PhD IN DATA SCIENCE AND COMPUTATION

DATA-STREAM DRIVEN FUZZY-GRANULAR APPROACHES FOR SYSTEM MAINTENANCE

Letícia Decker de Sousa

Bologna 2022

Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN
DATA SCIENCE AND COMPUTATION

Ciclo 33

Settore Concorsuale: 02/A1 - FISICA SPERIMENTALE DELLE INTERAZIONI FONDAMENTALI
Settore Scientifico Disciplinare: FIS/01 - FISICA SPERIMENTALE

DATA-STREAM DRIVEN FUZZY-GRANULAR APPROACHES FOR SYSTEM MAINTENANCE

Presentata da:
Letícia Decker de Sousa

Coordinatore Dottorato:
Andrea Cavalli

Supervisore:
Daniele Bonacorsi

Co-Supervisori:
Claudio Grandi
Daniel Furtado Leite

Esame finale anno 2022

I dedicate this Ph.D. Thesis to Antonio Guglielmi.

Acknowledgements

I would like to thank everyone who participated to my doctoral journey, directly or indirectly. To my professors and friends that helped me to get in, or made my days easier. To those who helped me to be someone able of making this commitment even with the fear of not being able to keep it until the end. To all the people who didn't let me give up. To the countless friends and colleagues I gained and lost during these years. To all the relatives who contributed on my upbringing. To my antagonists who encouraged me so much to move forward and overcome every adversity, my sincere thanks.

To my Ph.D. supervisors, Prof. Daniele Bonacorsi and Dr. Claudio Grandi, for all support and opportunities during these four years of hard work, being always available to support me through the crooked paths of the Italian immigration bureaucracy. To my co-supervisor, Prof. Daniel Leite, for the countless late nights writing articles together, running experiments, and making last-minute adjusting to A.I. model solutions. For having introduced me to Granular computing, changing the direction of my thesis and my life. For all attention, patience and for every insistence on improving my performance, believing in me and my potential, I am very grateful to all of you.

To my mother, Karin Decker de Sousa, and my father, Ronaldo Carvalho de Sousa, for all love and attention in my whole life. Specially, to my little brother, Alexander Decker de Sousa, to be present, being always amorous during our long talks about computer science, life and loves. Thank you for encouraging me to study and be a better person and professional.

To Antonio Guglielmi for all support and love, being close and away according with my needs, having patience and sensibility to recognise them. Thanks to be the best and loving companion possible in each moment, being always the most comforting part of my days. To Frenci and Jack Guglielmi, the sweetest people ever. To Marina Gherardi and Giuseppe Guglielmi that always treated me as a daughter, making me feel welcome. To you, my deepest recognition.

To Aline Ferreira, Milena Passini, and Lorena Pozzo for being the most faithful, patient, and supportive friends ever. To Pedro Silva Jr., for having ardently and insistently encouraged me to get a doctorate. To Chicca Maria for her unconditional love and daily companionship even in the countless isolation by COVID-19. And last but not least, to the INFN staff and colleagues, specially to Fabio Viola and Barbara Martelli for helping specially in the beginning of this work, and to Panagiota Dimopoulou (Iota) for helping in everything I needed in a superhuman efficiency way, thank you.

Abstract

Intelligent systems are currently inherent to the society, supporting a synergistic human-machine collaboration. Beyond economical, social, and climate factors, energy consumption is strongly affected by the performance of computing systems. The quality of software functioning may invalidate any improvement or optimisation attempt. In addition, data-driven machine learning algorithms are the basis for human-centered applications in the information age. Interpretability became one of the most important features of computational systems. Software maintenance is a critical discipline to support automatic and life-long operation of systems. A computing center maintenance system is a software that manages a set of software that composes a complex mosaic. As most software registers its inner events by means of logs, the analysis of logs is an approach to keep system operation. Logs are characterised as Big data assembled in large-flow streams. Logs are unstructured, heterogeneous, imprecise, and uncertain. This thesis addresses fuzzy and neuro-granular concepts and methods to provide accurate and understandable maintenance solutions. Anomaly detection and log parsing are approached by fuzzy and neuro-granular methods, which deal with the uncertainty inherent to streaming data. Anomaly detection identifies ideal time periods for detailed software analyses. Log parsing provides deeper semantics interpretation of the anomalous occurrences. The solutions given in this thesis evolve over time and are of general-purpose, being highly applicable, scalable, and maintainable. Granular classification models, namely, Fuzzy set-Based evolving Model (FBeM), evolving Granular Neural Network (eGNN), and evolving Gaussian Fuzzy Classifier (eGFC), are compared considering the anomaly detection problem. The evolving Log Parsing (eLP) method is proposed in this thesis to approach the automatic parsing applied to system logs. All the methods perform recursive mechanisms to create, update, merge, and delete information granules according with the data behavior. For the first time in the evolving intelligent systems literature, the proposed method, eLP, is able to process streams of words and sentences. Essentially, regