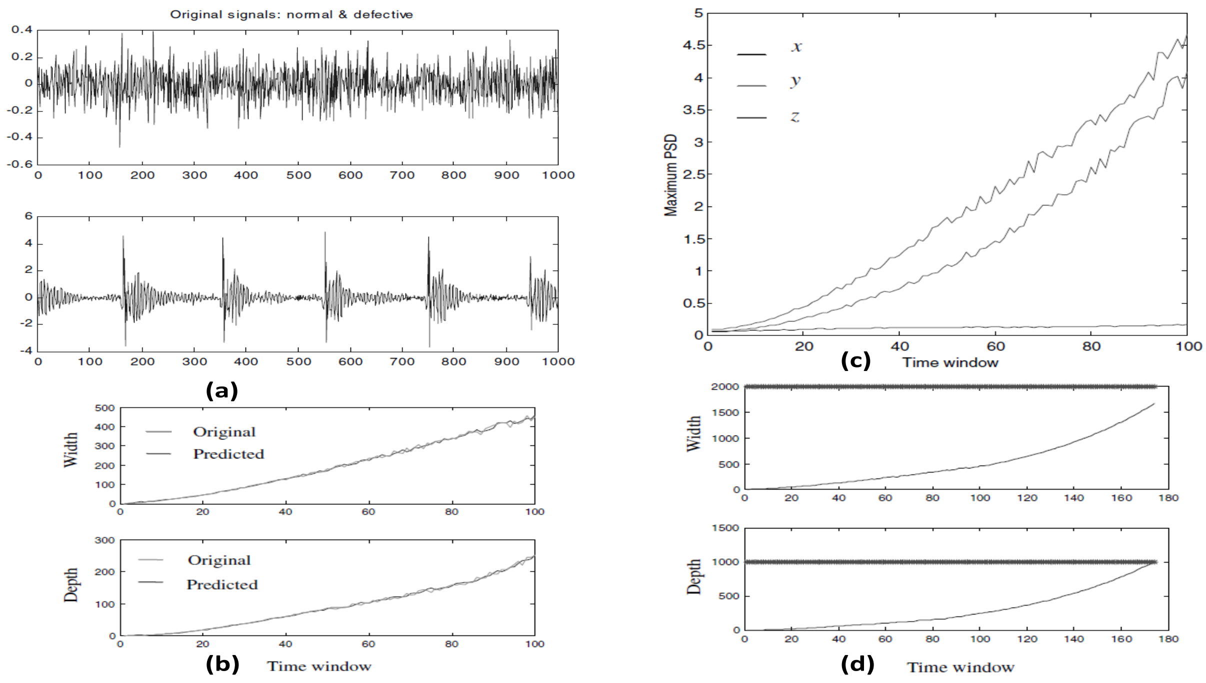


Figure 1.11: Neural network for RUL prediction: application example on the industrial chiller; (a) Vibration signals from a good and a defective bearing; (b) The crack growth predicted by the trained predictor within the 100^{th} time window ; (c) Maximum PSDs of the original signals; (d) The crack growth predicted by the trained predictor beyond the 100^{th} time window (Vachtsevanos et al., 2006).

© 2014 Tous droits réservés.

1.4.2.2 Successful and doubtful points of neural network based methods

The successful and doubtful points, when to consider and when to avoid ANN for prognosis modeling are given below.

Successful points:

– Complex and non-linear systems can be modeled.
– It is not necessary to understand the physical behavior of damage for a failure mode.
– Can handle big amounts of data.

Doubtful points:

– Large amount of data corresponding to the degradation behavior is required.
– Choosing the best ANN model parameters is based on trial and error and thus can be time consuming.
– Some networks cannot provide confidence intervals.

Another kind of methods that are guided by data are the stochastic models. They are presented in the next subsection.

1.4.3 Stochastic Models

A big area of research is associated to stochastic models for prognosis. They can be divided into two general sets of approaches: the reliability functions and the condition probability methods including the RUL Probability Density Function (PDF) and the Bayesian Networks (BN).

1.4.3.1 Reliability functions

The reliability approach consists in developing probabilistic models of the time to failure (which is the time of appearance of failure) on equipment. This is done by exploiting available historical data on the times to failure of a population of equipment. Then a hazard function and a PDF of the time to failure is determined for this population (M.Rausand and A.Hoyland, 2004). The computed PDF does not correspond to one incipient failure reaching a failure level. Instead, it represents the probability of occurrence of a failure on a system.

The hazard function (also called failure rate), can be interpreted as the probability that a component/subsystem, whose age is t, fails in the time interval $[t, t + dt]$. The lifetime of a component/subsystem is defined as its operating time until failure. Let $\lambda(t)$ be the hazard function. If $F(t)$ was the Cumulative Distribution Function (**CDF**) of lifetime, and $f(t)$ is its associated **PDF**, then the failure rate $\lambda(t)$ is given by:

© 2014 Tous droits réservés.

$$\lambda(t) = \frac{f(t)}{1 - F(t)}. \tag{1.10}$$

Several numbers of distributions are used to model the failure rate function. These distributions are the Exponential, Normal, Lognormal, Gaussian and Weibull functions. A commonly used hazard function is the bathtub curve (see Fig. (1.12)). It is developed in the example below.

1.4.3.2 Example of a reliability function: the bathtub curve

The bathtub curve, which is expressed as a sum of several Weibull distributions, is frequently used in the reliability engineering to model the hazard rate of an asset. In Fig. (1.12)), the bathtub curve is graphically illustrated. Three parts of the function are associated to three different ages of an asset. At an early age, the hazard rate is supposed high because of potential defective objects that are later discarded, and also because of early sources of potential failure such as handling and installation error. The second part is the useful life, which corresponds to the longest age. In this age, the failure rate is supposed to be low and constant. The third part corresponds to the age where the asset's life have passed its designed lifetime. Wear and fatigue are the major reasons of increase of the failure rate at this period.

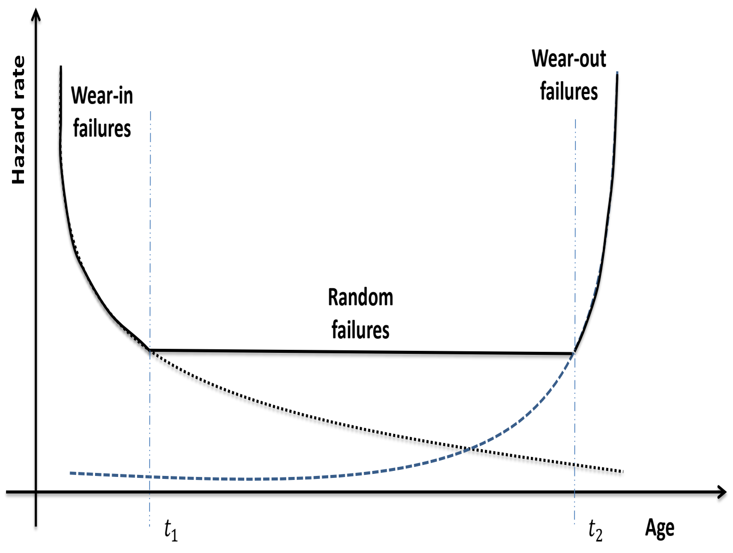


Figure 1.12: Bathtub curve.

1.4.3.3 Conditional RUL PDF or Bayesian model

The Bayesian model aims to represent the current state as a conditional probability function. Then, by using the Bayes theorem, the future behavior is estimated, in the probabilistic sense. The mainly used approaches included in the Bayesian models are the Gamma process and the Wiener process.

© 2014 Tous droits réservés. doc.univ-lille1.fr

1.4.3.4 Conditional RUL PDF - Gamma process

The Gamma process is a stochastic process that has the particularity to be monotonic, with positive independent increments. This particularity makes it adapted to model monotonic one-direction degradation. It is a natural model when deterioration gradually evolves with small positive increments. Examples of such deterioration are the wear process, the fatigue crack propagation, etc.

In theory, a Gamma process $\{Y(t), t \geq 0\}$ has two main properties:

1. The increments for any set of disjoint time intervals are independent random variables having a Gamma distribution. Particularly, the increment $Y(t_i) - Y(t_{i-1})$, for a given time interval $\Delta = t_i - t_{i-1}$ has a Gamma distribution,
2. $Y(0) = 0$ with probability equals to 1.

Let ω be a predefined threshold. By using a Gamma process, and by denoting X_{t_i} the RUL at time t_i, we have:

$$X_{t_i} = inf\{x_{t_i}; Y(t_i + x_{t_i}) \geq \omega | Y(t_i) < \omega\}. \tag{1.11}$$

Since the Gamma process is monotonic, Eq. (1.11) is equivalent to:

$$X_{t_i} = \{x_{t_i}; Y(t_i + x_{t_i}) \geq \omega | Y(t_i) < \omega\}. \tag{1.12}$$

In (Noortwijk, 2009), a thorough review of the application of Gamma processes to maintenance is presented. In addition, the mathematical properties of Gamma processes are well documented.

1.4.3.5 Conditional RUL PDF - Wiener process

Let $\{B(t), t \geq 0\}$ be the standard Brownian Motion process. This process has three properties:

1. $B(t_i)$ is a continuous function with respect to the time,
2. $\forall \tau_i \leq t_i$, $B(t_i) - B(\tau_i)$ is normally distributed,
3. $\forall \tau_1 \leq t_1 \leq \tau_2 \leq t_2 \leq \ldots \leq \tau_i \leq t_i$, $B(t_1) - B(\tau_1)$, $B(t_2) - B(\tau_2)$, ... , $B(t_i) - B(\tau_i)$ are independent.

The Wiener process $\{Y(t), t \geq 0\}$ (or also known as the Brownian Motion with Drift) can be represented by $Y(t) = \lambda t + \sigma B(t)$, where λ is a defined drift parameter, $\sigma \geq 0$ is a diffusion parameter and $B(t)$ is the standard Brownian Motion. The RUL, associated to a variable X_{t_i}, is the first passage time for which the Wiener process crosses a threshold ω (the same as in Eq. (1.11)). It is known that the PDF of the RUL of a Wiener process follows an inverse Gaussian distribution (Cox and Miller, 1965). Some applications of Wiener process in survival analysis in relation to lifetime modeling are presented in (Pennell et al., 2010; Lee et al., 2009).

1.4.3.6 Conditional RUL pdf - Markov processes

The **Markov models** assume that a component/system has a finite number of states in which it can be. In these models, a probability is associated to be in each state, and a probability of

© 2014 Tous droits réservés. doc.univ-lille1.fr

transitioning from one state to another is defined. The estimation of the future health state can be done using those two defined probabilities. When used for prognosis, several assumptions are made (M.Rausand and A.Hoyland, 2004):

– Probabilities of transition from one state to another is constant.
– The sojourn time in one state is exponentially distributed with a constant rate.
– The sum of all transition probabilities from each state to all other states is equal to one.

Unlike Markov models, **semi-Markov** models do not require that the sojourn time be exponentially distributed. It could be of any distribution. Thus, semi-Markov models are more accurate than Markov models for RUL estimation. An example of RUL estimation using Markov modeling is given in (Carlin and Chib, 1995). Examples of RUL estimation using semi-Markov models are given in (Montgomery et al., 2006; Montgomery and Jefferis, 2007).

In Markov models, all the states are supposed to be observable. An extension of Markov models, where this assumption is let down, is the Hidden Markov Models (HMM). Analogically, semi-Hidden Markov Models do not assume constant failure rate and exponential distribution of sojourn time in states. Thus, they also are more appropriate for the estimation of RUL. Examples of use of HMMs and semi-HMMs are given in (He et al., 2006; Bechhoefer et al., 2006; Ocak et al., 2007; Dong and He, 2007).

1.4.3.7 Successful and doubtful points of stochastic models for prognosis

Prognosis using stochastic reliability functions is well understood by the reliability engineering community. Confidence intervals are also available for RUL prediction. However, these methods are more adapted to preventive maintenance and not to predictive maintenance because they are offline. Thus, they do not require CM but require significant sample size pertaining to each failure mode.

Prognosis methodologies based on conditional RUL PDF models are simple. They also provide confidence intervals for RUL prediction. The accuracy and precision of the RUL prediction values increase as time to failure decreases. They also do not require CM data but require only the monitoring of the time before the occurrence of failure. However, these methodologies are also offline. They assume that hazard is only a function of operational time and do not take external risk factors. In addition, the available accuracy and precision of RUL prediction depend on forecasting interval. Thus, they require a statistically significant sample size for each failure mode in order to be able to have reliable RUL predictions.

Markov models have the advantage that they can model the different stages of degradations, corresponding to different failure modes in parallel. They also provide a confidence interval with the RUL prediction. However, these methodologies are very sensitive to the amount of historical data that are available for training.

© 2014 Tous droits réservés. doc.univ-lille1.fr

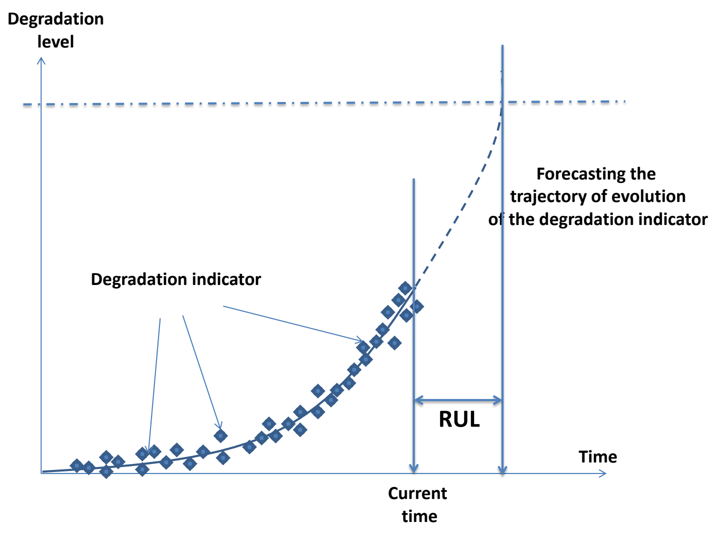


Figure 1.13: Example of evolution of a degradation indicator.

1.4.4 Data-Driven approaches - Statistical approaches

Statistical approaches attempt to calculate a RUL based on the CM data. CM data describe the underlying state of an asset. CM data are very versatile. In some cases, CM data reflects degradation directly and the estimation of the RUL is the estimation of this particular data to reach a predefined threshold. In this case, the CM data is referred to as DCM (D for direct) since it is directly used for RUL forecasting. However, in other cases, a health indicator should be extracted from CM data in order to reflect the evolution of degradation. In these cases, the CM data is referred to as ICM (I for indirect). In both cases, an indicator for degradation must be available and statistical approaches rely on it.

1.4.4.1 Trend evolution analysis - Regression based statistical methods

These methods, which are interesting for us, apply in the case where we have one indicator that reflects the degradation. Then the RUL is the interval of time between the actual instant and the estimated time on which this indicator reaches a threshold. An example is given in Fig. (1.13).

For forecasting and projection into the future, simple forecasting algorithms can be used. One can cite linear and non-linear regression techniques (Li et al., 1999; Peysson et al., 2008) as well as modeling times series using polynomial modeling. Regression techniques are inspired from the time series analysis domain, where a degradation indicator is considered as the time series. Auto-Regressive (AR) models, Moving Average (MA) models, a combination of these two Auto Regressive Moving Average (ARMA) models (Yan et al., 2004; Pham and Yang, 2010), Auto Regressive Integrated Moving Average (ARIMA), linear and quadratic polynomial models are known and widely used for forecasting time series (Brockwell and Davis, 2002a). The principle is that the regression and polynomial models are constructed on the degradation indicator, and then used to forecast future values until a predefined threshold is reached. The mathematical formulation of these models are

© 2014 Tous droits réservés.

detailed in the next chapter.

The ARMA model has been successfully used for prognosis. In (Yan et al., 2004), a logistic regression model was used to calculate the probability of failure for a given condition variable. The considered application was an elevator door motion system. The condition variable was the duration of an open/close cycle of the door. Big values of this duration meant that the system is starting to deteriorate. A threshold was defined for failure point identification. The ARMA model trended the evolution of the condition variable (cycle duration) until failure.

In (Pham and Yang, 2010), an ARMA model was used along with a GARCH (Generalized Auto-Regressive Conditional Heteroscedasticity (Engle, 1986)) to estimate the health state of a machine using vibration signals. Even though no explicit calculation of the RUL was given, their method allowed the estimation of the machine health into the future.

In (Lia et al., 2010), Principal Component Analysis (PCA) was executed on sensor measurements to compute degradation relevant features. A de-noising step using wavelet networks followed the PCA analysis. A Vector AR (VAR) model (which is the multivariate form of a AR model) was defined over the calculated features. The VAR model was recursively updated as new features arrive. The computation of the trends of these features was done using a multi-step prediction paradigm.
In (Wu et al., 2007), an improved ARIMA-based prediction method was used. The authors argued that the biggest disadvantages of auto regressive models are their sensitivity to noise and initial conditions, and that when forecasting, the error of prediction propagates since predicted values are used to predict also future values. To overcome this difficulty, the authors proposed to adapt the parameters of the model after each one step prediction, using the newly predicted value.

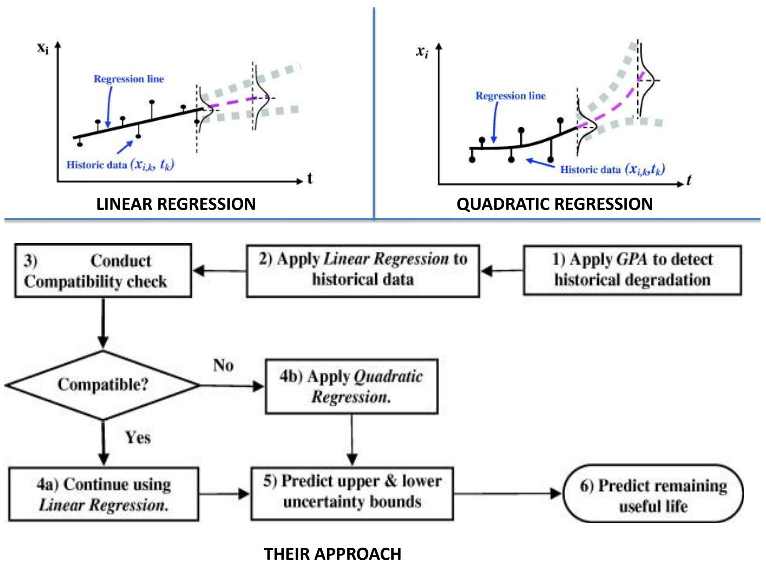
Another time series prediction method that is alternative to ARMA model is the Dempster-Shafer regression. To our best knowledge, the paper (Niu and Yang, 2009) seems to be the first example of this technique being applied to machinery prognosis. The use of the Dempster-Shafer regression has a lot of potential in modeling non-linear and chaotic temporal trend for prognosis, which ARMA techniques cannot model.

As stated earlier, polynomial models defined over a degradation indicator are also used for prognosis. An example of use of polynomial modeling for prognosis is shown below.

1.4.4.2 Example: polynomial modeling for prognosis

In (Li and Noilkitsaranont, 2009), the authors applied linear and quadratic regression to compute the remaining life. The application concerned the gas turbine performance prognosis. They argued, based on a literature review, that the degradation of gas turbines follows a linear trend in the beginning of the degradation then changes into a quadratic trend. So they started their trend evolution modeling by a linear trend then at a certain commutation instant T_c, they changed into quadratic regression. The Fig. 1.14 explains graphically pretty well their approach, and illustrates both linear

© 2014 Tous droits réservés. doc.univ-lille1.fr



x_i
Regression line
Historic data $(x_{i,k}, t_k)$
t
LINEAR REGRESSION
x_i
Regression line
Historic data $(x_{i,k}, t_k)$
t
QUADRATIC REGRESSION
3) Conduct Compatibility check
2) Apply Linear Regression to historical data
1) Apply GPA to detect historical degradation
Compatible?
No
4b) Apply Quadratic Regression.
Yes
4a) Continue using Linear Regression.
5) Predict upper & lower uncertainty bounds
6) Predict remaining useful life
THEIR APPROACH

Figure 1.14: Linear regression; Quadratic regression; Approach in (Li and Noilkitsaranont, 2009).

and quadratic regression. GPA in Fig. 1.14 means *Gas Path Analysis* is the diagnosis method they used to detect degradation in a gas turbine tube. Since we are interested in their prognosis approach, the GPA diagnosis method will not be discussed. T_c is determined by a compatibility check using hypothesis significance test on the error between the regression line initially learnt (in the beginning of the degradation) and the new incoming degradation data. Their method also allowed modeling of prognosis uncertainty by estimating the variance of this error. The authors obtained bounds for RUL estimation defining *pessimistic* RUL and *optimistic* RUL.

1.4.4.3 Kalman filter - Application to prognosis

Kalman filter is a state estimation technique that can be used for prognosis. It is a digital recursive filter that is used to estimate the state of a dynamical system from a series of noisy measurements. At each instant, the Kalman filter provides the state estimate and the error covariance. Given the actual state and observations, the Kalman filter can predict the next state in five steps (*see* Fig. (1.15)).
In fact, generalized in (Byington et al., 2002) and applied in (Batzel and Swanson, 2009), the idea behind using Kalman filter is to consider that the state of a system, denoted X is given by:

$$X = \begin{bmatrix} f_i \\ \dot{f}_i \\ \ddot{f}_i \end{bmatrix}, \tag{1.13}$$

where f_i is the degradation signal, $\dot{f}_i$ is the speed of variation of the degradation signal and $\ddot{f}_i$ is its acceleration, at time stamps t_i. The Kalman filter tracking equations are then applied on the state X.
The method of estimation of the RUL described in (Byington et al., 2002; Batzel and Swanson,

© 2014 Tous droits réservés. doc.univ-lille1.fr

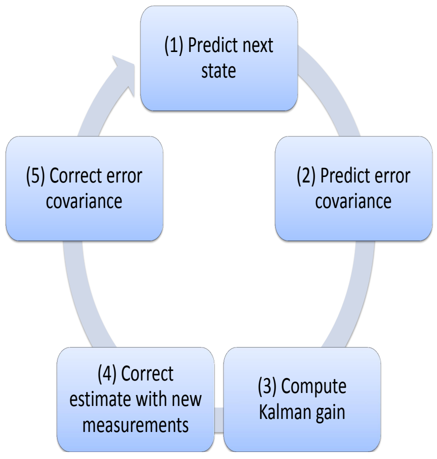

(1) Predict next state
(2) Predict error covariance
(3) Compute Kalman gain
(4) Correct estimate with new measurements
(5) Correct error covariance

Figure 1.15: Kalman filter process.

2009) uses the present states estimated by the Kalman filter, and assumes constant acceleration. Under this assumption, a Newtonian *kinematic* equation can then be applied on the degradation indicator f given by:

$$f_{i+1} = f_i + \dot{f}_i t_i + \ddot{f}_i t_i^2. \tag{1.14}$$

For a given degradation threshold f_d, RUL is equal to the value of i where equation (1.15) is satisfied:

$$|f_{RUL} - f_d| < \epsilon, \ \epsilon << 0. \tag{1.15}$$

1.4.4.4 Successful and doubtful points of regression based methods

The successful and doubtful points of the regression based statistical prognosis models are:

Successful points:

- Simple techniques to apply and understand.
- The amount of historical data corresponding to failure operating modes does not need to be exhaustive. In case of non-presence of historical failure data, expert knowledge can be used to set thresholds that defines failure conditions.
- Physical understanding of the failure mechanism is not necessary.
- Provide reliable short term predictions of RUL.

Doubtful points:

- Necessity to have a single degradation parameter with an obvious degradation trend.
- High sensitivity to initial conditions.
- The prediction of RUL becomes less and less reliable as the horizon of prediction becomes larger. However, it is interesting to note that this limit affects also physics-of-failure based

© 2014 Tous droits réservés.

prognosis models, but on a smaller scale.

1.5 Prognosis: performance evaluation of prognosis results

The ability of evaluating prognosis results is a necessary step for the analysis of the performance of prognosis models. For this reason, few questions arise: how to evaluate a prognosis result? What are the metrics that can be used for this purpose?

There are three categories of metrics that can be used to evaluate prognosis results. The first one is from an algorithmic point of view, as it concerns the performance of the forecasting algorithm. The second one is from a computational complexity point of view as it concerns the available resources needed for prognosis. And the third is from an economical point of view, as it concerns the profits to gain by implementing a prognosis system (Saxena et al., 2008, 2009).

1.5.1 Performance metrics of the forecasting algorithm

Performance defines the accuracy of the estimate obtained and of the confidence interval that can be attributed to a prediction, taking into account the uncertainties and disruptions inherent to the system. In order to evaluate the performance of a prediction algorithm, three metrics can be used. Let:

$$\begin{aligned} EOL &= \text{Time index of the End-Of-Life,} \\ r^l &= \text{Predicted RUL computed by the prediction algorithm,} \\ r^l_* &= \text{Real RUL,} \\ P &= \text{Time index at which the first prediction is made by the prediction algorithm,} \end{aligned}$$

The three metrics are:

1. The α-λ performance: α is called the accuracy modifier and λ the window (or more generally the time unit) modifier. It is used to evaluate if the prognosis result remains inside a precision interval, at each time stamp t_i. The length of the precision interval, determined by the value of α is also function of time. It is given an initial value at $t = t_p$ then it is linearly reduced until it reaches zero ($\alpha = 0$) at $t = t_{EOL}$. The value of λ is used to change the time index. Thus, the α-value is changed according to a λ-value as explained in Fig. (1.16). The α-λ performance metric is a Boolean value (True/False, 0/1) that takes the value true if:

$$(1-\alpha)r^l_*(t_i) \leq r^l(t_i) \leq (1+\alpha)r^l_*(t_i), \tag{1.16}$$
$$i = P + \lambda(EOL - P), \tag{1.17}$$
$$\lambda\text{: window modifier, } 0 \leq \lambda \leq 1.$$

2. Relative accuracy: it is defined as the difference between the real RUL value and the estimated RUL value, at time stamp t_i (*see* Fig. (1.17)). It is denoted RA and it is given by:

© 2014 Tous droits réservés. doc.univ-lille1.fr

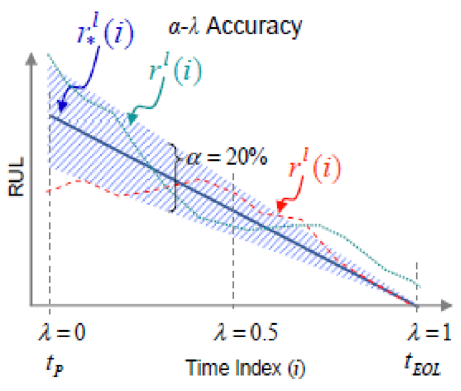


Figure 1.16: Graphical illustration of the α-λ performance metric (Saxena et al., 2008, 2009).

$$RA(t_i) = 1 - \frac{|r_*^l(t_i) - r^l(t_i)|}{r_*^l(t_i)}, \tag{1.18}$$

$$i = P + \lambda(EOL - P), \tag{1.19}$$

λ: window modifier, $0 \leq \lambda \leq 1$.

This metric is bounded by 1 such as: $RA(t_i) \leq 1$. The value 1 corresponds to a perfect score because it corresponds to $r_*^l(t_i) = r^l(t_i)$. The Cumulative Relative Accuracy (CRA) is the sum over all the time stamps from P to EOL. The CRA is given by:

$$CRA = \frac{1}{EOL - P + 1} \sum_{\lambda=P}^{\lambda=EOL} RA(t_\lambda). \tag{1.20}$$

In the same manner, the CRA metric is bound by 1, with 1 corresponding to perfect score.

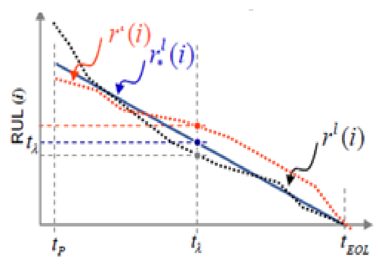


Figure 1.17: Graphical illustration of the relative accuracy performance metric (Saxena et al., 2008, 2009).

3. Convergence: this metric is defined to quantify the manner in which a curve converges to an ideal curve. In our case, the curve is the predicted RUL and the ideal curve is the real RUL. Given r_*^l and r^l the real RUL and the predicted RUL respectively, let:

$$M = r_*^l - r^l, \tag{1.21}$$

© 2014 Tous droits réservés. doc.univ-lille1.fr

be the resulting curve that illustrates the convergence of the predicted RUL to the real RUL. In Fig. (3.48), three curves are shown to be converging at different rates. It can be shown that the distance between the origin and the centroid of the area under the curve quantifies this convergence. The lower is this distance, the faster is the convergence. Convergence is a useful metric since we expect that the RUL given by a prognosis algorithm converges to the real value of RUL as time increases.

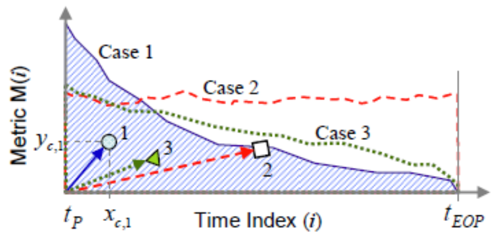


Figure 1.18: Graphical illustration of the convergence performance metric (Saxena et al., 2008, 2009).

Let (x_c, y_c) be the centroid of the area under the curve $M(i)$. Then, the convergence metric, denoted by C_M, can be represented by the Euclidean distance between the center of mass and $(t_P, 0)$. C_M is given by:

$$\begin{aligned} C_M &= \sqrt{(x_c - t_P)^2 + (y_c)^2}, \\ x_c &= \frac{1}{2}\frac{\sum_{i=P}^{EOL}(t_{i+1}^2 - t_i^2)M(i)}{\sum_{i=P}^{EOL}(t_{i+1} - t_i)M(i)}, \\ y_c &= \frac{1}{2}\frac{\sum_{i=P}^{EOL}(t_{i+1} - t_i)M(i)^2}{\sum_{i=P}^{EOL}(t_{i+1} - t_i)M(i)}, \end{aligned} \tag{1.22}$$

The metric C_M is stricly positive and unbounded, such as $C_M \in [0, \infty[$. The perfect score corresponds to the value 0.

The developed metrics are interesting since they all contain information about the performance of a prognosis model, but still have few differences between them. The CRA metric evaluates the global accuracy of the prognosis model. It gives the same importance to RUL predictions at all the time stamps, beginning from t_p until t_{EOL}. The α-λ performance metric supposes that as time passes, the RUL prediction must become closer to real RUL values (by reducing the α value). However, its Boolean nature makes all the points falling inside the precision borders (defined by the value of α) have the same value. The convergence metric C_M overcomes this problem but its value is intuitively less obvious then the α-λ performance metric. Consequently, in order to take into consideration all these metrics, a new metric will be developed and will be discussed in the next chapter.

© 2014 Tous droits réservés. doc.univ-lille1.fr

1.5.2 Other metrics: from an economical and computational point of views

As stated earlier, other metrics can be considered. From an economical point of view, two metrics can be calculated:

1. Life cycle cost with and without prognosis: the life cycle cost includes construction, acquisition and maintenance of a system. This metric evaluates the life cycle cost of a system that is equipped with a prognosis function, compared to a system that is not equipped with such function.
2. Investment gain: this metric evaluates the economic gains that are generated by the prognosis function.

Concerning the computational complexity, a metric to evaluate computational resource of an algorithm is measured by its calculation complexity, and described by the big 'O' notation. For example, if the time performance of an algorithm is $O(n^2)$, then the time needed to run the algorithm increases in a quadratic manner with the size of the input, if n is the number of inputs. It is sometimes important to take this factor into consideration, depending on the application.

1.6 Research orientation and conclusion

In this chapter, a thorough study concerning the thematic of prognosis has been made. It was presented as a key module in the CBM strategy. A state-of-the art on prognosis modeling was presented. Finally, different performance metrics that can be applied to a prognosis model have been given.

The study presented in this chapter is used to guide the research orientation for the rest of the work. In fact, in this thesis, and as it was stated in the general introduction, the aim is to develop an architecture that can provide supervision, diagnosis and prognosis results based on the exploitation of the data generated from a system. In addition, the prognosis model that is needed to be developed must have the following characteristics:

1. It should be a data-driven model, based on the exploitation of a degradation indicator generated from the data gathered from the system.
2. It should be reactive, in the sense of being online and adaptive. This particularity is very interesting since it means that the model is always updated as new data arrive.
3. It is considered that the available degradation data are not abundant. This means that the prognosis model must make use of the least possible knowledge on the degradation mechanisms that can affect different components or subsystems.

By making an analysis of the successful points as well as the doubtful points of the different prognosis methodologies, it can be stated that the regression techniques are an adequate choice when dealing with evolving systems with varying dynamics. They are simple techniques that do

© 2014 Tous droits réservés. doc.univ-lille1.fr

Thèse de Antoine Chammas, Lille 1, 2014

not require an exhaustive amount of historical data and could be adapted online. They do not also require any physical knowledge concerning the degradation mechanisms.

However these techniques are based on the necessity to have a degradation signal with an obvious trend. In addition, the prognosis model needs to be reactive, always updating as new values of this degradation signal are available. In order to tackle these difficulties, the system is considered as an environment that is generating non-stationary data. The non-stationarity is the result of the drifting faults that the system is subject to. In the next chapter, the problem of non-stationary environments is tackled. This problem is formally known as the concept drift problem.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Chapter 2

Concept drift: state-of-the-art

2.1 Introduction

A system is subject to drift when an incipient fault causes intrinsic changes in its parameters. This results in a change in the properties of the data that are generated by this system. Thus, the environment is non-stationary. In the context of CBM and prognosis, what is needed is an algorithm that can model data in non-stationary environments, in the aim of extracting indicators for health assessment and prognosis.

Conventional modeling algorithms proceed in an offline manner. A model is constructed from the historical data then applied online to the incoming data. In online manner, this model is used to fulfill its task, which could be prediction, decision making, etc. However, these conventional methods fail when the environment generating the data is subject to change. These **changing** environments or so called **non-stationnary** environments will induce incorrect outputs from the model. The changing in the environment is known as **Concept Drift**. It refers to a slowly changing environment. For abrupt changes, the term **Concept Shift** is used. For both "concept drift" and "concept shift", adaptation of the model is required. This gives rise to evolving modeling techniques that are designed to cope with changing environments. Their aim is to continuously give an authentic representation of the environment. Thus their structure as well as their parameters could persistently be subject to changes. Evolving models are also referred to adaptive models.

Evolving models are required to acquire new information from new data (update of the parameters and/or structure) and to forget their old characteristics. This aspect of learning new information and forgetting old information is known as the **stability-plasticity** dilemma or **learning-forgetting** dilemma. Abusive forgetting (high forgetting factor) could lead to what is known as catastrophic forgetting, which is the loss of old learnt relations inside the model. In this case, the plasticity is larger than stability. The opposite case, which corresponds to higher stability, could mean a less reactive model, requiring more time to adapt to abrupt or fast changes.

The modeling techniques involved in this thesis are part of the **Data Mining** techniques. Data

© 2014 Tous droits réservés. doc.univ-lille1.fr

mining is an important research field in which methods have been proposed to automatically extract useful information from data sets. These methods aim at building models given sets of Input-Output vectors of data. This particularity makes them more interesting than physical-based models in which a physical model is required. For complex systems, mathematical models (given for example with differential equations) are very hard to obtain, and in some cases impossible. This makes data-driven methods like the ones developed in data mining more adequate to such systems. They make use of **machine learning** algorithms in order to learn models from collected data.

In this chapter, the concept drift theory is presented and studied. In addition, the state-of-the-art on the adaptive modeling techniques is given, providing a complete comprehension on how to handle non-stationary changing environment. At the end of the chapter, a synthetic conclusion is gven in order to orientate the future research work that is needed to be done. In order to begin with the study on the concept drift theory, it is important to understand the nature of the data that the adaptive techniques must handle. This is the subject of the subsection below.

2.1.1 Data streams

Adaptive modeling techniques have to cope with the streaming nature of data. In fact, in today's information society, a huge amount of data related to a lot of activities is gathered for processing purposes in the form of *Data Streams*.

A data stream can be read only once or a small number of times using limited computing and storage capabilities. It is potentially unbounded in size, thus practical considerations have to be taken into account. The main characteristics of the data stream imply the following constraints (Bifet and Gavalda, 2007):

1. It is impossible to store all the data from the data stream. Only a small amount of data from a data stream can be stored.
2. The arrival speed of a data stream forces their processing step to be done in real time. The processing time must not be larger than the sampling time or time of arrival of new data.

Constraint 1 limits the amount of memory that algorithms operating on data streams can use, while constraint 2 limits the time in which an item can be processed. In (Gama and Castillo, 2006), streams of data were defined by a sequence $< S_1, S_2, ..., S_t, ... >$ in which each element is a set of instances generated by some stationary distribution D_i. They named each element S_i a **context**.

In stationnary environments, consecutive contexts are nearly the same. In non-stationnary environments, these contexts differ as time goes on creating the concept drift.

2.2 Concept drift theory

Let us begin with the definition of a concept. Let $X \in \mathbb{R}^d$ denotes a *feature vector* or a *pattern* containing d features. X_i, or $X(t_i)$, or X_{t_i} are three equivalent notations that denote a pattern

© 2014 Tous droits réservés. doc.univ-lille1.fr

appearing at time stamp t_i. X^j is the j^{th} feature and $X^j(t_i)$ is the j^{th} feature appearing at time stamp t_i. Let y_i be the target output of the feature X_i. y_i can be a class/cluster membership, or a real value. A concept is the description of the data in the feature space, completely defined by the joint probability $P(X_i, y_i)$. A concept drift (or shift) is a direct change of this joint probability.

2.2.1 Causes of Concept Drift

Concept drifts are related to changing patterns in datasets. In real world, concepts are often not stable but change with time. As an example, the customer's buying preferences. A customer can change his buying interests with time, in an unpredicted manner, due to some environmental change. This unpredicted change is due to what is called **hidden context** (Tsymbal, 2004). In this example, a hidden context could be the inflation rate, the availability of certain products, etc.

Another example in industrial processes, incipient faults and degradation of components are unwanted events that cause also drifts. The time of appearance of these process drifts is also unknown. Their fast detection and isolation is a major concern for companies.

Therefore, hidden contexts are the main cause of concept drifts. In real world, a high number of important properties on a specific domain can be hidden from view. Furthermore, these hidden properties may change over time. A lot of other examples can be cited but the dilemma remains the same, it is absolutely necessary to track these concept drifts and adapt models that were built on old data.

2.2.2 Concept Drift - Bayesian Point of View

Recall that X_i and y_i are respectively the feature vector and the output arriving at the time stamp t_i. A concept could be defined by the joint probability $P(X_i, y_i)$. We know from the Bayesian theory that:

$$P(X_i, y_i) = P(y_i|X_i).P(X_i) = P(X_i|y_i).P(y_i), \tag{2.1}$$

where:
- $P(X_i)$ is the unconditional Probability Density Function (PDF),
- $P(y_i|X_i)$ is the posterior probability,
- $P(y_i)$ is the prior probability,
- $P(X_i|y_i)$ is the class conditional PDF.

The formalization of the notion of concept drift can be detailed using Eq. (2.1) (Gao et al., 2007). According to Eq. (2.1), the analysis of concept drift yields four possible cases:

1. No change: neither $P(X_i)$ nor $P(y_i|X_i)$ change,
2. Change in $P(X_i)$: this type of change is called feature drift or *sampling shift* (Ikonomovska et al., 2011). The value of $P(y_i|X_i)$ remains the same,

© 2014 Tous droits réservés. doc.univ-lille1.fr

3. Change in $P(y_i|X_i)$: this type of change is called conditional change, or class drift. The value of $P(X_i)$ remains the same,
4. Change in both values $P(y_i|X_i)$ and $P(X_i)$.

2.3 Characterization of concept drift

2.3.1 Types of concept drift

Based on our experience, and the works done in the literature (Tsymbal, 2004; Zliobate, 2009; Minku et al., 2010), drifts can have several types as depicted in Fig. (2.1).

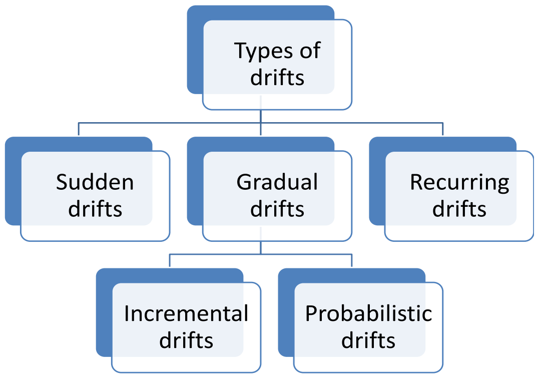


Figure 2.1: The different types of drifts.

Sudden drifts, known also as concept shifts, are in the case when the evolution from the old concept to the new concept is made in an abrupt way. Examples of sudden drifts are traffic jams due to accidents on a highway or communication loss due to network breakdown. From an industrial point of view, some kind of abrupt faults can drastically damage a process bringing it to a paralyzed state.

Gradual drifts are in the case where the migration of the old concept to the new concept is made in a slowly evolving way. A slowly wearing piece of factory equipment can have slowly drifting effect on the quality of its products. Slow drifts can be subdivided into two categories:

- **Probabilistic** drifts are in the case where the generated data comes from two (or more) active sources. As time passes, the probability of sampling from one source decreases and the probability of sampling from the other source(s) increases. As an example, let us consider the generation of electrical power from a hybrid wind-diesel generator (in this sense, hybrid means the connection of several electrical power sources, forming eventually one electrical power generator). If the wind speed was originally high and started to slowly decrease, the generated power from the diesel generator will slowly increase to higher and higher values. Fig. (2.2) highlights this type of drift graphically.
- **Incremental** drifts are another case of slow drifts. In this type of drifts, two (or more) sources

© 2014 Tous droits réservés.

are also considered to generate the data. However, the difference between the sources is very small. Thus, larger time periods are required to observe the totality of the drift. This kind of drifts is abundant in industrial processes. It could be referred to as a process drift. Fig. (2.3) illustrates this drift graphically.

Recurring drifts are in the case where old learnt active concepts may reappear. In industrial applications, recurring drifts appear usually in sensors. For example, icing on an anemometer sensor is a recurrent and is present in time of freezing temperatures.

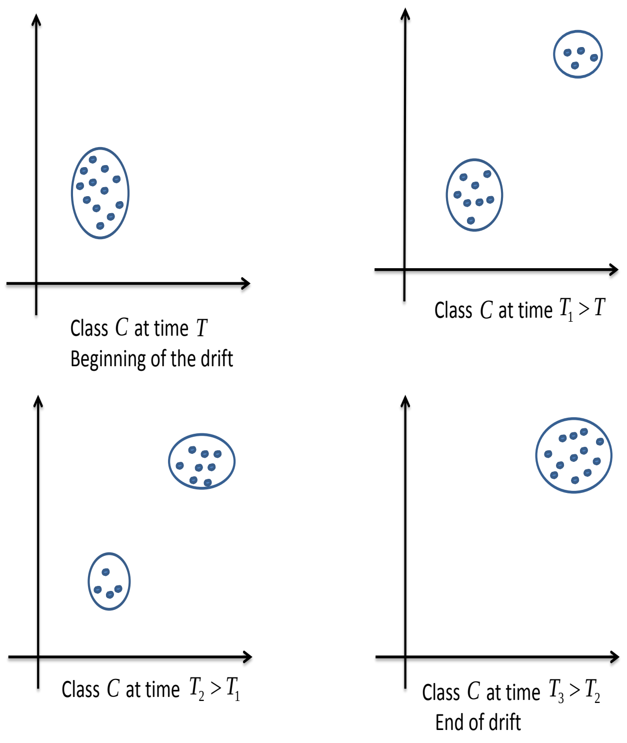

Class C at time T
Beginning of the drift
Class C at time $T_1 > T$
Class C at time $T_2 > T_1$
Class C at time $T_3 > T_2$
End of drift

Figure 2.2: Graphical illustration of a probabilistic drift.

2.3.2 Nature of concept drift

Concept drifts can have two natural identities: they can be *global* drifts or *local* drifts (Tsymbal, 2004; Tsymbal et al., 2008; Gama and Castillo, 2006). A local concept drift may be defined as changes in concept and data distribution occurring at specific subsets of the data. Formally speaking, it could be seen as a change occurring in some part of the instance space (a subset of the instance space). A global drift nonetheless affects the whole input instance space. This is why, it is considered to be on a data set level, because all instances inside a window of data represent a drift. In Fig. (2.4), the difference between a global and a local drift is illustrated.

In industrial processes, this difference between global and local drift becomes more clear in hybrid systems. In such cases, different operating modes are generally present. Some faults, such local drifts,

© 2014 Tous droits réservés.

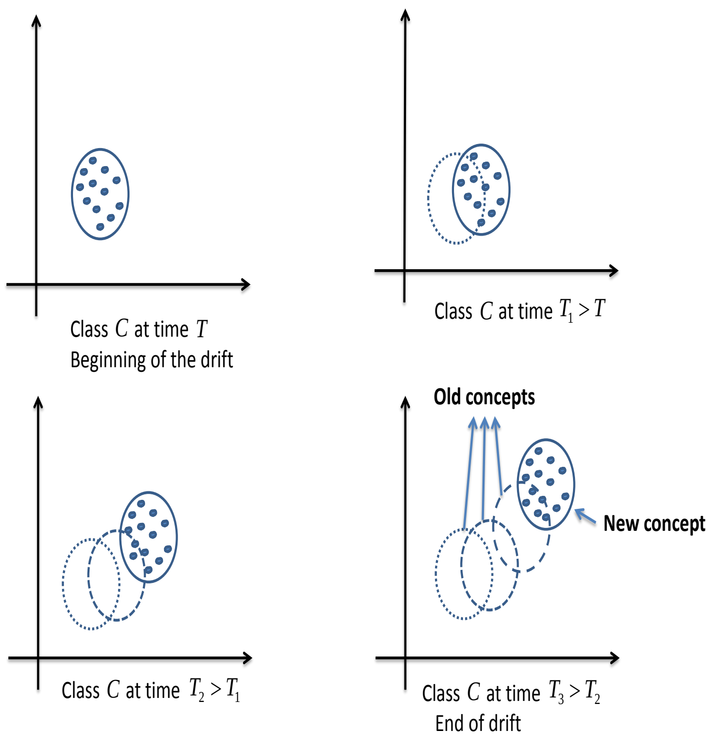

Class C at time T
Beginning of the drift
Class C at time $T_1 > T$
Class C at time $T_2 > T_1$
Old concepts
New concept
Class C at time $T_3 > T_2$
End of drift

Figure 2.3: Graphical illustration of a continuous drift.

could affect one operating mode but leave the other operating modes intact. On the contrary, some faults, like global drifts, could affect all the modes.

2.3.3 Characterization of a drift

A drift is an event that occurs in time, having a starting point and an ending point. The starting point of a drift is the time stamp in which the evolution from an old concept to a new concept begins. The ending point of a drift is when a relative stabilization on the new concept is achieved. This evolution can be seen as a dynamical process that must be characterized. Based on our experience, and the works that were done in the literature, four criteria are used to characterize a drift. They are summarized in Table (2.1). This table illustrates the four possible cases of drifts relatively to the overall speed and the overall severity criteria.

Fig. (2.5) shows an example of two scenarios of incremental drifts. In this example, we can see that:

– The beginning time is the same for both drifts: T_b.
– The ending times are $(T_e)_1$ and $(T_e)_2$ such as: $(T_e)_1 \leq (T_e)_2$.
– The durations of drifts are $(\Delta_T)_1$ and $(\Delta_T)_2$ such as: $(\Delta_T)_1 \leq (\Delta_T)_2$.
– The overall speeds of the drifts are $(O_{S_p})_1$ and $(O_{S_p})_2$ such as: $(O_{S_p})_1 \geq (O_{S_p})_2$.
– The overall severities of the drifts are $(O_{S_v})_1$ and $(O_{S_v})_2$ such as: $(O_{S_v})_1 \leq (O_{S_v})_2$.

© 2014 Tous droits réservés.

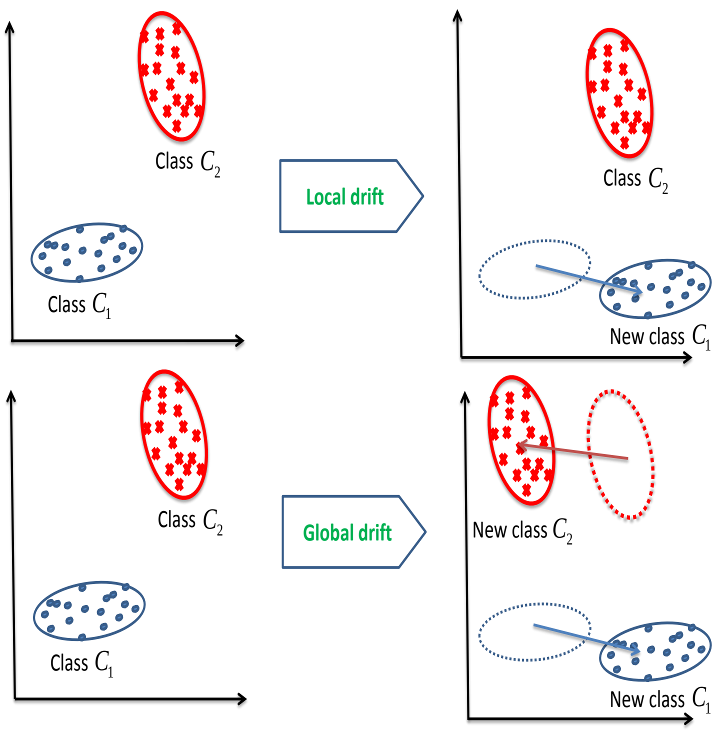


Figure 2.4: Different natures of concept drift; local drifts affect part of the input instance space whereas global drifts affect all this space.

Criterion	Notation	Meaning
Beginning time	T_b	The time stamp of the beginning of a drift.
Ending time	T_e	The time stamp of the ending of a drift. At this point, a relative stability of concepts is achieved.
Duration	Δ_T	The duration of a drift, given by: $\Delta_T = T_e - T_b$.
Overall speed	O_{S_p}	The overall speed of a drift. It could be defined as the inverse of time steps needed for a new concept to completely replace the old concept. It is given by: $O_{S_p} = \frac{1}{T_e - T_b}$.
Overall severity	O_{S_v}	The overall severity of a drift. This criterion is used to indicate the amount of change between concepts. A high severity means a high difference between the old and the new concept whereas a small severity means a low difference. As an example, a migration from a circle $\mathcal{C}(0,r) \rightarrow \mathcal{C}(3,3r)$ is more severe than $\mathcal{C}(0,r) \rightarrow \mathcal{C}(1,2r)$.

Table 2.1: Criteria to characterize a drift.

© 2014 Tous droits réservés.
doc.univ-lille1.fr

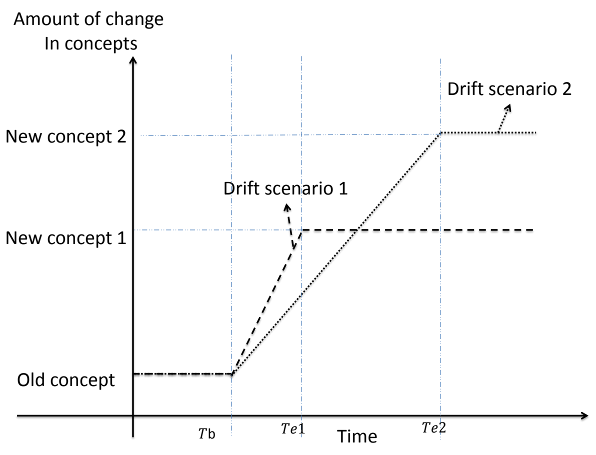


Figure 2.5: Example of two incremental drift scenarios.

Additionally, it is not sufficient to characterize a single drift in time. A more complete characterization must include criteria that classify also sequences of drifts. A sequence of drifts is usually seen on a large period of time. It corresponds to several drift events happening one after another. In order to characterize a sequence of drifts, three criteria are used: predictability, frequency and recurrence. They are listed in Table (2.2) below:

Criterion	Meaning	Possibilities
Predictability	It is an indicator to see if a sequence of drifts follows a certain pattern. It is equivalent to say if we can or not predict the occurrence of a sequence of drifts.	A sequence of drifts is either predictable or unpredictable.
Frequency	It is an indicator on the frequency of a drift over a period of time	A sequence of drifts could be periodic or non-periodic.
Recurrence	As stated before, recurrence is the possibility to return to old concepts.	A sequence of drift is either recurrent or non-recurrent. In case of presence of recurrence, the possibilities are that it can be recurrent in a cyclic way or in an unordered way. For example, power generation using solar panels induces recurrent drifts over a period of one day. This recurrence can be considered cyclic.

Table 2.2: Characterization of a sequence of drifts.

Examples of sequences of drifts are shown in Fig. (2.6). On these examples, we have:
– The upper example shows a sequence that is predictable, periodic and cyclic. It is predictable

© 2014 Tous droits réservés. doc.univ-lille1.fr

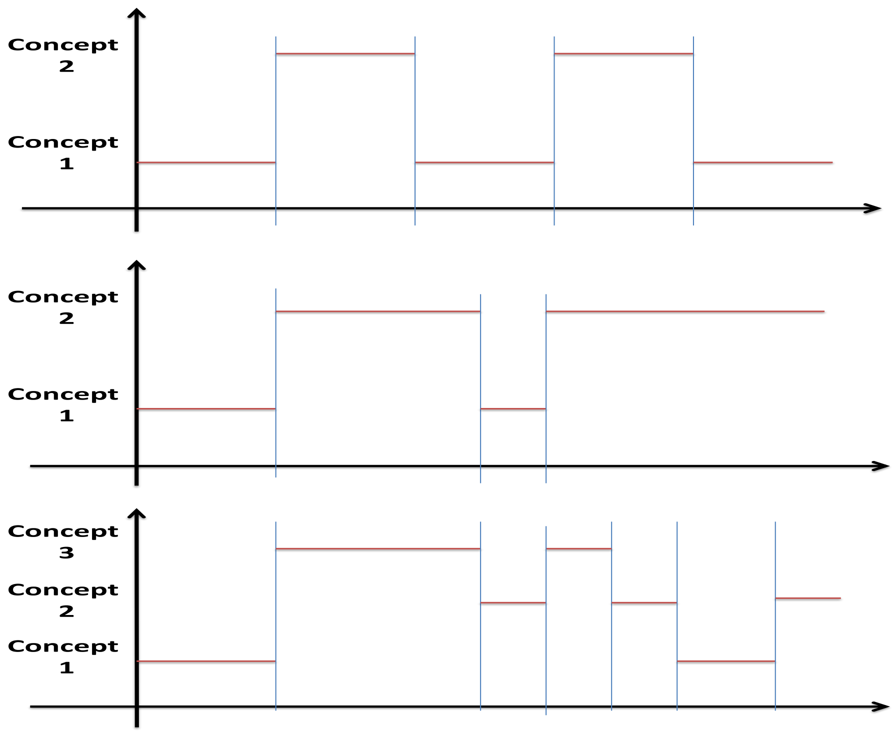


Figure 2.6: Examples of sequences of drift.

since over a period of time, we can observe several drifts. The notion of predictability is also associated to an application in which we may or may not be able to predict subsequent drift occurrences. It is obvious that it is also periodic. Additionally, it is cyclic since it is recurrent in an ordered way. The recurrence goes from concept1 to concept2 repetitively.

– The example in the middle shows a sequence that is predictable, non-periodic and cyclic sequence.
– The lower example shows a sequence that is predictable, non-periodic and recurrent but this time in an unordered manner.

The next section of this chapter gives a synthesized view on the state-of-the-art concerning the approaches for handling concept drift.

2.4 Evolving methods for Handling Concept Drifts

2.4.1 Introduction

Methods for handling concept drift have the general requirements that are related to the streaming nature of the data. They should take into consideration the stability-plasticity dilemma. This is done by adapting the current model using the new data and forgetting the old data, not in a catastrophic manner. They should use only new data and previously used data should not be used anymore in incoming stages once they were processed. The adaptation of the model should be done

© 2014 Tous droits réservés.

in low computational time and memory consumption. Additionally, the robustness to noise is very important to distinguish real change from noisy change.

In this work, modeling is done using machine learning techniques. Conventional machine learning techniques were developed for stationary environments. Later on, an adaptation of the conventional algorithms, and the creation of new ones, were dedicated for the non stationary case, and are called evolving algorithms. Thus, these evolving machine learning techniques have been developped for applications in which training data are available in streams. There are two major challenges concerning the handling of drifts:

– Practically, all the data are generated in a noisy environment and so the learner must be *robust* in order to differentiate between changes caused by noise and changes caused by concept drifting.
– Adapting with limited resources of time and memory. In industrial applications, changes in processes can be very critical and detection of change must be as quick as possible. For example, a leak in a nuclear power plant component must be detected as fast as possible, because the consequences can be catastrophic.

An important aspect to be taken into consideration is the nature of the incoming data. More specifically, learning algorithms depend on whether data arrive labeled or unlabeled, and on which kind of model we want to build on them. To this extent, it seems appropriate to distinguish between the two major typical techniques: classification and regression techniques.

Regression techniques aim at building a model to represent the relations between variables. Contrarily to classification models, regression models provide a quantitative output, and not a qualitative output as a membership to a class. Online versions of regression techniques and times series analysis were developed for streaming data (Kadlec et al., 2011; Ljung, 1999). Some major online recursive regression techniques are the recursive Least-Square (RLS), recursive Auto-Regressive (RAR) and the recursive Auto-Regressive with Moving-Average (RARMA).

In this work, drift modeling is done using classification techniques (regression techniques are used for prognosis as it will be seen later). Two cases have to be considered concerning classification techniques: supervised learning and unsupervised learning. In supervised learning, the incoming data are labeled. In case of concept drift, whether it is a class drift or feature drift, the instantaneous knowledge of the labels provides very useful information to detect concept drifts as we will see later. Formally speaking, we can define:

– M_C a classification model,
– $X_i(t)$: feature vector at time stamp t_i. $X_i \in \mathbb{R}^d$, where d denotes the dimension of the feature space.
– $y(t)$ and $\hat{y}(t)$: the true class label and the predicted class label. The predicted value is given by the classifier. It could be either true for a good classification, or false for a misclassification.
– $\mathbb{C} = \{C_1, C_2, ..., C_n\}$: set of class labels that each pattern could belong to.

© 2014 Tous droits réservés. doc.univ-lille1.fr

There are four possibilities relating a prediction outcome given by the classification model M_C to the real outcome. The terms true positives, true negatives, false positives, and false negatives are used to categorize these results. The terms positive and negative refer to the classifier's prediction, and the terms true and false refer to whether that prediction corresponds to the real outcome. Given a window W on which a classifier model is applied, let TP, TN, FP, and FN be the number of true positive, true negative, false positive and false negative predictions respectively. We define:

– Accuracy is the proportion of true results (both true positives and true negatives) in the population: Accuracy$=\frac{TP+TN}{TP+TN+FP+FN}$.

– Error-rate is defined as the probabilistic complement of the accuracy: Error-rate$=1-$Accuracy$=\frac{FP+FN}{TP+TN+FP+FN}$.
.

Under unsupervised learning, the incoming data are unlabeled. Clustering is used in this case to create **Similar** groups of data points correspond to what is defined as a cluster. Similarity measures are usually distance measures like Euclidean distance, Mahalanobis distance, etc. Online clustering, or evolving clustering, or dynamical clustering operate with streaming data. Three generic stages are applied to each new incoming data: **matching**, **accommodation** and **refinement**. Let us clarify those stages using an example of a dynamical clustering algorithm that operates on each single incoming instance or pattern:

– **Matching**: Comparison of the incoming point to the existing cluster database. This comparison requires a similarity measure.
– **Accommodation**: If the new point is assigned to a winning cluster, then the latter's structure should be updated. In the opposite case, where the new point does not belong to an existing cluster, than a new one should be formed, and possibly maintained in the future.
– **Refinement**: This last step contains mechanisms to enhance the general structure of the entire model. These mechanisms include merging clusters, splitting clusters or removing unrepresentative clusters.

The following of the chapter is directed towards evolving classification methods that should have specific properties, which are related to the general objectives of the thesis. The quest is to find an *ideal* evolving classification method that can be used for building an architecture for supervision, diagnosis and prognosis. For this, it is required that the evolving classification method:

– has the ability to detect a drift,
– gives the possibility to extract meaningful indicators to characterize a drift.

A selected set of evolving classification algorithms is presented as a state-of-the art. The selected algorithms are chosen because they are the closest for our study.

© 2014 Tous droits réservés. doc.univ-lille1.fr

2.4.2 Classification of evolving classification methods

In the last decade, the works on evolving methods that can handle non-stationary environments or concept drifts have known a great deal. One of the earliest works that deals with concept drift dates back to 1996 with Widmer and Kubat (Widmer and Kubat, 1996).

In order to give a synthetic description of the existing methodologies, few questions must be asked. The answers to these questions will allow the classification of the methods used for handling drifts. These questions are:

1. Under which form the data that is processed by the method is fed to it?
2. How does the method adapt the model with continuously incoming data?
3. When does the method adapt the model with continuously incoming data?
4. What was the information used for adaptation?
5. Does the incoming data have to be labeled (supervised learning) or not (unsupervised learning)?

The answers for these questions yield a particular combination of methods. A general graphical illustration of the family of methods for handling drifts is given in Fig. (2.7).

The possible answers for question 1 (Tsymbal, 2004) are:

– Instance by instance: in this case, each incoming instance or pattern is processed at a time. The approach is said to be instance based.
– Batches: in this case, the data is processed by batches or chunks. The approach is said to be batch based.

It is possible to transform streaming batches of data into streaming instances. However, this depends on the application, the speed of arrival of the streaming batches, and on their width or span. It could be too time consuming for example to process instance by instance. The reverse, which is to convert streaming instances to streaming batches, is also possible.

The answer for question 2 is very important to understand how the data is managed by the approach. The possible answers for question 2 are:

– Instance selection - Windowing techniques: these techniques (Widmer and Kubat, 1996; Sayed-Mouchaweh, 2011; Klinkenberg, 2004; Klinkenberg and Renz, 1998) are a natural way to cope with evolving data since only recent instances are kept. They are grouped inside a window, and the learner builds its model only using this recent window. Obviously, the size of the window is of an important deal. It reflects somehow the stability-plasticity dilemma. A large window increases stability since we can be surer that the concept is fully represented. However, it decreases plasticity and makes the model less reactive and slowly adaptive. From the point of view of the size of a window, we can distinguish between:

  1. Fixed size window: a sliding window of fixed size containing the latest examples (Widmer and Kubat, 1996).

© 2014 Tous droits réservés. doc.univ-lille1.fr

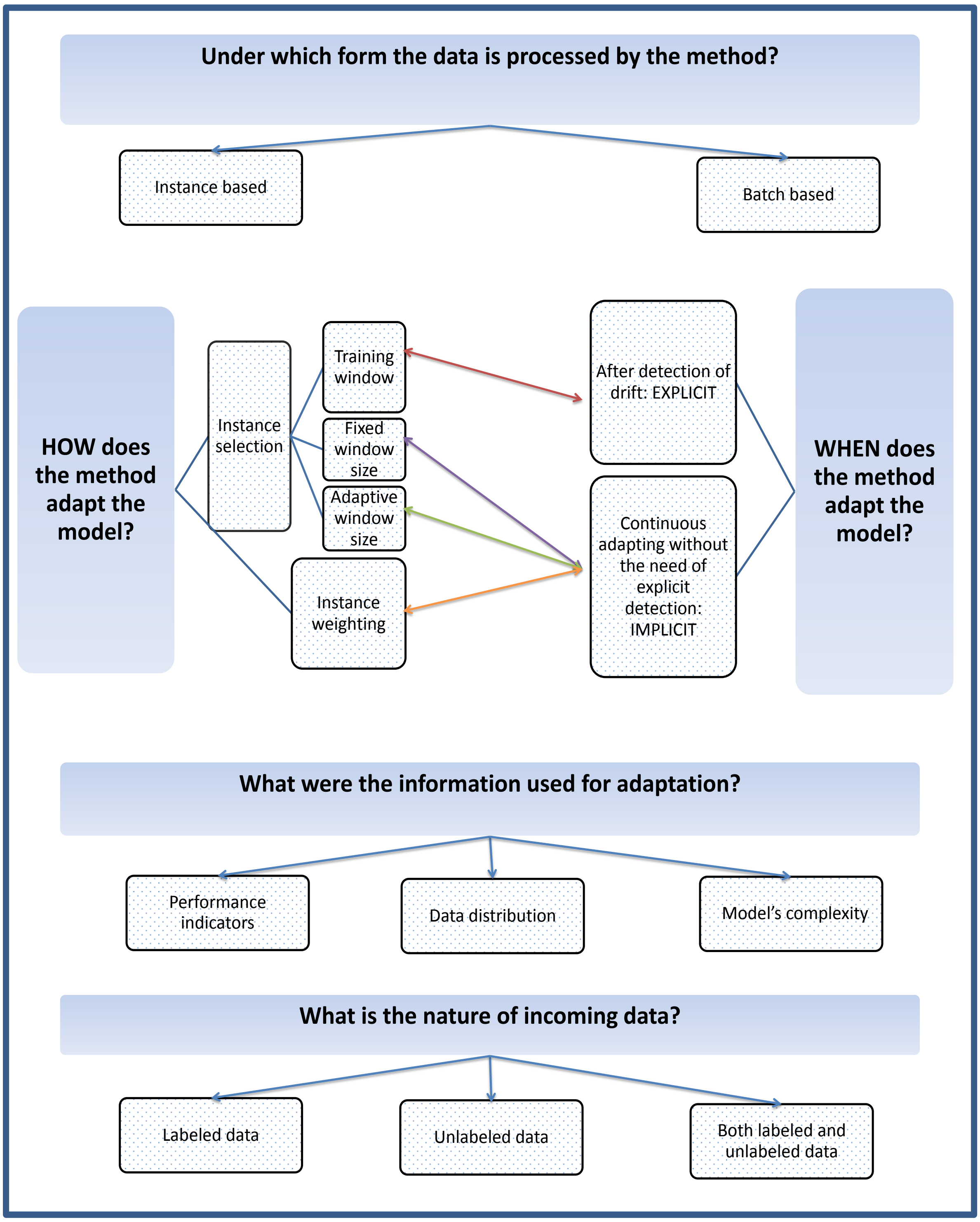

Under which form the data is processed by the method?
Instance based
Batch based
HOW does the method adapt the model?
Instance selection
Training window
Fixed window size
Adaptive window size
Instance weighting
After detection of drift: EXPLICIT
Continuous adapting without the need of explicit detection: IMPLICIT
WHEN does the method adapt the model?
What were the information used for adaptation?
Performance indicators
Data distribution
Model's complexity
What is the nature of incoming data?
Labeled data
Unlabeled data
Both labeled and unlabeled data

Figure 2.7: Taxonomy of methods that handle concept drifts.

© 2014 Tous droits réservés. doc.univ-lille1.fr

2. Adaptive size window: some methods change the size of the window when a concept drift is expected. The window's size grows when no concept drift is suspected and it shrinks whenever it is. The logic behind it is that when the concepts remain stationary, it is more useful to grow the window for better generalization (increasing stability). When concept drift occurs, shrinking the window size makes the model more reactive (increasing plasticity). However, making it too big can decrease the ability of the model to detect slight drifts (Sayed-Mouchaweh, 2011; Klinkenberg and Renz, 1998).
3. Training window: it is the construction of windows used by learners to retrain a model upon drift detection. The idea of training windows will be clearer once the third question is answered (Gama et al., 2004; Gama and Castillo, 2006).

– Instance weighting - Gradual forgetting: Another way of forgetting old knowledge and incorporating new knowledge is done by instance weighting (Klinkenberg, 2004). Decay functions are used to give less importance to older patterns and more importance to newer patterns inside a window. This weighting mechanism has the benefit to avoid a lot of data storage since only the latest example is needed. In (Cohena and Strauss, 2006), different decay functions were analyzed: exponential decay function w_{exp}, polynomial decay function w_{poly} and chordal decay function w_{chor}. Let W be a window of examples, $|W|$ its cardinality and λ a forgetting factor. These functions are:

$$w_{exp}(t) = \exp(-\lambda t),\ \lambda \geq 0 \tag{2.2}$$

$$w_{poly}(t) = \frac{1}{t^{\lambda}},\ \lambda \geq 0 \tag{2.3}$$

$$w_{chor}(t) = 1 - \frac{t}{|W|} \tag{2.4}$$

The question 3 (When does the method adapt the model with continuously incoming data?) corresponds to the temporal event of adaptation. The possible answers for question 3 are:

– Always: a lot of approaches for handling drifts are based on a continuous adaptation of the model. These approaches do not wait for an explicit detection of drift to update the model. They are given the name **implicit** methods (Hulten et al., 2001; Widmer and Kubat, 1996; Bifet and Gavalda, 2007; Black and Hickey, 1999; Cauwenberghs and Poggio, 2000; Klinkenberg and Renz, 1998).

– After detection of drift: these approaches update the system only after an **explicit** detection of drift. The model is reset upon detection, and rebuilt from scratch. The signal for drift detection is called the **trigger** and these methods are named **explicit** methods. They are associated with change detecting mechanisms, including statistical testing, control charts, fixed and adaptive thresholds (Basseville and Nikiforov, 1993; Alippi et al., 2009; Ikonomovska et al., 2009; Alippi and Roveri, 2008; Kuncheva, 2009; Lu et al., 2010; Ganti et al., 2002; Kifer et al., 2004; Ikonomovska et al., 2011; Dries and Ruckert, 2009; Sobhani and Beigy, 2011; Nishida and Yamauchi, 2007; Gama and Castillo, 2006).

At this point, it is interesting to comment the colored arrows in Fig. (2.7). These arrows relate the 'how' an approach adapt the model to the 'when' it does so. For implicit methods, the windowing techniques, including the fixed and the adaptive sized windows, are used. The learner is continuously

© 2014 Tous droits réservés. doc.univ-lille1.fr

adapted. However, explicit methods reset the system upon detection. Thus, they have to rebuild considering a training window.

Whether the method is implicit or explicit, it makes use of information for its adaptation strategy. For instance, for implicit methods, and as it was stated earlier, adaptive windowing techniques adjust the sizes of windows for adaptation purposes. To achieve this aim, they use information on the model. Additionally, explicit techniques have to use information and drift detectors to achieve adaptation. Thus, the possible information to be used corresponds to the answers of question 4 (What were the information used for adaptation?) and could be:

– Performance indicators from a classifier: a lot of methods use this criterion to evaluate the need for adjusting the window size, or the need to reset the system and re-learn (Gama et al., 2004; Baena-Garcia et al., 2006; Nishida and Yamauchi, 2007).
– Distribution of data in the feature space: this information could be used to detect variation in the distribution of the underlying data. They also make use of statistical testing (Ditzler and Polikar, 2011; Markou and Singh, 2003; Salganicoff, 1993).
– Evolution of the learner's complexity: as models continuously evolve, the parameters of the learners also evolve. These evolving parameters can be interesting for assessing the amount of change. In some cases, not only the parameters of a model change, but also its structure. This is the case for example in evolving clustering or dynamical clustering, where new clusters can appear, and other clusters can disappear (Su et al., 2008).

Finally, the nature of the data that an approach can handle is very application specific (question 5: what is the nature of the incoming data?). The possible answers for question 5 are:

– Labeled data: in this case, the instantaneous labeling of the data is available. This case calls for supervised learning techniques.
– Unlabeled data: in this case, the labeling is not straightforward. It could possibly be obtained after some time, or maybe never obtained. This case calls for unsupervised learning techniques.

In the next section, few methodologies taken from the literature are developed.

2.4.3 Algorithms for handling drifts

As stated earlier, few algorithms are chosen to be shown because they have properties that are close to those that are needed. This selection of algorithms falls into two categories. A first category of methods containing drift detection mechanisms is listed. Another category of implicit methods is also listed. Each listed category of methods is later followed by an analysis of its weak points and strong points. These analyses highlight interesting ideas for our work, and point out the directions of research that still have to be done.

The algorithms to be presented cover also different categories of methods from Fig. (2.7). Each algorithm developed in this section is given with a sort of identity card containing answers for the five questions introduced above. This allows positioning the algorithm inside the general scheme presented in Fig. (2.7).

© 2014 Tous droits réservés. doc.univ-lille1.fr

2.4.3.1 Algorithms with a drift detection mechanism

The first category of algorithms belong to the explicit family of algorithms. They contain explicit drift detection mechanism, which constitutes the basis of model adaptation. The algorithms to be considered are: (1) DDM, (2) EDDM, (3) STEPD, (4) CCDD and (5) HDDDM.

The first algorithm to be presented is called the DDM algorithm.

DDM: Drift Detection Method:

The DDM algorithm is presented in the Table (2.3). The logic behind this method is that when a

Name of the algorithm	DDM
Reference	(Gama et al., 2004)
How is the data processed?	Instance based
How is the adaptation done?	Training windows
When is the adaptation done?	Explicit method
What is the information used?	Classification accuracy Change detection: definition of warning and drift thresholds
What is the nature of the data?	Labeled

Table 2.3: DDM algorithm.

stable concept is being learnt, the error-rate of the classifier tends to be small and stay small, thus the accuracy of the classifier stays high. On each iteration, a new pattern is classified by a classifier (any classifier could be used). The result of the prediction of the classifier can be either *true* or *false*. Thus, for a set of examples, the error is a random variable from a Bernoulli trial. The number of classification errors is then modeled by a random variable with Binomial distribution. Let us denote the error rate at time stamp i by p_i and its standard deviation by s_i. At each time stamp i, we have:

$$p_i = p_{i-1} - \frac{p_{i-1}}{i}, \text{ if the prediction is true,} \tag{2.5}$$

$$p_i = p_{i-1} + \frac{1-p_{i-1}}{i}, \text{ if the prediction is false,} \tag{2.6}$$

$$s_i = \sqrt{\frac{p_i(1-p_i)}{i}}. \tag{2.7}$$

From Eq. (2.7), it can be seen that the error rate tends to 0 if only true predictions are done, and to 1 if only false predictions are done. For a sufficiently large number of examples n $(n > 30)$, the Binomial distribution is closely approximated by a Normal distribution with the same mean and variance. Two registers are kept in memory: p_{min} and s_{min}. They correspond respectively to the minimal value of the error rate and the minimal value of its standard deviation. These values are used to calculate a warning level α (see Eq. (2.8)) and a drift level β (see Eq. (2.9)). Each time a warning level is reached, examples are memorized in a separate window, to form a training window.

© 2014 Tous droits réservés. doc.univ-lille1.fr

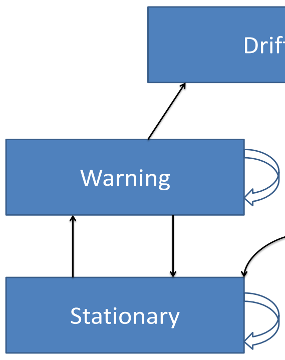


Figure 2.8: DDM algorithm state space transition (Gama and Castillo, 2006).

If afterwards the error rate falls again below the warning threshold, the warning is treated as a false alarm and the separate window is dropped. The condition to declare warning is:

$$W_i \ : \ p_i + s_i \geq p_{min} + \alpha s_{min}, \tag{2.8}$$

and the condition to declare drift is:

$$Dr_i \ : \ p_i + s_i \geq p_{min} + \beta s_{min}, \text{ with } \beta \geq \alpha. \tag{2.9}$$

The Figure 2.8 illustrates the transition steps of the DDM algorithm. The values α and β in the above conditions decide about the confidence levels at which the warning and alarm signals are triggered. The authors propose $\alpha = 2$ and $\beta = 3$ which approximately correspond to 95% and 99% confidence levels (the real values are 1.96 and 2.57 respectively) for warning and drift respectively.

The inconvenience of this method is its low adaptability to local concept drift. In fact, this method tends to relearn the algorithm from scratch when a drift has begun. In local concept drift cases, this is not useful since part of the instance space is still stable. In addition, it was shown, and we will see it later on, that this method is not very effective with slow drifts.

EDDM: Early Drift Detection Algorithm:

The authors in (Baena-Garcia et al., 2006) changed the algorithm DDM described just above by considering the distance between two consecutive false predictions or miss-classifications. The distance between two miss-classifications is the number of time stamps separating them. The logic behind their method is, when a concept is being learned, the time stamps elapsed between two consecutive false predictions should increase and stay high until drift occurs. For indicators, the authors considered the average distance between two miss-classification, which is denoted p_i, and its standard deviation, denoted s_i. The EDDM is called early drift detection method because this method is more suited to detect slow gradual drifts. A register is used to store a value that corresponds to the point where the distribution of distances between errors is maximum. This point corresponds to the value of $p_i + 2s_i$ reaching its maximum, obtaining p_{max} and s_{max}. Given that the warning

© 2014 Tous droits réservés. doc.univ-lille1.fr

Name of the algorithm	EDDM
Reference	(Baena-Garcia et al., 2006)
How is the data processed?	Instance based
How is the adaptation done?	Training windows
When is the adaptation done?	Explicit method
What is the information used?	Classification accuracy Change detection: definition of warning and drift thresholds
What is the nature of the data?	Labeled

Table 2.4: EDDM algorithm.

level is α and the drift level is β, for EDDM $\alpha > \beta$, since the average distance p_i is an indicator that decreases when drift occurs. The authors argued that the drift level and the warning level are triggered if:

- $\frac{p_i+2s_i}{p_{max}+2s_{max}} \leq \alpha$ a warning is triggered. Examples are stored in a separate training window.
- $\frac{p_i+2s_i}{p_{max}+2s_{max}} \leq \beta$ the drift is confirmed. The training window that was saved serves to re-learn the model from scratch.

α and β are the sensitivity values of the algorithms. Increasing the values of α and β induces faster detection of slow drifts but bigger number of false alarms whereas small values induce slower detection but bigger robustness to false alarms. Finally, we note that the training windows used to re-learn the model should contain at least 30 consecutive errors. Thus, EDDM starts detecting drifts after at least 30 consecutive errors have occurred. They chose 30 errors to calculate the values p_{max} and s_{max}. This choice is backed by the fact that statistical estimation is considered to be acceptable with at least 30 examples.

STEPD: Statistical Tests of Equal Proportions for Detection:

Name of the algorithm	STEPD
Reference	(Nishida and Yamauchi, 2007)
How is the data processed?	Instance based
How is the adaptation done?	Training windows
When is the adaptation done?	Explicit method
What is the information used?	Overall accuracy versus local accuracy Change detection: statistical testing
What is the nature of the data?	Labeled

Table 2.5: STEPD algorithm

In (Nishida and Yamauchi, 2007), the authors proposed an algorithm that makes use of the accuracy metric in two different ways. The idea is to consider two values: the recent accuracy and

© 2014 Tous droits réservés.

doc.univ-lille1.fr

the overall accuracy. The recent accuracy is the accuracy (see 2.4.1 for remind) of the classifier calculated on the most recent window containing W examples (size W). The overall accuracy is calculated from the beginning of the learning. Given Acc_o, Acc_r, W_o and W as the overall accuracy, recent accuracy, overall number of examples and recent examples respectively, they calculated the following statistic (p-value):

$$t_s = \frac{\mid \frac{Acc_o}{W_o} - \frac{Acc_r}{W} \mid -0.5(\frac{1}{W_o} + \frac{1}{W})}{\sqrt{P(1-P)(\frac{1}{W_o} + \frac{1}{W})}} \tag{2.10}$$

where P is given by:

$$P = \frac{Acc_o + Acc_r}{W_o + W}. \tag{2.11}$$

In this algorithm, there are also two levels to consider, the warning level (α) and the drift level (β). The calculated statistic is compared to the values of α (warning level) and β (drift level). Two cases are considered:

- If $\alpha \leq t_s \leq \beta$, the warning level is declared.
- If $t_s \geq \beta$, the drift level is considered.

Instances are stored when $\alpha \leq t_s \leq \beta$ (warning level is reached) and the classification model is rebuilt starting from these stored examples if $t_s \geq \beta$. The sored examples are removed if $t_s \leq \alpha$.

CCDD: Control Charts for Drift Detection:

Name of the algorithm	CCDD
Reference	(Kuncheva, 2009)
How is the data processed?	Batch based
How is the adaptation done?	Training windows
When is the adaptation done?	Explicit method
What is the information used?	Classifier's accuracy Change detection: control charts
What is the nature of the data?	Labeled

Table 2.6: CCDD algorithm.

Note that the authors in (Kuncheva, 2009) did not name their algorithm CCDD. This terminology has been adopted inside this thesis.

Control charts have always been used to monitor process quality. They are designed to help the operator knowing the actual state of a process. A state can be *out-of-control* or *in-control*, in proper process monitoring terminology. The *out-of-control* state is defined as the state where technical intervention to enhance the quality of products is required. The *in-control* state is declared when this intervention is not required.

© 2014 Tous droits réservés. doc.univ-lille1.fr

The data required to construct a control chart are data corresponding to in-control behavior. They are measurements taken on a process when it was sure that it was working correctly.

In (Kuncheva, 2009), two control charts were used as change detectors: Shewhart and CUSUM charts. The input data to their algorithm is considered to be batches B_i (arriving at time stamps t_i) containing W streaming values of 0 and 1. Each of these values is denoted $E_{i,j}$, $j = 1, \ldots, W$ such as a batch B_i contains W values. The value $E_{i,j} = 0$ corresponds to a good classification (real outcome is the same as the outcome predicted by the classifier) and the value $E_{i,j} = 1$ to a bad classification (real outcome is not the same as the outcome predicted by the classifier).

The idea of the Shewhart detection algorithm is to consider objects with probability p^* of an object being defective and with standard deviation σ^*. These values are considered to be known also, by experimenting or by given specifications (in-control behavior). Applied online, sequentially taken batches B_i of size W are inspected and the number of defective objects in these batches allows the estimation of $\hat{p}_i$. This is done by a simple division of the number of defective objects inside the batch B_i by W. If $\hat{p}_i \geq p^* + \gamma\sigma$ where γ is called the σ-limit (typically 3), a change is detected. γ can also be seen as a sensitivity parameter.

The CUSUM chart keeps the cumulative sum of errors, denoted $S(t_i)$, and cumulates its value from batch to batch. Then, a change detection test is applied on the signal $S(t_i)$. Given a batch B_i whose size is W, for detection of a significant **increase** of the error level, $S(t_i)$ is calculated by:

$$S(t_i) = max(0, S(t_{i-1})) + \sum_{i=1}^{W}(E_i - W\gamma), \tag{2.12}$$

where $\gamma = \frac{r_1}{r_2}$, $r_1 = -\ln\frac{1-p^*}{1-p}$, $r_2 = \ln\frac{p^*(1-p)}{p(1-p^*)}$ and p^* is the probability of defective objects. $S(t_i)$ is compared with a control limit h, and if $S(t_i) \geq h$, then alarm is launched. For detection of significant **decrease**, the *max* in Eq. (2.12) is replaced by *min* and the new equation becomes:

$$S(t_i) = min(0, S(t_{i-1})) + \sum_{i=1}^{W}(E_i - W\gamma), \tag{2.13}$$

The decision to launch a concept drift alarm becomes $S(t_i) \leq h$. Once the drift is confirmed (the alarm is launched), $S(t_i)$ is reinitialized.

HDDDM: Hellinger Distance based Drift Detection Method:

In (Ditzler and Polikar, 2011), the HDDDM (Hellinger Distance Drift Detection Method) algorithm is introduced. In probability and statistics, the Hellinger distance is used to quantify the similarity between two probability distributions. Let $P_{X^j}(t_i)$ and $Q_{X^j}(t_i)$ denote two continuous probability distributions with respect to a random variable X^j (we remind that X^j is the j^{th} feature of a feature

© 2014 Tous droits réservés. doc.univ-lille1.fr

Name of the algorithm	HDDDM
Reference	(Ditzler and Polikar, 2011)
How is the data processed?	Batch based
How is the adaptation done?	Training windows
When is the adaptation done?	Explicit method
What is the information used?	Monitoring the distribution of data in the input space Change detection: adaptive threshold
What is the nature of the data?	Labeled

Table 2.7: HDDDM algorithm.

vector X). The Hellinger distance between $P_{X^j}(t_i)$ and $Q_{X^j}(t_i)$ is defined as the quantity:

$$H(P_{X^j}(t_i), Q_{X^j}(t_i)) = \sqrt{\frac{1}{2}\int (\sqrt{\frac{dP_{X^j}(t_i)}{dX^j}} - \sqrt{\frac{dQ_{X^j}(t_i)}{dX^j}})^2 dX^j} \tag{2.14}$$

The Hellinger distance is a symmetric metric, and it is bounded:

$$0 \leq H(P_{X^j}(t_i), Q_{X^j}(t_i)) = H(Q_{X^j}(t_i), P_{X^j}(t_i)) \leq 1 \tag{2.15}$$

This metric can be computed between two histograms with the same number of bins. In (Ditzler and Polikar, 2011), batches of the same size W were considered and the number of bins B was taken as $B = \sqrt{W}$. With a modification to Eq. (2.14), the Hellinger distance computed on histograms for one feature is given by:

$$H(P_{X^j}(t_i), Q_{X^j}(t_i)) = \sqrt{\sum_{b=1}^{B} (\sqrt{f_P(b)} - \sqrt{f_Q(b)})^2} \tag{2.16}$$

where $f_P(b)$ and $f_Q(b)$ are the frequency counts of $P_{X^j}(t_i)$ and $Q_{X^j}(t_i)$ respectively in bin number b. For generalization purposes, the authors in (Ditzler and Polikar, 2011) averaged the Hellinger distance of all the features. Since there are d features, the authors used the metric $\delta_H(t)$ defined by:

$$\delta_H(t_i) = \frac{1}{d}\sum_{j=1}^{d} H_j(P_{X^j}(t_i), Q_{X^j}(t_i)) \tag{2.17}$$

where $H_j(P_{X^j}(t_i), Q_{X^j}(t_i))$ is the Hellinger distance calculated for the feature X^j. The idea was to consider a baseline batch to which the incoming batches are compared. The Hellinger distance was computed between batch $D_{t_{i-1}}$ and D_{t_i} and the error $\epsilon(t_i) = \delta_H(t_i) - \delta_H(t_{i-1})$ is computed. When this error exceeds a threshold $\beta(t_i)$, a drift was signaled and the classifier they used (Naive Bayes) was reset then retrained in their experimental set-ups.

The threshold $\beta(t_i)$ is a function of time because it is an adaptive threshold, automatically updated with time. In order to automatically update the threshold, they considered the average error of distances and their standard deviation until a change was detected. Let T_d the time where change

© 2014 Tous droits réservés. doc.univ-lille1.fr

was detected. They computed recursively $\hat{\epsilon}(t_i)$, the mean of $\mid \epsilon(t_i) \mid$ and the standard deviation $\hat{\sigma}(t_i)$ for t_i, $i = 1, 2, ..., d-1$. The formulas for recursive update are going to be detailed in few subsections later. At each time stamp, $\beta(t_i)$ is adjusted by the formula $\beta(t_i) = \hat{\epsilon}(t_i) + \gamma\hat{\sigma}(t_i)$, where γ is the sensitivity parameter. When $i = d$, the change was signaled and the error is reset. The baseline distribution becomes D_{T_d}.

The next step is to show the list of algoritms with implicit adaptation mechanisms.

2.4.3.2 Algorithms with implicit adaptation mechanisms

The algorithms to be shown are: (1) OLINDDA and (2) AuDyC.

OLINDDA: OnLIne Novelty and Drift Detection Algorithm:

Name of the algorithm	OLINDDA
Reference	(Spinosa et al., 2007)
How is the data processed?	Instance based
How is the adaptation done?	Instance selection - Fixed window size
When is the adaptation done?	Implicit method
What is the information used?	Similarity measures
What is the nature of the data?	Unlabeled

Table 2.8: OLINDA algorithm.

In (Spinosa et al., 2007), the authors developed a cluster-based approach for detecting novelty and concept drift in data streams. All the methods considered until now did not take into consideration the possible appearance of new concepts.

The OLINDDA algorithm stands for OnLIne Novelty and Drift Detection Algorithm. Given an intial data set labeled as normal, it builds an initial model, which represents the normal concept, using the k-means clustering algorithm. Given a set of examples of the normal concept and the number of clusters k, k-means returns k clusters and the positions of their centroids. Then, for each cluster, the maximum distance (Euclidean distance, which is the similarity measure) between examples and the centroid is calculated, i.e., the distance between the centroid and the example from which it is farthest. This allows the establishment of a decision boundary for each cluster. The union of the boundaries of all clusters is the global decision boundary which defines the model.

A new unseen example that falls inside this global boundary is consistent with the model and therefore considered normal; otherwise, it is labeled unknown and separated for further analysis. The possibilities are:

– These patterns belong to a new unidentified concept. This case is validated if the distance of these new points are considered as 'far enough' from existing concepts.

© 2014 Tous droits réservés. doc.univ-lille1.fr

– These examples are due to concept drift. This case is validated if the new appearing clusters are 'close enough' to already existing clusters.
– These examples are due to noise. This possibility is omitted by defining a minimum number of instances inside a concept. This threshold will make sure no noisy concepts are kept.

More information on how the thresholds for distinguishing between a concept drift and the formation of a new concept can be found in (Spinosa et al., 2007). OLINDDA periodically updates the model to ensure that it represents the current definition of the normal concept. In order to do that, it maintains examples of validated clusters that have been identified as concept drift in a short-term memory of normal data. The update is performed whenever a certain number of examples, defined by the user, are waiting in that memory. By incorporating new examples to the model, the technique is capable of learning in an online manner, which is needed for its application in data streams.

AuDyC: Auto adaptive Dynamical Clustering

Name of the algorithm	AuDyC
Reference	(Lecoeuche and Lurette, 2003; Boubacar et al., 2005)
How is the data processed?	Instance based
How is the adaptation done?	Instance selection - Fixed window size
When is the adaptation done?	Implicit method
What is the information used?	Parameters of the evolving model
What is the nature of the data?	Unlabeled

Table 2.9: AuDyC algorithm.

AuDyC is an online algorithm that stands for Auto-Adaptive Dynamical Clustering. It uses a technique that is based on a mixture of Gaussian models. The features are modeled using Gaussian *prototypes*, or clusters, characterized by a center and a covariance matrix. Each prototype (cluster) can be composed of several Gaussian prototypes, forming one class. AuDyC is adaptive by nature. This makes it a valuable choice to handle drifts

AuDyC models a cloud of patterns by one or several Gaussian prototypes. Each of the Gaussian prototypes P_j is characterized by a center μ_{P_j} and a covariance matrix Σ_{P_j}. By misuse of language, a prototype P_j can thus be mathematically written as $P_j = (\mu_{P_j}, \Sigma_{P_j})$.

The PDF of a Gaussian prototype, denoted $G(X, P_j)$ is given by:

$$G(X,P_j) = \frac{1}{(2\pi)^{\frac{d}{2}} det(\Sigma_{P_j})^{\frac{1}{2}}} exp(-\frac{1}{2}(X-\mu_{P_j})^T \Sigma_{P_j}^{-1}(X-\mu_{P_j})) \tag{2.18}$$

Learning using AuDyC is done according to four procedures that match the general requirements of online unsupervised learning defined in the beginning of the section. The matching is done according to similarity criteria, based on the membership of the sample X_i to existing P_j. Refinement is

© 2014 Tous droits réservés. doc.univ-lille1.fr

done according to three different procedures. These procedures are: class creation, class adaptation and class merging. Finally, an evaluation phase is done, in which an analysis of the structures of the created classes is done. This analysis allows getting rid of parasite classes, or more generally non representative classes.

The functionalities of AuDyC will be studied very precisely later in the incoming subsections. The reason is that AuDyC is chosen to be as the best algorithm candidate for our studies. The analysis leading to this conclusion is shown next.

2.4.4 Discussion, analysis and research orientation

Let us remind that the idea is not to make an extensive review on algorithms on handling drifts, but to show the ones that can be efficiently applied for the drift detection. In fact, the major strong point of the first category of methods presented above is that they are equipped with change detection mechanisms. Different explicit drift detection methodologies were shown: fixed threshold, adaptive threshold, statistical testing, and control charts.

Additional successful points of these methodologies are their fast reactivity to drifts. More particularly, in (Dries and Ruckert, 2009), the DDM, EDDM and STEPD algorithms were compared on artificial datasets. The results showed that the performance of STEPD and EDDM were comparable for slow drifts, but for sudden drifts, STEPD outperformed EDDM. They both outperformed DDM in sudden and gradual drifting cases.

The major drawback of these methodologies is that they can only treat labeled data. Concerning industrial processes, having instantly labeled data is not possible. Another major drawback is the high sensitivity of these models to false alarms, since it is necessary to rebuild the model if the alarm is triggered.

The major strong point of the second list of algorithms (dynamical clustering techniques) is their ability to work in an unsupervised manner. They have the ability to recognize drifting concepts from the appearance of new one. In order to be able to create an architecture for supervision, diagnosis and prognosis of industrial systems, unlabeled data are gathered online from sensor networks.

OLINDDA, AuDyC and other algorithms like SAKM (Self-Adaptive Kernels Machine) (Boubacar, 2005), CDL (Clustering and Labelling Detection) (Eltoft, 1998), which were not presented in this section, are all adequate for treating unlabeled data, and could constitute a possible algorithm that can be used for our purposes. However, amongst all these algorithms, AuDyC seems to be the most adequate as it has proven interesting abilities in previous experiments (Boubacar et al., 2005; Boubacar, 2005). In fact, AuDyC uses Gaussian prototypes to model patterns in the feature space. Gaussian prototypes have nice qualities that can be used to create indicators for health assessment and monitoring. At the same time, it is not a restrictive option since data are usually provided by sensors that are corrupted with Gaussian noise. Also, with an adequate signal processing, Gaussian

© 2014 Tous droits réservés. doc.univ-lille1.fr

attributes can be extracted from data to form adequate features to be trated by AuDyC.

However, all these algorithms, including AuDyC, do not have a detection mechanism. This constitutes a major drawback. At the same time, there is a need to develop a mechanism for the extraction of meaningful indicators that can help us achieve the goals that were set in the introduction of this section. This triggers future research in this direction in the sense that equipping AuDyC with functionalities as drift detection and characterization (for meaningful indicator extraction), as well as prognosis model constitute the main contributions of the work in this thesis. These contributions are tackled in chapters 3 and 4.

The rest of this chapter is dedicated to formally present AuDyC, its structure and its functionalities.

2.5 AuDyC algorithm

This section is dedicated to the AuDyC algorithm. The structure of AuDyC, and its learning mechanisms are presented hereafter.

2.5.1 Structure of AuDyC

AuDyC is a dynamical clustering algorithm that has a neural network structure (*see* Fig. (2.9)). It has three layers:

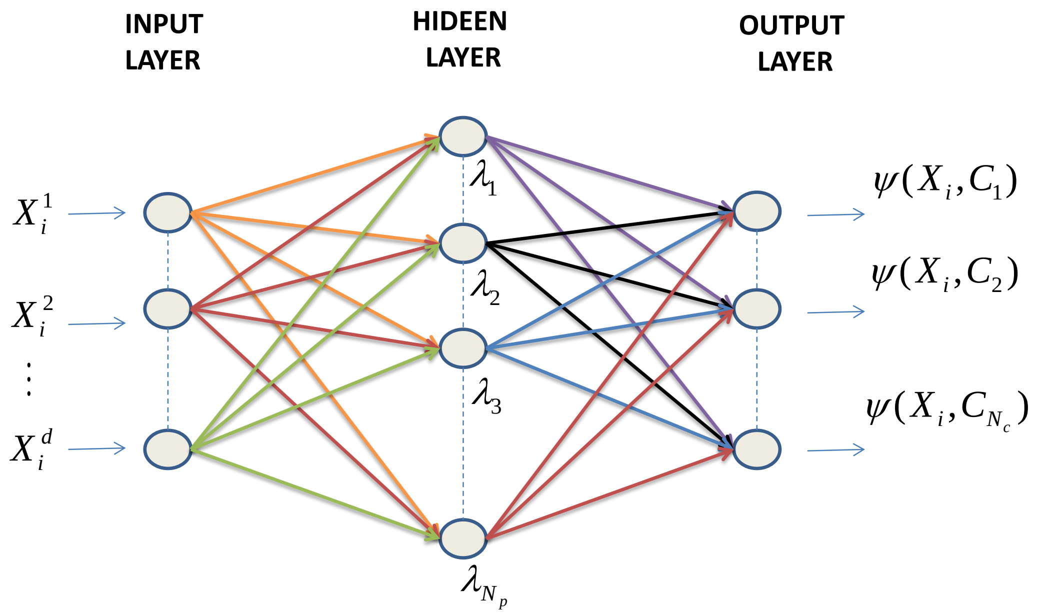


Figure 2.9: Neural structure of AuDyC.

1. Input layer: given that the input at time t_i is the feature vector X_i, the input layer contains as many neurons as components of this vector. If d is the dimensionality of the feature space,

© 2014 Tous droits réservés. doc.univ-lille1.fr

then $X_i = (X_i^1, X_i^2, \ldots, X_i^d)^T$, and the input layer contains d neurons.

2. Hidden layer: it contains as many neurons as the number of existing prototypes (or clusters). This number is denoted N_p (such that $j = 1, \ldots, N_p$). The activation function of each of these neurons is the membership degree of the input X_i to each of these prototypes. The membership degree of a feature vector to a prototype, denoted $\lambda(X_i, P_j)$ is also a similarity measure of this vector to this prototype. Let $d_M(X_i, P_j)$ be the Mahalanobis distance between the incoming feature vector X_i and a prototype P_j, given by:

$$d(X_i, P_j) = \sqrt{(X_i - \mu_{P_j})^T)\Sigma_{P_j}^{-1}(X_i - \mu_{P_j})}. \tag{2.19}$$

   The similarity measure $\lambda(X_i, P_j)$ is given by:

$$\lambda(X_i, P_j) = \lambda_j = exp(-\frac{1}{2}(d(X_i, P_j))^2). \tag{2.20}$$

   From Eq. (2.20), it is clear that the more the feature vector is distant from the cluster P_j, the lower its membership degree to this prototype is.

3. The output layer: it contains as many neurons as the number of classes. This number is denoted N_c. As stated earlier, a class C_k $(k = 1, \ldots, N_c)$ can contain several prototypes. The synapses relating the neurons of the hidden layer to the neurons of the output layer define the membership of the clusters to the classes. The values of these connections are saved in a binary matrix W_{kj}^C, whose dimension is $N_c \times N_p$. The association of a cluster P_j to a class C_k yields $W_{kj} = 1$, else $W_{kj} = 0$. Finally the value of the output is the membership degree of the input feature vector X_i to a class C_k. This membership function is denoted $\psi(X_i, C_k)$ and can be calculated by:

$$\psi(X_i, C_k) = min(1, \sum_{P_j \in C_k} \lambda(X_i, P_j)). \tag{2.21}$$

By applying AuDyC, several prototypes, which correspond also to clusters, can form one class. Thus, a class is a set of clusters or prototypes that groups similar data instances. AuDyC contains four essentials functionalities: class creation, evolution, merging and elimination. These different functionalities, along with the similarity measures are developed in the next subsections.

2.5.2 Learning with AuDyC

In order to classify the new incoming feature vector X_i, AuDyC calculates its similarity measure, computed with respect to all the existing prototypes. Each similarity measure is compared to two thresholds λ_m and λ_M. The threshold λ_m can be interpreted as the minimal threshold for a feature vector to belong to a class. The threshold λ_M constitutes the minimal threshold for a feature vector to belong to a prototype inside a class. λ_m defines the boundary of the class whereas λ_M defines the boundary of a prototype. The Table (2.10) shows the different learning rules that are used by AuDyC. The subsections below will explain each of these rules.

© 2014 Tous droits réservés. doc.univ-lille1.fr

	If	Then
1	$\exists P_j \in C_k,\ \lambda_M \leq \lambda(X_i, P_j)$	Adaptation of P_j using Eq. (2.22) or Eq. (2.23)
2	$\exists P_j \in C_k,\ \lambda_m \leq \lambda(X_i, P_j) \leq \lambda_M$	Creation of a new prototype $P_{new} \in C_k$
3	$\forall j,\ \lambda(X_i, P_j) \leq \lambda_m$	Creation of new prototype and a new class
4	$\exists P_{j1} \in C_{k_1} \cup \exists P_{j2} \in C_{k_2},\ \lambda_m \leq \lambda(X_i, P_{j1})$ and $\lambda_m \leq \lambda(X_i, P_{j2}) \leq \lambda_M$	Ambiguity $\rightarrow X_i \in X_{amb} \rightarrow$ possibility of using a merging mechanism

Table 2.10: AuDyC learning rules.

2.5.2.1 Adaptation rules

If the membership degree $\lambda(X_i, P_j)$ of a new incoming feature vector X_i to a prototype P_j is larger than λ_M (case 1 in the Table (2.10)), then the feature vector is affected to the prototype P_j. In this case, AuDyC updates the parameters of the prototype P_j, *i.e.* the mean and the covariance matrix. In fact, AuDyC does the update by forgetting the last feature vector of the prototype and by accepting the newest one. Then, recursive update formulas of the parameters are used to avoid recalculating them after each affectation. A total of N_w feature vectors are used to define a prototype. Before reaching N_w inside a prototype, the update of the parameters are done only by adding the new patterns to this prototype. Formally, if n_{P_j} is the cardinal of the prototype P_j at a certain time stamp t_i, then we have:

1. If $n_{P_j} \leq N_w$, the parameters of the prototype P_j are updated using these formulas:

$$\left\{ \begin{array}{r} \mu_{P_j}(t_i) = \frac{n_{P_j}}{n_{P_j}+1}\mu_{P_j}(t_{i-1}) + \frac{1}{n_{P_j}+1}X_i, \\ \Sigma_{P_j}(t_i) = \frac{n_{P_j}-1}{n_{P_j}}\Sigma_{P_j}(t_{i-1}) + \frac{1}{n_{P_j}+1}(X_i - \mu_{P_j}(t_{i-1}))(X_i - \mu_{P_j}(t_{i-1}))^T. \end{array} \right. \tag{2.22}$$

This corresponds only to an information adding since the cardinal of the prototype did not reach N_w yet.

2. If $n_{P_j} \geq N_w$, the parameters of the prototype P_j are updated using these formulas:

$$\left\{ \begin{array}{r} \mu_{P_j}(t_i) = \mu_{P_j}(t_{i-1}) + \frac{1}{N_w}(X(t_i) - X(t_{old})), \\ \Sigma_{P_j}(t_i) = \Sigma_{P_j}(t-1) + \Delta X \begin{pmatrix} \frac{1}{N_w} & \frac{1}{N_w(N_w-1)} \\ \frac{1}{N_w(N_w-1)} & \frac{-(N_w+1)}{N_w(N_w-1)} \end{pmatrix} \Delta X^T, \end{array} \right. \tag{2.23}$$

where

$$\left\{ \begin{array}{lcl} X_{old} & = & \text{the oldest pattern that used to belong to the prototype } P_j, \\ \Delta X & = & \Delta X = \left[X_i - \mu_{P_j}(t_{i-1}) \quad X_{old} - \mu_{P_j}(t_{i-1}) \right], \end{array} \right. \tag{2.24}$$

© 2014 Tous droits réservés. doc.univ-lille1.fr

In Fig. (2.10), the adaptation mechanism of a cluster or a prototype is graphically illustrated in the case $n_{P_j} \geq N_w$. The parameter N_w is a critical parameter, as in all adaptation mechanisms based on a sliding window scheme. As stated in the first part of this chapter, the choice of this parameter must solve the stability-plasticity dilemma. A large value for this parameter means a lesser reactivity to drifts but a good representation of the concepts in the decision space. A small value means more reactivity to drifts but a lesser representation of concepts, meaning less stability.

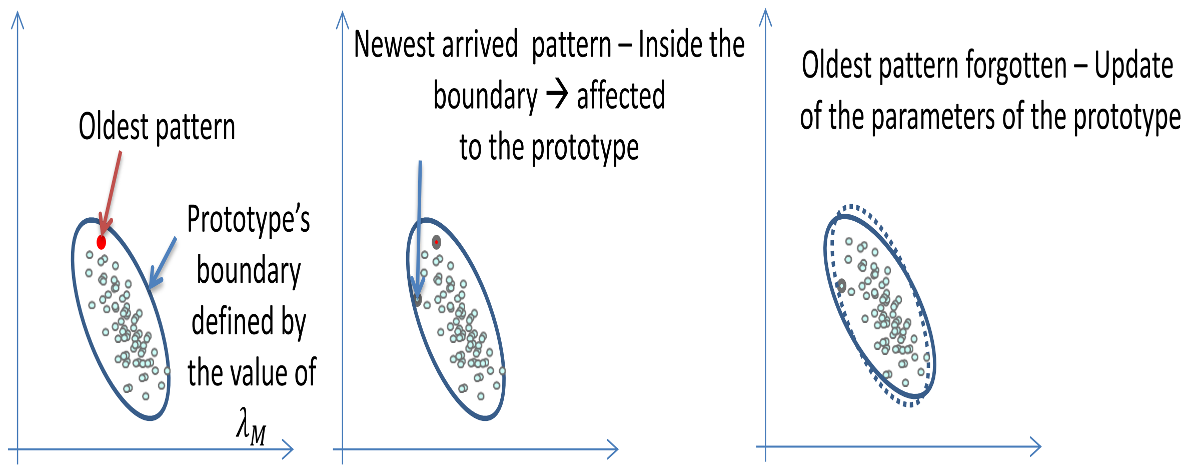

Oldest pattern
Prototype's boundary defined by the value of λ_M
Newest arrived pattern – Inside the boundary → affected to the prototype
Oldest pattern forgotten – Update of the parameters of the prototype

Figure 2.10: Update of the parameters of the clusters by AuDyC.

2.5.2.2 Creation of new prototypes/classes

As clarified in Table (2.10) (case 2), a new class is created with one prototype. AuDyC initializes the parameter of this prototype in such a way that:

- The mean of the new prototype is centered around the new pattern X_i that created it: $\mu_{P_{new}} = X_i$.
- The covariance matrix of the newly created prototype is given an initial value: $\Sigma_{P_{new}}^{init} = \sigma_{init}\mathbb{I}$. σ_{init} is user-defined parameter and $\mathbb{I}$ is the identity matrix. However, this does not mean that it is the only way to initialize the value of the covariance matrix. In some applications, the availability of historical data and/or expert knowledge helps choosing the best initialization value.

  Another possibility, which is the case 3 of the Table (2.10), a new prototype is created inside an already existing class. This is the case when the pattern falls inside the class but outside all the prototypes of this class. The mathematical translation of this case is highlighted in Table (2.10).

In Fig. (2.11), the creation of a new class/prototype is graphically illustrated.

© 2014 Tous droits réservés. doc.univ-lille1.fr

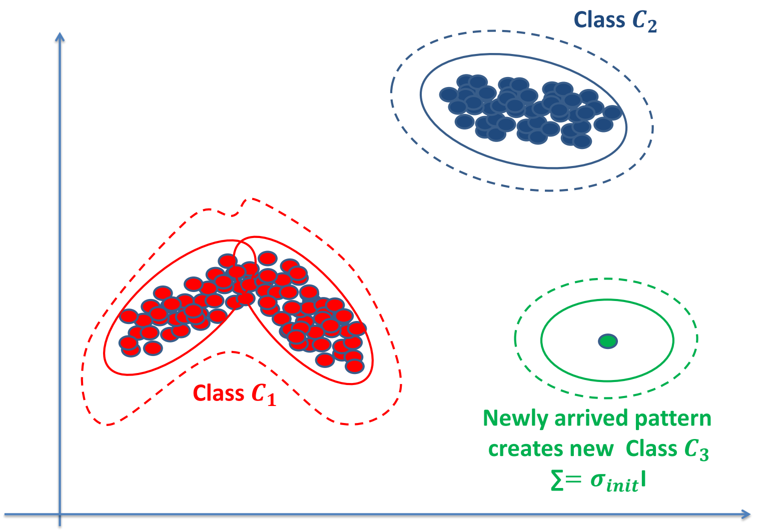


Figure 2.11: Class creation procedure by AuDyC.

2.5.2.3 Merging mechanism

AuDyC contains also a merging mechanism whose role is to deal with ambiguous data, as shown in Table (2.10) (case 4). These data belong to the **ambiguity** set, denoted X_{amb}. The ambiguity set is composed of patterns belonging to several classes. Thus, these datum points cannot be decided directly and clearly to which class they belong.

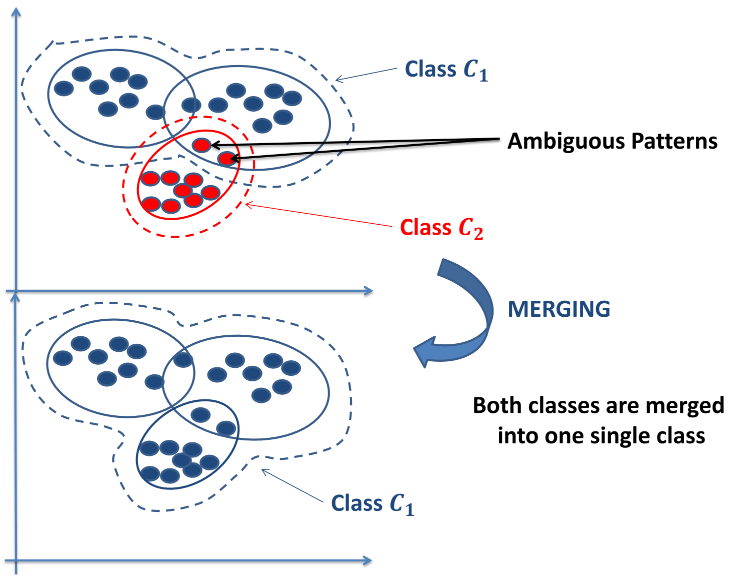


Figure 2.12: Merging mechanism of AuDyC.

Considering Gaussian prototypes, the merging mechanism is set as follows:

– First step: detection of candidate prototypes for merging. This is done thanks to an ambiguity criterion, a merging acceptance criterion and the definition of a parameter N_{amb}. The criterion

© 2014 Tous droits réservés. doc.univ-lille1.fr

is:

$$Card\{X \in P_i \& X \in P_j\} > N_{amb} \Rightarrow P_i \text{ and } P_j \text{ candidates.}$$

- Second step: evaluation of the merging acceptance criterion and then adaptation of the adequate prototype.
- Third step: merging of classes in case it is needed.

In Fig. (2.12), the notion of ambiguity is well clarified with an example in which $N_{amb} = 1$. The two classes in this example were merged to form one single class, since there were two points in ambiguity between a prototype from the class C_1 and the prototype of the class C_2.

2.5.2.4 Evaluation phase

Lastly, it is important to note that the creation of a new prototype can be sometimes risky. The reason is that these newly created prototype can be the product of noise, thus they are no more than parasite prototypes. The evaluation phase is important to eliminate them. To detect those not-representative prototypes, this phase is based on their cardinality. A created prototype will not be considered representative unless newly arrived data fall in its scope.

Let L_{pa} be a user-defined parameter that determines the number of pattern acquisition times. After L_{pa} acquisition, the cardinalities of the prototypes of all the classes are evaluated. Then, they are compared to a user-defined threshold: $N_{P_{min}}$. This threshold corresponds to the minimum cardinality that a prototype should have so that it is considered as representative.

In order to avoid deleting at each L_{pa} acquisition newly created prototypes (or in other terms, prototypes that did not have time to achieve their representativeness), the data contained in the parasite prototypes could be passed again to the classification process after a certain number of iterations.

It is important to remind that, in this work, AuDyC is used as a modeling tool to achieve condition monitoring, health assessment and prognosis. This section presented a brief description about the AuDyC algorithm. Other functionalities, as well as more detailed explanation about AuDyC, that are not going to be used in this work, can be found in (Lecoeuche and Lurette, 2003; Boubacar et al., 2005). In the next section, the way AuDyC is used for health monitoring, according to few assumptions made in this work, is shown.

2.6 Conclusion

In this chapter, many notions concerning the theory of concept drifts have been developed. At the beginning, a formal definition of this theory is given. The different forms, types and natures of concept drifts are defined and illustrated. Secondly, a formal characterization of the drift, seen as an event was developed. Finally, the last part of the chapter tackled the methods that handle concept

© 2014 Tous droits réservés. doc.univ-lille1.fr

drifts.

Concepts are defined by the joint probability of the input features and a possible output for them (class membership for instance). A concept drift is a change affecting either the input feature space (feature distribution) or the output values, or both. It is a consequence of non-stationary environments. Modeling concepts is about finding a relationship between those input features and their output. Modeling concepts that are subjects to drift should thus be able to handle such changes.

Drifts are events that occur in time, thus having a beginning point and an ending point. Also, a drift can differ from another by its speed and its magnitude or severity. Additionally, it is important to see how a sequence of drifts behaves over a long period of time. In this chapter, these points were tackled and a general characterization of single drifts and sequence of drifts was given.

Methods that handle concept drifts have gained a lot of importance in the last two decades. This abundance in literature requires a classification of the possible methods into a larger scheme. In this chapter, a state-of-the-art of these methods has been developed, resulting in a general scheme regrouping all these methods. Later on, few methodologies were selected according to our research interest. Finally, an analysis of successful points and disadvantages of methods allowed us to point out in which direction our research should go. This was also aided by the definition of the interesting characteristics we are waiting from a methodology.

This analysis resulted in choosing a dynamical clustering algorithm that can handle concept drift. The algorithm AuDyC was chosen to be the most adequate according to our demands. The next chapters of this thesis contain the work contribution that was developed. It is related to AuDyC in the sense that the latter must be associated with more functionalities, as drift detection, drift characterization and to prognosis. This association's goal is to create the complete architecture for supervision, diagnosis and prognosis.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Thèse de Antoine Chammas, Lille 1, 2014

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Chapter 3

Tools for drift detection, diagnosis and prognosis

3.1 Introduction

In Chapter 2, some problems related to evolving non-stationary environments have been studied. The study concluded in choosing an algorithm for handling drifts which is AuDyC. In this chapter, some of the contributions this thesis are going to be shown. These contributions concern several modules, defining different functionalities, which are going to be developed, and associated to AuDyC with the aim of creating an architecture for supervision, diagnosis and prognosis. These modules are the drift detection, drift characterization and prognosis modules.

The drift detection module is the one to alert the system of the presence of a change in the current functioning. In this work, the detection module is based on the fact that a failure mode, initiating a drift, will cause the evolving operating mode to deviate from the normal operating mode. The deviation of the classes representing these modes will be the basis of the drift detection test.

The drift characterization module is responsible for giving meaningful indicators that help to isolate the failure mode behind the drift, and to provide an assessment of the health state of the system. Thus, there are two aspects that are going to be tackled by the drift characterization module: the isolation aspect for which a direction indicator is developed, and a health assessment aspect for which a severity indicator is developed.

The prognosis module, being related to these two former modules, provides an estimate of the Remaining Useful Life and associates a confidence interval to it. In fact, once a failure mode triggering a drift is identified, the prognosis module uses the health assessment given by the severity indicator values in order to estimate a RUL and a confidence interval. The idea is to study the trend of the severity indicator, and project it into the future.

In the first section, a formulation of the problem relating the concept drift to the prognosis con-

© 2014 Tous droits réservés. doc.univ-lille1.fr

text is shown. The subsections that come after show the different tools that are developed (drift detection, drift characterization and online prognosis).

3.2 Problem formulation: from concept drifts to process faults

A system can have different operating modes. Namely, these modes could be the normal, degraded or failure modes. Normal operating modes correspond to the normal functioning of a system, in which case is fulfilling its task. A degraded operating mode is the functioning mode where the system still operates but not at its optimal efficiency. A failure mode corresponds to the breakdown of the system.

Systems are equipped with sensors that produce continuous streams of data. These data are preprocessed and features are extracted from them. Features are meaningful characteristics of the operating modes of the system. These features are essential for building a discriminative model for diagnosis/prognosis. They are used to build a feature space. They were described in Chapter 1 (Section 1.2.2.2) along with the ideal properties they should have.

Consider a system that is described by a normal operating mode and several failure modes. In our approach, the modeling consists in estimating clusters in the feature space from patterns that either come from historical data, or online (treated sensor) data. Then, a decision space is deduced from the feature space by exploiting a clustering tehcnique and by assigning an operating mode to each cluster. In other words, each cluster is demarcated by a decision boundary and the region inside this boundary corresponds to an operating mode. The transition from the feature space to the decision space is illustrated in Fig. (3.1), in case of Gaussian classes.

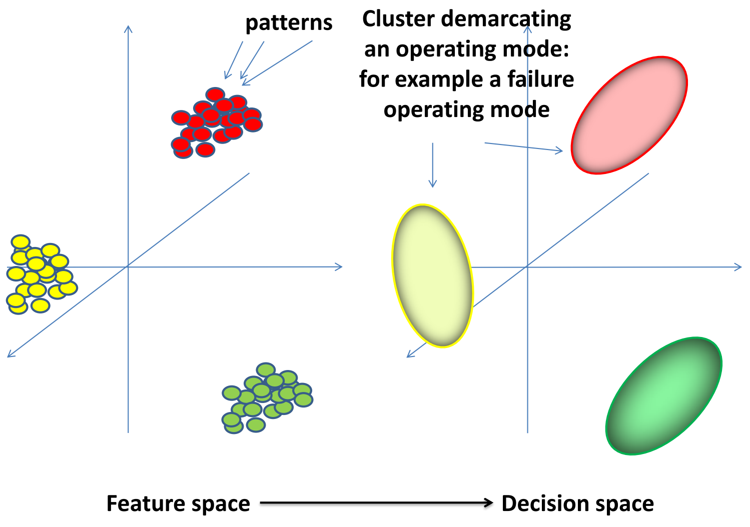

patterns
Cluster demarcating an operating mode: for example a failure operating mode
Feature space
Decision space

Figure 3.1: Transition from the feature space to the decision space.

© 2014 Tous droits réservés. doc.univ-lille1.fr

The step of building an initial model is done using dysfunctional analysis methods along with available historical database. This analysis is realized according to available measurements and an expert knowledge on the process. The expert is able to label the decision space. In the example shown in Fig. (3.1), it is possible that the green class corresponds to the normal mode, the red and the yellow classes are two failure modes.

Once the clustering technique model is used in an online manner, and the system is operating without faults, the patterns should fall in this case in the region corresponding to the 'normal class'. For example, in the case of the classes in Fig. (3.1), the patterns will fall in the green class under no fault conditions. When patterns fall in the yellow or the red classes, this means that a failure mode is reached. It is important to note that the classes that demarcate a failure mode correspond in reality to the phase of time where the system is about to break down. The reason is that when the system is really shut down, there is few or no data that can be gathered. Thus, a failure class is in reality a pre-breakdown class.

A process drift is an incipient fault that can affect a system, a sensor or an actuator. However, whatever is the affected process, the incipient fault is a slowly evolving fault leading the system to a failure state through a degraded state.

A process drift causes a gradual change in the distribution of patterns in the feature space, which is characterized by evolving non-stationary data. This is how an incipient fault behaves, affecting an operating process. It can be translated, from a scientific point of view, to a concept drift problem, as defined in Chapter 2.

At the end of Chapter 2, AuDyC was chosen to be the algorithm used for handling drifts. In the next subsection, it will be shown how this tool can be converted to a health monitoring tool.

3.3 Health monitoring based on AudyC

AuDyC is an essential tool that is used for modeling continuous streams of data. It is the key stone solution to handling non-stationary environments. Thus, it is the monitoring tool used in this work. The classes that are built by AuDyC, and which are continusouly updated, correspond to the operating modes of the system. Treating the parameters of these classes constitutes the essential tool that our architecture is built on. Few assumptions accompanying the rest of the work can be made at this point:

1. The classes considered in this work are homogeneous and contain only one prototype. This yields that an operating mode can be represented by a Gaussian prototype, or cluster.
2. The parameters λ_m and λ_M, that define respectively the boundaries of a prototype and a class, are equal (Chapter 2, Section 2.5.2). This assumption completes the first by combining the boundaries of the class and its prototype.

© 2014 Tous droits réservés. doc.univ-lille1.fr

3. An assumption of the type of drift should be made. In Chapter 2, Section 1.3.1, different types of drifts have been defined. In this work, the drift is assumed to be continuous. This means that it initiates the migration of the evolving current class from one operating mode to another, but in small changes (the reader can also refer to Fig. (1.4), Chapter 2, Section 1.3.1). Also, it is supposed that there is no recovery mechanisms. This means that once a drift starts, it will continue until failure, with no possibility for an intermediate 'healing'.

As stated earlier, AuDyC is suited to model drifting concepts since it is dynamical and auto-adaptive by nature. When data evolves slowly, the evolution of classes can be visually seen. The decision space is continuously updated. The ability to continuously represent the current concept is interesting for our application since we essentially need to monitor the condition of systems. AuDyC seems as an appropriate tool to tackle the challenges defined in the previous section. At this point, it is important to supervise the system and to detect anomalous events as soon as they occur. Before being able to do this, the acquired historical knowledge on the system must be modeled. All the steps, beginning from the initial historical learning phase to the online drift detection are detailed in the subsections below.

3.3.1 Initial offline learning

The offline learning of the operating modes of the system is made using historical data, labeled thanks to expert knowledge.

At this point, it is interesting to clarify the hypothesis made about the availability of this initial knowledge. In fact, in this work, it is considered that the features used to build the feature space span a decision space containing one operating mode to represent normal behavior. It is considered that more than one normal operating mode out of the scope of the work. The decision space contains also several known failure modes. However, as stated earlier, it is possible that few failure modes are not known at the beginning. An example of such offline models is given in Fig. (3.2).

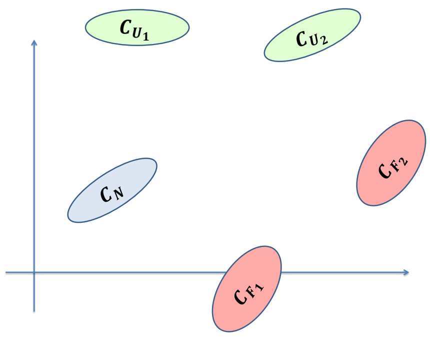


Figure 3.2: Initial offline modeling using AuDyC.

The initial construction of the knowledge base (initial decision space) is done using AuDyC. Initially, we are given:

© 2014 Tous droits réservés. doc.univ-lille1.fr

- S_n: a data set corresponding to normal operating mode. Let $N_{S_n} = card(S_n)$ be the cardinality of S_n.
- $S_{f_l}, l = 1, \ldots, n_f$: sets of data corresponding to different failure modes. $n_f + 1$ is the number of sets available.

We start the training process with an empty decision space. The first training set used for learning is S_n. The first feature vector is inserted in the database as the initial cluster, denoted $C_n(t_0)$. Each subsequent input feature vector is assigned to $C_n(t_i)$ and the parameters (center and covariance matrix) of the latter are updated (see chapter 2). The process iteratively continues on each feature vector of the training set S_n. After a certain time, the cluster $C_n(t_i)$ converges to a region in the feature space. At this point, we define $C_N = C_n(t_{N_{S_n}})$ the last cluster that is covering the region corresponding to normal operating mode. The cluster C_N is characterized by its mean, denoted μ_N and its covariance matrix, denoted Σ_N. The same is done to the training sets S_{f_i}, $i = 1, \ldots, n_f$ and the result is the obtainment of C_{f_i}, $i = 1, \ldots, n_f$, the last clusters covering failure modes (see Fig. (3.2)).

At this point, it is important to note that the availability of the data sets is crucial for the definition and the characterization of failure classes. In fact, concerning these failure modes, two cases can be considered:

- There is enough data to define and characterize a failure class C_{F_j}, $j = 1, \ldots, n_f$. In this case, the parameters of the classes C_{F_j} are supposed to be well estimated.
- There is not enough data to define and characterize the failure classes C_{F_j}. However, it can always be considered that the available knowledge can be used, even if the parameters of these failure classes are not very well estimated.

In addition to these known failure modes, there exist unknown ones. These are failure modes that could appear during the operational time of a system. The unknown characteristic of these failure modes can be attributed to single faults which were not covered by the knowledge stemmed from the historical data. It can also be due to multiple faults occurring in the same time. These unknown regions of the decision space are denoted C_{v_i}, $i = 1, \ldots, n_u$ (see Fig. (3.2)).

3.3.2 Online operating process

Once the offline model is built, it can be used in an online manner. It is supposed that patterns are arriving in the form of a data stream.

The current functioning mode of the process is modeled in the decision space and it's called the **evolving current class**, and denoted $C_e(t_i)$. Under normal operating conditions, patterns will fall in the domain of C_N. Thus, under normal conditions, the evolving class is approximately equal to the normal class C_N. Formally, this is mathematically equivalent to:

$$C_e(t_0) = C_N. \tag{3.1}$$

© 2014 Tous droits réservés. doc.univ-lille1.fr

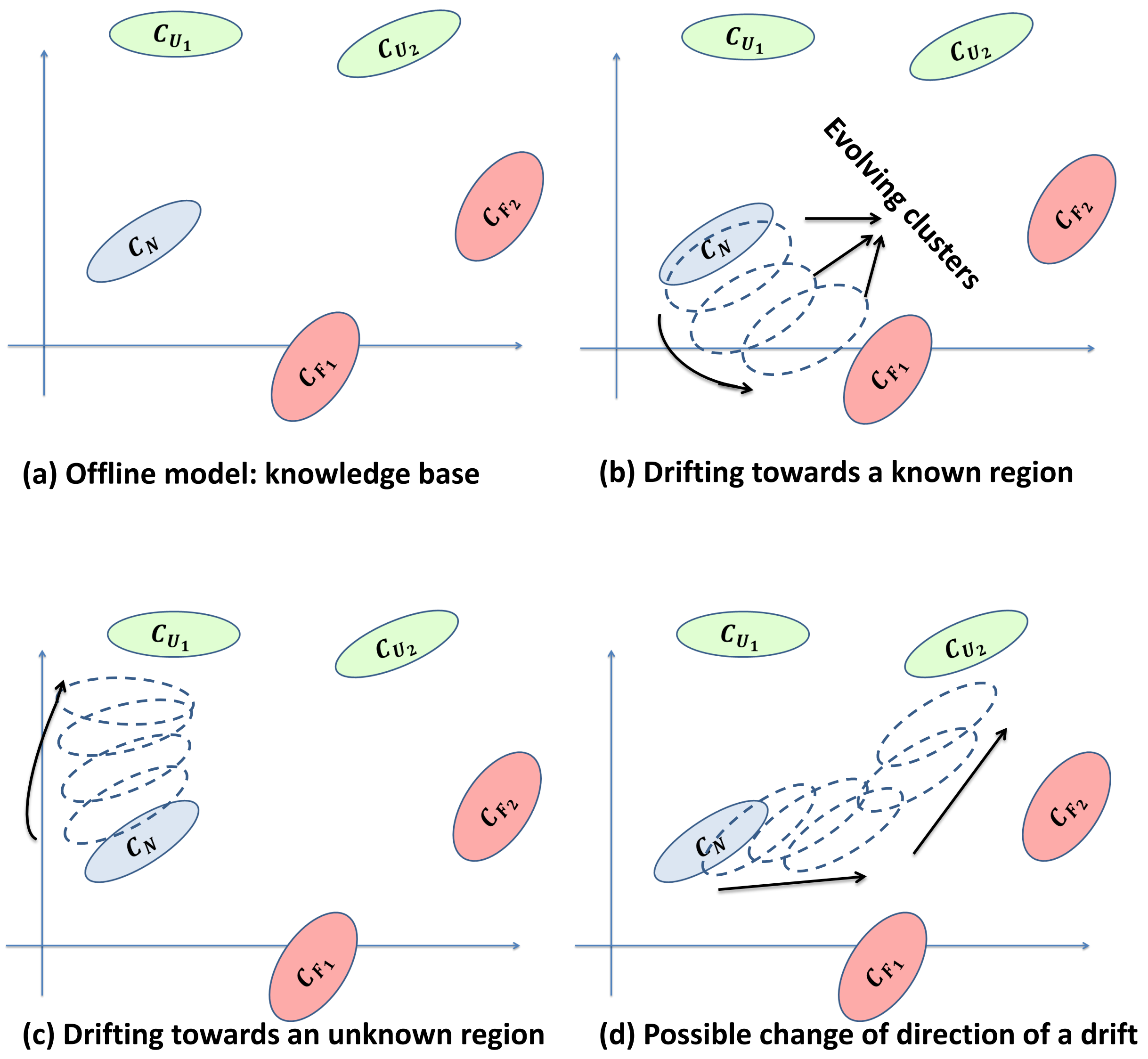

C_{U_1}
C_{U_2}
C_N
C_{F_2}
C_{F_1}
(a) Offline model: knowledge base
Evolving clusters
(b) Drifting towards a known region
(c) Drifting towards an unknown region
(d) Possible change of direction of a drift

Figure 3.3: Class evolution procedure by AuDyc.

As long as no anomalies take place, the current evolving class will be approximately in the same region as the normal class learnt offline, such that:

$$C_e(t_i) \approx C_e(t_{i-1}) \approx \ldots \approx C_e(t_0) = C_N. \tag{3.2}$$

At this point, let's remind an assumption made in the beginning of the chapter concerning the type of drift. In Chapter 2, Section 1.3.1, different types of drifts have been defined. In this work, the drift is assumed to be continuous. This means that it initiates the migration of the evolving current class from one operating mode to another, but in small changes (the reader can also refer to Fig. (1.4), Chapter 2, Section 1.3.1). Also, it is supposed that there is no restoring mechanisms. This means that once a drift starts, it will continue until failure, with no possibility for an intermediate 'healing'.

© 2014 Tous droits réservés.

The evolution of the evolving class can have three possible outcomes. In Fig. (3.3, (b), (c) and (d)), these possibilities are:

– If a known failure mode is causing the drift, the evolving class will begin to migrate to the known region in the decision space which is covered by the failure class.

– If an unknown failure mode is causing the drift, then in this case, the evolving class begins to migrate towards an unknown region. In this case, it is necessary to identify and interpret the meaning of this unknown region in the feature space. To do so, expert knowledge may be required.

– It is possible in some situations that a drift towards a known failure mode changes direction. This is due to the influence of another failure mode. This change in direction puts the monitoring system in an unknown state also, requiring possibly an expert intervention.

It is important to note that in the first case, where the evolving class migrates toward a known region (characterized by a failure class), there are two dynamical models to be considered separately:

1. The first model concerns the dynamical update of the parameters of the evolving class. This is done by AuDyC.

2. The second model concerns the trajectory of the center of the evolving class. In fact, in this work, it is supposed that the migration of the evolving class to the failure class can take a linear or a slightly non-linear (parabolic) path. These two dynamical models are shown separately in Fig. (3.4). Later on in this chapter, more details about the shape of the trajectories are going to be shown (Fig. (3.17)).

The orientation for the work to be developed is concerned with building tools that will be used for the architecture for supervision, diagnosis and prognosis. The next steps are:

– Equipping AuDyC with a drift detection mechanism. This step allows having an alarm launching mechanism that detects anomalous events. The drift detection is associated to the supervision process.
– Equipping AuDyC with indicators for health assessment. In this step, a full real-time characterization of a drift must be provided. The drift characterization is associated to the diagnosis process, in which the failure mode behind a fault is isolated, and a severity assessment of the health state, are given.
– Developing a prognosis algorithm that can be integrated to AuDyC. In fact, the drift characterization block associated to AuDyC provides an image of the dynamics of the drift. The prognosis algorithm needs to take advantage of this information in order to provide an estimate of the RUL, as well as a confidence interval associated to it.

© 2014 Tous droits réservés. doc.univ-lille1.fr

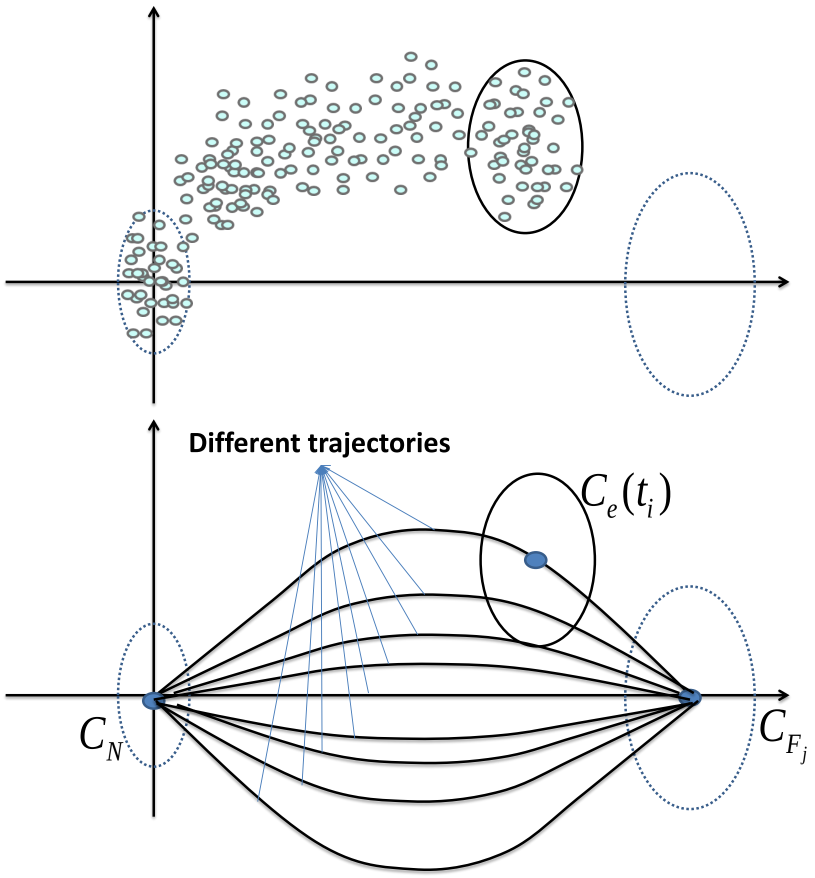


Figure 3.4: Upper figure: a drift scenario from a normal mode to a failure mode showing the dynamical update of the parameters of an evolving class done by AuDyC; Lower figure: a sketch of this drift scenario showing the class labels and the different possible trajectories of the evolving class.

3.4 Drift detection

In any case of drift (independantly from the failure mode triggering it), there is a migration of the evolving cluster from a normal mode to another mode. Thus, the evolving cluster will deviate from the normal mode. Drift detection is about detecting this deviation. The detection of a drift is evaluated by its detection time and the associated number of false alarms. In this work, the detection is based on the assumption that the drift is continuous. The migration of the evolving cluster from the normal cluster will thus cause the mean of the evolving cluster to significantly deviate from the mean of the normal cluster. Consequently, a convenient drift detection test can be a hypothesis test on the means of the normal cluster and the evolving cluster (Johnson and Wichern, 2012; Saporta, 2011). The random variable to be used in this test is the online coming feature vector, or pattern $X(t_i)$. The dimension of the feature space, spanned by the patterns $X(t_i)$ is d.

© 2014 Tous droits réservés. doc.univ-lille1.fr

The drift detection test needs to formulate some hypothesizes. Formally speaking, the hypothesis of interest is that there is no drift versus the other hypothesis of drift has occurred. Mathematically, this can be written as:

$$H_0 : \mu_e(t_i) = \mu_N \text{ vs. } H_1 : \mu_e(t_i) \neq \mu_N, \tag{3.3}$$

where, to recall, μ_N and $\mu_e(t_i)$ are the means of the normal cluster and the evolving cluster respectively. The test will decide if the hypothesis H_0 is true or not based on the computation of a statistic. Then, the computed statistic is compared to a p-value that is determined according to a confidence level, also called critical level and usually denoted by α. The choice of the statistic depends on the distribution of the vectors from the sample that is inducing the value $\mu_e(t_i)$. Let S_{C_e} denote that sample. S_{C_e} contains the patterns that are defining the current evolving class. The cardinality of S_{C_e} is the AuDyC parameter N_w (Chapter 2, Section 2.5).

Since we are in a d-dimensional decision space, it is necessary to consider a multivariate hypothesis test and not d different univariate tests. The reasons are:

1. The use of d different tests increases the error rate α, whereas the multivariate test preserves the exact α-level. For example, if we have $d = 10$ separate univariate tests at the $\alpha = 0.05$ level, the probability of at least one false rejection is greater than 0.05. If the variables are independent (which is not always the case), we would have, under H_0:

$$\begin{aligned} P(\text{at least one rejection}) &= 1 - P(\text{all 10 accepts } H_0) && (3.4)\\ &= 1 - (0.95)^{10} = 0.4. && (3.5)\end{aligned}$$

   The level 0.4 is not an acceptable confidence level.

2. Multivariate tests take into consideration the correlation between variables something that univariate tests ignore.

The test to be used in this work is the **Hoteling T^2 test**. It will allow us to test the veracity of the hypothesis H_0 versus the hypothesis H_1 (Eq. (3.3)). The reasons why we can use this test are:

1. The patterns belonging to S_{C_e} come from a normal distribution,
2. These patterns are statistically independent.

In this work, the Hoteling T^2 test is used in the case of a one sample test (Johnson and Wichern, 2012; Saporta, 2011), since we are comparing the mean of the sample S_{C_e} to μ_N, which is supposed to be known from historical data. The estimates of the mean and the covariance matrix of S_{C_e} are instantly given by AuDyC. The Hoteling test invloves calculating a statistic, denoted by $T^2_{\nu_1,\nu_2}(t_i)$, where ν_1 and ν_2 are the degress of freedom associated to the statistic, and t_i is the time stamp (te statistic is calculated at each instant). It is given by:

$$T^2_{\nu_1,\nu_2}(t_i) = N_w \times (\mu_e(t) - \mu_N)^T \times \Sigma_e^{-1}(t_i) \times (\mu_e(t) - \mu_N). \tag{3.6}$$

There are some key properties for the Hoteling statistic:

© 2014 Tous droits réservés. doc.univ-lille1.fr

- We must have: $N_w - 1 \geq d$. Otherwise, $\Sigma_e(t_i)$ is singular and $T^2(t_i)$ cannot be computed.
- The degrees of freedom of the $T^2_{\nu_1,\nu_2}(t_i)$ statistic are $\nu_1 = N_w - 1$ and $\nu_2 = d$ (dimension).
- The statistic $T^2_{\nu_1,\nu_2}(t_i)$ can be converted to an F-statistic, where 'F' refers to the Fisher-Snedecor distribution. This is done as follows:

$$\frac{\nu - \nu_2 + 1}{\nu_1 \nu_2} T^2_{\nu_1,\nu_2} = F_{\xi_1,\xi_2}, \tag{3.7}$$

  where $\xi_1 = \nu_2 = d$ and $\xi_2 = \nu_1 - \nu_2 + 1 = N_w - d$. Note that the dimension d of the T^2 statistic becomes the first of the two degrees of freedom parameters of the F statistic.

The particularity of the relation between the Hoteling distribution and the Fisher-Snedecor distribution will be used to compute a threshold for drift detection (*see* Eq. (3.7)). In fact, at each time stamp t_i, the parameters of the evolving class $C_e(t_i)$ are adapted. Thus, new value of $\mu_e(t_i)$ are constantly available. Consequently, at each time stamp t_i, the value $T^2_{\nu_1,\nu_2}(t_i)$ is computed for the values of ν_1 and ν_2 defined by the values of N_w and d. Let us define:

$$T_h = \frac{\nu_1 \nu_2}{\nu_1 - \nu_2 + 1}. \tag{3.8}$$

For a user-defined confidence level α, there is a high chance of drift if:

$$T^2_{\nu_1,\nu_2}(t_i) > T_h \times F_{(\xi_1,\xi_2)|\alpha}, \tag{3.9}$$

where, to recall, $\xi_1 = \nu_2 = d$ and $\xi_2 = \nu_1 - \nu_2 + 1 = N_w - d$. By replacing the values of ξ_1 and ξ_2 in Eq. (3.9), we have $F_{(\xi_1,\xi_2)|\alpha} = F_{(d,N_w-d)|\alpha}$. The value of $F_{(d,N_w-d)|\alpha}$ can be found in tables in statistics books (Johnson and Wichern, 2012).
However, the relation in Eq. (3.9) cannot confirm, in a deterministic manner, if a drift has occurred or not. In fact, it confirms the occurrence of a drift with a certain confidence limit. To ensure the occurrence of a drift, a new parameter is added. It is denoted K_h and it is used in Eq. (3.9), in such a way that drift is confirmed if:

$$T^2_{\nu_1,\nu_2}(t_i) > K_h \times T_h \times F_{(d,N_w-d)|\alpha}, \text{ where } K_h \geq 1. \tag{3.10}$$

The threshold for detection is thus: $T_h \times F_{(d,N_w-d)|\alpha}$. The value of K_h is user-defined and can be tuned using historical data in such a way that false alarms are limited. In fact, in this thesis (and as it will be detailed more in Chapter 4), it is considered that false alarms should be completely avoided. The role of K_h is to make sure this condition is verified.

K_h amplifies the value of the threshold set by the hypothesis test. The idea is that a first value of the threshold is determined according to the value of α. Then, by using the data corresponding to normal operating functioning, the value of K_h is tuned in such a way no value of $T^2_{\nu_1,\nu_2}(t_i)$ never crosses the value of the threshold for detection. In Chapter 4, more discussion and scenarios are shown (in the experimentation part of Chapter 4) concerning the choice of α and K_h.

© 2014 Tous droits réservés. doc.univ-lille1.fr

3.5 Drift characterization

In the case of detection of a drift, a signal confirming this event (an alarm) will be launched. Upon detection, a characterization of the drift is needed. The aim behind characterizing a drift is to compute meaningful indicators that will be used for fault isolation and condition assessment. Thus, the drift characterization contains these related to the isolation of a drifting fault and to the condition assessment.

In Chapter 2, few criteria for characterizing drifts have been given (Chapter 2, Section 2.3.3). The notions of beginning time, ending time, duration, overall speed and overall severity of drifts are given. However, these criteria describe a drift as a general event. What is needed are more particular criteria that extend those in terms of fault diagnosis. The indicators that are seeked must be computed in an online manner, in order to provide an online monitoring of the system's health. For this reason, we define these two indicators:

- Direction indicator: this indicator is related to the aspect of isolation of a failure mode that is responsible for the process drift. It is the module that generates indicators that are used for isolation.

- Severity indicator: given a drifting scenario towards a known failure mode, this indicator will indicate the current severity of the drift. It is different than the overall severity defined in Chapter 2. The former is an image of how much the drift has evolved compared to the failure level. The overall severity indicator, defined in Chapter 2, concerns the main difference that exists between concepts. For example, let's consider a leak in a reservoir. The leak is caused by cracks on the surface of the reservoir. These cracks are slowly growing up. The severity indicator reflects the size of the current cracks. The overall severity reflects the limit of the acceptable crack size.

The next subsections are dedicated to formally define and detail those characterization indicators.

3.5.1 Direction indicator

An evolving cluster will migrate away from the cluster representing the normal operating mode, once a failure mode triggers this movement. The direction indicator is used to help finding this failure mode, *i.e.* to pinpoint the cause of the fault by studying the direction of the movement of the evolving cluster.

Amongst the failure classes, let's consider that C_{F_j} (the j^{th} one) is the one triggering the drift. Then, the direction indicator must have a value that is equal to the failure's mode number j. Let $C_e(t_i) = (\mu_e(t_i), \Sigma_e(t_i))$ and $C_e(t_{i-N_p}) = (\mu_e(t_{i-N_p}), \Sigma_e(t_{i-N_p}))$ the parameters of the evolving clusters at time t_i and t_{i-N_p} respectively, where $N_p \in \mathbb{N}$ is a user defined parameter.

The idea behind defining N_p is to consider a time horizon between the centers of two clusters

© 2014 Tous droits réservés. doc.univ-lille1.fr

for the calculation of the direction indicator. By choosing a small value for N_p, the direction indicator will be highly sensitive to the evolution of the cluster $C_e(t_i)$ (high plasticity, small stability). By choosing a large value for N_p, the direction indicator will indicate the direction of evolution of $C_e(t_i)$ over a larger time horizon. Thus, it will be more robust to noisy evolution of $C_e(t_i)$ but less reactive and slower to detect direction changes (high stability, less plasticity). For this reason, this parameter must be well chosen in order to optimize this choice.

Let μ_{f_j} be the center of the failure cluster corresponding to failure number j. The idea is to consider a vector that is an image of the movement direction, and a vector that is an image of the destination direction, and then to study the angle between those two vectors. Let:

- $D_e(t_i) = \frac{\mu_e(t_i)-\mu_e(t_{i-N_p})}{||\mu_e(t_i)-\mu_e(t_{i-N_p})||}$ be the unitary vector relating the centers of the clusters $C_e(t_i)$ and $C_e(t_{i-N_p})$. $D_e(t_i)$ is an image of the movement direction. They are called **movement vectors**.

- $D_j(t_i) = \frac{\mu_{f_j}(t_i)-\mu_e(t_i)}{||\mu_{f_j}(t_i)-\mu_e(t_i)||}$ be the unitary vector relating the centers of $C_e(t_i)$, and the center of the failure cluster C_{F_j}, $j = 1, ..., n_f$. $D_j(t_i)$ is an image of the destination direction. They are called **direction vectors**.

The algorithm for the computation of the direction indicator is as follows. At each step, let $p_j(t_i) = D_e^T(t_i).D_j(t_i)$, and $\Gamma(t_i) = \underset{1\leq j\leq n_f}{arg(max}(p_j(t_i)))$. $p_j(t_i)$ is the scalar product between $D_e(t_i)$ and the $D_j(t_i)$. The closer its value is to 1, the more $D_e(t_i)$ and $D_j(t_i)$ are closer to be colinear. The reason is that the scalar product is directly related to the angle relating those two vectors. In fact, the mathematical expression of $p_j(t_i)$ is:

$$p_j(t_i) = D_e^T(t_i)D_j(t_i) = ||D_e^T(t_i)|| \times ||D_j(t_i)|| \times \cos(D_e(t_i), D_j(t_i)). \tag{3.11}$$

Since the vectors $D_e(t_i)$ and $D_j(t_i)$ are unitary vectors, their amplitude is equal to 1. The Eq. (3.11) yields:

$$p_j(t_i) = D_e^T(t_i)D_j(t_i) = \cos(D_e(t_i), D_j(t_i)). \tag{3.12}$$

Thus, if $p_j(t_i) = 1$, then the angle $(D_e(t_i), D_j(t_i)) = 0$. In this case, the vectors $D_e(t_i)$ and $D_j(t_i)$ are colinear.

The values of the direction indicator and the rules of assignment at each step follow this algorithm:

1. First calculate $p_j(t_i)$ and $\Gamma(t_i)$,
2. If $p_\Gamma(t_i) \geq p_M$, where p_M is a user-defined threshold, then it is safe to say that the movement is towards the failure cluster whose number is Γ. In this case, we give a value for direction such that $direction = \Gamma$.

   For example, if $p_\Gamma(t_i) = 1 \geq p_M = 0.8$, then the drift is towards the failure mode number Γ (at this time stamp) and it is linear, *i.e.* the trajectory of the centers of the cluster $C_e(t)$ is lined up with the fault.

© 2014 Tous droits réservés. doc.univ-lille1.fr

3. In the case where $p_\Gamma(t) < p_M$, the movement of the drift is considered to be towards an unknown region. The value of direction in this case is 0. It is important to remind that the value of the direction indicator prior to drift detection is also equal to zero.

In order to better view the role of the parameter p_M, let us consider the example of an evolving class, migrating towards a failure class C_{F_n} (see Fig.(3.6)). In fact, the value of p_M defines a certain hypercone starting from the center of the evolving cluster $\mu_e(t_i)$. The base angle of this hypercone is defined by the value of p_M. This angle can be calculated by using Eq. (3.12). The axe passing through the vector $D_n(t_i)$ is a symmetry axe. If the vector $D_e(t_i)$, translated to the center of the evolving cluster $\mu_e(t_i)$, falls inside the hypercone, then the direction of the drift is towards the failure cluster C_{F_n}.

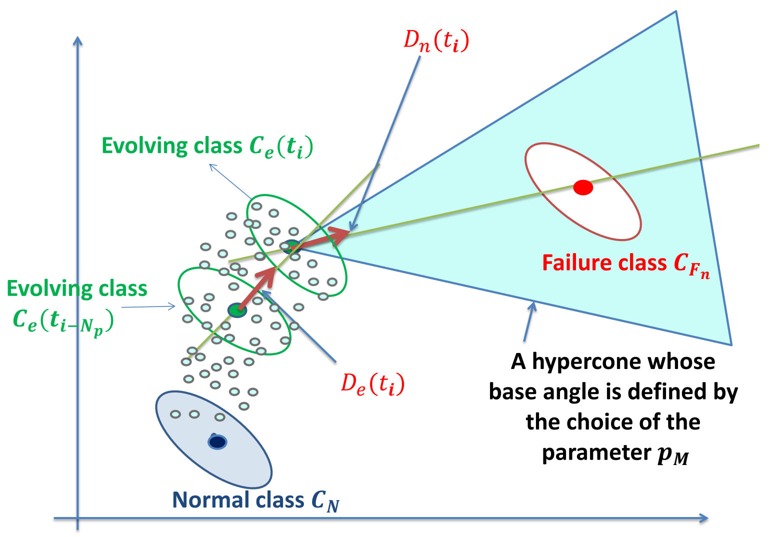

$D_n(t_i)$
Evolving class $C_e(t_i)$
Evolving class $C_e(t_{i-N_p})$
Failure class C_{F_n}
$D_e(t_i)$
A hypercone whose base angle is defined by the choice of the parameter p_M
Normal class C_N

Figure 3.5: A hypercone whose symmetry axe relates the center of the evolving cluster to the center of the failure cluster, and whose base angle is defined by the value of p_M.

The threshold p_M is user defined and depends on the application. The larger its value is, the more the user is assuming a linear drift. The choice of p_M must thus be based on experience or some *a priori* knowledge on the degradation mechanisms. In Fig.(3.6), a graphical illustration of the impact of p_M is given.

Another discussion concerning the value of p_M is also related to the colinearity of the failure classes. In fact, it is possible to have, in a feature space, two failure classes that are close one to the other. In Fig. (3.7), an example is shown with two failure modes, $C_{F_{j1}}$ and $C_{F_{j2}}$, that are close one to the other. In this case, the hypercones overlap (the pink region in Fig. (3.7)). This case is still acceptable since the value of the direction indicator corresponds to the failure class wherewith the angle between the direction vector and the movement vector is the smallest. However, in this case, the chances of erroneous results are higher because the movement vector and the direction vectors

© 2014 Tous droits réservés. doc.univ-lille1.fr

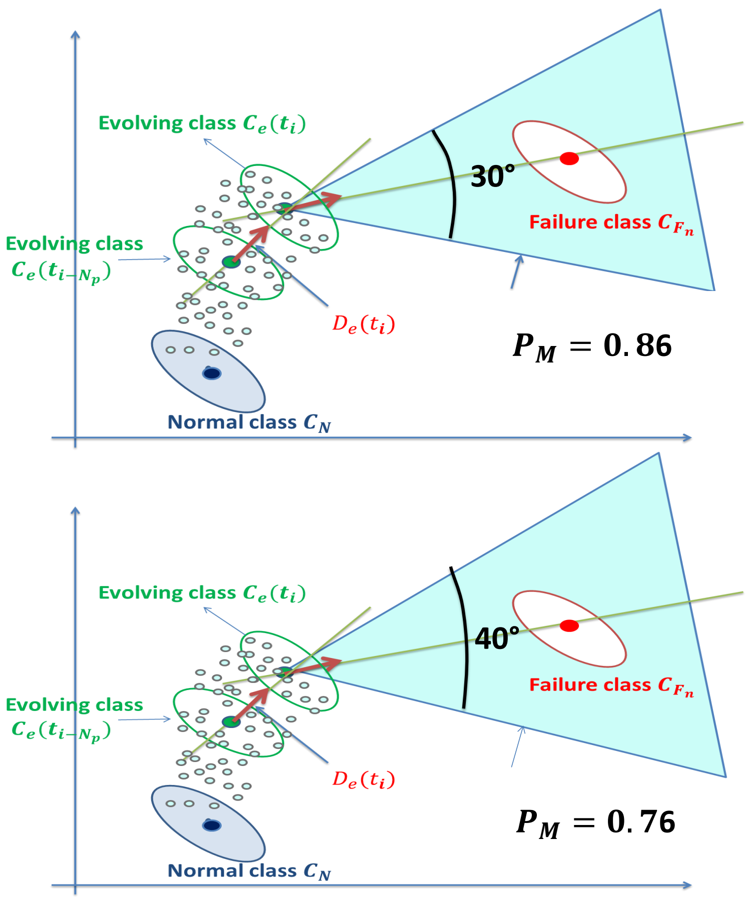


Figure 3.6: Effect of the value of p_M on the base angle of the hypercone.

corresponding to both $C_{F_{j_1}}$ and $C_{F_{j_2}}$ are all close together. Finally, this case is even harder when the center of the two failure classes $C_{F_{j_1}}$ and $C_{F_{j_2}}$ are colinear with the movement vector. This case is shown in Fig. (3.8). It presents a challenge to the current direction indicator computation algorithm since it is not possible to know which is the failure mode triggering the drift.

The next subsection the second characterization indicator developed in this work, which is the severity indicator.

3.5.2 Severity indicator

Suppose we are given a drift scenario as depicted in Fig. (3.9). The severity indicator must reflect how far the evolving class $C_e(t_i)$ is from the normal class and how close is it getting to a failure class. In the example given in Fig. (3.9), the failure's class number is 1.

In order to compute this indicator, we thus need a certain metric between classes. Since the clusters are Gaussians, the symmetrized Kullback-Leibler divergence metric (Boubacar et al., 2004) seems to be appropriate. The equation of this divergence between two multivariate Gaussian prototypes $P_1 = (\mu_1, \Sigma_1)$ and $P_2 = (\mu_2, \Sigma_2)$, denoted $d_{KL}(P_1, P_2)$, is:

© 2014 Tous droits réservés.

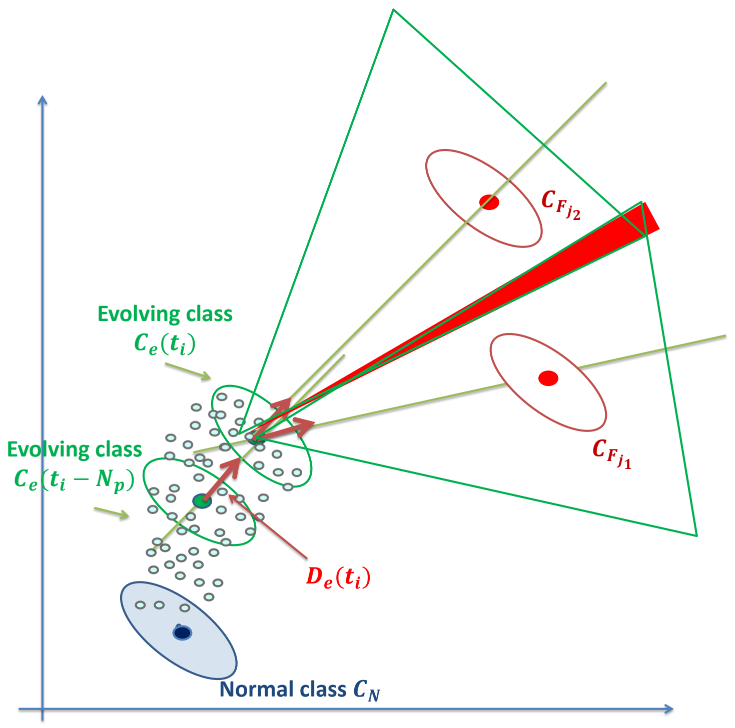


Figure 3.7: Computation of the direction indicator in the case where two failure modes $C_{F_{j_1}}$ and $C_{F_{j_2}}$ are close together but not colinear.

$$d_{KL}(P_1, P_2) = d_{KL}(P_2, P_1) = \frac{1}{2} \times (tr[(\Sigma_1)^{-1} \times \Sigma_2 + (\Sigma_2)^{-1} \times \Sigma_1] + (\mu_1 - \mu_2)^T((\Sigma_1)^{-1} + (\Sigma_2)^{-1}) \times (\mu_1 - \mu_2)) - d, \tag{3.13}$$

where d is the dimension of the feature space. The choice of the Kullback-Leibler metric as a divergence metric is backed-up by its adequacy to Gaussian prototypes. In fact, it takes into consideration the distance between the centers of each of the Gaussian prototypes to the other Gaussian prototype. In addition, it takes into consideration their shape and orientation. Two cases are possible:

– The direction of drift is towards a known failure cluster, that is characterized by a failure class C_{F_j}: in this case the severity indicator, calculated at each time stamp t_i, is denoted $I^j_{sv}(t_i)$ and is given by:

$$I^j_{sv}(t_i) = \frac{d_{KL}(C_N, C_e(t_i))}{d_{KL}(C_N, C_e(t_i)) + d_{KL}(C_e(t), C_{F_j})}. \tag{3.14}$$

– The direction of the drift is towards an unknown region in the decision space, or towards a known region but not characterized by a failure class: in this case, no severity indicator can be calculated.

From Eq. (3.14), it is clear that a severity indicator $I^j_{sv}(t_i)$ can take values ranging from 0 to 1, in case of a drift towards a known class. Under normal operating conditions, $C_e(t_i) \approx C_N$ thus

© 2014 Tous droits réservés. doc.univ-lille1.fr

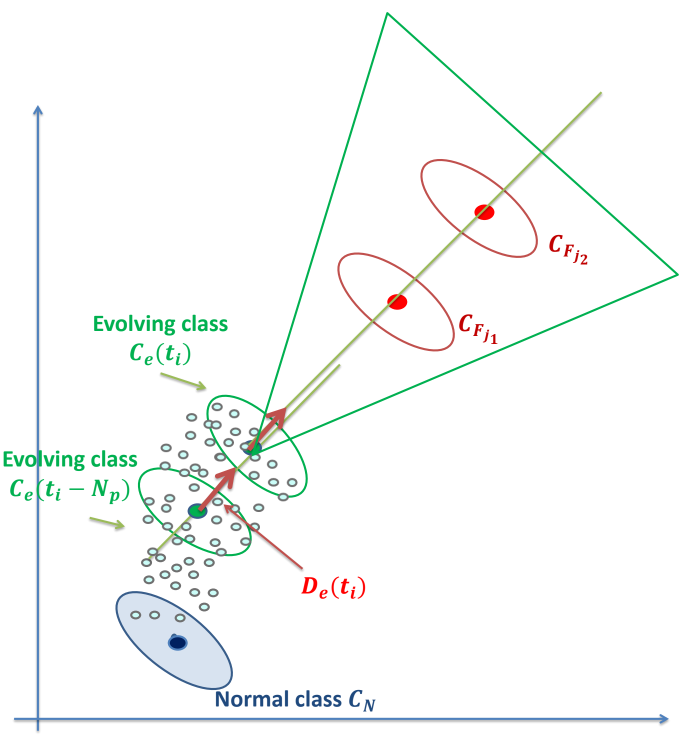


Figure 3.8: Computation of the direction indicator in the case where two failure modes $C_{F_{j_1}}$ and $C_{F_{j_2}}$ are colinear.

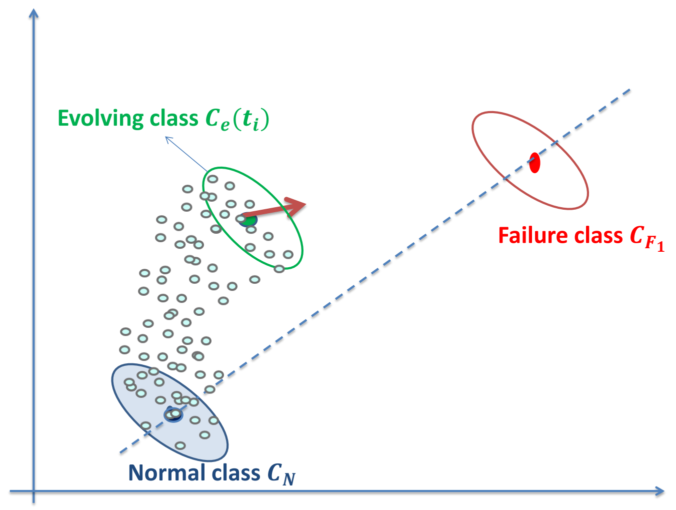


Figure 3.9: Drift scenario from the normal mode to a knwon region characterized by the failure mode number 1.

$I_{sv}^{j}(t_i) \rightarrow 0$. Under failure operating conditions (failure j), $C_e(t_i) \approx C_{F_j}$ thus $I_{sv}^{j}(t_i) \rightarrow 1$. In Fig. (3.10), the migration of an evolving class from normal to failure is shown at several levels along with

© 2014 Tous droits réservés.

the severity indicator.

The shape of the signal of the severity indicator depends on the speed of drift and of the form of the trajectory of the drift. If the drift is fast, the values of the severity indicator tend to reach the value 1 in a fast way. If the drift is slow, this tendancy is thus slow, and more time is needed to reach the value 1. The form of the trajectory affects also the shape of the signal. Concerning this shape, three different stages can be considered:

1. A convexity in the beginning, which is associated to a start of an increase in the values of the severity indicator (Fig. (3.10,a)).
2. As drift continues, the severity indicator is considered as a non stationary times series with a trending behavior (Fig. (3.10,b)).
3. A concavity at the end, because the value of the severity indicator tends to 1, as the evolving class reaches the failure class (Fig. (3.10,c)).

It is also important to state that the intrinsic nature of the severity indicator means that when it is sufficiently close to the value 1, failure level is reached. However, in practical cases, this level is not desirable at all. A threshold for the value of the severity indicator must be defined. The next subsection will treat this case.

3.5.2.1 Threshold on the value of the severity indicator

Let us consider the example of the fuel level in a car. The warning light for fuel is not activated when there is no more fuel in the car. It is lighted before in order to give the driver few more miles to reach a petrol station. Analogously, when the value of the severity indicator tends to 1, it means that the evolving cluster is approaching more and more the failure mode cluster. It is often undesirable to reach that level. For this reason, the threshold to be used acts as the undesirable level of severity of drift. Let I_{th}^j be the threshold for the severity indicator $I_{sv}^j(t_i)$. There are two ways of defining I_{th}^j:

1. Heuristic method: based on expert knowledge or on maintenance specifications, a value can be given to I_{th}^j. For instance, given an industrial component subject to a degradation, one can consider that a value $I_{th}^j = 70\%$ is the maximum undesirable degradation level to reach.
2. Mathematical method: a mathematical approximation of the undesired level to reach can be calculated. In this work, this mathematical approximation of I_{th}^j is based on two assumptions:
   - The undesired level to reach corresponds to the first intersection of the boundaries of the evolving cluster and the failure cluster. In Fig. (3.11), this intersection is graphically illustrated.
   - The covariance matrix of the evolving cluster at this level is equal to the covariance matrix of the corresponding failure mode. This hypothesis is not necessarily very restrictive since when the evolving mode gets closer to the failure mode, it can be considered that the parameters of the two modes get also closer one to another.

   This is mathematically equivalent to:

© 2014 Tous droits réservés. doc.univ-lille1.fr

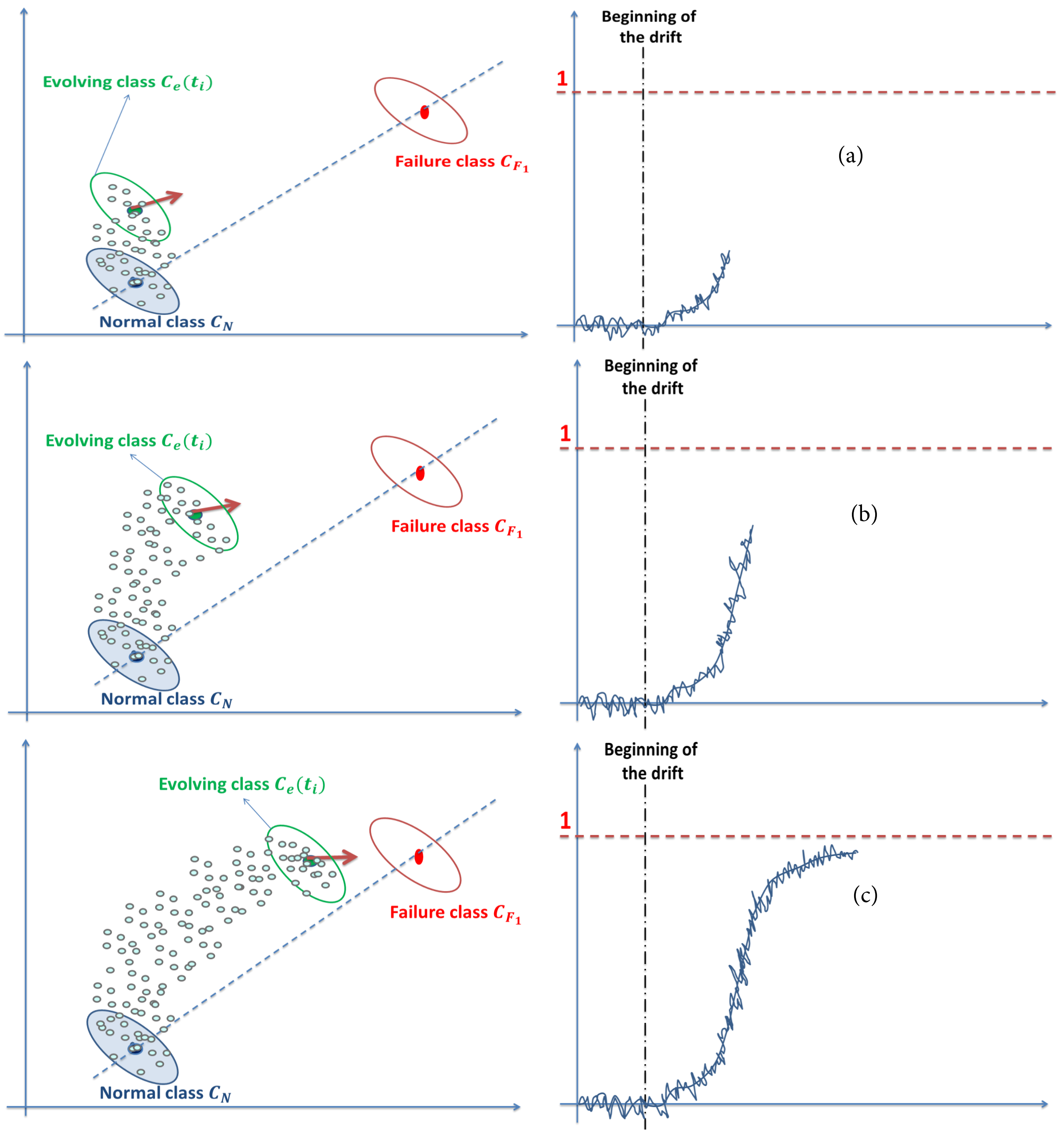


Figure 3.10: Evolution of the severity indicator in a drifting scenario.

$$
\begin{aligned}
I &= min\{i, Card(C_e(t_i) \cap C_{F_n}) = 1\}, & (3.15)\\
\Sigma_e(t_I) &= \Sigma_{F_n}. & (3.16)
\end{aligned}
$$

Given these assumptions, let (ζ) be the set of points on which $\mu_e(t_I)$ can be so that we have one intersection point between $C_e(t_I)$ and C_{F_n}. This intersection point is $X(t_I)$, or simply X_I for easier notations. The boundaries of $C_e(t_I)$ and C_{F_n} are defined by the parameter λ_M

© 2014 Tous droits réservés. doc.univ-lille1.fr

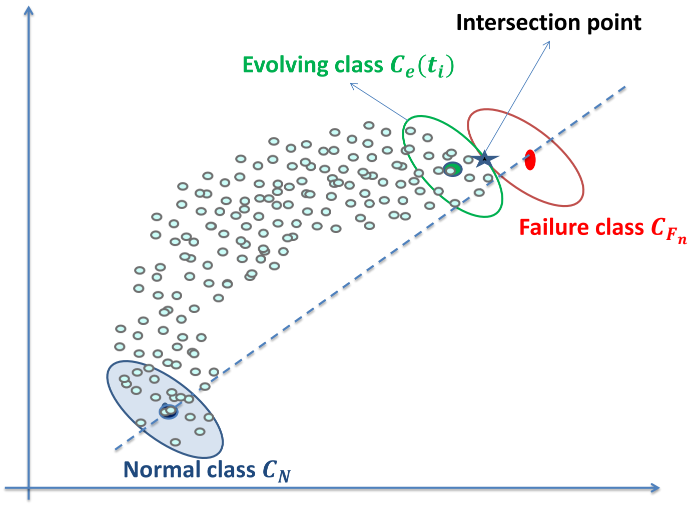


Figure 3.11: First intersection between the evolving cluster and the failure cluster as a basis for threshold calculation.

given to the similarity measure. To recall, $\lambda(X_i, C_j)$ is the similarity measure whose definition is given in Eq. (2.20) (chapter 2, section 2.5.1). Let also $\Sigma_e(t_I) = \Sigma_{F_n} = \Sigma_*$.

$\forall X_I \in (\zeta)$, we have:

$$\begin{cases} \lambda(X_I, C_e(t_I) = \lambda_M \\ \lambda(X_I, C_{F_n}) = \lambda_M \end{cases} \tag{3.17}$$

$$\Rightarrow \begin{cases} (X_I - \mu_e(t_I))^T \Sigma_*^{-1} (X_I - \mu_e(t_I)) = -2\ln(\lambda_M) \\ (X_I - \mu_{F_n})^T \Sigma_*^{-1} (X_I - \mu_{F_n}(t_i)) = -2\ln(\lambda_M) \end{cases} \tag{3.18}$$

$$\begin{aligned} \Rightarrow & (X_I - \mu_e(t_I))^T \Sigma_*^{-1} (X_I - \mu_e(t_I)) - \\ & (X_I - \mu_{F_n})^T \Sigma_*^{-1} (X_I - \mu_{F_n}(t_i)) = 0 && (3.19) \\ \Rightarrow & (2X_I - (\mu_{F_n} + \mu_e(t_I)))^T \Sigma_*^{-1} (2X_I - (\mu_{F_n} + \mu_e(t_I))) = 0 && (3.20) \\ \Rightarrow & 2X_I = \mu_{F_n} + \mu_e(t_I) && (3.21) \\ \Rightarrow & X_I = \frac{1}{2}(\mu_{F_n} + \mu_e(t_I)), && (3.22) \end{aligned}$$

By replacing the value of X_I in Eq. (3.18), we have :

$$\frac{1}{4}(\mu_e(t_I) - \mu_{F_n})^T \Sigma_*^{-1} (\mu_e(t_I) - \mu_{F_n}) = -2\ln(\lambda_M). \tag{3.23}$$

Thus, (ζ) is an ellipse centered at μ_{F_n}, and defined by the equation:

$$F_\zeta(X) = (X - \mu_{F_n})^T \Sigma_*^{-1} (X - \mu_{F_n}) = -8\ln(\lambda_M). \tag{3.24}$$

© 2014 Tous droits réservés. doc.univ-lille1.fr

The last step is the calculation of the threshold I_{th}^j. In order to explain how it is computed, let us remind the formula to calculate it, given in Eq. (3.14):

$$I_{sv}^j(t_I) = \frac{d_{KL}(C_N, C_e(t_I))}{d_{KL}(C_N, C_e(t_I)) + d_{KL}(C_e(t_I), C_{F_j})}. \tag{3.25}$$

By dividing by $d_{KL}(C_N, C_e(t_I))$, the expression of $I_{sv}^j(t_I)$ in Eq. (3.25) becomes:

$$I_{sv}^j(t_I) = \frac{1}{1 + \frac{d_{KL}(C_e(t_I), C_{F_j})}{d_{KL}(C_N, C_e(t_I))}}. \tag{3.26}$$

Under the constraint of intersection between $C_e(t_I)$ and C_{F_j}, corresponding to $\mu_e(t_I) \in (\zeta)$ (whose equation is given in Eq. (3.24)), the value of $d_{KL}(C_e(t_I), C_{F_j})$ is fixed. This result can be easily seen by using the equation of the Kullback-Leibler divergence (Eq. (3.13)) and the assumption about the equality of the covariance matrices of both $C_e(t_I)$ and C_{F_j}. In fact:

$$\begin{aligned} d_{KL}(C_e(t_I), C_{F_j}) &= \frac{1}{2} \times (tr((\Sigma_*)^{-1} \times \Sigma_* + (\Sigma_*)^{-1} \times \Sigma_*) \\ &+ (\mu_e(t_I) - \mu_{F_n})^T((\Sigma_*)^{-1} + (\Sigma_*)^{-1}) \times (\mu_e(t_I) - \mu_{F_n})) - d, \\ &= \frac{1}{2} \times (d.I + d.I) + 2 \times (\mu_e(t_I) - \mu_{F_n})^T \Sigma_*^{-1}(\mu_e(t_I) - \mu_{F_n}) - d, \\ &= -16 \ln(\lambda_M), \\ &= cste \end{aligned} \tag{3.27}$$

Thus, the only remaining variable affecting the value of $I_{sv_j}(t_i)$ is $d_{KL}(C_N, C_e(t_I))$. In order to find the appropriate value of $d_{KL}(C_N, C_e(t_I))$, a last observation should be made. The migration of the evolving cluster towards the failure cluster can be linear or slightly non-linear (but always within the scope of the direction indicator), as illustrated in Fig. (3.4).

Since the degradation trajectory is not known *a priori*, and in order to have a higher security level for the value of I_{th}^j, it can be argued that minimizing the Eq. (3.26) is required. But the minimization of the Eq. (3.26) means the minimization of $d_{KL}(C_N, C_e(t_I))$. This is achieved when:

$$\mu_e(t_I) \in (\zeta) \cap (\mu_e(t_I)\mu_{F_j}), \tag{3.28}$$

where $(\mu_e(t_i)\mu_{F_j})$ is the axe relating the centers $\mu_e(t_I)$ to μ_{F_j}. Once this value is calculated, the value of I_{th}^j can be calculated.

This achieves the threshold estimation or calculation for the severity indicator. Once the thresholds I_{th}^j, $j = 1, \dots n_F$ have been chosen or estimated, the prediction for prognosis can be made. Prognosis modeling is the subject of the next section.

© 2014 Tous droits réservés. doc.univ-lille1.fr

3.6 Prognosis modeling - Prediction

Being able to provide a RUL (Remaining Useful Life) estimate is another one main contribution of the work in this thesis. The prognosis has a sense only when the failure mode causing a drift is known, and can be characterized by a failure class. In order to be able to give an estimate of the RUL, the dynamics of the evolution of the drift must be approximated. Once done, the knowledge of these dynamics allows us to make predictions onto the future, allowing us thus to estimate a RUL. The question arises: how to use the indicators developed above to make the prediction?

The direction indicator is used to pinpoint the failure mode. It is the severity indicator that gives the actual health state of the process. Thus, in order to predict future healthy states, the dynamics of the severity indicator must be modeled. Thus, the focus is on the severity indicator.

The severity indicator is computed at each instant. In fact, up until now, three types of models have been seen. The first is concerned about the dynamical modeling of the evolving class. The second type of model was the trajectories of the drift, when an evolving cluster is migrating towards a failure cluster (see Fig. (3.4)). Given some dynamics of evolution of a cluster, defined over a model of trajectory, the severity indicator is calculated at each instant, giving a third, time series model, which will be used for prognosis.

The search for the best prognosis model is certainly related to the challenges that were stated earlier. The prognosis model that needs to be developed must be compatible with unknown degradation mechanisms and reactive to operating conditions. Thus, there are two main difficulties that need to be tackled:

1. The model to choose: The prognosis model that is developed in this work is based on regression and polynomial techniques for prediction. The idea is to model the time series that is the severity indicator, and project its values into the future. By using the threshold defined for the severity indicator, a RUL can be estimated. The choice of these models is backed up by the fact that they are simple techniques, and that they can be used as a solution in our work. This difficulty is related to the advantages that a model can provide compared to the other since the degradation mechanisms are unknown.
2. The model's reactivity and parameter update: the RUL estimation depends on the parameters of a prognosis model up to the current time. In order to be reactive, the parameters of a prognosis model should always updated as new data arrive. The aspect of the continuous update of the prognosis model's parameters is possible because of the use of a dynamical decision space. The biggest advantage of using a dynamical decision space is that it allows building a prognosis model characterized by a high adaptability to following the evolution of drifts. Using the dynamics of the severity indicator, another big advantage is that an extensive knowledge of the degradation mechanisms is not required. The difficulty in this case is related to the knowledge of which is the best update scheme of the prognosis model.

The next subsections are dedicated to:

© 2014 Tous droits réservés. doc.univ-lille1.fr

- Studying polynomial and regression models that can be used to make predictions. The estimation of RUL using these models is also shown.
- Studying the update schemes that can be used for the prognosis model.
- Showing the prognosis model that is developed in this work.

3.6.1 Regression and polynomial models

Regression and polynomial models models have long been used for predicting the value of a variable (Box et al., 2008; Enders, 2009; Brockwell and Davis, 2002b). The main techniques are the linear regression, the quadratic regression, the Auto-Regressive (AR), the Auto-Regressive Moving Average (ARMA) and the Auto-Regressive Integrated Moving Average (ARIMA) models. Before showing some related work for prognosis using these models, let us show their mathematical definitions.

3.6.1.1 Linear and quadratic regression

Consider a time series $z(t_i)$. Creating a *linear* regression model for prediction is based on the hypothesis that the variables z_{t_i} and the time t_i are related by a linear relationship. Suppose we have a sample of n observations of the couple (t_i, z_{t_i}), $i = 1, \ldots, n$. The hypothesis means that there exists a parameter vector, denoted θ, such as:

$$\hat{z}(t_i) = f(t_i) = \theta^T Z_{t_i}, \tag{3.29}$$

where $Z_{t_i} = \begin{bmatrix} t_i \\ 1 \end{bmatrix}$ $(i = 1, \ldots n)$, $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ and $\hat{z}(t_i) = f(t_i)$ is the prediction function. In Fig. (3.12,(A)), the points and the linear model relating the variables are illustrated.

The *quadratic* regression model supposes that the variables t_i and z_{t_i} are related by a quadratic relationship (see Fig. (3.12,(B))). In this case, $Z_{t_i} = \begin{bmatrix} t_i^2 \\ t_i \\ 1 \end{bmatrix}$ and $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$. The prediction function $\hat{y} = f(z)$ is supposed to be: $\hat{y} = f(z) = \theta^T Z_{t_i}$.

There exist different techniques to estimate the parameter vector θ (Box et al., 2008; Enders, 2009; Brockwell and Davis, 2002b). The most common technique is the Least Square (LS) method that consists in minimizing the LS criterion:

$$J_{LS} = \sum_{i=1}^{n} (z_{t_i} - \hat{z}_{t_i})^2. \tag{3.30}$$

For more details on the LS method (and other estimation methodologies), the reader is invited to consult the references (Box et al., 2008; Enders, 2009; Brockwell and Davis, 2002b).

© 2014 Tous droits réservés.

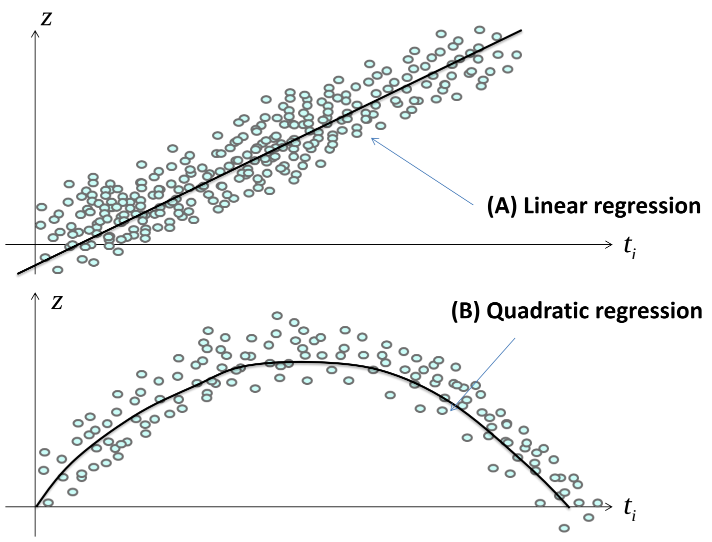


Figure 3.12: Linear and quadratic regression.

3.6.1.2 AR, ARMA and ARIMA models

Let $z(t_i)$ (or also z_{t_i}) be a time series defined at time stamps t_i, $i = 1, \ldots, n$. An AR(p) model, of order p, is defined by:

$$z(t_i) = \theta^T Z(t_i) + a(t_i), \tag{3.31}$$

where

$$\begin{aligned}
p &= \text{the AR model order,}\\
\theta &= \begin{bmatrix}\theta_1\\ \theta_2\\ \vdots\\ \theta_p\end{bmatrix}, \text{ the AR parameter vector,}\\
Z(t_i) &= \begin{bmatrix}z(t_{i-1})\\ z(t_{i-2})\\ \vdots\\ z(t_{i-p})\end{bmatrix}, \text{ the vector containing the last } p \text{ values of the time series,}\\
a(t_i) &= \text{a random component following a Gaussian distribution with zero mean.}
\end{aligned}$$

An ARMA(p,q) process, of orders p and q is given by:

$$z(t_i) = \theta^T Z(t_i) + \Psi^T A(t_i), \tag{3.32}$$

where

© 2014 Tous droits réservés. doc.univ-lille1.fr

$$
\begin{aligned}
p, q &= \text{the ARMA model orders,} \\
\theta &= \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix}, \text{ the AR parameter vector,} \\
\Psi &= \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_q \end{bmatrix}, \text{ the MA parameter vector,} \\
Z(t_i) &= \begin{bmatrix} z(t_{i-1}) \\ z(t_{i-2}) \\ \vdots \\ z(t_{i-p}) \end{bmatrix}, \text{ the vector containing the last } p \text{ values of the time series,} \\
A(t_i) &= \begin{bmatrix} A(t_{i-1}) \\ A(t_{i-2}) \\ \vdots \\ A(t_{i-q}) \end{bmatrix} \text{ a random vector component following a Gaussian distribution with zero mean.}
\end{aligned}
$$

AR and ARMA models are more suited to model stationary series (Brockwell and Davis, 2002b). Their mathematical definition allows us to define the ARIMA model, which is more suited to model non-stationary time series. In fact, a non-stationary time series process has no fixed level, but has a drifting or trending behavior. A non-stationary process can be converted to a stationary process by differentiating. For example, upward trend (which is the case of the severity indicator developed in this work) is a case of non-stationarity that can be flattened by taking the difference of successive values. Let Δ be the differencing operator, we have:

$$\Delta z(t_i) = z(t_i) - z(t_{i-1}). \tag{3.33}$$

The differencing operator just defined can be applied several times, in such a way that:

$$\Delta^D z(t_i) = \Delta(\Delta^{D-1} z(t_i)). \tag{3.34}$$

An ARIMA(p,D,q) model can then be defined as:

$$z(t_i) = \sum_{j=1}^{p} \theta_j \Delta^D z_{t_{i-j}} + \Psi^T A(t_i) \tag{3.35}$$

The natural question that comes now concerns the prediction of the RUL that is done using these models. It is the subject of the next subsection.

© 2014 Tous droits réservés. doc.univ-lille1.fr

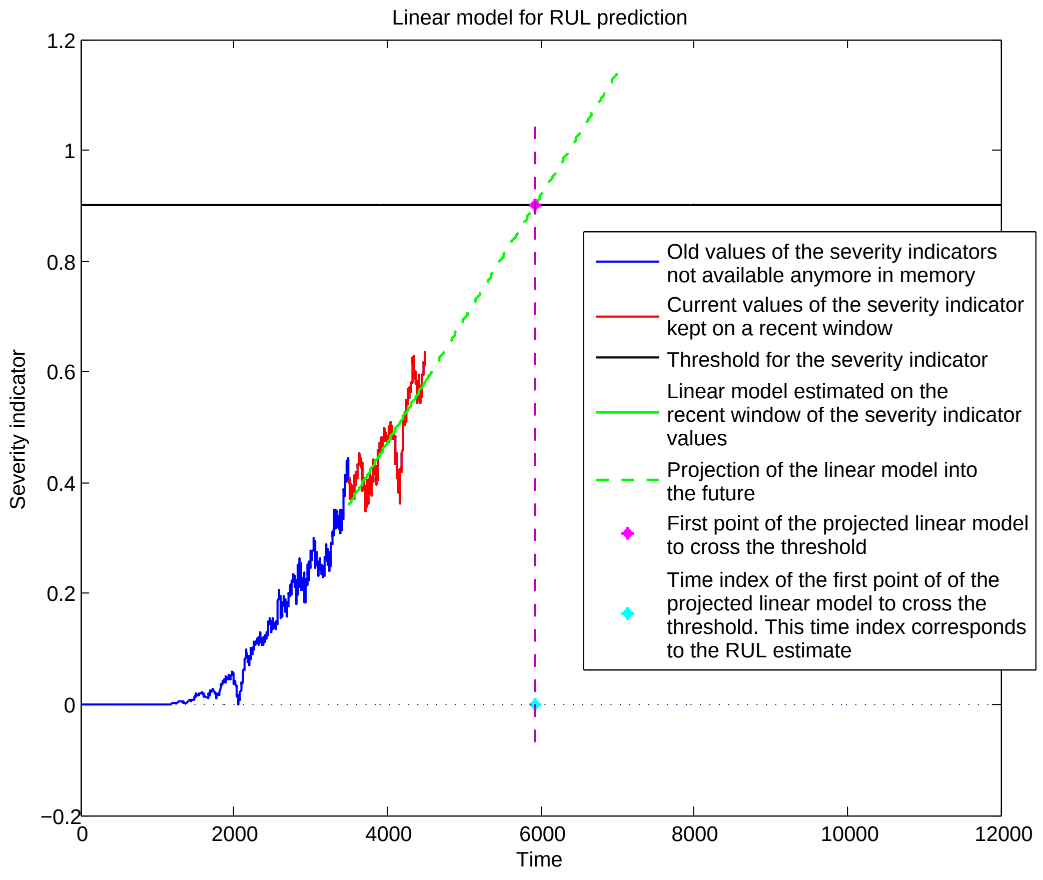


Figure 3.13: Prediction using a linear model.

3.6.1.3 Prediction

In this section, the prediction using each of the linear, quadratic and ARIMA model is shown. The prediction of the RUL is done by defining a model (linear, quadratic and ARIMA) on the severity indicator's most recent window of values in order to capture its most recent dynamics. The prediction of RUL that is done using these models depends on the model's parameters at each time it is computed, as well as the threshold that was set for the severity indicator. Thus, given these parameters and the thresholds, how is the prediction done?

In the linear and quadratic regression cases, the prediction of the remaining life is done by considering the prediction function detailed in Eq. (3.29). Then, a projection into future values is done until the threshold is reached. The time stamps t_i are the inputs, and the estimated severity indicator values are the output.

Let M_l and M_q denote respectively a linear model and a quadratic model, defined on a recent

© 2014 Tous droits réservés. doc.univ-lille1.fr

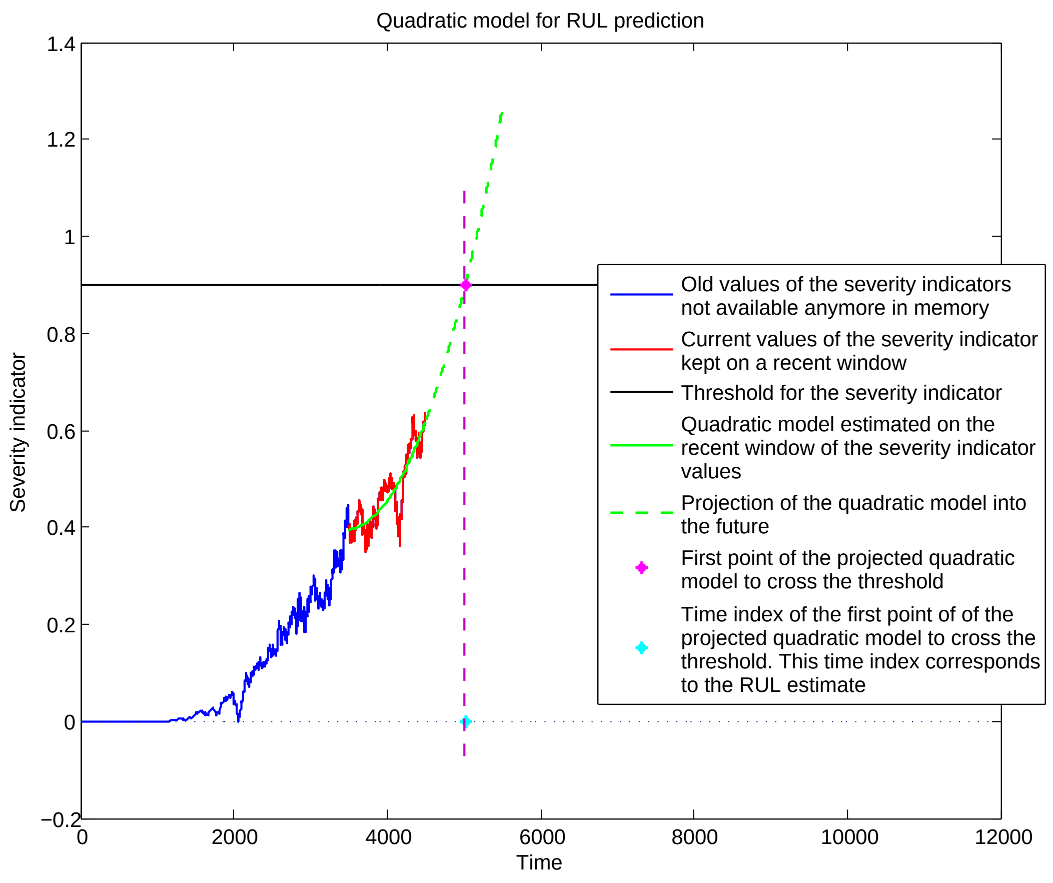


Figure 3.14: Prediction using a quadratic model.

window W. The value of the severity indicator projected k values into the future is denoted by $\hat{I}^j_{sv}(t_{i+k})$. Let $R_{t_i}(M_l)$ and $R_{t_i}(M_q)$ denote the estimated value of the RUL given respectively by the linear model and the quadratic model (see Figs. (3.13,3.14)). The estimated values of the RUL are mathemtically defined as:

$$R_{t_i}(M_l) = t_r, r = min\{k \in \mathbb{N}; \hat{I}^j_{sv}(t_{i+k}) > I^j_{th}\}. \tag{3.36}$$

The prediction using AR and ARMA models are not detailed. The reason is that these two types of models are, as stated earlier, suitable for modeling stationary time series.

The prediction using the ARIMA model is based on the n-step ahead prediction algorithms, also known as the multi-step prediction algorithms. The idea consists in estimating, iteratively, the future values of the time series. Each estimated value for a horizon k is used to make prediction for the horizon $k+1$. This is equivalent to making predictions on a horizon k, which is later on incrementally increased. For more details on the iterative multi-step prediction algorithm using ARIMA models, the reader can be referred to (Brockwell and Davis, 2002b). The increase of the prediction horizon

© 2014 Tous droits réservés. doc.univ-lille1.fr

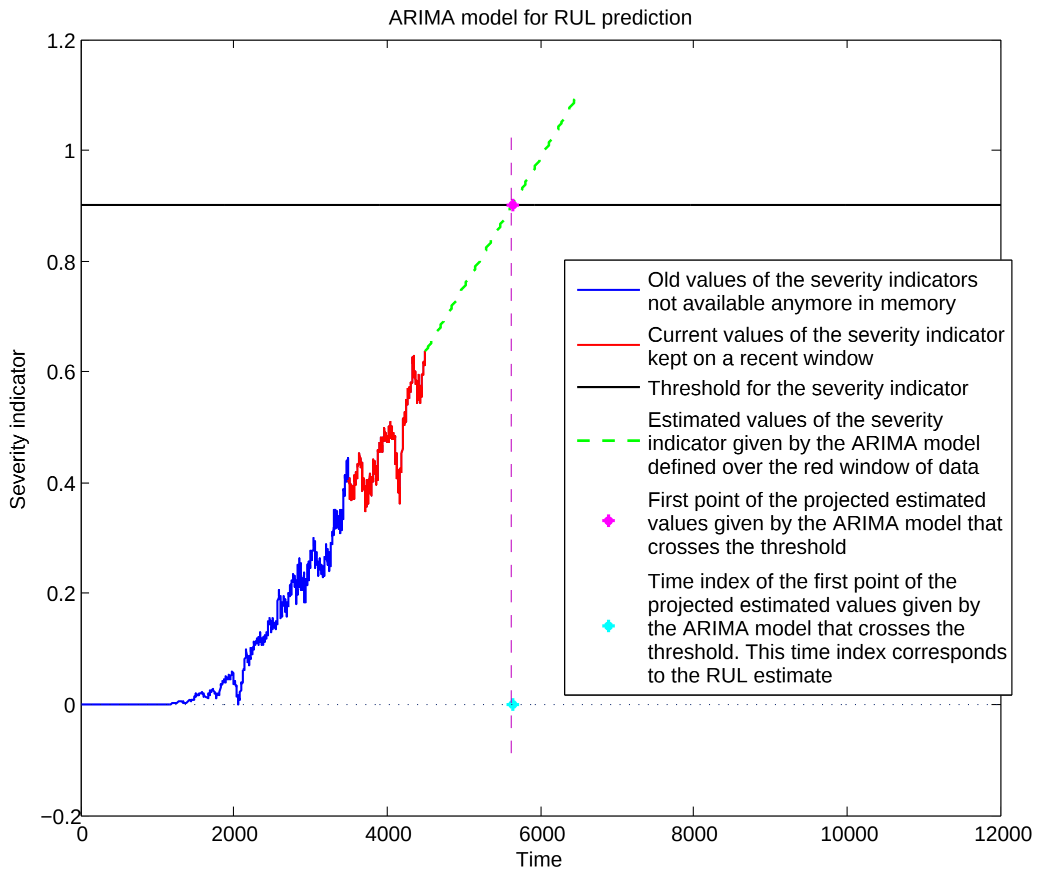


Figure 3.15: Prediction using an ARIMA model.

k continues until the condition in Eq. (3.37) is satisfied (see Fig. (3.15)). However, it is necessary to define a majoring bound for its value. The reason is to prevent multistep prediction towards infinity, in case the condition in Eq. (3.37) was not satisfied. This majoring bound is a user-defined parameter and is denoted H_M. It is chosen by the user according to the application. Let $R_{t_i}(M_a)$ denote the estimated value of the RUL given by the linear model. The condition to be satisfied is:

$$R_{t_i}(M_a) = t_r, r = min\{k \in \mathbb{N}; \hat{I}^j_{sv}(t_{i+k}) > I^j_{th}, k \leq H_M\}. \tag{3.37}$$

In case the majoring bound is reached and the condition is still not satisfied, the RUL will be given the value -1. In the same manner, for any case in which the RUL has no sense, it will be given the value -1. An example of such a case is when there is no detected drift. Another example is when there is a detected drift but the failure mode is unknown.

The next section is dedicated to finding the prognosis model that can answer for the challenges of the work in the thesis.

© 2014 Tous droits réservés. doc.univ-lille1.fr

3.6.2 Update scheme of a prognosis model

As stated earlier, the second difficulty is related to the reactivity of the model. In fact, the prediction that is done using any prognosis regression model depends on its parameters at each time it is computed. The reactivity of a prognosis model is achieved by continuously updating its parameters so that they reflect the most recent degradation's dynamics. This update of the parameters is done as new data corresponding to the degradation mechanism is available. There are two ways for update:

1. Moving window: by following the same logic of the moving window schemes, a model is defined on the most recent time window of data. As new data windows are available, the old model is dropped and a new model is created on them.
2. Recursive update: it is a instance-based update scheme in which each acquired data point contributes to the update of the parameters of the model. The update makes use of recursive formulas that can be found in several references, as for example (Brockwell and Davis, 2002b; Kadlec et al., 2011; Ljung, 1999).

The recursive update scheme requires the definition of a forgetting factor. It also requires defining initial conditions (initial parameters). The choice of these could affect the model's parameter estimation (Ljung, 1999). Consequently, there are two drawbacks associated with these recursive models:

– The forgetting factor is necessary to be used. Without it, the algorithm does not forget, thus the impact of the newly arrived data will tend to zero (no more impact).
– The initial conditions affect a lot the speed of convergence of the parameters. If initially the parameters are very different then the real parameters, the convergence can take too much time, which is an undesirable thing.

In this work, the recursive update scheme is abandoned. This allows avoiding these two main drawbacks of the recursive update scheme. Instead, the sliding window update scheme.

Using the sliding windows of data poses the problem of the window size. The latter is related to the stability-plasticity dilemma. A bigger window takes more data points and thus is adequate to represent slower evolutions of the degradation indicator. A smaller window represents better rapid evolution of the degradation indicator. Thus, the choice of the window is crucial and represents the major drawback of this update scheme. However, in this work, it is the sliding window update scheme that will be used because of its simplicity. Later, we will see how the problem of the window size is tackled.

The developed prognosis model is designed to take into consideration all the drawbacks cited above. It is presented in the next subsection.

© 2014 Tous droits réservés.
doc.univ-lille1.fr

3.6.3 Prognosis model

The prognosis model to be developed must take into consideration the two major difficulties that were defined. The first difficulty is related to the compatibility of the model to the unknown degradation mechanisms. The shape of the degradation signal, which in our case is the severity indicator, is supposed to be unknown. It was said earlier that, when a normal mode (characterized by the normal cluster) is migrating towards a failure mode (characterized by a failure cluster), the associated severity indicator tends asymptotically to 1. This yields that the shape of the severity indicator tends to have a certain concavity after a certain time. However, this does say nothing on the initial behavior of this signal, which can be linear. Thus, linear, quadratic and ARIMA models can be considered.

Considering only one regression model deprives the prediction result from the advantages of using another model. The polynomial models (linear and quadratic) are interesting to be taken into consideration because of the simplicity of the created models. In the same manner, if the behavior of the severity indicator becomes less linear for a long period of time, the quadratic regression model performs better in future projections. Analogously, linear regression models and ARIMA models perform better if the actual degradation behavior is linear for a certain period of time. The interest for the use of an ARIMA model was discussed earlier. ARIMA models are suited to model time series with drifting behavior. The differentiation removes this drifting behavior, making the time series stationary.

The second difficulty concerns the update scheme. The study shown above was concluded by choosing a sliding window update scheme. The major drawback concerned the choice of the size of the window.
In order to tackle these difficulties, the prognosis model developed in this work is based on **merging** several linear, quadratic and regression models, defined over several window sizes. This idea behind using several regression and polynomial models is to combine the advantages of each of the models, allowing the final merged model to be more robust to unknown degradation mechanisms. In addition, in order to tackle the problem of the window size, several window sizes are defined. The lack of knowledge associated to the speed of drift makes the choice of a single window size hard to do. Thus, choosing several window sizes is one way to cope with that difficulty. Each single model will be called *trending model*, since it can be used to estimate the trend of the severity indicator, allowing a RUL estimation.

Let:
- K_M be the number of trending models,
- K_W be the number of data window sizes,
- W^z, $z = 1, \ldots, K_W$ the different data windows.

The number of trending models is divided over the three families of methods to be used: ARIMA, linear and quadratic models.

© 2014 Tous droits réservés. doc.univ-lille1.fr

- K_{M_a} the number of ARIMA models,
- K_{M_l} the number of linear regression models,
- K_{M_q} the number of quadratic regression models,
- M_a^z, $z = 1, \ldots, K_{M_a}$ denote the different ARIMA models used. Let $R_{t_i}(M_a^z)$, $z = 1, \ldots, K_{M_a}$ denote the different RUL estimate obtained by the ARIMA models at time stamps t_i,
- M_l^z, $z = 1, \ldots, K_{M_l}$ denote the different linear regression models used. Let $R_{t_i}(M_l^z)$, $z = 1, \ldots, K_{M_l}$ denote the different RUL estimate obtained by the linear regression models,
- M_q^z, $z = 1, \ldots, K_{M_q}$ denote the different quadratic regression models used. Let $R_{t_i}(M_q^z)$, $z = 1, \ldots, K_{M_q}$ denote the different RUL estimate obtained by the quadratic regression models.

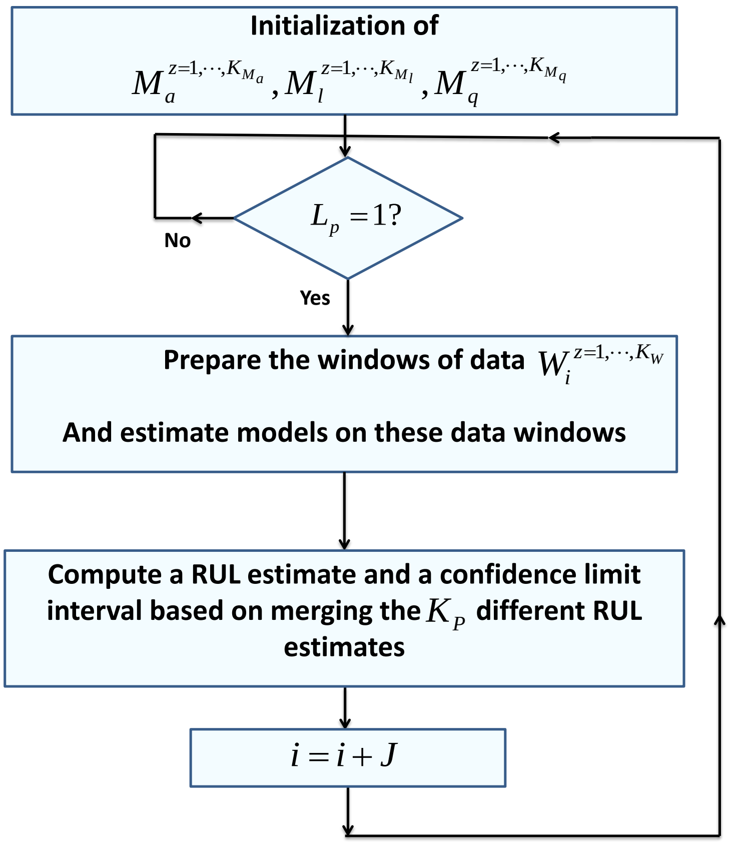

Initialization of
$M_a^{z=1,\cdots,K_{M_a}}, M_l^{z=1,\cdots,K_{M_l}}, M_q^{z=1,\cdots,K_{M_q}}$
$L_p = 1?$
No
Yes
Prepare the windows of data $W_i^{z=1,\cdots,K_W}$
And estimate models on these data windows
Compute a RUL estimate and a confidence limit interval based on merging the K_P different RUL estimates
$i = i + J$

Figure 3.16: Prognosis model algorithm.

The prognosis algorithm is depicted in Fig. (3.16). This algorithm computes a prediction of the RUL each J time steps, where J is a user-defined parameter. In fact, the severity indicator values are available to the prognosis algorithm within a sampling time. There is a difference between the sampling time of the severity indicator and the sampling time that corresponds to the computation of a prognosis result. For example, if the sampling time of the severity indicator is in seconds, and the drift horizon is around months, it is definitely not needed to calculate a RUL estimate each second.

© 2014 Tous droits réservés. doc.univ-lille1.fr

The parameter J insures the consistency between the differences in sampling period interpretation.

The initialization phase corresponds to the definition of the parameters. An external binary signal, denoted L_P, launches the algorithm, which starts to compute RUL results. When the value of L_P goes from 0 to 1, the prognosis algorithm is launched. The value of L_P is related to the detection of a drift, and to the isolation of the failure mode. Thus, its value is not equal to one unless the two conditions of detection and isolation are verified. The time stamp of the launch of the prognosis algorithm is denoted P, with respect to the notations of Chapter 1. Then the rest of the algorithm is constituted of three steps that are repeated:

1. Estimate a linear, quadratic and ARIMA models on the most recent windows of data (these windows are the ones referred to as W^z, $z = 1, \ldots, K_W$). The data inside these windows are values of the severity indicator $I_{sv}^j(t_i)$ corresponding to the migration of the normal operating mode to the isolated failure mode j.
2. Compute an estimate of the RUL as well as a confidence interval associated to it. In fact, since there are K_M different models and K_W different windows, then there is $K_P = K_M \times K_W$ different RUL estimates. The final RUL estimate is a merging of all these values. There are several possibilities for merging rules. A common method is to consider a *linear combination* in which the values of the different RUL estimates are given equal weights. This is equivalent to taking the **mean** of all the estimated RUL values, given by the different models. Some authors (Firmino et al., 2014) consider correction mechanisms on the estimated models that are used to adjust the values of the weights. However, in order to do so, correction mechanisms must be based on residuals, *i.e.* the difference between the estimated values and the real values. In our application, this is not possible since the values of the real RUL are unknown. Another possibility is to consider the final RUL estimate as a function (**min** or **max** for instance) of all the estimated RUL values. However, this could lead to taking extreme values of RUL estimates, which is unwanted specially in case where one of the models give results that are not very similar to the other models. Consequently, in this work, the merging RUL that is considered is the **mean** rule. In the next subsection, more reasons justifying this choice are presented. Also, the merging rule and the equations for the RUL and the confidence interval estimation are shown.
3. Once the RUL estimate and the confidence interval are computed, the algorithm proceeds by waiting for new data. This is equivalent to increasing the time index by the jump factor J, which was defined earlier.

3.6.3.1 Estimation of a RUL and a confidence interval

As stated earlier, K_P RUL values are computed every J time steps, starting from the launching time stamp P. At each of these time steps, the RUL value computed by the prognosis algorithm developed in this work is based on merging those K_P values. The merging rule is the *mean* rule because the *min* or the *max* rules tend to choose the extreme RUL estimates results. This is not acceptable since if one of the models gave a RUL estimate which is further then the other models,

© 2014 Tous droits réservés. doc.univ-lille1.fr

only the *mean* merging rule can overcome this difficulty. The equation giving the final RUL estimate is thus:

$$RUL(t_i) = \frac{1}{K_P}(\sum_{z=1}^{K_{M_a}} R_{t_i}(M_a^z) + \sum_{z=1}^{K_{M_l}} R_{t_i}(M_l^z) + \sum_{z=1}^{K_{M_q}} R_{t_i}(M_q^z)). \tag{3.38}$$

Let $I = [U_b, L_b]$ be the confidence interval, associated to a RUL estimate. The upper bound of the interval is denoted by U_b. The lower bound of the interval is denoted by L_b. These bounds are calculated by using the different K_P RUL estimates given by the different single models. Let $\sigma_R(t_i)$ be the standard deviation of the different RUL estimates given by the different single models. We define:

$$\begin{aligned} U_b(t_i) &= RUL(t_i) + \gamma_s \sigma_R(t_i), & (3.39)\\ L_b(t_i) &= RUL(t_i) - \gamma_s \sigma_R(t_i), & (3.40)\end{aligned}$$

where γ_s is a user-defined sensitivity factor, verifying: $\gamma_s \in \mathbb{R}^+$. γ_s indicates how many standard deviations of change around the mean estimate of RUL are accepted to form the boundaries. The calculation of $\sigma_R(t_i)$ is done using these equations:

$$\begin{aligned} \Delta R_{t_i}(M_a^z) &= R_{t_i}(M_a^z) - RUL(t_i), & (3.41)\\ \Delta R_{t_i}(M_l^z) &= R_{t_i}(M_l^z) - RUL(t_i), & (3.42)\\ \Delta R_{t_i}(M_q^z) &= R_{t_i}(M_q^z) - RUL(t_i), & (3.43)\\ \Delta_{t_i}^a &= \sum_{z=1}^{K_{M_a}} (\Delta R_{t_i}(M_a^z))^2, & (3.44)\\ \Delta_{t_i}^l &= \sum_{z=1}^{K_{M_l}} (\Delta R_{t_i}(M_l^z))^2, & (3.45)\\ \Delta_{t_i}^q &= \sum_{z=1}^{K_{M_q}} (\Delta R_{t_i}(M_q^z))^2, & (3.46)\\ \sigma_R(t_i) &= \sqrt{\frac{1}{K_P}(\Delta_{t_i}^a + \Delta_{t_i}^l + \Delta_{t_i}^q)}, & (3.47)\end{aligned}$$

In order to show the effectiveness of the proposed methodology, in the next subsection, simulations of different scenarios of drift are shown.

3.6.4 Drift scenarios simulation for prognosis validation

In order to show the effectiveness of the prognosis model, several scenarios of a single drift are used. The drift corresponds to the migration of a normal mode C_N to a failure mode C_{F_j}. The difference between the scenarios is governed by the change of few parameters corresponding to the form of the trajectory of this drift and its speed.

© 2014 Tous droits réservés. doc.univ-lille1.fr

The parameters concerning the form of the trajectory of the drift are illustrated in Fig. (3.17). These parameters are: the angle β between the tangent to the curve at the center of the normal class and the axe relating the centers of the classes (normal and failure), the summum $M(x_M, y_M)$, the parameters (mean and covariance matrix) of the normal class C_N and the failure class C_{F_j} ($F(x_f, y_f)$ is the center of C_{F_j}).

The parameter controlling the speed of drift is the duration of drift, denoted by Δ_d. A slow drift corresponds to a big value of Δ_d. A fast drift corresponds to a small value of Δ_d. The ending of the duration is equivalent to the fact that the evolving class is overlapping the failure class.

The different parameters used to simulate these different scenarios are summarized in Table 3.1. The sampling period is 1. In these scenarios, we have $t_i = i$ since the sampling period is 1. All drifts start at 1000. The threshold has been approximated to $I_{th}^j = 0.9$ by applying the formulas shown in Section (1.5.1.1). Let P and eol be the time indexes corresponding to the launching time of the algorithm and the end of life (with respect to the notations defined in Chapter 1). The algorithm is launched for the values of t_P which are indicated in Table 3.1. Also, the values of t_{eol} are calculated for each scenario. For each of the scenarios, they correspond to the time point in which the severity indicator crosses the threshold I_{th}^j. The parameters of the classes corresponding to the normal operating mode and the failure mode (C_N and C_{F_j}), and other parameters that are common to all scenarios are:

$$\begin{aligned}
C_N &= \left((0,0), \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}\right) \\
C_{F_j} &= \left((x_f,0), \begin{bmatrix} 0.2 & 0 \\ 0 & 0.4 \end{bmatrix}\right) \\
I_{th}^j &= 0.9 \\
(x_f, y_f) &= (10, 0)
\end{aligned}$$

The evaluation of the prognosis model is based on the use of the three metrics defined in Chapter 1, with a little modification. The metrics are:

- $Met1$: it is based on the cumulative relative accuracy (CRA) metric. The CRA metric verifies $CRA \in [-\infty, 1]$ with one corresponding to the highest performance. Met1 is the same as CRA but it will be bounded by 0, such as $Met1 \; in \, [0, 1]$, with 0 corresponding to the worst score, and 1 to the highest.

- $Met2$: it is based on the α-λ performance metric, which is a True/False metric. Let True be given the value 1 and False the value 0. Met2 is the average of the values of True/False over all the period stretching from t_P to t_{eol}. Thus Met2 is bounded by 0 and 1 such as $Met2 \; in \, [0, 1]$, with 0 corresponding to the worst score, and 1 to the highest.

© 2014 Tous droits réservés. doc.univ-lille1.fr

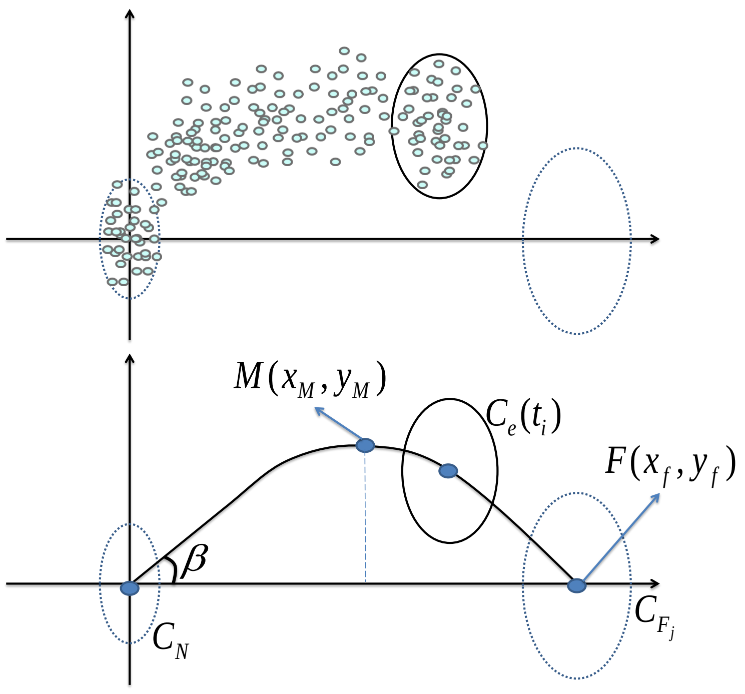


Figure 3.17: Parameters of the form of drift scenarios.

– $Met3$: this metric is related to the convergence metric (C_M). The higher the value of C_M is, the less successful is the prognosis model. Thus, it goes in the opposite way as the first two metrics. In addition, the convergence metric is unbounded by size contrarily to the first two metrics which are bounded by 0 (lowest score) and 1 (highest score). Consequently, in order to have the same order of magnitude for the three metrics, Met3 will be defined by:

$$Met3 = \frac{1}{1 + C_M}. \tag{3.48}$$

Since $C_M \in [0, \infty[$, with 0 corresponding to the best score, $Met3 \in [0, 1]$ with 0 corresponding to the worst score, and 1 to the highest. In this manner, the three metrics have the same order of magnitude.

Three sliding windows W^1, W^2 and W^3 were used in this simulation. Their sizes are respectively 500, 1000 and 1500. The jump factor $J = 100$, meaning that a RUL result is given by each of the models every 100 time steps. The numbers of models used in this simulation are:

– Three ARIMA models: M_a^1, M_a^2 and M_a^3, defined over W^1, W^2 and W^3.
– Three linear models: M_l^1, M_l^2 and M_l^3, defined over W^1, W^2 and W^3.
– Three quadratic models: M_q^1, M_q^2 and M_q^3, defined over W^1, W^2 and W^3.
– The ensemble model based on merging the results of these 9 models, denoted M_{en}.

© 2014 Tous droits réservés.

Table 3.1: Parameters corresponding to different drift scenarios.

Drift Scenario	β	y_M	t_{eol}	Δ_d	t_P
Scenario 1.1	30	2	5210	6000	2500
Scenario 1.2	20	1	5061		
Scenario 1.3	10	0.5	5109		
Scenario 2.1	30	2	6183	7000	2600
Scenario 2.2	20	1	5920		
Scenario 2.3	10	0.5	5914		
Scenario 3.1	30	2	6875	8000	2700
Scenario 3.2	20	1	6643		
Scenario 3.3	10	0.5	6475		
Scenario 4.1	30	2	7586	9000	2800
Scenario 4.2	20	1	7377		
Scenario 4.3	10	0.5	7188		
Scenario 5.1	30	2	8497	10000	2900
Scenario 5.2	20	1	7990		
Scenario 5.3	10	0.5	8004		

In Fig. (3.18), an example of a simulation of one scenario is shown. The real values of the RUL, the estimated values and the confidence interval are also shown in Fig. (3.18). The real values of the RUL are calculated on the period stretching from t_P to t_{eol}. The equation of the real RUL is:

$$realRUL(i) = \begin{cases} 0 \text{ if } i < P \\ -i + eol \text{ if } P \leq i \leq eol \\ 0 \text{ if } i \geq eol \end{cases} \tag{3.49}$$

The estimated RUL values is the result of the merging of different RUL estimates, computed each J time steps. The boundaries are calculated as explained earlier. In the example shown in Fig. (3.18), the value of γ_s is 2. It can be seen that the real values of the RUL are often inside the boundaries defined by the green lines. The latter define an envelop surrounding the real values. However, at some interval of times, the real values of the RUL were present outside this envelop. A way to overcome this difficulty is by increasing the value of the sensitivity factor γ_s. Nevertheless, the value of γ_s should not be chosen to be very high. If this is the case, the envelop becomes very wide meaning that the confidence interval looses its importance.

After the simulation of each of these 15 scenarios, the performance metrics defined above are calculated at the end of the simulation, over the period stretching from t_P to t_{eol}, in each of these scenarios. The results of the metrics of each of the scenarios are summarized in Table (3.2). In addition, the mean values of the metrics, calculated over the 15 scenarios are shown in Table (3.2). The highest scores were highlighted in bold.

© 2014 Tous droits réservés. doc.univ-lille1.fr

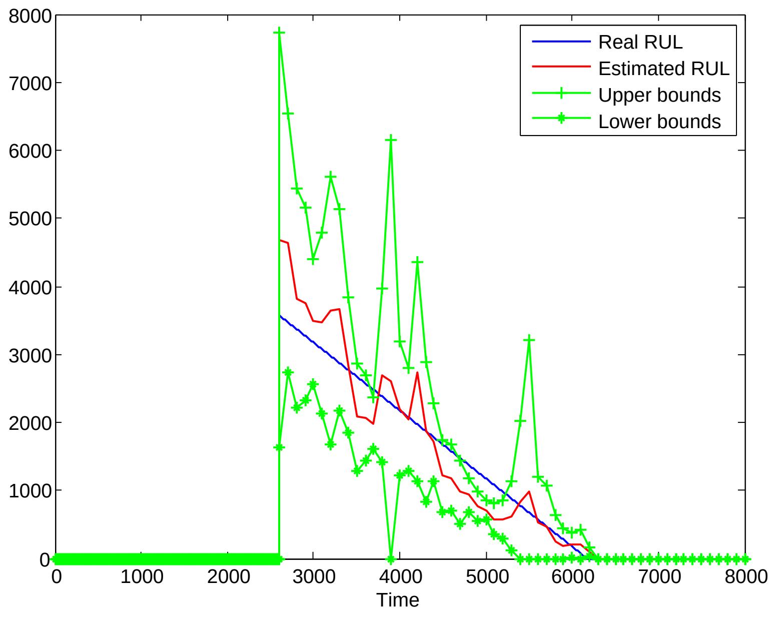


Figure 3.18: An example of simulation of one scenario showing the real values of the RUL, the estimated values and the confidence interval.

Table 3.2: Results of simulation of 15 scenarios for different prognosis models. The values in bold correspond to the highest scores.

Scenario 1.1

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,827	0,817	0,803	0,723	0,722	0,735	0,831	0,828	0,833	**0,836**
Met2	**0,607**	0,357	0,321	0,393	0,393	0,357	0,643	**0,607**	**0,607**	0,500
Met3$\times 10^3$	**1,112**	0,983	0,894	0,901	0,867	0,797	0,961	0,984	0,913	0,979

Scenario 1.2

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,846	0,828	0,797	0,799	0,783	0,743	0,880	**0,886**	0,830	0,860
Met2	0,308	0,385	0,346	0,500	0,462	0,385	**0,654**	0,615	0,423	0,523
Met3$\times 10^3$	0,858	0,870	1,035	0,752	0,844	1,020	0,844	1,094	**1,113**	1,007

Scenario 1.3

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,814	0,801	0,779	0,745	0,739	0,705	0,810	**0,865**	0,794	0,810
Met2	0,370	0,333	0,333	0,407	0,407	0,333	**0,704**	0,630	0,407	0,407
Met3$\times 10^3$	0,912	0,962	1,030	0,804	0,874	0,980	1,009	**1,104**	1,063	1,029

Scenario 2.1

© 2014 Tous droits réservés. doc.univ-lille1.fr

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,747	0,824	0,805	0,806	0,798	0,766	0,579	0,831	0,746	**0,836**
Met2	0,361	0,528	0,500	0,528	0,543	0,528	0,361	**0,556**	**0,556**	**0,556**
Met3$\times 10^3$	0,671	0,674	**0,805**	0,601	0,567	0,747	0,509	0,627	0,781	0,723

Scenario 2.2

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,684	0,684	0,690	**0,745**	0,718	0,619	0,687	0,772	0,671	0,725
Met2	0,324	0,265	0,265	**0,441**	0,353	0,265	0,353	0,412	0,324	0,324
Met3$\times 10^3$	0,782	0,742	0,718	0,867	**0,854**	0,542	0,813	0,845	0,504	0,770

Scenario 2.3

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,757	**0,784**	0,755	0,763	0,709	0,627	0,614	0,776	0,633	0,729
Met2	0,382	0,382	0,324	0,441	0,412	0,294	0,471	**0,559**	0,382	0,471
Met3$\times 10^3$	0,806	**0,811**	0,771	0,796	0,706	0,469	0,758	0,696	0,446	0,729

Scenario 3.1

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,635	0,699	0,674	0,736	0,740	0,699	0,259	0,588	0,713	**0,742**
Met2	0,262	0,310	0,238	0,405	0,357	0,262	0,333	0,381	0,262	**0,405**
Met3$\times 10^3$	0,544	**0,576**	0,568	0,351	0,709	0,521	0,260	0,596	0,501	0,571

Scenario 3.2

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,682	**0,711**	0,699	0,703	0,639	0,532	0.48	0,575	0,518	0.672
Met2	0,325	0,325	0,225	**0,475**	0,400	0,275	0,350	0,375	0,275	0.400
Met3$\times 10^3$	0,612	0,618	**0,625**	0,298	0,385	0,268	0,015	0,399	0,265	0.407

Scenario 3.3

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,771	0,797	0,769	**0,804**	0,759	0,688	0,661	0,623	0,642	0,764
Met2	0,500	0,474	0,395	0,553	0,421	0,368	0,368	0,474	0,447	**0,553**
Met3$\times 10^3$	0,652	**0,769**	0,704	0,557	0,630	0,418	0,640	0,637	0,414	0,609

Scenario 4.1

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,626	0,786	0,783	0,803	**0,829**	0,780	0,410	0,702	0,693	0,798
Met2	0,333	0,500	0,417	0,542	0,563	0,500	0,313	**0,583**	0,563	0,500
Met3$\times 10^3$	0,473	0,558	0,540	0,453	0,570	0,533	0,247	0,555	0,457	**0,581**

Scenario 4.2

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}

© 2014 Tous droits réservés. doc.univ-lille1.fr

Met1	0,708	0,752	0,736	**0,781**	0,716	0,643	0,705	0,767	0,710	0,772
Met2	0,304	0,370	0,261	**0,478**	0,370	0,348	0,413	0,457	0,391	0,391
Met3$\times 10^3$	0,531	0,552	0,542	**0,621**	0,505	0,358	0,557	0,516	0,343	0,520

Scenario 4.3

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,795	0,787	0,772	0,820	0,772	0,699	0,786	0,797	0,762	**0,825**
Met2	0,341	0,318	0,318	**0,500**	0,409	0,295	0,455	0,364	0,409	0,455
Met3$\times 10^3$	0,614	0,602	0,568	**0,728**	0,638	0,438	0,649	0,636	0,426	0,640

Scenario 5.1

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,580	0,678	0,701	0,795	0,800	0,769	0,493	0,788	**0,812**	0,804
Met2	0,196	0,339	0,357	0,554	0,571	0,500	0,357	0,571	**0,643**	0,607
Met3$\times 10^3$	0,411	0,448	0,441	0,463	0,396	0,475	0,379	0,445	**0,541**	0,457

Scenario 5.2

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,692	0,700	0,729	**0,791**	0,723	0,622	0,594	0,633	0,596	0,730
Met2	0,392	0,294	0,333	**0,549**	0,451	0,353	0,255	0,373	0,392	0,353
Met3$\times 10^3$	0,476	0,476	**0,480**	0,468	0,348	0,226	0,378	0,357	0,222	0,327

Scenario 5.3

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,601	0,650	0,627	**0,763**	0,711	0,615	0,708	0,727	0,673	0,665
Met2	0,327	0,365	0,327	**0,577**	0,519	0,423	0,423	0,538	0,442	0,442
Met3$\times 10^3$	0,488	**0,503**	0,477	0,372	0,315	0,205	0,302	0,265	0,194	0,273

Mean metric values

	M_a^1	M_a^2	M_a^3	M_l^1	M_l^2	M_l^3	M_q^1	M_q^2	M_q^3	M_{en}
Met1	0,718	0,753	0,741	0,772	0,746	0,683	0,556	0,744	0,708	**0,778**
Met2	0,356	0,370	0,331	0,489	0,445	0,366	0,430	**0,500**	0,435	0,463
Met3$\times 10^3$	0,663	0,676	**0,680**	0,502	0,614	0,533	0,555	0,560	0,545	0,658

Also, as stated earlier, two bounds are calculated: the upper bound and the lower bound, forming a confidence interval. The percentage of times the real RUL values has fallen inside the confidence interval is summarized in Table (3.3) for three different values of γ_s. The variable denoting this percentage is $P_{cli}^{\gamma_s}$.

The results obtained from these simulations show these major results:

© 2014 Tous droits réservés. doc.univ-lille1.fr

Scenario	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2	3.3	4.1	4.2	4.3	5.1	5.2	5.3	Avg
$P_{cli}^{\gamma_s=1}$	43	53	48	70	43	56	81	78	60	86	59	62	75	71	62	64
$P_{cli}^{\gamma_s=2}$	70	76	76	86	68	78	88	86	77	90	79	84	90	81	80	80.6
$P_{cli}^{\gamma_s=3}$	75	80	82	89	73	91	93	87	85	98	84	89	97	85	87	86.3

Table 3.3: Percentage of times the estimated RUL has fallen inside the confidence interval.

1. The overall performance of the prognosis model based on an ensemble of trending models is quite high. The model showed the highest CRA score and relatively high scores in the other two metrics when the averages of all of these metrics were calculated (Last part of the Table (3.2)).
2. The scenarios covered different drifting speeds. It can be seen from the results that different models performed more or less better than others for a particular scenario. However, the scores changed when the scenario changed. This means that a prognosis model, based on a single trending algorithm, can perform well in one case but less in another. The reason is that the combination of the update scheme and the size of the moving window can be more adapted to one case then to the other. The prognosis model based on the ensemble of the trending algorithms had the highest scores in some cases, and close to the highest in other cases.
3. The confidence interval showed different results according to the value of the sensitivity parameter. A 100% score of the real RUL values falling inside the limits defined by the boundaries (see Fig. (3.18 for a graphical illustration) was never reached. However, the bigger the sensitivity factor is, the higher scores for these percentages are. This result is natural since by increasing the sensitivity factor, the boundaries are more wide. The scores for the percentages showed acceptable results starting from the value $\gamma_s = 2$, in which the mean percentage score is above 80%.

3.7 Conclusion

In this Chapter, different tools were developed to answer the challenges defined in the beginning of this thesis. These tools covered different aspects, namely drift detection, drift characterization and prognosis modeling.

The drift detection module is based on the evolution of the operating modes. When the change becomes significant, the drift detection module must signal this change as fast as possible. In this work, this module is based on a hypothesis test, namely the hoteling T^2 statistic.

The drift characterization module is developed to assess the severity of a drift and the direction of the drift. The direction indicator is used to pinpoint the failure mode behind the drift. Once a failure mode is known to cause a drift, the severity indicator serves to assess the severity of this drift. A drift becomes more and more severe when the evolving operating mode approaches the failure mode.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Thèse de Antoine Chammas, Lille 1, 2014

The prognosis model uses the severity indicator to estimate the RUL. In addition, a confidence interval is associated to the estimated RUL. The prognosis model developed in this work makes use of an ensemble of regression models. The basic idea of this approach is to minimize the disadvantages of individual regression models, and to maximize their advantages by combining them together. Several simulated drift scenarios were made to show the interest of the prognosis approach of this work.

Although all these modules have been developed, it does not mean that the challenges are met. There is yet an assembly work to be done. In fact, the next step will be to combine all these modules together, to form a supervision, diagnosis and prognosis architecture that is adequate to the CBM architecture. This will be the subject of the next Chapter.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Chapter 4

Architecture for supervision, diagnosis and prognosis

4.1 Introduction

In the last chapter, we have developped tools for drift detection, dritft characterization and prognosis. The drift detection and drift characterization tools were based on the dynamical modeling of classes. The drift characterization tools resulted in creating an indicator for health assessment, the severity indicator, that later on was used by a prognosis model to provide RUL estimate, as well as a confidence interval associated to it.

The aim of this chapter is to propose an architecture of supervision, diagnosis and prognosis using these tools. This architecture comes as an answer to the challenges that were defined in this thesis. To remind, these challenges are:

1. How to integrate an alarm launching system, which detects a beginning of drift?
2. How to assess the health of the process?
3. Given a drift scenario towards an identified failure mode, and given the actual drift dynamics, how much time left before this failure mode is reached?
4. What happens if a drift is actually happening but the failure mode behind this drift is not identified, or just unknown?

The architecture to be created is made up of several modules that regroup the tools of drift detection and characterization as well as a prognosis module. These modules are arranged together in such a way that they create the architecture itself. The starting point of this architecture is dynamical modeling using AuDyC. The next section is dedicated to the description of this architecture. The sections that come after are concerned with the experimental part of the thesis, which is used to test the developed methodology.

© 2014 Tous droits réservés. doc.univ-lille1.fr

4.2 Architecture for supervision, diagnosis and prognosis

A complex system is made up of several subsystems connecting with each other. A monitoring architecture that can provide online supervision, diagnosis and prognosis results should be global to an entire system. The architecture that is to be proposed in this work is based on features that extracted from historical data gathered on a complex system, as well as features that are generated online from data acquired on this system.

The historical data gathered on a system are used to build wit the help of the proposed architecture. These models constitute the basis for the generation of indicators for drift detection, drift characterization and prognosis. There are two scientific difficulties related to the treatment of the data:

- High amount of data corresponding to the different subsystems forming a complex system. The difficulty lies in treating high amounts of data. As a consequence, it is convenient to create smaller sets of data out of the intial large set. Let:
    1. X is the set of feature vectors, such as $X \in \mathbb{R}^{N \times d}$. N is the number of observations and d is the size of feature vectors or the dimension of the feature space,
    2. X_{sb}^{i} is a subset of X, whose index is i, such as $X_{sb}^{i} \subset X$,
    3. X_{hist} is the set of features acquired from historical data,
    4. X_{onl} is the set of features that are generated online,
    5. E_{sb}^{i} is a feature space spanned by the subset X_{sb}^{i}.
- Availability of failure data corresponding to different failure modes. There are failure modes for which historical data are available. There are failure modes for which these data do not exist. In the case where historical data are available for a failure mode, then it is possible to characterize this failure mode by a failure class in a feature space. In the opposite case, where no historical data are available for a failure mode, then this characterization is impossible.

The proposed architecture for supervision, diagnosis and prognosis is depicted in Fig. (4.1). It is denoted G-arch. The strategy of the G-arch is based on two mechanisms:

- Separation of the set of features $X = \{X_{hist}, X_{onl}\}$. This mechanism will result in the creation of smaller, easier to treat data sets, denoted by X_{sb}^{i}, $i = 1, \ldots, N_s$.

- Treating subsets of features using smaller architectures, which are embedded to the G-arch. These smaller architectures are denoted E-arch, for Elementary architecture. They are referred to by $E - arch_i$, $i = 1, \ldots, N_{ea}$. E-archs regroup different modules: (a) a modeling module using AuDyC, (b) a drift detection module, (c) a drift characterization module, and (d) a prognosis module.

Before developing and detailing those two mechanisms, it is important to show the reasons why the separation mechanism is needed. In fact, the separation mechanism comes as a first step to

© 2014 Tous droits réservés. doc.univ-lille1.fr

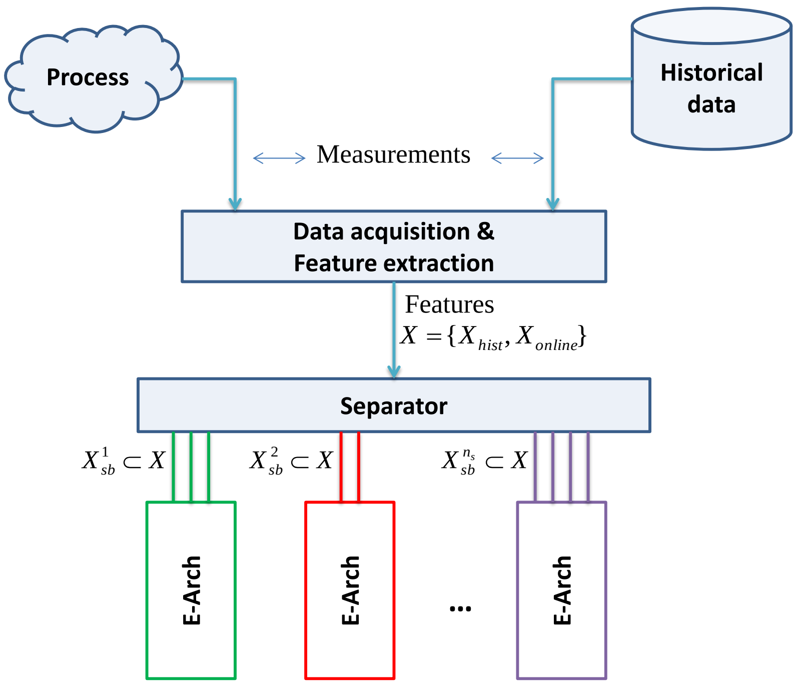


Figure 4.1: Global architecture for supervision, diagnosis and prognosis.

overcome the two difficulties stated above. A complex system is made of several subsystems and generating high amount of data (and consequently features) corresponding to various failure modes. Thus, the issues are the presence of large amounts of data and two kinds of failure modes: a first kind for which a failure class can be characterized (available failure data) and a second kind that cannot be characterized by a failure class. An interesting approach is to divide data into smaller, easier to study subsets. In addition, it is more suitable to make a difference between failure modes for which failure data is available, and failure modes for which failure data are not available. More formally:

1. For a given subset of faults, there exists a subset of features that are sensitive to these faults. A feature is sensitive to a fault if its value deviates from a normal value when the fault exists. Thus, it is not needed to treat the whole input feature space to tackle a subset of faults. In fact, given a set of features, a failure mode can affect several features from this set, and each feature can be sensitive to different failures. This relationship can be converted to the form of a *signature matrix* (Gertler, 1991, 1998; Gertler and Cao, 2005). The columns of these matrices are the failure modes, represented by their status codes. The rows of the matrices are the features. The intersection of the rows and columns is a binary number, which is given the value 1 in case the feature (at the corresponding row) is sensitive to a failure mode (at the corresponding column). Otherwise, it is given the value 0.

   For example, in Table (4.1) below, an example of 3 features associated to 5 faults (described by 5 status codes) is given. From this example, it can be seen that:

© 2014 Tous droits réservés. doc.univ-lille1.fr

– The features (x_1, x_3) are sensitive to the failure mode number 1. The features (x_1, x_2) are sensitive to the failure mode number 2.
– The features (x_2, x_3) are sensitive to both failure modes number 3 and 4.
– The failure mode number 5 cannot be detected and isolated since there is no feature that is sensitive to it. In this case, more features should searched for in order to treat this case.

Status codes / Features	Sc 1	Sc 2	Sc 3	Sc 4	Sc 5
x_1	1	1	0	0	0
x_2	0	1	1	1	0
x_3	1	0	1	1	0

Table 4.1: Example of a signature matrix.

2. As stated earlier, there exists a subset of failure modes for which failure data exists. Formally speaking, this is equivalent to: $\exists\ \mathbb{F}_k \subset \mathbb{F}$, such as $\forall F \in \mathbb{F}_k$, $\exists\ X_{sb}^F \subset X$ that characterizes a failure class for this failure F. Let $X_{\mathbb{F}_k}$ be the set of features sensitive to the failure modes inside $\mathbb{F}_k$. In this work, saying that failure data is available means that there is an initial learning database that can provide prior knowledge about some failure modes.

   There exists also a subset of failure modes for which no initial failure data provide knowledge on failure modes. This subset is denoted $\mathbb{F}_u$. Let $X_{\mathbb{F}_u}$ be the set of features sensitive to the failure modes inside $\mathbb{F}_u$. In this work, saying that failure data is not available means that there is not an initial learning database that can provide prior knowledge to some failure modes. In order to treat this case, it is supposed that only expert knowledge is available. The role of this expert knowledge is to find a subset of features sensitive to a failure mode for which no failure data exists.

   In order to correctly treat those two cases (availability and non-availability of failure data), features that are sensitive to these two types of failures must be separated.

In the next subsections, the different mechanisms of the G-arch, namely the separation mechanism, the E-arch designed to treat subsets of the feature space and the decision mechanism are going to be developed and explained for each of the two case: 1) available failure data, 2) non-available initial failure data.

4.2.1 Case 1: available failure data and prior knowledge of failure classes

The idea is that, for each failure mode $F \in \mathbb{F}_k$, there exist a feature space, spanned by features that are sensitive to this failure mode, and in which a failure class can be characterized. There are general instructions to be respected for the separation mechanism:
– The dimension of each feature space spanned by a subset X_{sb}^i must not exceed a certain limit. High dimension feature spaces have two major drawbacks; (a) they are complex making the

© 2014 Tous droits réservés. doc.univ-lille1.fr

interpretation of the indicators they generate harder (b) additional computational time cost from the operation of AuDyC. In this work, the limit for the dimension of a feature space spanned by a subset X^i_{sb} will be considered to be equal to 3,

- Features that have different sampling times cannot be put together.
- $\forall i \neq j$, $E^i_{sb} \not\subset E^j_{sb}$, meaning that there is no space that is a subset of another,
- The number of spaces must be minimized.

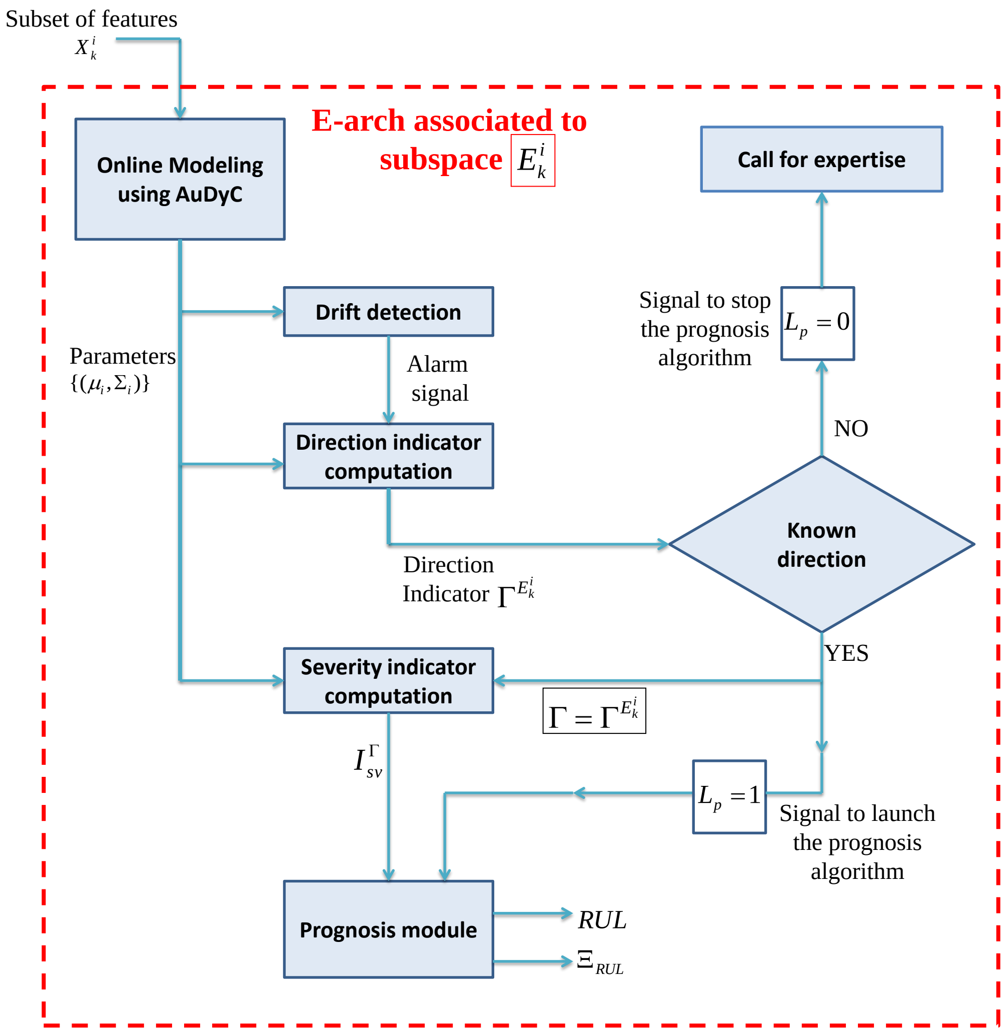

Subset of features
X^i_k
E-arch associated to subspace E^i_k
Online Modeling using AuDyC
Call for expertise
Drift detection
Alarm signal
Parameters $\{(\mu_i, \Sigma_i)\}$
Signal to stop the prognosis algorithm
$L_p = 0$
NO
Direction indicator computation
Known direction
Direction Indicator $\Gamma^{E^i_k}$
YES
Severity indicator computation
$\Gamma = \Gamma^{E^i_k}$
I^{Γ}_{sv}
$L_p = 1$
Signal to launch the prognosis algorithm
Prognosis module
RUL
Ξ_{RUL}

Figure 4.2: E-arch: elementary architecture in case of available failure data.

Once the separation is done, each subset of features is used to span a feature space on which a decision model is built using AuDyC. Each set of features is denoted $\mathbb{E}_k$, such as $\mathbb{E}_k = \{E^1_k, E^2_k, \ldots, E^{N_k}_k\}$, where N_k is the cardinal of $\mathbb{E}_k$. For each of these subspaces, the modeling is associated with three different modules. These are the drift detection module, the drift characterization modules and the prognosis module. For each feature subset, the assembly of all these modules, forms the E-arch. Several E-archs can be created using several subsets of features, taken from the original set of features.

© 2014 Tous droits réservés.

In Fig. (4.2), an E-arch is graphically depicted. An E-arch contains several modules connecting with each other. Individually, these modules are:

1. Modeling module using AudyC. AuDyC is an online tool. Incipient faults on the system are translated into concept drifts, which AuDyC is built to treat. The parameters of the classes will serve as the basis of the architecture since they are mainly used to provide detection and characterization indicators, and then contribute to achieve diagnosis and prognosis.

   This module is depicted in Fig. (4.3). Fed by a set of features coming from the separation mechanism, this block is responsible for modeling the different operating modes by Gaussian classes (Chapter 2, Section 2.5). It begins by the historical learning phase in which a model is developed using an initial set of given data. This leads to the definition and to the characterization of the normal mode and of several failure modes, since we are considering in this case that failure data is available. Labeling the different failure modes is done thanks to expert knowledge.

   Once learning from historical data is done, the model can be used in online manner as explained in Chapter 3, Section 3.2. The inputs for this module become the patterns that are being generated online. The outputs of this module are the parameters of clusters that were created by AuDyC, and that are dynamically updated.

2. Drift detection module. This module is depicted in Fig. (4.4). It consists in applying the drift detection test. We remind that in this work, only one normal operating mode is considered to exist. The drift detection is based on the fact that, when a drift occurs, the evolving class, representing the current operating mode, will deviate from the class representing the normal mode.

   Thus, the inputs of this module are the parameters of the dynamically updated clusters (the output of the modeling with AuDyC module). The output of this module is an alarm signal indicating that a significant deviation from the normal operating mode has been detected. This alarm signal triggers the functioning of the drift direction computation module. The reader is invited to consult Chapter 3, Section 3.4, for the technical details of the drift detection test.
3. Drift characterization module - drift direction computation. This module is depicted in Fig. (4.5). Based on the values of the means of the dynamically updated evolving class, the movement vectors can be calculated at each time stamp t_i (Chapter 3, Section 3.5.1). Based on the values of the means of the evolving class and the means of each of the failure classes, the direction vectors can be calculated (Chapter 3, Section 3.5.1). Since the definition of the movement vectors and the direction vectors is possible, the algorithm for calculation of the direction indicator is thus possible.

   From the subspace E_k^i, this module will extract the direction indicator $\Gamma^{E_k^i}$. Since failure classes can be characterized in these feature subspaces, the value of the direction indicator computed inside them refer to a failure mode. The value of the direction indicator computed

© 2014 Tous droits réservés. doc.univ-lille1.fr

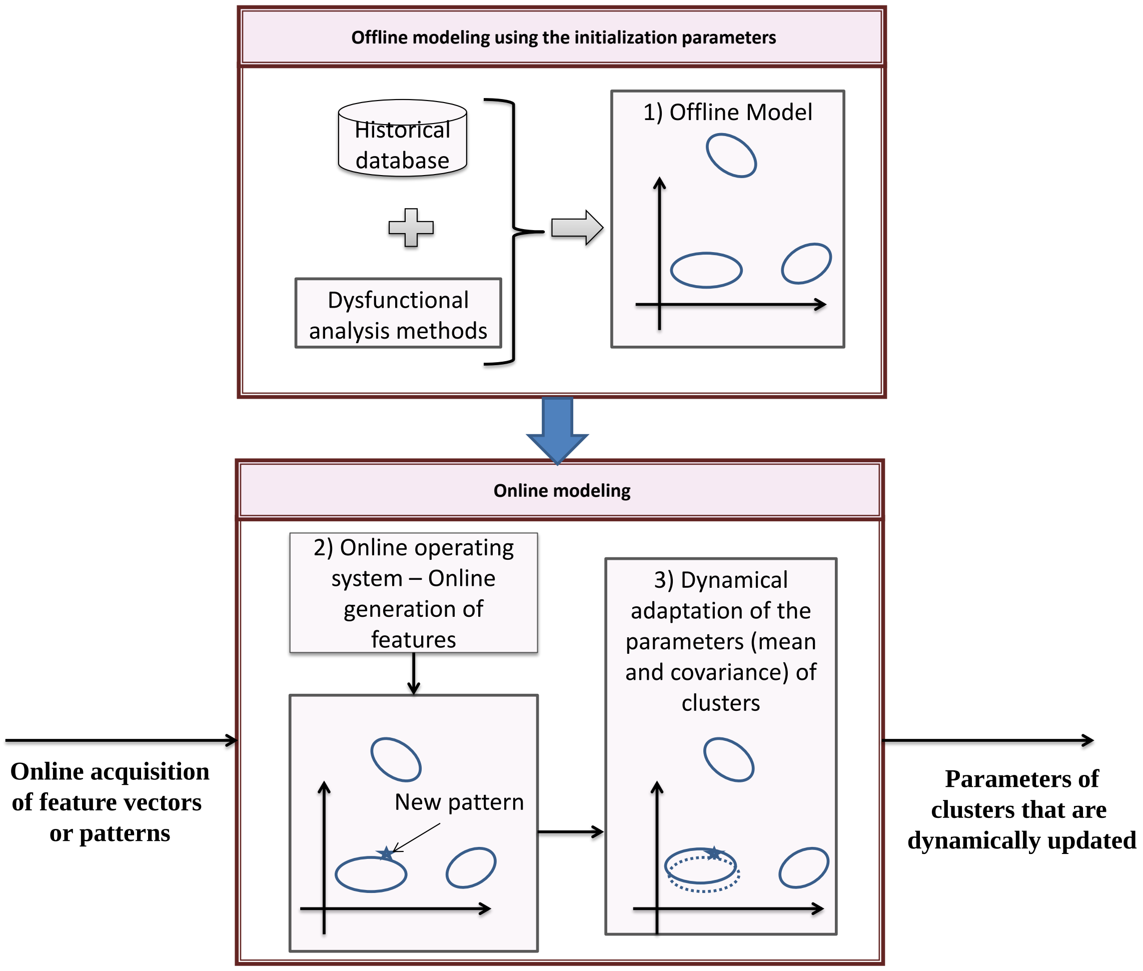

Offline modeling using the initialization parameters
Historical database
Dysfunctional analysis methods
1) Offline Model
Online modeling
2) Online operating system – Online generation of features
New pattern
3) Dynamical adaptation of the parameters (mean and covariance) of clusters
Online acquisition of feature vectors or patterns
Parameters of clusters that are dynamically updated

Figure 4.3: Module: modeling with AuDyC.

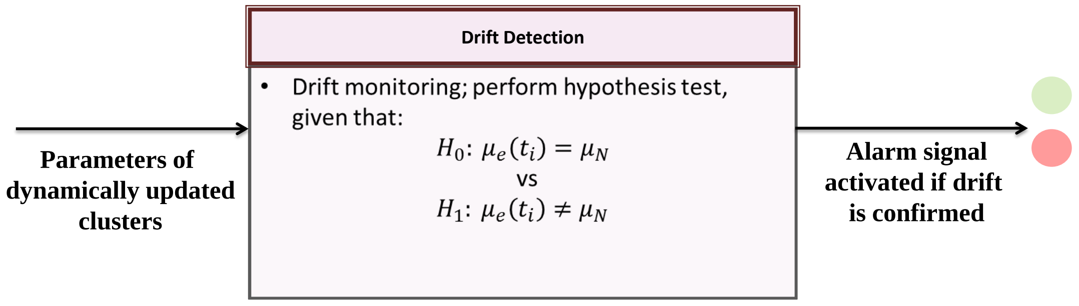

Drift Detection
• Drift monitoring; perform hypothesis test, given that:
H_0: $\mu_e(t_i) = \mu_N$
vs
H_1: $\mu_e(t_i) \neq \mu_N$
Parameters of dynamically updated clusters
Alarm signal activated if drift is confirmed

Figure 4.4: Module: drift detection test.

will be the same as the status code of the failure mode that can be diagnosed in this space.

4. Drift characterization module - severity indicator computation. This module is depicted in

© 2014 Tous droits réservés.

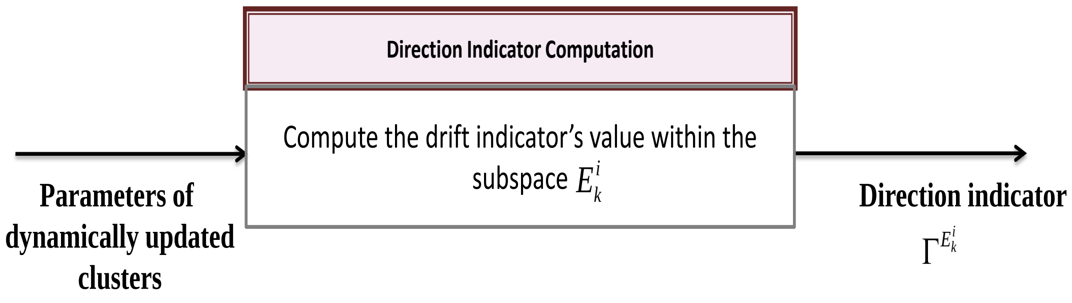

Direction Indicator Computation
Compute the drift indicator's value within the subspace E_k^i
Parameters of dynamically updated clusters
Direction indicator $\Gamma^{E_k^i}$

Figure 4.5: Module: direction indicator computation.

Fig. (4.6). It consists in calculating the severity indicator corresponding to an already isolated failure mode. In order to be able to calculate a severity indicator, it is necessary thus to have an E-arch in which a feature space contains failure classes. Without a failure class, which occupies a region in the feature space, it is impossible to calculate this indicator.

It was stated earlier that for such kind of faults, the isolation is straightforward. In fact, once this fault happens, the E-arch that contains the set of features that are sensitive to it provides an alarm signal. The value of the direction indicator calculated inside this E-arch gives directly the status code of the failure mode. Using Eq. (3.14) in Chapter 3, Section 3.5.2, several severity indicators are calculated according to all the known failure classes inside this E-arch. However, only the value of the severity indicator that corresponds to the isolated failure mode is kept, and considered as the one that represents the health state of the system.

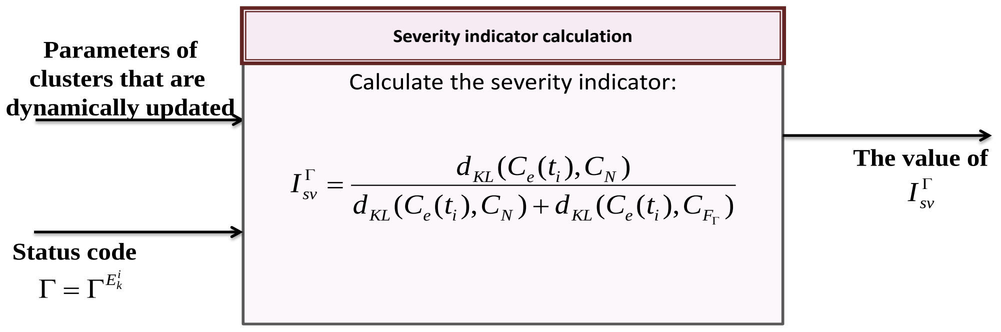

Severity indicator calculation
Calculate the severity indicator:
$I_{sv}^{\Gamma} = \frac{d_{KL}(C_e(t_i), C_N)}{d_{KL}(C_e(t_i), C_N) + d_{KL}(C_e(t_i), C_{F_\Gamma})}$
Parameters of clusters that are dynamically updated
Status code $\Gamma = \Gamma^{E_k^i}$
The value of I_{sv}^{Γ}

Figure 4.6: Module: severity indicator computation.

5. Prognosis module. This module is depicted in Fig. (4.7). It consists in providing an estimate of the remaining useful life. It does so by using the prognosis algorithm developed in this work, and explained in Chapter 3, Section 3.5.

   Obviously, the prognosis algorithm is based on the severity indicator I_{sv}^{Γ} provided by the pre-

© 2014 Tous droits réservés.

ceding module. This is natural in the sense that once a failure mode has been isolated, and can be characterized, the prognosis algorithm can be launched to provide estimates of the RUL and a corresponding confidence limit interval. When one of the two obligations is not satisfied, no RUL can be calculated. Thus, to ensure that both obligations are satisfied, a signal is sent to the prognosis algorithm that will launch it. This signal is denoted L_P with respect of notations in Chapter 3. Launching the prognosis algorithm is done by setting the value of L_P to 1.

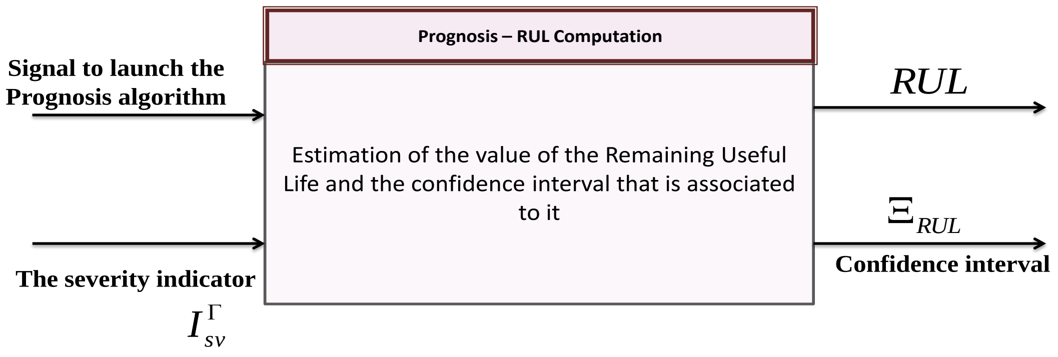

Prognosis – RUL Computation
Signal to launch the Prognosis algorithm
Estimation of the value of the Remaining Useful Life and the confidence interval that is associated to it
RUL
Ξ_{RUL}
Confidence interval
The severity indicator
I_{sv}^{Γ}

Figure 4.7: Module: prognosis.

The different modules that constitute the G-arch were detailed in the case when failure data is available. The next case to be treated when there is no initial failure data corresponding to some failure modes, belonging the set $\mathbb{F}_u$.

4.2.2 Case 2: unavailable initial failure data and prior knowledge on failure classes

In order to treat this case, it is supposed that only expert knowledge is available. The expert knowledge can be used to find subset of features sensitive to a failure mode for which no initial failure data is available.

For a failure mode $F \in \mathbb{F}_u$ triggering a drift, as for any failure mode, the ultimate goal is to diagnose it and then to provide a RUL estimate using a prognosis model. However, for the cases of non-availability of failure data, no severity indicator can be calculated because of the lack of knowledge on the initial failure modes. Thus no prognosis can be made. In this work, treating the case of non-available failure data is limited to using drift detection and characterization in order to diagnose the failure mode triggering a drift.

The detection of a drift in case of non-available failure data is possible since the drift detection test requires only the parameters of the normal mode, as well as the parameters of the evolving mode. The isolation process is harder since it is based on the direction indicator computation. The latter requires the definition of movement vectors and direction vectors. The movement vectors are possible to calculate since they depend only on the parameters of the evolving class (see Chapter 3, Section 3.5.1). The direction vectors cannot be calculated if at least the value of the mean of the

© 2014 Tous droits réservés. doc.univ-lille1.fr

failure class is not available, which the case is now.

In order to overcome this difficulty, expert knowledge will be used. The role of the expert knowledge is to find, for a failure mode $F \in \mathbb{F}_u$, a subset of features X_{sb}^F that is sensitive to it. The result of the separation mechanism is the obtainment of different subsets of features, generating different feature spaces. Let $\mathbb{E}_u$ denote the set of these feature spaces, such as $\mathbb{E}_u = \{E_u^1, E_u^2, \ldots, E_u^{N_u}\}$, where N_u is the cardinal of $\mathbb{E}_u$. The different subset of features generating E_u^i, $i = 1, \ldots, N_u$, are also the basis of E-archs. An E-arch generated by a feature space E_u^i does not contain any severity indicator calculation block, nor a prognosis module block. In Fig. (4.8), an E-arch for the case of non-available failure data is shown.

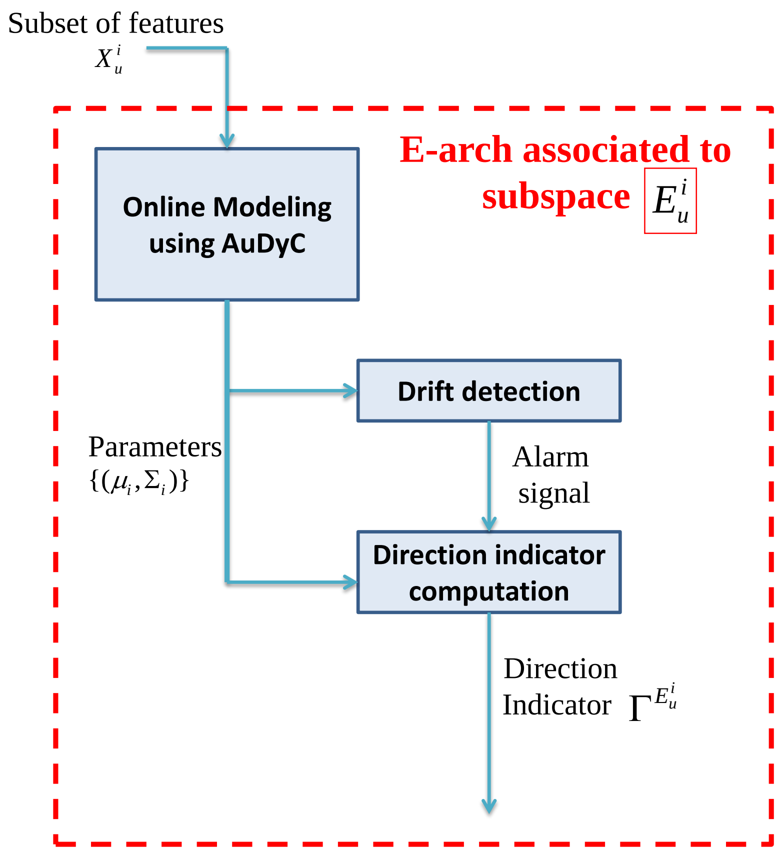

Subset of features
X_u^i
E-arch associated to subspace $\boxed{E_u^i}$
Online Modeling using AuDyC
Drift detection
Parameters $\{(\mu_i, \Sigma_i)\}$
Alarm signal
Direction indicator computation
Direction Indicator $\Gamma^{E_u^i}$

Figure 4.8: Elementary architecture in case of non-available failure data.

For each subspace, a feature corresponds to an axe of this subspace. The direction indicator inside each subspace is given a number according to each axe. If a fault affects this feature, then the direction indicator will have a number that denotes this feature. Then the decision is made with a set of if-$then$ statements, in order to find the status code, using the signature matrix.

Each axe in a feature space has a unitary vector. The movement of the evolving class can thus

© 2014 Tous droits réservés. doc.univ-lille1.fr

be in the same direction as the unitary vector, or in the opposite direction. In any case, since the evolving class' movement will be on this axe, then a direction vector can be calculated because it can be sure that the failure class is present on this axe. However, its exact position remains unknown, in the sense that the mean of the failure class can be in the positive direction of the axe, or in the negative direction of the axe. Thus, two direction vectors are considered: the unitary vector, and its opposite.

For better illustration, let us consider the example shown in Table (4.2). Five features are considered: x_1, x_2, x_3, x_4 and x_5. 5 faults, described by 5 status codes are also considered: Sc 1, Sc 2, Sc 3, Sc 4 and Sc 5. Two feature spaces can be built using those five features; one feature space whose dimension is 3, denoted E_1 and another feature space whose dimension is 2, denoted E_2. Let us consider those two possible configurations (there are other possible configurations but only these two are considered because they are enough for this example):

- Configuration 1: $E_1 = span\{x_1, x_2, x_3\}$ and $E_2 = span\{x_4, x_5\}$,
- Configuration 2: $E_1 = span\{x_2, x_3, x_4\}$ and $E_2 = span\{x_1, x_5\}$.

In configuration 1, the condition of having for each fault, a subset of sensitive features in which each feature belong to a different space is satisfied. In configuration 2, this condition is not verified. The reason is that, in configuration 2, the features x_2 and x_3 are present in space E_1. Thus, if failure 3 (Sc 3) occurs, then the evolving class in space E_1 will not move along an axe, but inside the plane $P_l = span\{x_2, x_3\} \subset E_1$. In configuration 1, if failure 3 occurs, the evolving class in space E_1 moves along the axe corresponding to the feature x_3, and the evolving class in space E_2 (always in configuration 1) will move along the axe corresponding to the feature x_4.

Now, by considering the configuration 1, direction vectors are defined for each axe of each feature space, and a direction indicator value is associated to the direction vectors (see Fig. (4.9)). In this example, the decision mechanism is composed of a set of if-$then$ statements such as:

- If the failure 1 is causing the drift, then only the feature x_1 is sensitive. Thus, the evolving class in space E_1 will move along the axe x_1. This movement is detectable by using the drift detection mechanism described in Chapter 3, Section 3.4. Then, the drift direction indicator is computed and is denoted Γ^{E_1}. Different values are associated to the direction indicator Γ^{E_1} according to the different direction vectors. In Fig. (4.9), these different values are shown. Thus, we have: $If\ \Gamma^{E_1} = 1$, $THEN\ sc = 1$.
- For the failure 2, and following the logic described above, we have: $If\ \Gamma^{E_1} = 2$, $THEN$ $sc = 2$.
- For the failure 3: $If\ \Gamma^{E_1} = 3\ AND\ \Gamma^{E_2} = 1$, $THEN\ sc = 3$.
- For the failure 4: $If\ \Gamma^{E_2} = 2$, $THEN\ sc = 4$.

A graphical illustration of the generation of the status code by using if-$then$ statements is shown in Fig. (4.10). The generation of a status code is a decision mechanism. The result of this decision mechanism is a valid status code, or just 0. In case a drift is detected and the status code always shows 0, then this means that either it is an unknown failure mode, or more time is needed for isolation. In order to decide which the case is, expert knowledge is called for.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Features \ Status codes	Sc 1	Sc 2	Sc 3	Sc 4
x_1	1	0	0	0
x_2	0	1	0	0
x_3	0	0	1	0
x_4	0	0	1	0
x_5	0	0	0	1

Table 4.2: Another example of a signature matrix.

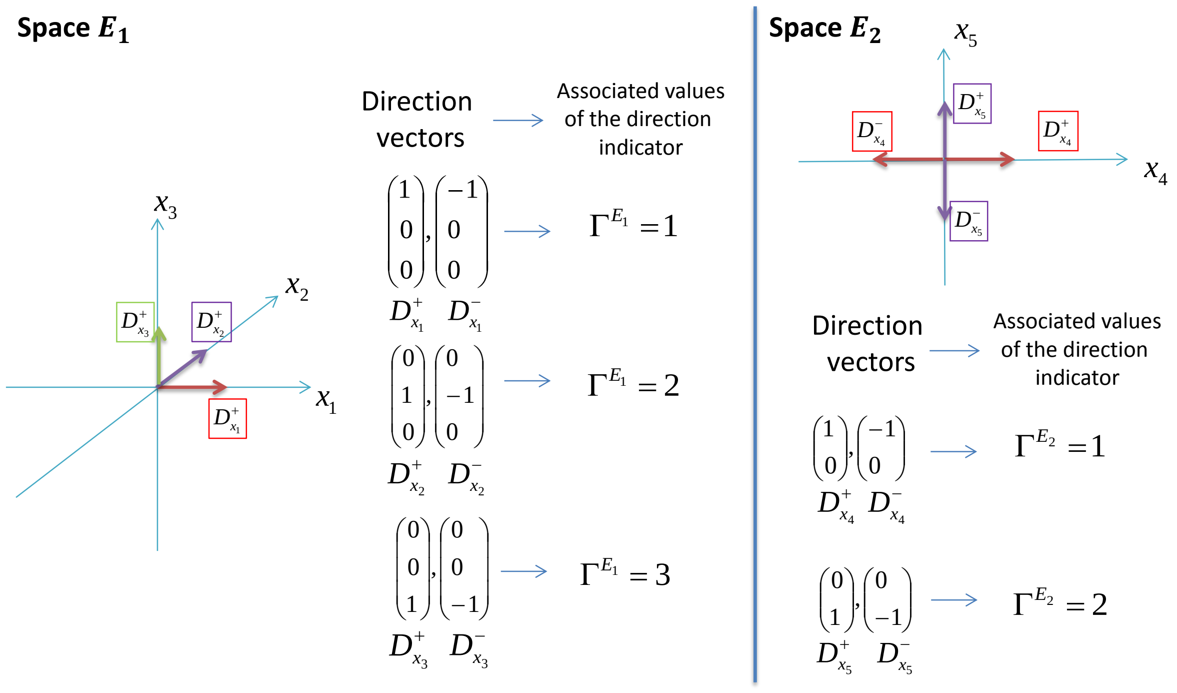

Space E_1
x_3
x_2
x_1
$D^+_{x_3}$
$D^+_{x_2}$
$D^+_{x_1}$
Direction vectors
Associated values of the direction indicator
$\begin{pmatrix}1\\0\\0\end{pmatrix},\begin{pmatrix}-1\\0\\0\end{pmatrix}$ $D^+_{x_1}$ $D^-_{x_1}$ $\Gamma^{E_1}=1$
$\begin{pmatrix}0\\1\\0\end{pmatrix},\begin{pmatrix}0\\-1\\0\end{pmatrix}$ $D^+_{x_2}$ $D^-_{x_2}$ $\Gamma^{E_1}=2$
$\begin{pmatrix}0\\0\\1\end{pmatrix},\begin{pmatrix}0\\0\\-1\end{pmatrix}$ $D^+_{x_3}$ $D^-_{x_3}$ $\Gamma^{E_1}=3$
Space E_2
x_5
x_4
$D^+_{x_5}$
$D^-_{x_4}$
$D^+_{x_4}$
$D^-_{x_5}$
Direction vectors
Associated values of the direction indicator
$\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}-1\\0\end{pmatrix}$ $D^+_{x_4}$ $D^-_{x_4}$ $\Gamma^{E_2}=1$
$\begin{pmatrix}0\\1\end{pmatrix},\begin{pmatrix}0\\-1\end{pmatrix}$ $D^+_{x_5}$ $D^-_{x_5}$ $\Gamma^{E_2}=2$

Figure 4.9: Spaces, direction vectors and direction indicators for the example above.

4.3 Experimentation

In this section, the functioning of the E-arch and the G-arch are going to be shown. For this reason, a set of drifting scenarios will be created in order to show the functionality of all the modules.

This experimentation part of the thesis is divided into two case studies. The first case study concerns a water tank system that is subject to different kind of drifting faults. The second case study is a wind turbine system on which also different kind of drifting faults are given. The first case study aims to better explain the functionality of the methodology in case of available data for failure modes. The second case study aims to show the functionality of the methodology in case of non-available failure data.

© 2014 Tous droits réservés. doc.univ-lille1.fr

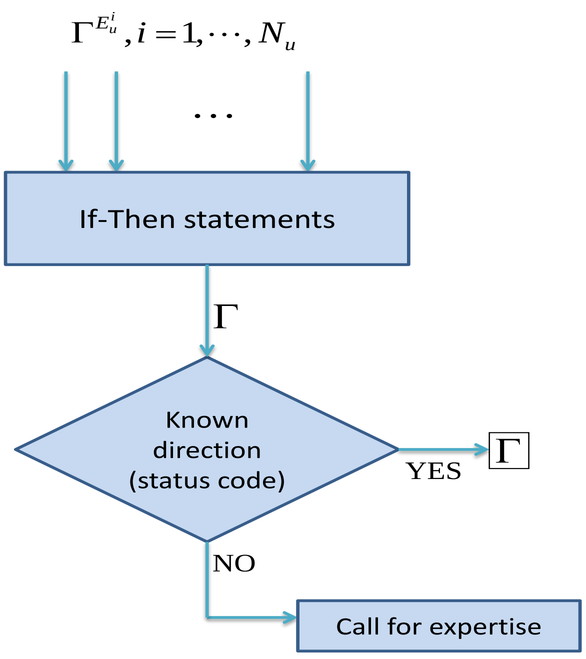

$\Gamma^{E_u^i}, i = 1, \cdots, N_u$
...
If-Then statements
Γ
Known direction (status code)
YES
Γ
NO
Call for expertise

Figure 4.10: Generating the status code in case of non-available prior knowledge on failure modes.

In the first case study, three drifting scenarios are considered. Their aims are to highlight:

- The relation and the connections between modules in case of known failure modes, that can be characterized by failure classes. The results for the detection and characterization of drifts, as well as the prognosis results are given.
- The case of a detection of an unknown functioning mode, due to multiple failure occurrences, is shown. The idea is to show the behavior of the architecture in case of an unknown functioning mode.

In the second case study, five drifting scenarios are shown. Their aims are to highlight:

1. The construction of the G-arch in order to achieve supervision and diagnosis results,
2. The separation/grouping mechanisms and how they are used for diagnosis. The separation/grouping mechanisms are drawn according to a fault signature matrix that is built.
3. The connections between different modules in case of known failure modes that cannot be characterized by failure classes, but by direction vectors, are shown.

4.3.1 Case study 1: water tank system

The water tank system is depicted in Fig. (4.11). Under normal operating mode, the level of water is kept between two thresholds, h_1^{HIGH} and h_1^{LOW}. When the level of water reaches h_1^{HIGH}, the pump P_1 is closed and the valve V_2 is opened. When the level of water reaches h_1^{LOW}, the pump

© 2014 Tous droits réservés. doc.univ-lille1.fr

P_1 is opened and the valve V_2 is closed. The valve V_1 is used to simulate leak in the tank. The surface of the valves V_1 and V_2 is the same: $S_{V_1} = S_{V_2} = S_V$ and the surface of the pump pipe is S_P. The instrumentation used consists of only one sensor for the level of water in the tank. It is denoted by h_1.

4.3.1.1 Considered faults

In order to test our architecture, three faults are considered, two known faults and one unknown:

1. Fault1 (known): gradual increase of the surface of the valve V_1 leading to a gradual increase of the flow of water leaking from the tank. This surface increases from $0\% \times S_V$ to $30\% \times S_V$ considered as the maximum intensity of leakage. At this stage, the system is considered in failure. When the surface is between $1\% \times S_V$ and $30\% \times S_V$, the system is faulty (degraded operation). The fault is considered to be known in the sense that we can characterize the failure by a class that will be denoted C_{F_1}. Thus, we consider C_{F_1} to be the failure class associated to this fault.

2. Fault2 (known): clogging of the pump P1 meaning that the flow of water that the pump is delivering is decreasing with time. The same principle for the simulation of V_1 is used. A clogging of $30\% \times S_P$ corresponds to failure. The failure level of this fault can also be characterized by a failure class C_{F_2}.

3. Fault3 (unknown): clogging of the valve V_2. The same principle for the fault 2 with the valve V_1 is used. A clogging of $30\% \times S_V$ of the valve V_2 means failure. The failure level of this fault is considered as unknown. Thus, it cannot be characterized by a failure class.

4.3.1.2 Feature extraction

At the beginning, three data sets are given. One data set corresponds to normal operating mode, and two data sets correspond to failure modes 1 and 2, *i.e.* C_{F_1} and C_{F_2}. In Fig. (4.12), we show the sensor measurements under normal operating mode. Let a cycle be a sequence of a filling period followed by a draining period (see Fig. (4.12)). The features extracted from this signal are the time required for the filling T_1 and the time required for draining T_2. At the end of each cycle j, one feature vector $X(j) = [T_1(j)\ T_2(j)]^T$ is extracted. Thus, the considered time unit will be the index of the cycles, which are denoted by $cy(j), j = 1, \ldots$.

4.3.1.3 Simulation scenarios

Three simulation scenarios are considered:
– Scenario1: only fault1 is considered. The drift is simulated linearly from 0% to 30% in $26cy$.
– Scenario2: only fault2 is considered. The drift is simulated linearly from 0% to 30% in $60cy$. The drift in this scenario takes more time to be finished than in scenario 1. Thus, the speed of this drift is lower.

© 2014 Tous droits réservés. doc.univ-lille1.fr

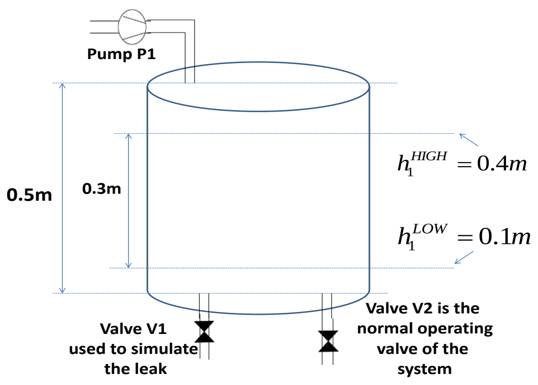

Pump P1
0.5m
0.3m
$h_1^{HIGH} = 0.4m$
$h_1^{LOW} = 0.1m$
Valve V1 used to simulate the leak
Valve V2 is the normal operating valve of the system

Figure 4.11: The tank system.

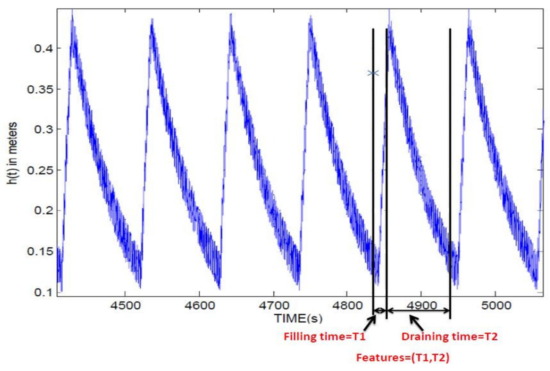


Figure 4.12: Six filling/draining cycles under normal operating conditions.

– Scenario3: In this scenario, fault2 and fault3 are considered. At the beginning, only fault2 is activated. Then, fault3 is activated with some delay and both faults are thus affecting the system together. The result is a change in the direction of the drift and the idea behind it is to test the ability of our algorithm to detect this case.

4.3.1.4 Simulation results

In this case study, there is no need for the separation mechanism since there are only two features. These two features span only one feature space which is used to make the supervision, diagnosis and prognosis. The knowledge of the failure modes (available initial failure mode data) allows the definition of the direction vectors $D_j(t_i)$, $j = 1, 2$ since there are two known failure modes (as explained in Chapter 3, Section 1.4.1). Thanks to these direction vectors, the fault isolation is straightforward. It is done directly according to the value of the direction indicator.

The parameters of the overall algorithm are: $\lambda_m = 0.001$ and $N_{win} = 30$ (AuDyC parameters, Chapter 2, Section 2.5), $\alpha = 99.5\%$, $T_h = 2.1$, $F_{(2,28)|\alpha)} = 3.44$, $p_M = 0.85$, $N_p = 4$ (Chapter 3, drift detection and characterization indicators, Section 3.5) and $H_M = 300$ (Chapter 3, Section 3.6). In all scenarios, drift was started at $t_{drift} = 10cy$.

© 2014 Tous droits réservés. doc.univ-lille1.fr

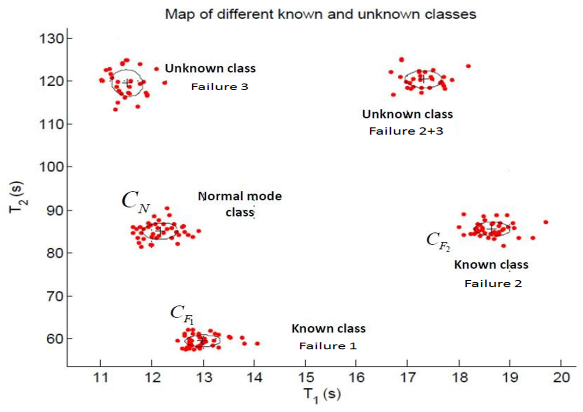


Figure 4.13: Map of different known and unknown regions corresponding to normal and failure modes.

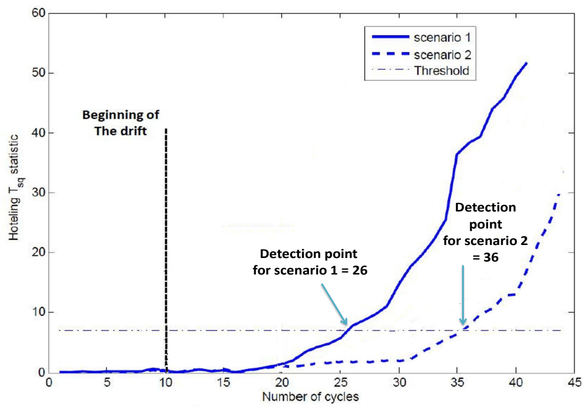


Figure 4.14: Drift detection results for scenarios 1 and 2.

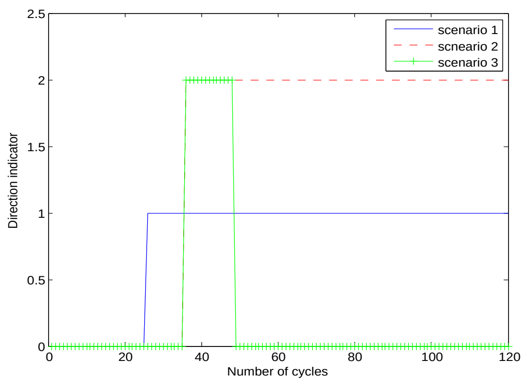


Figure 4.15: Direction indicator results for scenarios 1, 2 and 3.

The map showing normal mode and different known and unknown modes is presented in Fig. (4.13).

© 2014 Tous droits réservés. doc.univ-lille1.fr

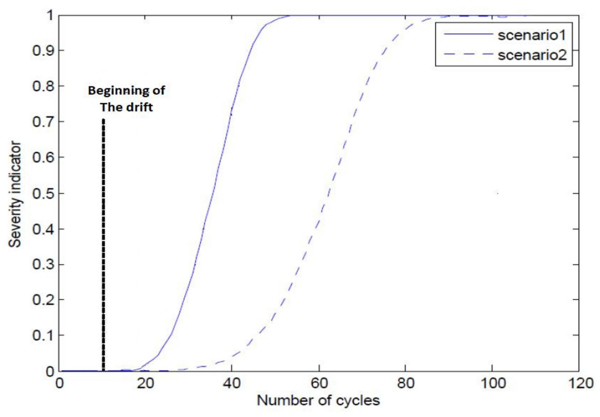


Figure 4.16: Severity indicator results for scenarios 1 and 2.

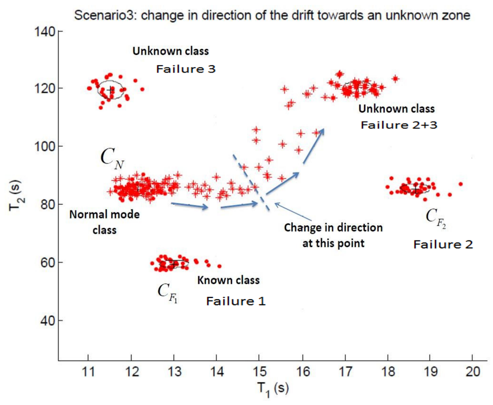


Figure 4.17: Illustration, in the feature space, of the direction change in scenario 3.

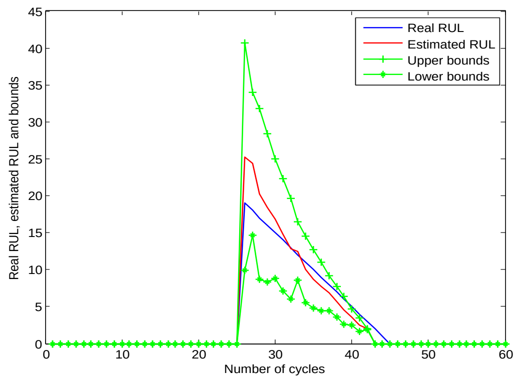


Figure 4.18: Real RUL vs estimated RUL for scenario1.

Classes corresponding to the known failure modes as well as unknown failure modes are shown. In scenarios 1 and 2, the drift was towards a known region. The drift was correctly detected in both scenarios. We notice that the detection time is higher in scenario 2 than in scenario 1 (Fig. (4.14).

© 2014 Tous droits réservés. doc.univ-lille1.fr

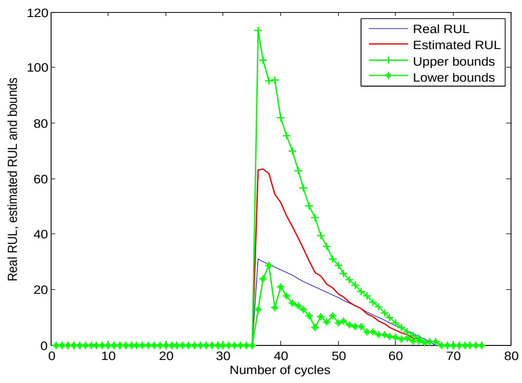


Figure 4.19: Real RUL vs estimated RUL for scenario2.

The reason is that the speed of drift in scenario 2 is lower than the speed of drift in scenario 1. Upon detection, the direction indicator module is requested. For both scenarios, the direction indicator is directly calculated according to the algorithm that was explained in Chapter 3, Section 3.4.1. It can be seen in Fig. (4.15) that the direction indicator's value, in scenario 1, takes the value 1 upon detection. This means that the migration of the class is from the normal mode, towards the failure mode C_{F_1}. The same goes for the case in scenario 2. The value of the direction indicator goes to 2 upon detection (see Fig. (4.15)).

Once the failure modes have been isolated in scenarios 1 and 2, the conditions to calculating the severity indicators are present. In fact, as stated earlier, the values for the severity indicators are calculated at each instant. Upon drift detection, and once the isolation is done thanks to the direction indicator, the value of the severity indicator corresponding to the failure mode is chosen. In scenario 1, it is the value of the severity indicator corresponding to the migration towards C_{F_1}. In scenario 2, it is the value of the severity indicator corresponding to the migration towards C_{F_2}. The severity indicators in these two scenarios are shown in (Fig. (4.16)). It can be seen from this figure that in the case for scenario 1, the value of the severity indicator reaches the value faster then in the case of scenario 2. The reason is that the drift is faster in the first scenario then the one in the second.

As the values of the severity indicators are being calculated during these two scenarios, the conditions to launch the prognosis algorithm are verified. These conditions are verified in scenarios 1 and 2 upon detection. Thus the prognosis algorithm is launched in both cases. In this case study, two data widows were chosen for prognosis (remind Chapter 3, Section 3.5 for details). The first window's length is 5, and the second window's length is 10. It is reminded that the unit is the cycle. The results for prognosis are detailed in Figs. (4.18,4.19). It can be seen that the prognosis results are considered satisfying because in both scenarios, the estimated value of the RUL converges to the real value. Also, it can be seen that the confidence interval associated to the estimate of the RUL provides an envelope in which the real values of the RUL are, almost during all the time (except for the last few cycles).

© 2014 Tous droits réservés.

In scenario 3, the direction indicator change values from 2 to 0. This reflects a change in the direction of the evolution of classes in the feature space. This can only mean that there is another fault affecting the drift (Fig. (4.15) and Fig. (4.17)). The same unknown zone is detected even when two known faults are simulated. Thus, in this case, the unknown zone reflects the need to update the dysfunctional analysis using expert knowledge. The drift detection for this scenario is the same as in scenario 2 because fault 2 was simulated in the beginning and so it wasn't presented in Fig. (4.14).

The obtained results confirm the efficiency of the architecture. The methodology was based on monitoring the parameters of dynamically updated clusters. Then, the drift detection module was used to detect drifts. In case of positive detection, two drift characterization indicators were computed. These indicators are the direction indicator and the severity indicator. In case of a known failure mode behind the incipient fault is detected, a prognosis module is triggered to provide a RUL estimate. Few points are to discuss:

- This methodology is designed to work in any dynamical environment where pertinent features can be obtained. A decision space with overlapping classes corresponding to normal and failure operating modes will limit the capacity of this method to generate efficient results. Thus the feature extraction part is very important.
- The architecture allows detecting new unknown failure modes. These modes correspond to either unknown failure modes or multiple failures. A multiple failure mode is caused by a combination of known and unknwon faults. In this work, whatever is the case, the resulting detection of an unknown mode paralyses the ability of the methodology to estimate a RUL because of the limited knowledge.

4.3.2 Case study 2: wind turbine

In this section, the wind turbine diagnosis problem is tackled. A benchmark of wind turbine is used for this purpose. The paper (Odgaard et al., 2009) presents this wind turbine benchmark model containing sensors, actuators and system faults. The benchmark simulates a realistic generic three blade horizontal variable speed wind turbine with a full converter coupling. However, since little or no drifting faults were available in the paper (Odgaard et al., 2009), an effort was made to define and to include drifting faults. As stated earlier, this case study aims to highlight the functionning of the methodology in the case of non existing failure data.

The rest of the section is as follows. The first part concerns the description of the wind turbine, its physical model and the control scheme used in it. The second part is concerned about the application of the architecture for the FDI problem.

4.3.2.1 Wind turbine description

Wind turbines, transform the power of the wind into electrical power. The most common wind turbines are the three blade horizontal turbines. In Fig. (4.20), a description of the wind turbine is

© 2014 Tous droits réservés.

doc.univ-lille1.fr

graphically depicted [1].

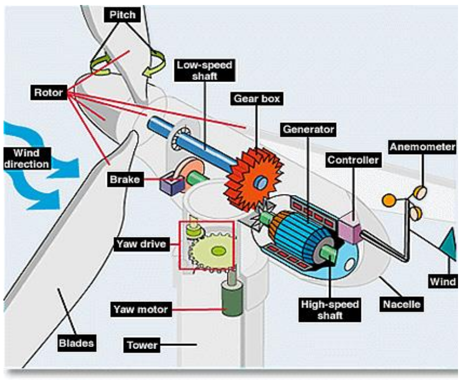


Figure 4.20: Description of the conventional three blade horizontal wind turbine.

The energy of the wind is a kinetic energy. The flow of the wind through the wind turbine causes the rotation of its blades, which in turn causes the rotation of the motor to a generator that produces electricity. The only required power is thus the wind power, that is proportional to the wind speed. Wind turbines provide clean energy. Once manufactured, they do not release pollution componenets into the environment, like the carbon dioxide for example.

4.3.2.2 Wind turbine modeling

The wind turbine of the benchmark is a $5MW$ three blade horizontal turbine, with converter coupling. It is a variable speed turbine as well. The objective of the control system is to follow the power reference in case of high wind speeds. In case of lower wind speeds, the objective is to minimize the reference error ((Odgaard et al., 2009)). The functionality is based on the fact that the control of the energy flow from wind to electrical is done by controlling the aerodynamics of the wind turbine. In order to be able to do this control scheme, the wind turbine is modeled as in Fig. (4.21) ((Odgaard et al., 2009)). The rotor and the generator are related with a drive train. The latter can be used to increase control the speed of the rotor using the speed of the generator. The control of the generator speed is done using the converter, which sets the generator torque. In case of high wind speeds, controling the aerodynamics of the wind turbine is not enough. It is also necessary to use the pitch subsystems whose aim is to change the pitch angle of the blade, reducing the amount of energy harvested. Doing that prevents the wind turbine from breaking down due to an extra power extraction. Fig. (4.21) shows the relations between the blade and pitch system, drive train, generator, converter, and the controller. The wind turbine is equipped with sensors. The instrumentation of the wind turbine model is resumed in Table (4.3).

4.3.2.3 Wind turbine control

In commercial wind turbines, there are four zones of operation for the controller of the wind turbines. Zone 1 is startup of the turbine, Zone 2 is power optimization, Zone 3 is constant power

1. Fig. (4.20) is taken from (http://homeguides.sfgate.com/explanation-wind-turbines-79607.html)

© 2014 Tous droits réservés.

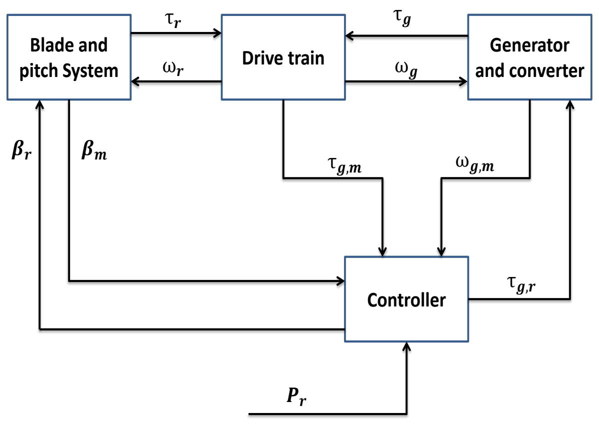


Figure 4.21: Wind turbine model (Odgaard et al., 2009).

Variable	Number of sensors	Notation
Generator speed ω_g	2	ω_{g,m_1}, ω_{g,m_2}
Rotor speed ω_r	2	ω_{r,m_1}, ω_{r,m_2}
Pitch position measurements β_1, β_2, β_3	2 sensors/blade	β_{1,m_1}, β_{1,m_2}, β_{2,m_1}, β_{2,m_2}, β_{3,m_1}, β_{3,m_2}
Power P_g	1	$P_{g,m}$
Generator torque τ_g	1	$\tau_{g,m}$
Wind speed	1	v_{wind}

Table 4.3: Different variables and their correspondent sensors.

production, and Zone 4 is high wind speed. Zones 1 and 4 are the extreme zones in which no power is produced. For high wind speeds for instance, the turbine is forced to stop in the aim of protecting it from burning down. In the benchmark model, only zones 2 and 3 are considered.

The control mode switches from zone 2 to zone 3 if:

$$P_g(t_i) \geq P_r \ OR \ \omega_g(t_i) \geq \omega_{nom}.$$

The control mode should switch from zone 3 to 2 if:

$$\omega_g(t_i) \leq \omega_{nom} - \omega_\Delta,$$

Where ω_Δ is a constant defined for the controller.

The tip speed ratio of a wind turbine, denoted λ_t, is defined as in Eq. (4.1), where R is the radius of the blades, v_w is the wind speed, and ω_r is the angular rotor speed:

$$\lambda_t = \frac{R\omega_r}{v_w}. \tag{4.1}$$

Considering the wind turbine benchmark, in zone 2, the turbine is controlled in such a way to obtain optimal power production. In this zone, pitch angles of all three blades are set to 0 degrees,

© 2014 Tous droits réservés.

by setting the reference value of these angles, denoted β_r, to 0 ($\beta_r = 0$). The tip speed ratio, in this zone, is constant at its optimal value. The optimal value of λ_t, which is denoted λ_t^{opt}, is found as the optimum point in what is called the power coefficient mapping table of the wind turbine (Manwell et al., 2010). This optimal value is achieved by setting the reference torque to the converter, $\tau_{g,r}$.

The torque in this power optimization zone is found as:

$$\begin{aligned} \tau_{g,r} &= K^{opt}\omega_r^2, && (4.2) \\ K^{opt} &= \frac{1}{2}\rho A R^3 \frac{C_{p_{max}}}{(\lambda_t^{opt})^3}, && (4.3) \end{aligned}$$

where ρ is the air density, A is the area swept by the turbine, $C_{p_{max}}$ is the maximal value of the power coefficient C_p (given by the power coefficient table), relating to λ_t^{opt}. The reader is invited to consult (Manwell et al., 2010) fore more details about the definition of C_p and about the power coefficient mapping.

In zone 3, the control objective is to follow the power reference, P_r. This is done by controlling β_r, such that the coefficient C_p is decreased. Decreasing C_p is done by pitching the blade, which is done by changing the value of β_r. A PI controller is used to do that and thus to keep ω_r at the rated value.

4.3.2.4 Fault scenarios

In the benchmark (Odgaard et al., 2009), several fault scenarios were proposed. These faults covered several sensor and actuator faults, and few system faults. These faults are abrupt faults and these scenarios contain little drifting faults. Thus, in this work, more fault scenarios are defined and simulated. Details on how these new scenarios were created is given next. The drifting scenarios are summarized in Table (4.4) above.

Scenario's number	Notation	Description
Scenario 1 (sensor drift)	Sc-Dr 1	Sensor drift affecting the generator speed sensor 2
Scenario 2 (actuator drift)	Sc-Dr 2	Converter's drift (Actuator); loss of efficiency
Scenario 3 (sensor drift)	Sc-Dr 3	Sensor drift affecting the pitch position sensor β_{1,m_1}
Scenario 4 (sensor drift)	Sc-Dr 4	Sensor drift affecting the pitch position sensor β_{2,m_2}
Scenario 5 (system drift)	Sc-Dr 5	Change in the dynamics of the pitch subsystem (number 3).

Table 4.4: Sensor, actuator and system drift fault scenarios in a wind turbine.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Before giving the simulation information about the scenarios, let us remind the mathematical modeling of a drift. Given a variable θ (that can be a sensor measurement, a system parameter, etc.), let:

- θ_b be this variable before the occurrence of a drift,
- θ_a be this variable after the occurrence of a drift,
- T_b the beginning of the drift,
- T_e the end of the drift,
- δ the drift coefficient.

Then, the drift model is:

$$\begin{cases} \theta_a = \theta_b & \text{if} \quad t \leq T_b, \\ \theta_a = \theta_b + \delta(t - T_b) & \text{if} \quad T_b \leq t \leq T_e, \\ \theta_a = \theta_b + \delta(T_e - T_b) & \text{if} \quad t \geq T_e. \end{cases}$$

The information about the simulation times of these scenarios are summarized in Table (4.5) below. For all these scenarios, false alarms must be avoided. Thus, from a statistical point of view, the probability of having a 'wrong' detection must be equal to zero.

Scenario	Parameter	T_b	T_e	Duration	δ	Status code
Sc-Dr 1	ω_{g,m_2}	$1000s$	$1500s$	$500s$	$\delta = \frac{10rad.s^{-1}}{500}$	1
Sc-Dr 2	τ_g	$1000s$	$2000s$	$1000s$	$\delta = \frac{50N.m}{1000}$	2
Sc-Dr 3	β_{1,m_1}	$1000s$	$2000s$	$1000s$	$\delta = \frac{3^\circ}{1000}$	3
Sc-Dr 4	β_{2,m_2}	$1000s$	$2000s$	$1000s$	$\delta = \frac{3^\circ}{1000}$	4
Sc-Dr 5	(ξ, ω_n) Eq. (4.5)	$1000s$	$1500s$	$500s$	$\delta = \frac{10}{500}$	5

Table 4.5: Information concerning the five scenarios of drift in the wind turbine case study.

The next step will be the data acquisition and the feature extraction.

4.3.2.5 Data acquisition and feature extraction

The feature extraction in the wind turbine case must result in providing pertinent features that can achieve fault detection and isolation results. The faults cover different subsystems of the wind turbine. The extraction of features should make use of the limited knowledge that is available on the system. For instance, measurements constitute a major source of data that can be used for feature extraction. The latter is done by using different techniques discussed in Chapter 1, Section 1.2.2. In addition to measurements, few knowledge corresponding to an analytical description of a subsystem should also be used. Analytical models could help to select good features. For instance, the converter's models as well as the blade pitch subsystem models are known to be first order and second order models respectively. Thus, incorporating this knowledge can be very useful for the feature extraction part.

© 2014 Tous droits réservés. doc.univ-lille1.fr

The converter dynamics can be modeled by a first order transfer function:

$$\frac{\tau_g(s)}{\tau_{g,r}(s)} = \frac{\alpha_{gc}}{s+\alpha_{gc}}, \tag{4.4}$$

where α_{gc} is a converter specific parameter. It is given by the constructor. The blade pitch system is a hydraulic system modeled by a second order transfer function. The model is:

$$\frac{\beta(s)}{\beta_r(s)} = \frac{\omega_n^2}{s^2+2\xi\omega_n+\omega_n^2}, \tag{4.5}$$

where ξ and ω_n are the damping coefficient and the natural frequency respectively. The parameters ξ and ω_n are also given by the constructor. The variables that are considered to be known are the measured outputs listed in Table (4.3), as well as the control inputs τ_{g_r} and β_r. Using these control inputs and the models of the converter and the blade pitch subsystems (Eqs. (4.4) and (4.5) above), the values of τ_g, β_1, β_2 and β_3 can be reconstructed using analytical redundancy (Gertler, 1991). The reconstructed values are respectively denoted by $\hat{\tau}_g(\tau_{g_r})$, $\hat{\beta}_1(\beta_r)$, $\hat{\beta}_2(\beta_r)$ and $\hat{\beta}_3(\beta_r)$.

Considering the reconstructed values, the set of measurements and the set of redundant sensors, the features that can be extracted are:

$$\begin{aligned}
rwg &= \omega_{g,m_1} - \omega_{g,m_2}, && (4.6)\\
rwr &= \omega_{r,m_1} - \omega_{r,m_2}, && (4.7)\\
rwg1 &= \frac{P_{g,m}}{\tau_{g,m}} - \omega_{g,m_1} + 0.15 \times v_{wind}, && (4.8)\\
rwg2 &= \frac{P_{g,m}}{\tau_{g,m}} - \omega_{g,m_2} + 0.15 \times v_{wind}, && (4.9)\\
r_{\beta_1} &= \beta_{1,m_1} - \beta_{1,m_2} && (4.10)\\
r_{\beta_{1c}} &= \beta_{1,m_2} - \hat{\beta}_1(\beta_r) && (4.11)\\
r_{\beta_2} &= \beta_{2,m_1} - \beta_{2,m_2} && (4.12)\\
r_{\beta_{2c}} &= \beta_{2,m_2} - \hat{\beta}_2(\beta_r) && (4.13)\\
r_{\beta_3} &= \beta_{3,m_1} - \beta_{3,m_2} && (4.14)\\
r_{\beta_{3c}} &= \beta_{3,m_2} - \hat{\beta}_3(\beta_r) && (4.15)\\
r_{\tau} &= \tau_{g,m} - \hat{\tau}_g(\tau_{g_r}) && (4.16)
\end{aligned}$$

$$Pr_{\beta_{1c}}, Pr_{\beta_{2c}}, Pr_{\beta_{3c}} \text{ see Appendix A} \tag{4.17}$$

The online calculation of these features is straightforward thanks to these formulas. The first four features (rwg, rwr, $rwg1$ and $rwg2$) are used to detect faults related to the drive train. The others are used to detect faults related to the blade pitch subsystems (r_{β_1}, $r_{\beta_{1c}}$, r_{β_2}, $r_{\beta_{2c}}$, r_{β_3}, $r_{\beta_{3c}}$) and the converter (r_{τ}).

In addition to these features, three more are considered which are $Pr_{\beta_{1c}}$, $Pr_{\beta_{2c}}$ and $Pr_{\beta_{3c}}$. They are respectively the number of peaks in the signals $r_{\beta_{1c}}$, $r_{\beta_{2c}}$ and $r_{\beta_{3c}}$, computed on a moving time

© 2014 Tous droits réservés. doc.univ-lille1.fr

window $w(t_j)$ of size N_w. In Fig. (4.22), an example of a signal containing 4 peaks between two time stamps t_i and t_f is shown. The size of this time window is the same used by AuDyC as it will be seen later. The online calculation of the number of peaks in a signal on a moving time window is very time consuming. However, in this work, and in order to solve this issue, an iterative method is used for the computation of this number. The method is detailed in Appendix A.

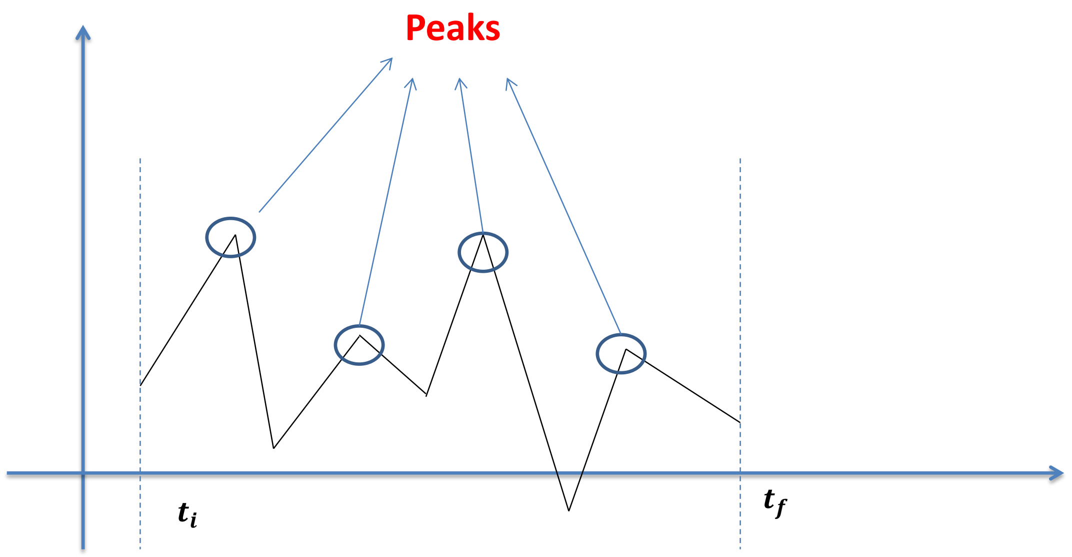


Figure 4.22: An example of a signal containing 4 peaks between two time stamps t_i and t_f.

In this case study, it is supposed that there are only data available in normal operating modes, and no failure data exist. These data are the measurements from the different sensors. Features are extracted from these measurements according to the equations above. Given these features, the next step will be the building of the G-arch, in order to achieve diagnosis results. It is noteworthy to remind that no prognosis result can be achieved when there is no knowledge on failure classes, which is the case in this case study.

4.3.2.6 Building the G-arch

The first step is the separation mechanism. As stated earlier, the separator block separates the feature space into several smaller feature spaces, each spanning its own E-arch. The combination of the signals coming from all the elementary architectures is done later in the grouping mechanism, which is based on the logic used in the separator block.

Before deciding which features to span which spaces, an analysis of the sensitivity of the features to the faults must be done. This analysis results in the creation of a signature matrix depicted in Table (4.6) below. The number '1' means that the feature is sensitive to the fault and '0' means the opposite. A feature is sensitive to a fault if the occurrence of this fault deviates the value of the feature from its mean value. Some feature in this example do not show any sensitiveness to any of

© 2014 Tous droits réservés. doc.univ-lille1.fr

Features \ Scenarios	Sc-Dr 1	Sc-Dr 2	Sc-Dr 3	Sc-Dr 4	Sc-Dr 5
$rwg1$	0	0	0	0	0
$rwg2$	1	0	0	0	0
r_{τ}	0	1	0	0	0
r_{β_1}	0	0	1	0	0
$r_{\beta_{1c}}$	0	0	1	0	0
r_{β_2}	0	0	0	1	0
$r_{\beta_{2c}}$	0	0	0	1	0
r_{β_3}	0	0	0	0	0
$r_{\beta_{3c}}$	0	0	0	0	0
rwg	1	0	0	0	0
rwr	0	0	0	0	0
$Pr_{\beta_{1c}}$	0	0	0	0	0
$Pr_{\beta_{2c}}$	0	0	0	0	0
$Pr_{\beta_{3c}}$	0	0	0	0	1

Table 4.6: Fault signature matrix for the wind turbine case study.

the considered faults. However, they are kept and used here because the idea is to show how to find a configuration of feature spaces that can be used to diagnose the existing faults.

For example, the sensor redundancy provides valuable features that can be used to know if one of the sensors is subject to a fault. Since two sensors are measuring the same variable, the difference between the values given by the two sensors must be equal to 0. Thus, if a sensor starts to show a drifting behavior, this can be directly seen on the difference between the values given by the two sensors. For instance, the features rwg, rwr, r_{β_1}, r_{β_2} and r_{β_3} are all very sensitive to any sensor fault affecting $\omega_{g,m_{1,2}}$, $\omega_{r,m_{1,2}}$, $\beta_{1,m_{1,2}}$, $\beta_{2,m_{1,2}}$ and $\beta_{3,m_{1,2}}$ respectively. In Fig. (4.23), it is seen that the measures coming from sensors ω_{g,m_1} and ω_{g,m_2} are moving away one from the other. Thus, the difference between those two values, which is rwg, shown in Fig. (4.24), presents a clear trend. Under the hypothesis that only one sensor is drifting, the latter cannot be determined by only observing this difference between those signals (*i.e.* rwg). The reason is that the trend present in this difference can be caused by the drift of one of the sensors (ω_{g,m_1} or ω_{g,m_2}). Thus, another signal is needed. Later on, this signal is showed.

The logic of the separation in this case must follow the conceptual logic defined earlier in this Chapter. In this case study, 5 spaces are spanned using the fourteen features. They are summarized in Table (4.7).

In each space, features define each of the dimensions. Direction vectors are defined and different direction indicator values are associated to them. In Fig. (4.25), the spaces, the direction vectors and the direction indicator values are all summarized. It is inmportant to note that in Fig. (4.25), conical shapes were surrounding the axis in both direction. It is because the direction vectors are

© 2014 Tous droits réservés. doc.univ-lille1.fr

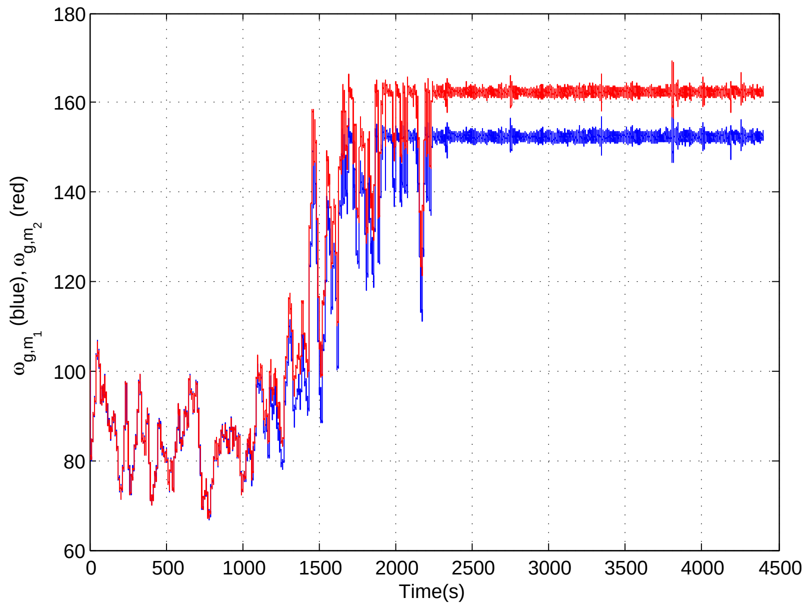


Figure 4.23: Sensor measurements ω_{g,m_1} and ω_{g,m_2} showing that one is clearly drifting from the other.

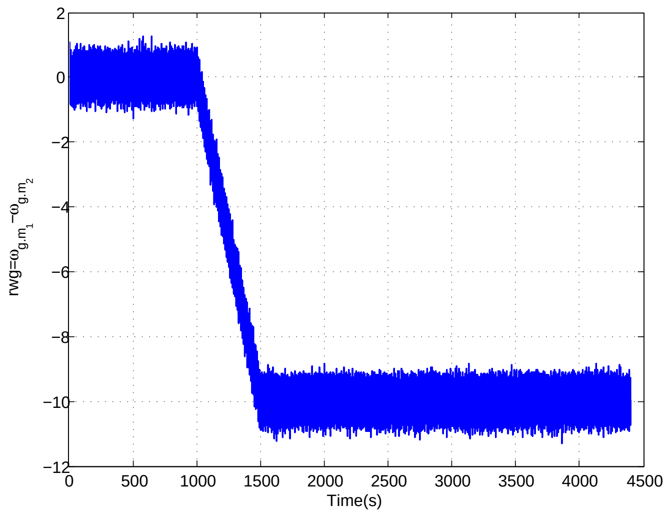


Figure 4.24: Features $rwg1$ and $rwg2$ with one ($rwg2$) showing a clear deviation.

along those axes, and because of the definition of the parameter p_M, the conical shapes appear (refer to chapter 3 for more explanation).

The second step is the treatment of each of the feature subspaces. In fact, an E-arch is built on each of these five subspaces. Each one of these E-archs has a structure that is depicted in Fig. (4.8). The modeling using AuDyC results in having one class corresponding to the normal operating mode.

© 2014 Tous droits réservés.

Name of the space	Features used to span it	Γ^{E_i}: direction indicaor
E_1	$(rwg,\, rwr,\, r_\tau)$	1: rwg; 2: rwr; 3: r_τ
E_2	$(rwg1,\, rwg2)$	1: $rwg1$; 2: $rwg2$
E_3	$(r_{\beta_1},\, r_{\beta_2},\, r_{\beta_3})$	1: r_{β_1}; 2: r_{β_2}; 3: r_{β_3}
E_4	$(r_{\beta_{1c}},\, r_{\beta_{2c}},\, r_{\beta_{3c}})$	1: $r_{\beta_{1c}}$; 2: $r_{\beta_{2c}}$; 3: $r_{\beta_{3c}}$
E_5	$(Pr_{\beta_{1c}},\, Pr_{\beta_{2c}},\, Pr_{\beta_{3c}})$	1: $Pr_{\beta_{1c}}$; 2: $Pr_{\beta_{2c}}$; 3: $Pr_{\beta_{3c}}$

Table 4.7: Different feature subspaces generated by the separated features.

When a drift occurs, this class will start migrating and its parameters deviate from the parameters of the normal class. This deviation is detected using the drift detection mechanism.

Then, the drift direction indicator is computed. The computation is done according to Table (4.7) and to Fig. (4.25). By doing the separation mechanism explained above, a drift will cause the migration of one or more classes, in different feature subspaces, to migrate in the direction of one of the axes. The values of the different direction indicator are all then calculated and used within a set of if-$then$ statements to find the failure mode behind the drift, as explained in Section 1.2.2. Following this logic, the set of if-$then$ statements corresponding to the list of drift faults presented in this Chapter:

Algorithm 1 if-$then$ statements for status code generation

```
if $\Gamma^{E_1} = 1$ AND $\Gamma^{E_2} = 1$ then
    $sc =$Sc-Dr 1
end if
if $\Gamma^{E_1} = 3$ then
    $sc =$Sc-Dr 2
end if
if $\Gamma^{E_3} = 1$ AND $\Gamma^{E_4} = 1$ then
    $sc =$Sc-Dr 3
end if
if $\Gamma^{E_3} = 2$ AND $\Gamma^{E_4} = 2$ then
    $sc =$Sc-Dr 4
end if
if $\Gamma^{E_5} = 3$ then
    $sc =$Sc-Dr 5
end if
```

The spaces E_3, E_4 and E_5 are not sensitive to the failure mode to Sc-Dr 1. The spaces E_2, E_3, E_4 and E_5 are not sensitive to the failure mode Sc-Dr 2. The spaces E_1, E_2 and E_5 are not sensitive to Sc-Dr 3. The spaces E_1, E_2 and E_5 are not sensitive also to the failure mode Sc-Dr 4. The spaces E_1, E_2, E_3 and E_4 are not sensitive to the failure mode Sc-Dr 5.

© 2014 Tous droits réservés. doc.univ-lille1.fr

The next section will show the results of simulation of these scenarios.

4.3.2.7 Simulation results

The parameters of the overall algorithm were set to: $\lambda_m = 0.001$ and $N_{win} = 100$, $\alpha = 99.5\%$, $F_{(2,98|\alpha)} = 5$, $F_{(3,97|\alpha)} = 4$, $p_M = 0.85$ and $N_p = 4$. More information about the drifts can be found in Table (4.5).

The first scenario corresponds to the drift of the sensor ω_{g,m_2}. The Figs. (4.23) and (4.24) show the behavior of the sensor measurements and the feature rwg respectively. In Fig. (4.26), several signals were shown. Fig. (4.26, (a)) shows the behavior of the signals $rwg1$ and $rwg2$. Fig. (4.26, (b)) shows the F-statistic computed during the first scenario. The F-statistic is directly proportional to the T^2. As explained in Chapter 3, Section 3.3.1, an additional parameter is used to ensure that there are no false alarms. This parameter is K_h and its tuning is made using historical data corresponding to normal operating mode. Its initial value is 1, and its increased until there is no point, in Fig. (4.26, (b)), that crosses the threshold. For the space E_1, its value was found to be $K_h = 37.5$ and the threshold is thus: $Th = 37.5 \times F_{(3,97|\alpha)} = 37.5 \times 4 = 150$. Once defined, the first point to cross the threshold triggers the direction indicator Γ^{E_1}. The signal Γ^{E_1} is drawn in Fig. (4.26, (c)). Its value is 1 (around $t \approx 1007s$), indicating that feature rwg is deviating (see Fig. (4.25)). The same goes to Fig. (4.26, (d)) and Fig. (4.26, (e)) respectively. In Fig. (4.26, (d)), the F-statistic within space E_2 is shown. In the same manner as for the space E_1, the value of K_h was tuned and it equals $K_h = 12$. The threshold is thus equal to 60 in space E_2. Also, the first point to cross the threshold triggers the direction indicator Γ^{E_2} (around $t \approx 1010$). The latter indicates the value 1, meaning that the feature $rwg1$ is deviating. Once the values of Γ^{E_1} and Γ^{E_2} are decided, the grouping algorithm 1 has its first 'if' statement verified. This statement yields that the status code behind this is 1. The value of the status code is shown in Fig. (4.26, (f)). Its value goes to 1 as soon as the value of Γ^{E_2} goes to 1 (around $t \approx 1010$). The reason is that the value of Γ^{E_2} has changed after the value of Γ^{E_1}.

The second scenario corresponds to the drift of the actuator, giving a slowly growing bias of the generator torque value. In Fig. (4.27), several signals were shown. Fig. (4.27, (a)) shows the behavior of the signals rwg and $rwgr$ and r_τ. From this figure, it can be seen that the feature r_τ is sensitive to this fault. Fig. (4.27, (b)) shows the F- statistic within the space E_1 is computed during this scenario. For the space E_1, the threshold was already computed and it equals 150 (in the paragraph above, it was explained how). The first point to cross the threshold triggers the direction indicator Γ^{E_1} (around $t \approx 2560s$). The latter indicates the value 3, meaning that the feature r_τ is deviating. As soon as its value goes to 3, the value of the status code goes to 2, because the second 'if' statement of the grouping mechanism (algorithm 1) is verified (Figs. (4.27, (c,d))).

The third scenario corresponds to the drift of the sensor $\beta_{1,m1}$. In Fig. (4.28), several signals were shown. Fig. (4.28, (a)) shows the behavior of the signals r_{β_1} and $r_{\beta_{1c}}$. From this figure, it can be seen how these two features are sensitive to this fault. Fig. (4.28, (b)) shows the F- statistic within space E_3 computed during this scenario. The parameter K_h is tuned to be equal to 42.5. The

© 2014 Tous droits réservés. doc.univ-lille1.fr

threshold is thus equal to 170. The first point to cross the threshold triggers the direction indicator Γ^{E_3}. The signal Γ^{E_3} is drawn in Fig. (4.28, (c)). Its value is 1 (around $t \approx 1035s$), indicating that feature r_{β_1} is deviating (see Fig. (4.25)). The same goes to the figures in Fig. (4.28, (d)) and Fig. (4.28, (e)) respectively. In Fig. (4.28, (d)), the F-statistic within space E_4 is shown. In the same manner, the value of K_h was tuned and it equals $K_h = 42.5$. The threshold is thus equal to 170. Also, the first point to cross the threshold triggers the direction indicator Γ^{E_4} (around $t \approx 1035$). The latter indicates the value 1, meaning that the feature $r_{\beta_{1c}}$ is deviating. Once the values of Γ^{E_3} and Γ^{E_4} are decided, the grouping algorithm 1 has its third 'if' statement verified. This statement yields that the status code behind this is 3. The value of the status code is shown in Fig. (4.28, (f)).

The fourth scenario corresponds to the drift of the sensor $\beta_{2,m2}$. In Fig. (4.29), several signals were shown. Fig. (4.29, (a)) shows the behavior of the signals r_{β_2} and $r_{\beta_{2c}}$. From this figure, it can be seen how these two features are sensitive to this fault. Fig. (4.29, (b)) shows the F- statistic within space E_3 computed during this scenario (the threshold is equal to 170 as seen in the paragraph above). The first point to cross the threshold triggers the direction indicator Γ^{E_3}. The signal Γ^{E_3} is drawn in Fig. (4.29, (c)). Its value is 2 (around $t \approx 1010s$), indicating that feature r_{β_2} is deviating (see Fig. (4.25)). The same goes to Fig. (4.29, (d)) and Fig. (4.29, (e)) respectively. In Fig. (4.29, (d)), the F-statistic within space E_4 is shown (the threshold is 170). Also, the first point to cross the threshold triggers the direction indicator Γ^{E_4} (around $t \approx 1040$). The latter indicates the value 2, meaning that the feature $r_{\beta_{2c}}$ is deviating. Once the values of Γ^{E_3} and Γ^{E_4} are decided, the grouping algorithm 1 has its fourth 'if' statement verified. This statement yields that the status code behind this is 4. The value of the status code is shown in Fig. (4.29, (f)).

The fifth signal corresponds to the drifting system fault of the pitch subsystem. In Fig. (4.30), several signals were shown. Fig. (4.30, (a)) shows the behavior of the signals $Pr_{\beta_{1c}}$, $Pr_{\beta_{2c}}$ and $Pr_{\beta_{3c}}$. From this figure, it can be seen that the feature $Pr_{\beta_{3c}}$ is sensitive to this fault. Fig. (4.29, (b)) shows the F- statistic within the space E_5 is computed during this scenario. For this space, the value of K_h is equal to 22.5 and the threshold is equal to 90. The first point to cross the threshold triggers the direction indicator Γ^{E_5} (around $t \approx 3660s$). The latter indicates the value 3 meaning that the feature $Pr_{\beta_{3c}}$ is deviating. As soon as its value goes to 3, the value of the status code goes to 5, because the fifth 'if' statement of the grouping mechanism (algorithm 1) is verified (Figs. (4.30, (c,d))).

The proposed approach is destined to work with a very high number of faults. The use of multiple feature spaces gives few advantages, namely:

– The other advantage of the approach is its robustness to noise since in each space, the current operating mode is modeled using AuDyC. This modeling provides a natural filter for data and since the modules inside the architecture are based on the parameters of classes, robustness to noise is thus obtained. In addition, the parameter K_h enhances this robustness by increasing the threshold for detection.

© 2014 Tous droits réservés. doc.univ-lille1.fr

- The ability of AuDyC to treat parasite data provides an additional robustness against outliers.
- It is possible that there could be detection in a space that does not lead to isolation of the fault. Then the possibilities are, as stated earlier, an unknown failure mode, a multiple failure mode or more time is needed for isolation. A deeper study is required to differentiate between those three cases, thus requiring the call for expertise.
- The faults considered in this Chapter are all drifting faults that were developed and applied on the benchmark model presented in (Odgaard et al., 2009). However, in reality a lot of abrupt faults are also present and a lot of scenarios of abrupt faults were defined in (Odgaard et al., 2009). This architecture was tested on the case of abrupt faults and the results are reported in (Chammas et al., 2013). The obtained results confirm that the methodology can be also applied for the abrupt fault case. In order to achieve a higher level architecture, it is however needed to improve the detection mechanism in such a way that it can differentiate between drifting faults and abrupt faults.

4.4 Conclusion

In this Chapter, a global architecture for supervision, diagnosis and prognosis has been established. This global architecture comes as an answer to the challenges that were presented in the general introduction of the thesis. It is built by the association of several modules, namely the data acquisition and feature extraction block, the separator block and the elementary architectures blocks.

The data acquisition and feature extraction block is where the extraction of features from raw measurements is done. Then, these features are separated into subsets such as each subset generates an elementary architecture. The elementary architecture joins together the modules of drift detection, characterization (severity indicator computation and direction indicator computation) and prognosis in case of available failure data. In case of non-available intial failure data and prior knowledge on failure modes, only the modules of detection and direction indicator computation are kept. These modules were developed and explained in Chapter 3.

The methodology is based on monitoring the parameters of dynamically updated clusters. Then, the drift detection module was used to detect drifts. In case of positive detection, and available failure classes, two drift characterization indicators were computed. These indicators are the direction indicator and the severity indicator. In case of a known failure mode behind the incipient fault is detected, a prognosis module is triggered to provide a RUL estimate. In case of positive drift detection but with non-available failure data, several direction indicators were calculated coming from several feature spaces, then regrouped together to find the failure mode causing the drift.

This chapter contained also the experimentation part of the work in this thesis. Two case studies were considered. The first case study is a water tank system. In this first case study, three drifting scenarios were considered in order to highlight the functioning of the elementary architecture. The second case study is a little bit more complex and concerns a wind turbine system. Five

© 2014 Tous droits réservés. doc.univ-lille1.fr

drifting scenarios were considered in this case study. The general aim of it was to highlight the building and the operation of the global architecture for supervision, diagnosis and prognosis.

This chapter was the last one of the thesis. What comes next is the general conclusion of this work and the perspectives for future work.

© 2014 Tous droits réservés. doc.univ-lille1.fr

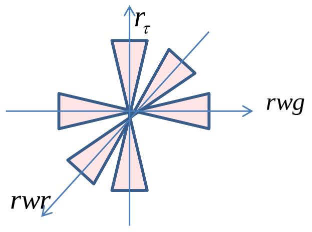
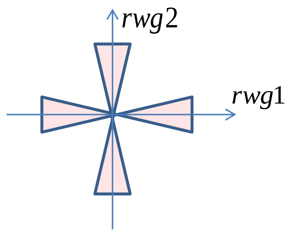
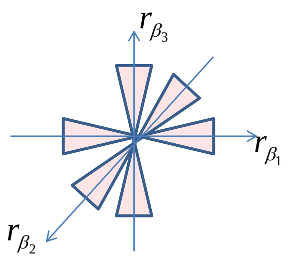
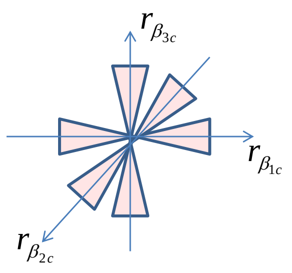
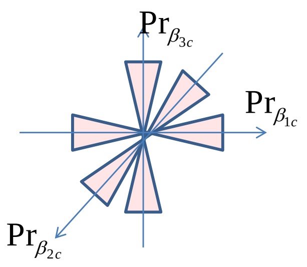
Space	Direction vectors and the associated direction Indicator values	Form of the space
E_1	$\left.\begin{matrix}D_{1+}^{E_1}=(1,0,0)\\ D_{1-}^{E_1}=(-1,0,0)\end{matrix}\right\}\Rightarrow\Gamma^{E_1}=1$ $\left.\begin{matrix}D_{2+}^{E_1}=(0,1,0)\\ D_{2-}^{E_1}=(0,-1,0)\end{matrix}\right\}\Rightarrow\Gamma^{E_1}=2$ $\left.\begin{matrix}D_{3+}^{E_1}=(0,0,1)\\ D_{3-}^{E_1}=(0,0,-1)\end{matrix}\right\}\Rightarrow\Gamma^{E_1}=3$	
E_2	$\left.\begin{matrix}D_{1+}^{E_2}=(1,0)\\ D_{1-}^{E_2}=(-1,0)\end{matrix}\right\}\Rightarrow\Gamma^{E_2}=1$ $\left.\begin{matrix}D_{2+}^{E_2}=(0,1)\\ D_{2-}^{E_2}=(0,-1)\end{matrix}\right\}\Rightarrow\Gamma^{E_2}=2$	
E_3	$\left.\begin{matrix}D_{1+}^{E_3}=(1,0,0)\\ D_{1-}^{E_3}=(-1,0,0)\end{matrix}\right\}\Rightarrow\Gamma^{E_3}=1$ $\left.\begin{matrix}D_{2+}^{E_3}=(0,1,0)\\ D_{2-}^{E_3}=(0,-1,0)\end{matrix}\right\}\Rightarrow\Gamma^{E_3}=2$ $\left.\begin{matrix}D_{3+}^{E_3}=(0,0,1)\\ D_{3-}^{E_3}=(0,0,-1)\end{matrix}\right\}\Rightarrow\Gamma^{E_3}=3$	
E_4	$\left.\begin{matrix}D_{1+}^{E_4}=(1,0,0)\\ D_{1-}^{E_4}=(-1,0,0)\end{matrix}\right\}\Rightarrow\Gamma^{E_4}=1$ $\left.\begin{matrix}D_{2+}^{E_4}=(0,1,0)\\ D_{2-}^{E_4}=(0,-1,0)\end{matrix}\right\}\Rightarrow\Gamma^{E_4}=2$ $\left.\begin{matrix}D_{3+}^{E_4}=(0,0,1)\\ D_{3-}^{E_4}=(0,0,-1)\end{matrix}\right\}\Rightarrow\Gamma^{E_4}=3$	
E_5	$\left.\begin{matrix}D_{1+}^{E_5}=(1,0,0)\\ D_{1-}^{E_5}=(-1,0,0)\end{matrix}\right\}\Rightarrow\Gamma^{E_5}=1$ $\left.\begin{matrix}D_{2+}^{E_5}=(0,1,0)\\ D_{2-}^{E_5}=(0,-1,0)\end{matrix}\right\}\Rightarrow\Gamma^{E_5}=2$ $\left.\begin{matrix}D_{3+}^{E_5}=(0,0,1)\\ D_{3-}^{E_5}=(0,0,-1)\end{matrix}\right\}\Rightarrow\Gamma^{E_5}=3$	 $\Pr_{\beta_{3c}}$ $\Pr_{\beta_{1c}}$ $\Pr_{\beta_{2c}}$

Figure 4.25: Different subspaces; their forms, the different direction vectors and the direction indicator indexes that are associated to them.

© 2014 Tous droits réservés. doc.univ-lille1.fr

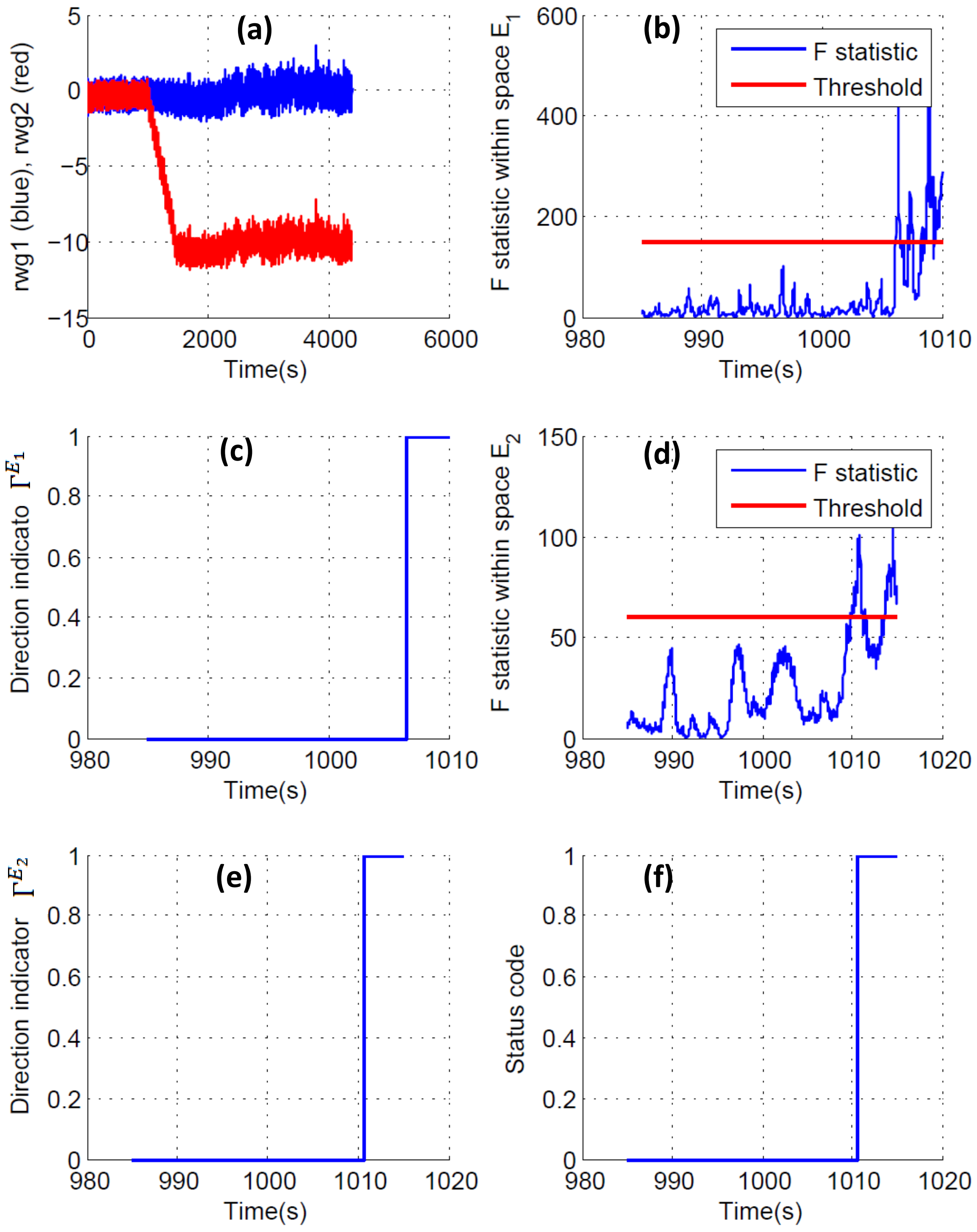


Figure 4.26: Sc-Dr 1 results.

© 2014 Tous droits réservés.

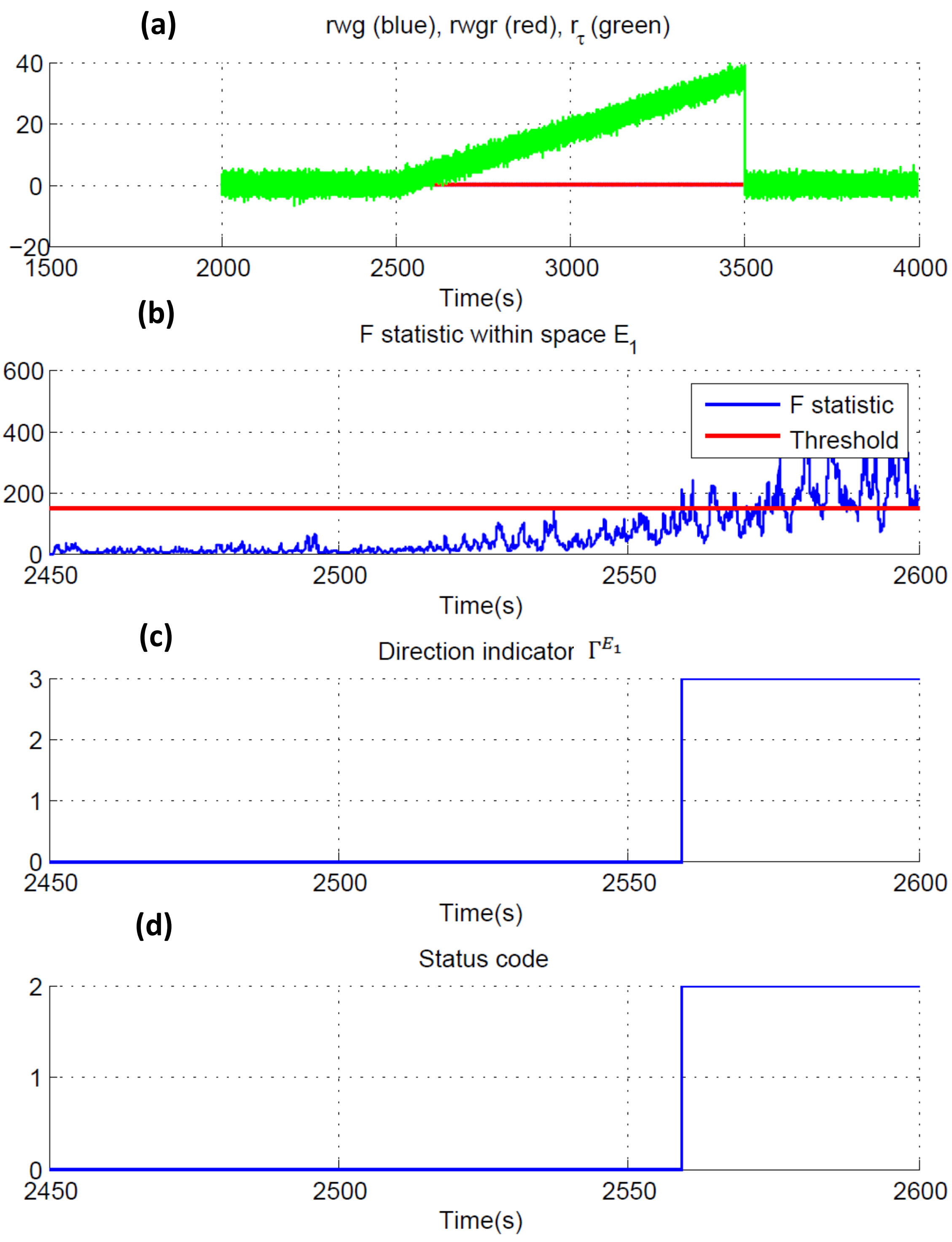


Figure 4.27: Sc-Dr 2 results.

© 2014 Tous droits réservés.
doc.univ-lille1.fr

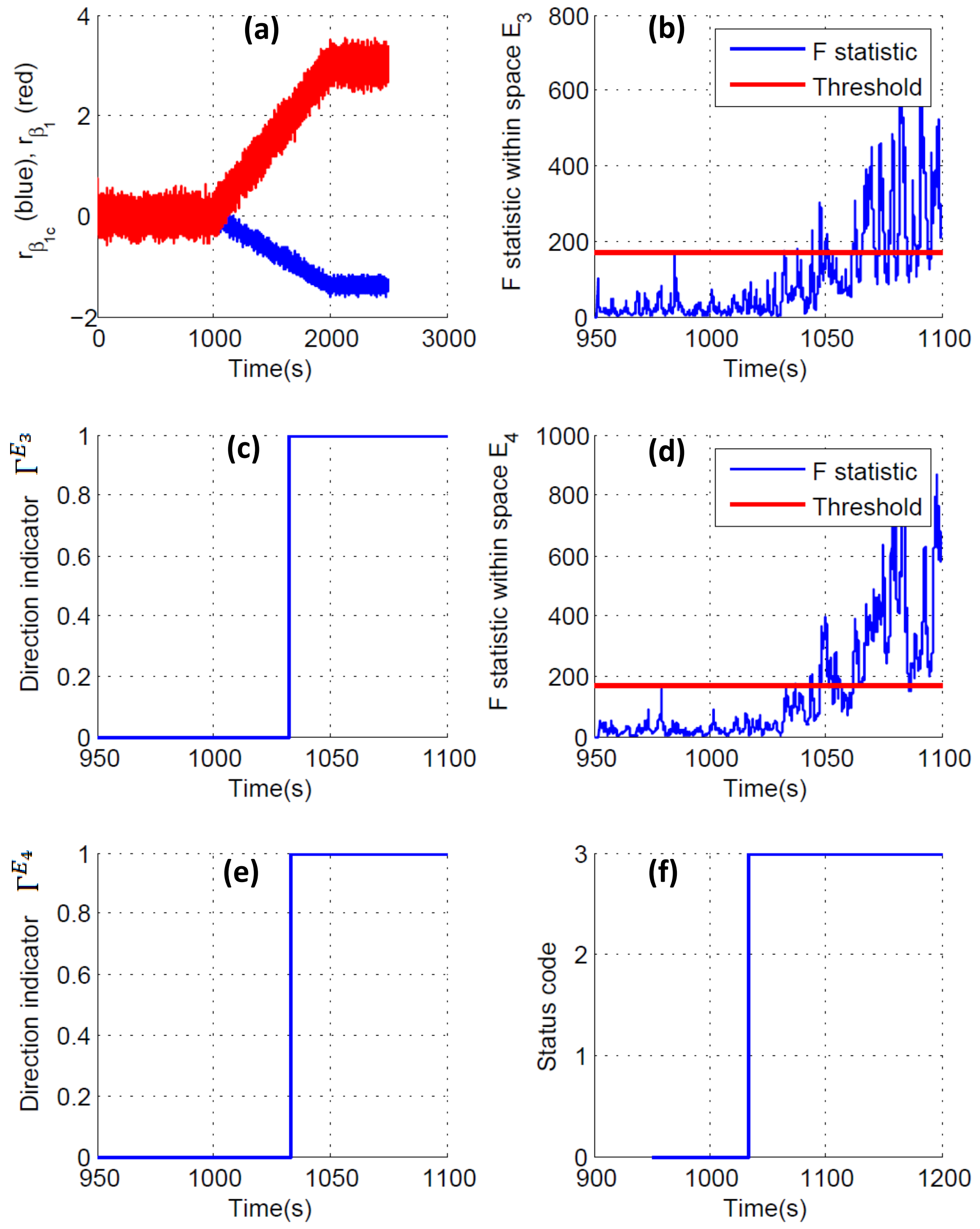


Figure 4.28: Sc-Dr 3 results.

© 2014 Tous droits réservés.

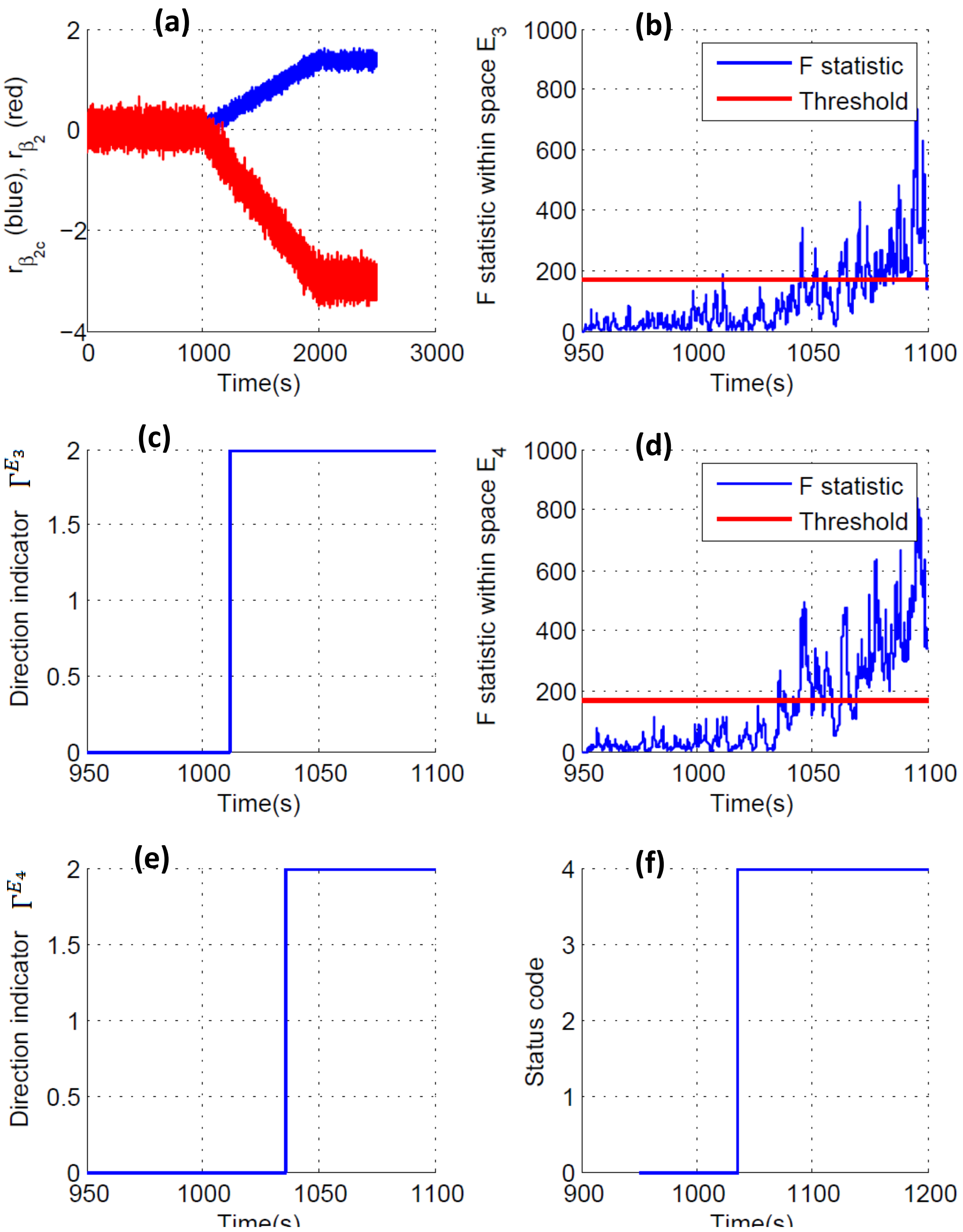


Figure 4.29: Sc-Dr 4 results.

© 2014 Tous droits réservés.

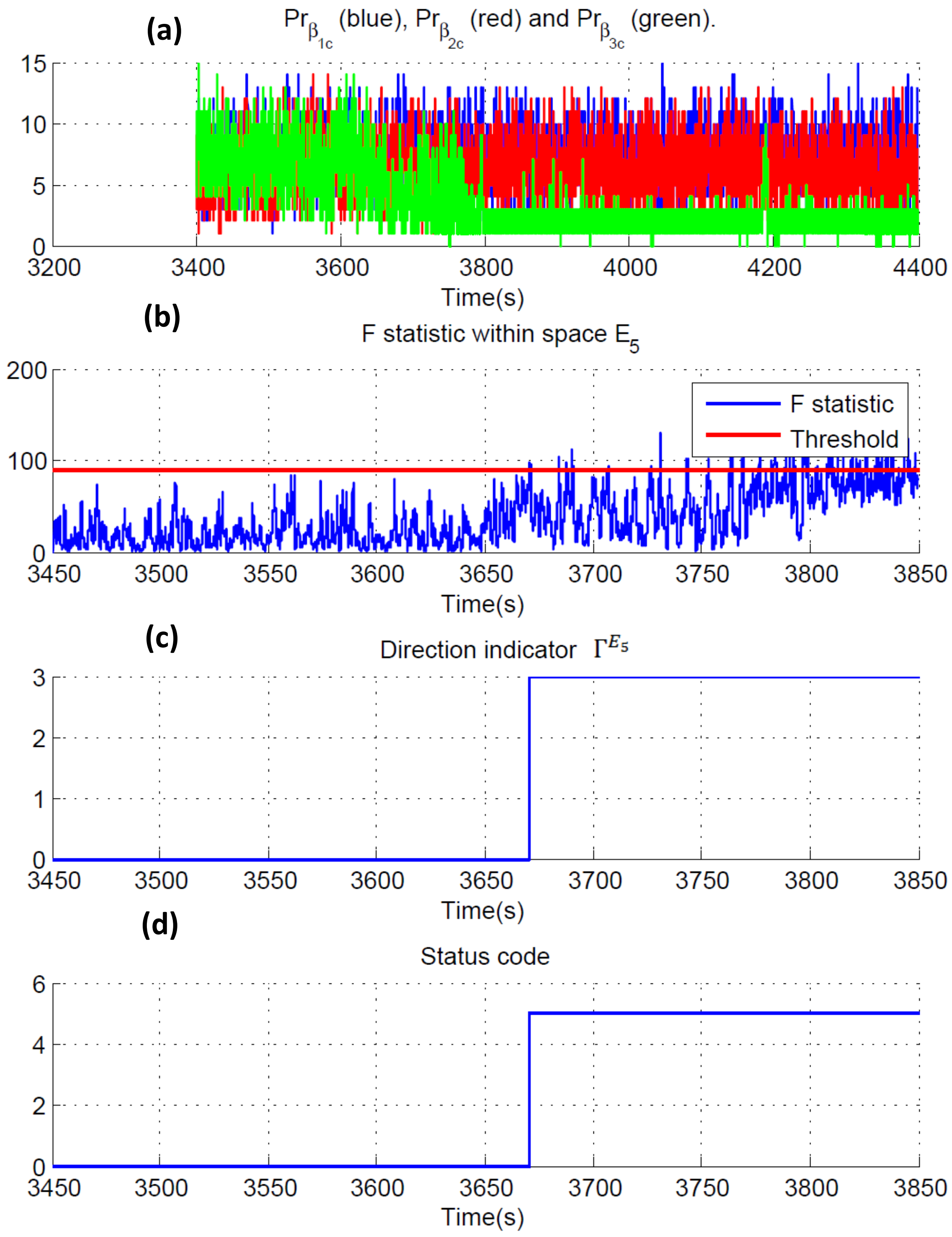


Figure 4.30: Sc-Dr 5 results.

© 2014 Tous droits réservés.

General conclusion and perspectives

The increasingly complexity and large exploitation constraints have marked the conception and the operation of modern industrial processes. These constraints have led to an increase of the attention given to maintenance strategies.

The work presented in this thesis deal with the integration of diagnosis and prognosis for optimizing maintenance of complex systems. The particular kind of faults taken into consideration in this work is process drifts. The diagnosis can determine line components that need to be repaired, when fault occurs. It is based on handling the drift by considering indicators for detection and characterization. Then, the prognosis can anticipate system failures by providing information on future states of the components on which preventive maintenance actions may be considered.

A complex system is composed of several heterogeneous components. The knowledge of the mechanisms related to the operational modes of these diverse components can be very different from one to another. Different techniques for condition monitoring, health assessment and prognosis are thus necessary to cover these diversities. However, a unified structural maintenance strategy, known as the condition-based maintenance is considered in this work. This maintenance strategy is universal, and it clearly defines the role of the supervision, diagnosis and prognosis architecture.

In chapter 1, the condition-based maintenance strategy has been detailed and its benefits were also shown. Then, a state-of-the-art study on prognosis techniques has shown that a new prognosis technique based on a dynamical decision space is needed, since a process drift creates a non-stationary evolving system. This issue is related to what is known as the concept drift problem.

The study of the concept drift problem is dealt with in chapter 2. A thorough study of the existing techniques resulted in a general classification of the algorithms that are able to handle concept drifts. The study was concluded by choosing an algorithm to be used as a modeling tool. The algorithm's name is AuDyC and it stands for Auto-adaptive and Dynamical clustering. The choice of AuDyC was backed by its characteristics that were judged adequate with the objectives of this work.

In chapter 3, modules have been developed to develop the main bricks that are used to create the architecture. These modules covered the different areas related with the challenges of the thesis work which were defined in the general introduction. They correspond to drift detection, drift

© 2014 Tous droits réservés. doc.univ-lille1.fr

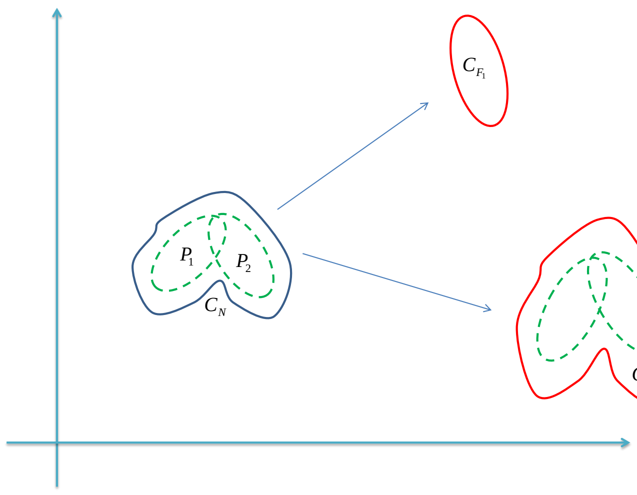


Figure 4.31: Non-convex class can drift possibly towards another non-convex class, as well as towards a convex class.

characterization and a prognosis model. The drift detection and drift characterization modules are built in the aim of calculating valuable indicators for the detection of drifts and the for the follow-up of their evolution. The prognosis module contains a prognosis algorithm that makes use of these indicators. It was built in such a way that it answers for few challenging characteristics that are associated to the objectives of the thesis.

In Chapter 4, a combination of these modules allowed the creation of an elementary architecture for supervision, diagnosis and prognosis based on drift detection and characterization. A set of elementary architectures formed a more global architecture that can be installed on a system level. An elementary architecture however is more adequate to component/subsystem level. Finally, in this Chapter, two case studies were shown on which the architecture was tested. The first case study is a tank system in which the role of an elementary architecture is highlighted. The second case study is a wind turbine in which a global architecture was built. The different steps before being able to create such an architecture were highlighted.

Discussion and perspectives:

Throughout the development of the work in this thesis, few hypotheses were made at some stages. The discussion and the perspective for future work are around those hypotheses. It can be roughly said: what if these hypotheses were not made?

The discussion and the points to be addressed in future work are:

– One major hypothesis emitted in Chapter 3, concerned the classes and their Gaussian nature. It was stated that a class can contain several Gaussian prototypes but that in this work only

© 2014 Tous droits réservés. doc.univ-lille1.fr

one prototype by class is considered. This means that an operational mode can be represented by a Gaussian class. Since the developed architecture is based on the features extracted from these measurements, this hypothesis can become restrictive in some cases. In fact, considering industrial process, measurements are made with sensors which are usually corrupted with Gaussian noise. However features that are extracted from these Gaussian measures can make them lose this property. In this case, non-convex classes have to be dealt with.

In Fig. (4.31), an example of incremental concept drift is illustrated in the case of non-convex classes. A non-convex class could drift towards a convex class, as well as towards another non-convex class. This shows that there must be adaptation of:

1. Drift detection module: The one developed in this work is based on the distancing of the evolving mode to the normal mode. It is translated by the crossing of the Hoteling T^2 value a threshold. This method is based on the fact that the distributions are Gaussian. The remedy is to develop another drift detection method that is capable of treating non-convex class drifts.

   One possibility is to consider the following metric (denoted M_{DD}) as a basis for drift detection:

   $$M_{DD} = d_{kl}(C_N, C_e(t_i)),$$

   where C_N and $C_e(t_i)$ are non-convex classes here, composed of several Gaussian prototypes. Thus, an evident difficulty is the adaptation of the Kullback-Leibler divergence to the classes. This difficulty can be overcome by using the fact a class is composed of several Gaussian prototypes. Another difficulty will be the definition of a threshold for detection. Further investigation into the nature of the constructed Kullback-Leibler divergence metric must be done in order to overcome this difficulty.
2. Severity indicator calculation module: the adaptation of this module is straightforward once the Kullback-Leibler divergence metric is adapted to the non-convex case.

– The second discussion and perspectives point is about a possible expansion of the architecture on hybrid systems. It has been supposed throughout the work of the thesis that there is one normal mode. In hybrid systems, there might be more than one normal mode. In addition, there might failure modes that are common to several operating modes, and other failures that are particular to one operating mode. This fact creates a lot of complexities considering the management of all these modes.

– Prognosis module: the idea of using several models to make online prognosis has shown to overcome basic difficulties concerning the choice of the sizes of sliding windows as well as model orders. It also allowed giving a confidence limits interval which, as discussed earlier, important for decision-making processes for maintenance.

© 2014 Tous droits réservés. doc.univ-lille1.fr

However, in this work, it was supposed that the degradation model is completely unknown. Thus, the model did not allow the incorporation of possible old knowledge on this degradation. This eliminated the choice of considering recursive update scheme for prognosis modeling, since the initial state are essential for the convergence of these models. Thus, a possible way to incorporate the knowledge of degradation mechanisms is by using them to initialize prognosis models that are update under a recursive scheme.

In addition to this first point, another point concerning prognosis can be considered and it is the combination rule that is used. In fact, in this work, the combination rule used is the mean value of several prognosis models, each based on a regression model and a window size. However, another combination rules can be considered. For instance, the median value as a combination technique could be interesting to use because it is not affected by the extreme values that can be given by an individual prognosis model. These extreme values affect however the mean value and thus make it more vulnerable to erroneous individual prognosis model.

– Unknown modes: in the work developed in this thesis, it was clear that the occurrence of a drift towards an unknown mode has not an evident solution. It is an interesting subject since it was considered that an unknown mode can be associated to a single unknown failure mode or a multiple fault scenario including the combination of known and/or unknown failure modes.

  A possible direction for future work at this subject is to consider possible interactions between features at the offline modeling stage. The idea is to characterize modes that could correspond to a multiple failure mode scenario, which combines two or more known failure modes. Then, in some situations, it could be possible to say that a multiple failure mode, combining these knwon failure modes, is causing the drift. If confirmed, the result would be having more possibilities to calculate a severity indicator and a RUL estimate, thus enhancing the decision-making process. Thus, more robust considerations for condition-based maintenance can be made.

– Finally, a last perspective for this work concerns the widening of the natures of the treated faults. In this work, only drifts were considered. The architecture developed in this work must also be capable to adjust to concept shifts, or abrupt changes, caused by abrupt faults. In order to do so, new detection mechanisms should be included in order to differentiate between abrupt faults and drifting faults. Once this differentiation is done, new tools should be developed to treat the abrupt fault case.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Thèse de Antoine Chammas, Lille 1, 2014

Appendices

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Thèse de Antoine Chammas, Lille 1, 2014

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Appendix A

The online calculation of the number of peaks in a signal on a moving time window is very time consuming. However, in this work, and in order to solve this issue, an iterative method is used for the computation of this number. In fact, given a signal $S(t_j)$ and a window $w(t_j) = [t_i : t_j]$, such as $i \leq j$, the number of data points inside this window is $N_w = j - i + 1$, let:

– $pk(S(w(t_j)))$ be the number of peaks in the signal in the time window $w(t_j)$,

– $pk(S(w(t_j))) = pk(S(t_i : t_j)) = pk(i : j) = pk(t_j)$ are four equivalent notations.

Thus, we have two basic formulas:

$$\begin{aligned} pk(i:j) &= pk(i-1:j) - pk(i-1:i+1), && \text{(A.1)} \\ pk(i:j) &= pk(i:j-1) + pk(j-1:j+1) && \text{(A.2)} \end{aligned}$$

Given these formulas, the iterative formula to calculate the peaks in a signal on a moving window is:

$$\begin{aligned} pk(t_{j+1}) &= pk(i+1:j+1) \\ &= pk(i+1:j) + pk(j-1:j+1) \\ &= pk(i:j) - pk(i:i+2) + pk(j-1:j+1) \\ &= pk(t_j) - pk(i:i+2) + pk(j-1:j+1) && \text{(A.3)} \end{aligned}$$

This result is used to iteratively calculate the values of the peaks in the signals $Pr_{\beta_{1c}}$, $Pr_{\beta_{2c}}$ and $Pr_{\beta_{3c}}$.

© 2014 Tous droits réservés. doc.univ-lille1.fr

© 2014 Tous droits réservés. doc.univ-lille1.fr

Bibliography

Alippi, C., G. Boracchi, and M. Roveri (2009). Just in time classifiers: managing the slow drift case. *Proceedings of International Joint Conference on Neural Networks*.

Alippi, C. and M. Roveri (2008). Just-in-time adaptive classifiers–part i: Detecting nonstationary changes. *IEEE Transactions on Neural Networks 19*, 1145–1153.

Almeida, R. D., S. D. S. Vicente, and L. Padovese (2002). New technique for evaluation of global vibration levels in rolling bearings. *Shock and Vibration 9*, 225–234.

An, D., J.-H. Choi, and N. H. Kim (2013). Prognostics101: A tutorial for particle filter-based prognostics algorithm using matlab. *Reliability Engineering & System Safety 115*, 161–169.

Andrade, F., I. Esat, and M. Badi (Chipping Norton, 1999). Gearbox fault detection using statistical methods, time-frequency methods (stft and wigner-ville distribution) and harmonic wavelet - a comparative study. In *Proceedings of the COMADEM*.

Aoufir, H. E. and D. Bouami (2003). Place des modèles d'optimisation dans le processus d'aide à la décision en maintenance. *Revue Francaise de Gestion Industrielle 22(3)*, 61–76.

Baena-Garcia, M., J. del Campo-vila, R. Fidalgo, and A. Bifet (2006). Early drift detection method. *ECML PKDD International Workshop on Knowledge Discovery from Data Streams*, 77–86.

Baillie, D. and J. Mathew (1996). A comparison of autoregressive modeling techniques for fault diagnosis of rolling element bearings. *Mechanical Systems and Signal Processing 10*, 1–17.

Baruah, P. and R. Chinnam (2005). Hmm's for diagnostics and prognostics in machining processes. *International Journal of Production Research 43(6)*, 1275–1293.

Basseville, M. and I. Nikiforov (1993). *Detection of Abrupt Changes: Theory and Application*. Prentice Hall, Inc.

Batzel, T. and D. Swanson (2009). Prognostic health management of aircraft power generators. In *IEEE Transactions on Aeorspace and Electronic Systems*.

Baydar, N. and A. Ball (2001). A comparative study of acoustic and vibration signals in detection of gear failures using wigner-ville distribution. *Mechanical Systems and Signal Processing 15*, 1091–1107.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Bechhoefer, E., A. Bernhard, D. He, and P. Banerjee (May 9-11, 2006). Use of hidden semi-markov models in the prognostics of shaft failure. In *Presented at the American Helicopter Society 62th Annual Forum, Phoenix, AZ*.

Bifet, A. and R. Gavalda (2007). Learning from time-changing data with adaptive windowing. *Siam International conference on Data Mining*.

Black, M. and R. J. Hickey (1999). Maintaining the performance of a learned classifier under concept drift. *Intelligent Data Analysis 3*, 453–474.

Boubacar, H. A. (2005). *Classification Dynamique de Données Non-Stationnaires Apprentissage et Suivi de Classes Évolutives*. Ph. D. thesis, Université des Sciences et Technologies de Lille.

Boubacar, H. A., S. Lecoeuche, and S. Maouche (2005). Audyc neural network using a new gaussian densities merge mechanism. In *7^{th} International Conference on Adaptive and Natural Computing Algorithms. Coimbra, Portugal*, pp. 155–158.

Boubacar, H. A., S. Lecoueuche, and S. Maouche (2004). Audyc neural network using a new gaussian densities merge mechanism. In *7th Conference on Adaptive and Neural Cmputing Algorithms*, pp. 155–158.

Box, G., G. Jenkins, and G. Reinsel (2008). *Time Series Analysis: Forecasting and Control*.

Brockwell, P. and R. Davis (2002a). *Introduction to Time Series and Forecasting*. Springer-Verlag New York, Inc.

Brockwell, P. J. and R. A. Davis (2002b). *Introduction to Time Series and Forecasting, Second Edition*. Springer.

Byington, C. S., M. J. Roemer, and T. Galie (2002). Prognostic enhancements to diagnostic systems for improved condition-based maintenance [military aircraft]. In *IEEE Aeorspace Conference Proceedings*.

Carlin, B. and S. Chib (1995). Bayesian model choice via markov chain monte carlo methods. *Journal of the Royal Statistical Society B57(3)*, 473–484.

Cauwenberghs, G. and T. Poggio (Denver, Colorado, 2000). Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing*.

Celaya, J. R., A. Saxena, S. Saha, and K. F. G. 4 (2011). Prognostics of power mosfets under thermal stress accelerated aging using data-driven and model-based methodologies. *Annual Conference of the Prognostics and Health Management Society, Montreal, Canada*.

Chammas, A., E. Duviella, and S. Lecoeuche (2013). Fault diagnosis for wind turbines based on dynamical culstering. In 52^{th} *International Conferene on Decision and Control, CDC*, Florenzia, Italy.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Cohena, E. and M. J. Strauss (2006). Maintaining time-decaying stream aggregates. *Journal of Algorithms 59(1)*, 19–36.

Cox, D. R. and H. D. Miller (1965). *The theory of stochastic processes*. Methuen and Company, London.

Dalpiaz, G., A. Rivola, and R. Rubini (May 2000). Effectiveness and sensitivity of vibration processing techniques for local fault detection in gears. *Mechanical Systems and Signal Processing 14(3)*, 387–412.

Ditzler, G. and R. Polikar (2011). Hellinger distance based drift detection for nonstationary environments. *CIDUE, Symposium on Computational Intelligence in Dynamic and Uncertain Environments*, 41–48.

Dong, M. and D. He (2007). A segmental hidden semi-markov model (hsmm)-based diagnostics and prognostics framework and methodology. *Mechanical Systems and Signal Processing 21(5)*, 2248–2266.

Dragomir, O. E., R. Gouriveau, F. Dragomir, E. Minca, and N. Zehrouni (2009). Review of prognostic problem in condition-based maintenance. In *European Control Conference, ECC*.

Dries, A. and U. Ruckert (2009). Adaptive concept drift detection. *Statistical Analyse and Data Mining Wiley Online Library*.

Dzakowic, J. E. (Virginia Beach, VA, 2007). Advanced techniques for the verification and validation of prognostics and health management capabilities. In *Machinery Failure Prevention Technologies, MFPT*.

Eltoft, T. (1998). A new neural network for cluster detection and labeling. In *IEEE Transactions on Neural Networks*.

Enders, W. (2009). *Applied Econometric Times Series*.

Engle, R. (1986). Autoregressive conditional heteroscedasticity. *Journal of Econometrics 31*, 307–327.

Firmino, P. R. A., P. S. de Mattos Neto, and T. A. Ferreira (2014). Correcting and combining time series forecasters. *Neural Networks 50*, 1–11.

Gama, J. and G. Castillo (2006). Learning with local drift detection. *Advanced Data Mining and Applications 4093*, 42–55.

Gama, J., P. Medas, and G.Castillo (2004). Learning with drift detection. *Lecture Notes in Computer Science 3171*.

Ganti, V., J. Gehrke, and R. Ramakrishnan (2002). Mining data streams under block evolution. *SIGKDD Explorations 3(2)*, 1–10.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Gao, J., W. Fan, J. Han, and P. S. Yu (2007). A general framework for mining concept-drifting data streams with skewed distributions. In *SIAM International Conference on Data Mining*.

Garga, A., B. Elverson, and D. Lang (Haymarket, 1997). Ar modeling with dimension reduction for machinery fault classification. In *Critical Link: Diagnosis to Prognosis*.

Gertler, J. (1991). Analytical redundancy methods in fault detection and isolation. In *Proceedings of IFAC/IAMCS Symposium on Safe Process*.

Gertler, J. (1998). *Fault Detection and Diagnosis in Engineering Systems*. Taylor and Francis.

Gertler, J. and G. Cao (2005). Design of optimal structured residuals from partial principal component models for fault diagnosis in linear systems. *Journal of Process Control 15(5)*, 585–603.

He, D., W. Shenliang, P. Banerjee, and E. Bechhoefer (2006). Probabilistic model based algorithms for prognostics. In *IEEE Aerospace Conference Proceedings, Big Sky, MT, UnitedStates, IEEEComputer Society, Piscataway*, NJ, United States, pp. 1–10.

Heng, A., A. C. C. Tan, J. Mathew, N. Montgomery, D. Banjevic, and A. K. S. Jardine (2009). Intelligent condition-based prediction of machinery reliability. *Mechanical Systems andSignal Processing 23(5)*, 1600–1614.

Heng, A., S. Zhang, A. C. Tan, and J. Mathew (2009). Rotating machinery prognostics: State of the art, challenges and opportunities. *Mechanical Systems and Signal Processing 23*, 724–739.

Herzoga, M., T. Marwalab, and P. Heynsa (February 2009). Machine and component residual life estimation through the application of neural networks. *Reliability Engineering and System Safety 94(2)*, 479–489.

Hess, A., G. Calvello, and P. Frith (2005). Challenges, issues, and lessons learned chasing the "big p". real predictive prognostics. part 1. In *IEEE Aerospace Conference*.

Huanga, R., L. Xia, X. Lib, C. R. Liuc, H. Qiud, and J. Leed (January 2007). Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods. *Mechanical Systems and Signal Processing 21(1)*, 193–207.

Hulten, G., L. Spencer, and P. Domingos (2001). Mining timechanging data streams. In *7th ACM SIGKDD international conference on knowledge discovery and data mining*.

Ikonomovska, E., J. Gama, and S. Dzeroski (2011). Learning model trees from evolving data streams. *Data Mining Knowledge Discovery 23*, 128–168.

Ikonomovska, E., J. Gama, R. Sebastião, and D. Gjorgjevik (2009). Regression trees from data streams with drift detection. *Lecture Notes in Computer Science 5808*, 121–135.

Jardine, A. K., D. Lin, and D. Banjevic (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing 20*, 1483–1510.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Jaw, L. C. (1999). Neural networks for model-based prognostics. In *IEEE Aerospace Conference*.

Johnson, R. A. and D. W. Wichern (2012). *Applied Multivariate Statistical Analysis*. Prentice Hall.

Jouin, M., R. Gouriveau, D. Hissel, M.-C. Péra, and N. Zerhouni (2014). Prognostics of pem fuel cell in a particle filtering framework. *International Journal of Hydrogen Energy 39*, 481–494.

Jover, P. and H. Hyotyniemi (2004). Signal processing of vibrations for condition monitoring of an induction motor. In *ISCCSP, First International Symposium on Control, Communications and Signal Processing*, New York.

Kadlec, P., R. Grbic, and B. Gabrys (2011). Review of adaptation mechanisms for data-driven soft sensors. *Computers and chemical Engineering 35*, 1–24.

Kifer, D., S. Ben-David, and J. Gehrke (Toronto, Canada, 2004). Detecting changes in data streams. In *Proceedings of the* 30^{th} *VLDB Conference*.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis, Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift 8*, 281–300.

Klinkenberg, R. and I. Renz (1998). Adaptive information filtering: Learning drifting concepts. In *In AAAI98/ICML-98 Workshop Learning for Text Categorization*.

Kuncheva, L. (2009). Using control charts for detecting concept change in streaming data. Technical report, Technical Report BCS-TR-001.

Lebold, M., K. Reichard, C. S. Byington, and R. Orsagh (2002). Osa-cbm architecture development with emphasis on xml implementations. In *MAINTENANCE AND RELIABILITY CONFERENCE*.

Lebold, M. and M. Thurston (2001). Open standards for condition-based maintenance and prognosis systems. *In the* 5^{th} *Annual Maintenance and Reliability Conference, Gatlinburg, USA*.

Lecoeuche, S. and C. Lurette (2003, June). Auto-adaptive and dynamical clustering neural network. In *ICANN'03 Proceedings*, Istanbul, Turkey.

Lee, M. L. T., G. A. Whitmore, F. Laden, J. E. Hart, and E. Garshick (2009). A case-control study relating railroad worker mortality to diesel exhaust exposure using a threshold regression model. *Journal of Statistical Planning and Inference 139*, 1633–1642.

Leobold, M., K. Reichard, and D. Boylan (2003). Utilizing dcom in an open system architecture framework for machinery monitoring and diagnostics. In *Aerospace conference proceedings*.

Li, G., J. Qin, Y. Ji, and D.-H. Zhou (2010). Reconstruction based fault prognosis for continuous processes. *Control Engineering Practice 18*, 1211–1219.

Li, Y., S. Billington, C. Zhang, T. Kurfess, S. Danyluk, and S. Liang (1999). Adaptive prognostics for rolling element bearing condition. *Mechanical Systems and Signal Processing 13*, 103–113.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Li, Y. and P. Noilkitsaranont (2009). Gas turbine performance prognostic for condition-based maintenance. *Applied Energy 86*, 2152–2161.

Lia, G., S. J. Qinb, Y. Jic, and D. Zhoua (2010). Reconstruction based fault prognosis for continuous processes. *Control Engineering Practice 18*, 1211–1219.

Liu, Z., X. Yin, Z. Zhang, D. Chen, and W. Chen (2004). Online rotor mixed fault diagnosis way based on spectrum analysis of instantaneous power in squirrel cage induction motors. *IEEE Transactions on Energy Conversion 19*, 485–490.

Ljung, L. (1999). *System Identification Theory for the User*. PTR Prentice Hall Information and System Sciences Series.

Lu, N., Guangquan, and J. Lu (2010). Detecting change via competence model. *Case-Based Reasoning, Research and Development*, 201–212.

Lugtigheid, D., A. Jardine, and X. Jiang (2007). Optimizing the performance of a repairable system under a maintenance and repair contract. *Quality and Reliability Engineering International 23(8)*, 943–960.

Manwell, J. F., J. G. McGowan, and A. L. Rogers (2010). *Wind Energy Explained: Theory, Design and Application*.

Markou, M. and S. Singh (2003). Novelty detection: a review–part i: statistical approaches. *SIGNAL PROCESSING 83*, 2481–2497.

Mehala, N. (2010). *Condition Monitoring and Fault Diagnosis of Induction Motor Using Motor Current Signature Analysis*. Ph. D. thesis, National Institute of Technology, Kurukshetra, India.

Miller, A. (1999). *A new wavelet basis for the decomposition of gear motion error signals and its application to gearbox diagnostics*. Ph. D. thesis, The pennsylvania State University, State College, PA.

Minku, L., A. White, and X. Yao (2010). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering 22(5)*, 730–742.

Montgomery, N. and T. Jefferis (2007). The effect of minor maintenance on condition-based maintenance models, icoms. In *Engineers Australia*, Melbourne, Australia.

Montgomery, N., T. Lindquist, M. Garnero, R. Chevalier, and A. Jardine (2006). Reliability functions and optimal decisions using condition data for edf primary pumps. In *International Conference on Probabilistic Methods Applied to Power Systems, PMAPS*, Stockholm, Sweden.

M.Rausand and A.Hoyland (New Jersey, 2004). *System Reliability Theory: Models, Statistical Methods and Applications*. Wiley-Interscience, Hoboken.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Muller, A. (Juin 2005). *Contribution à la maintenance prévisionnelle des systèmes de production par la formalisation d'un processus de pronostic*. Ph. D. thesis, Thèse de doctorat, Université Henri Poincaré Nancy I, France.

Nima, G., M. Lin, M. Murthy, Y. Prasad, and D. Yong (2009). A review on degradation models in reliability analysis. In *Proceedings of the* 4^{th} *World Congress on Engineering Asset Management*.

Nishida, K. and K. Yamauchi (2007). Detecting concept drift using statistical testing. *Discovery Science-Springer*.

Niu, G. and B.-S. Yang (2009). Dempster-shafer regression for multi-step-ahead time-series prediction towards data-driven machinery prognosis. *Mechanical Systems and Signal Processing 23*, 740–751.

Noortwijk, V. (2009). A survey of the application of gamma process in maintenance. *Reliability Engineering & System Safety 94*, 2–21.

Ocak, H., K. Loparo, and F. Discenzo (2007). Online tracking of bearing wear using wavelet packet decomposition and probabilistic modeling: a method for bearing prognostics. *Journal of Sound and Vibration 302(4-5)*, 951–961.

Odgaard, P. F., J. Stoustrup, and M. Kinnaert (Barcelona, Spain, June 30 - 3 July 2009). Fault tolerant control of wind turbines - a benchmark model. In 7^{th} *IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*.

Pandian, A. and A. Ali (2010). A review of recent trends in machine diagnosis and prognosis algorithms. *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM) 2*, 320–328.

Paris, P., M. Gomez, and W. Anderson (1961). A rational analytic theory of fatigue. *Trends Eng. 13*, 9–14.

Pennell, M. L., C. Whitmore, and M. L. T. Lee (2010). Bayesian random-effects threshold regression with application to survival data with nonproportional hazards. *Biostatistics 11(1)*, 111–126.

Peysson, F., M. Oulasdine, R. Outbib, J.-B. Leger, O. Myx, and C. Allemand (2008). Damage trajectory analysis based prognostic. In *International Conference on Prognostics and Health Management*.

Pham, H. T. and B.-S. Yang (2010). Estimation and forecasting of machine health condition using arma/garch model. *Mechanical Systems and Signal Processing 24*, 546–558.

Pugnoa, N., M. Ciavarellab, P. Cornettia, and A. Carpinteria (2006). A generalized paris' law for fatigue crack growth. *Journal of the Mechanics and the Physics of Solids 54*, 1333–1349.

Ray, A. and S. Tangirala (1996). Stochastic modeling of fatigue crack dynamics for on-line failure prognostics. *IEEETransactions on Control Systems Technology 4*, 443–451.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Ribot, P. (Décembre 2009). *Vers l'intégration diagnostic/pronostic pour la maintenance des systèmes complexes*. Ph. D. thesis, Thèse de doctorat, Université de Toulouse, France.

Romeu, J. L. (2001). Statistical analysis of reliability data, part 1: random variables, distribution parameters, and data. *Journal of the Reliability Analysis Centre*, 9–14 firstquarter.

Salganicoff, M. (1993). Density-adaptive learning and forgetting. In *Proceedings of the 10^{th} International Conference on Machine Learning*.

Saporta, G. (2011). *Probabilités, Analyse des données et Statistique*.

Saxena, A., J. Celaya, E. Balaban, S. Saha, B. Saha, and M. Schwarbacher (Denver, CO, USA, 2008). Metrics for evaluating performance of prognostics techniques. In *2008 International Conference on Prognostics and Health Management*.

Saxena, A., J. Celaya, B. Saha, S. Saha, and K. Goebel (2009). On applying the prognostic perfomance metrics. In *Proceedings of the 1^{th} Annual Conference of the Prognostics and Health Management Society*, San Diego, CA, USA.

Sayed-Mouchaweh, M. (2011). Adaptive time window size to track concept drift. In *International Conference on Machine Learning and Applications and Workshops, ICMLA*.

Schoen, R. and T. Habetler (1995). Effects of time-varying loads on rotor fault detection in induction machines. *IEEE Transactions on Industry Applications 31*, 900–906.

Senechal, O. (2004). *Pilotage des systèmes de production vers la performance globale, Habilitation à Diriger des Recherches.* Ph. D. thesis, L'Université de Valenciennes et du Hainaut Cambresis.

Si, X.-S., W. Wang, C.-H. Hu, and D.-H. Zhou (2011). Remaining usefil life estimation - a review on the statistical data driven approaches. *European Journal of Operational Research 213*, 1–14.

Sikorska, J., M. Hodkiewicz, and L. Ma (2011). Prognostic modeling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing 25*, 1803–1836.

Sobhani, P. and H. Beigy (2011). New drift detection method for data streams. *Adaptive and Intelligent Systems-Springer*.

Spinosa, E. J., A. P. de Leon, and J. Gama (2007). Olindda: a cluster-based approach for detecting novelty and concept drift in data streams. In *Proceedings of the 2007 ACM symposium on Applied Computing*.

Su, B., Y.-D. Shen, and W. Xu (2008). Modeling concept drift from the perspective of classifiers. In *IEEE Conference on Cybernetics and Intelligent Systems*.

Todinov, M. (Chichester England, 2005). *Reliability and Risk Models-Setting Reliability Requirements,*. John Wiley & Sons Ltd.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Tonshoff, H., X. Li, and C. Lapp (2003). Application of fast haar transform and concurrent learning to tool-breakage detection in milling. *IEEE/ASME Transactions on Mechatronics 8*, 414–417.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. *Trinity College, Dublin, Ireland, TCD-CS-2004-15*.

Tsymbal, A., M. Pechenizkiy, P. Cunningham, and S. Puuronend (2008). Dynamic integration of classifiers for handling concept drift. *Information Fusion 9*, 56–68.

Vachtsevanos, G., F. Lewis, M. Roemer, A. Hess, and B. Wu (2006). *Intelligent Fault Diagnosis and Prognosis for Engineering Systems.* Jhon Wiley and Sons, Inc.

Venkatasubramanian, V., R. Rengaswamy, K. Yin, and S. Kavuri (2003a). A review of process fault detection and diagnosis. part i: Quantitative model-based methods. *Computers and chemical Engineering 27*, 293–311.

Venkatasubramanian, V., R. Rengaswamy, K. Yin, and S. Kavuri (2003b). A review of process fault detection and diagnosis. part ii: Qualitative models and search strategies. *Computers and chemical Engineering 27*, 313–326.

Venkatasubramanian, V., R. Rengaswamy, K. Yin, and S. Kavuri (2003c). A review of process fault detection and diagnosis. part iii: Process history based methods. *Computers and chemical Engineering 27*, 327–346.

Volov, A. (2014). Verification and validation of prognostic and health management. *Prognostics 2 17*.

Wanga, W. Q., M. Golnaraghib, and F. Ismail (July 2004). Prognosis of machine health condition using neuro-fuzzy systems. *Mechanical Systems and Signal Processing 18(4)*, 813–831.

Widmer, G. and M. Kubat (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning 32(2)*, 1–35.

Wu, W., J. Hu, and J. Zhang (2007). Prognostics of machine health condition using an improved arima-based prediction method. In 2^{nd} *IEEE Conference on Industrial Electronics and Applications, ICIEA 2007*.

Yan, J., M. Koc, and J. Lee (2004). A prognostic algorithm for machine performance assessment and its application. *Production Planning and Control 15*, 796–801.

Zille, V. (Janvier 2009). *Modélisation et évaluation des stratégies de maintenance complexes sur des systèmes multi-composants.* Ph. D. thesis, Thèse de doctorat, Université de Technologie de Troyes, France.

Zliobate, I. (2009). Learning under concept drift: an overview. Technical report, Vilnius University.

© 2014 Tous droits réservés.

doc.univ-lille1.fr

© 2014 Tous droits réservés.

doc.univ-lille1.fr

List of Publications

Publication(s) in international journals:

- *Drift Detection and Characterization for Condition Monitoring- Application to Dynamical Systems with Unknown Failure Modes*, IMA, Oxford Journal of Management Mathematics, 2014

Publication(s) in international conferences:

- *Fault Diagnosis of Wind Turbine Drive Train Faults based on Dynamical Clustering* , 52nd IEEE Conference on Decision and Control, Florence, Italy, December 10-13, 2013.

- *Prognosis based on handling drifts in dynamical environments - Application to a wind turbine benchmark*, International Conference on Machine Learning and Applications, ICMLA: Workshop: Machine Learning Algorithms, Systems and Applications, Boca Raton, Florida, USA, December 12-15, 2012.

- *Condition monitoring architecture for maintenance of dynamical systems with unknown failure modes*, 2nd IFAC Workshop on Advanced Maintenance Engineering, Services and Technology, Seville, Spain, November 22-23, 2012.

- *Drift detection and characterization for fault diagnosis and prognosis of dynamical system*, International Conference on Scalable Uncertainty Management, SUM, Marburg, Germany, September 17, 2012.

Supervision of switching systems based on dynamical classification approach, European Safety and Reliability Conference, ESREL, Troyes, France, September 9, 2011.

© 2014 Tous droits réservés. doc.univ-lille1.fr

Thèse de Antoine Chammas, Lille 1, 2014

© 2014 Tous droits réservés.

doc.univ-lille1.fr

Abstract

The work presented in this thesis deal with the integration of diagnosis and prognosis for optimizing maintenance of complex systems. The particular kind of faults taken into consideration in this work is process drifts. A generic architecture for supervision, diagnosis and prognosis based on handling drifts is developed. The supervision/diagnosis can determine line components that need to be repaired, when fault occurs. It is based on handling the drift by considering indicators for detection and characterization. Then, the prognosis can anticipate system failures by providing information on future states of the components on which preventive maintenance actions may be considered. The developed methodology is based on treating data collected on a system, dropping the need for a mathematical description of the system. Using historically saved data and online generated data, a decision space is constructed, in which different classes corresponding to normal and failure operating modes exist. An incipient fault will cause a drift in the decision space, and consequently a change in the parameters of the classes. Thus, the decision space needs to be dynamically updated in order to treat those drifts. Then, indicators for drift detection and characterization, based on the parameter changes, are calculated. The aim of drift detection it is to detect, as soon as possible, the occurrence of a drift. Drift characterization deals with finding the failure mode causing the drift, and with calculating an indicator that reflects the health state. Prognosis models the evolution of the actual health system indicator, and estimates a RUL (Remaining Useful Life) as well as a confidence interval associated to it. All these aspects are combined together to form the generic architecture for supervision, diagnosis and prognosis.

Keywords: *preventive maintenance, supervision, diagnosis, prognosis, dynamical systems, dynamical classification*

Résumé

Le travail présenté dans cette thèse traite l'intégration de diagnostic et de pronostic pour l'optimisation et l'aide à la décision d'actions de maintenance de systèmes complexes. Les défauts considérés dans ce travail sont les dérives. Une architecture générique pour la surveillance, le diagnostic et le pronostic basée sur le traitement des dérives est développée. La surveillance et le diagnostic permettent de déterminer le composant en défaillance et le mode opérationnel de défaut qui initie la dérive. Ensuite, le pronostic permet d'anticiper les défaillances du système en fournissant des informations sur les futurs États des composants. En se basant sur le retour du pronostic, des actions de maintenance préventive peuvent être alors considérées. Les méthodes employées sont basées sur les données. Ceci est d'autant plus intéressant parce que l'hypothèse de connaissance de modèle physique n'y est pas toujours vérifiée. A partir des données mesurées est construit (ou sont construits) un (des) espace(s) de représentation et de décision dans lequel sont matérialisées ou modélisées les classes de fonctionnement. L'environnement dynamique dans lequel se trouve le système peut (va) provoquer des dérives de fonctionnement qui auront comme conséquence un changement dans la structure des classes. Les techniques proposées sont sensibles à ces dérives et permettent de calculer des indicateurs de détection et de caractérisation de la dérive. La détection de dérive vise à détecter, dès que possible, l'apparition d'une dérive. La caractérisation de dérive a pour but de trouver le mode de défaillance entraînant la dérive et de calculer un indicateur de dégradation qui reflète l'état de santé. Le pronostic est fait en modélisant l'évolution de l'indicateur de dégradation du système et permet l'estimation d'RUL (durée de vie utile) ainsi qu'un intervalle de confiance qui lui est associé. Tous ces aspects sont combinés ensemble pour former l'architecture générique pour la surveillance, de diagnostic et de pronostic.

Mots-clés : *maintenance prévisionnelle, surveillance, diagnostic, pronostic, systèmes dynamiques, classification dynamique*

© 2014 Tous droits réservés.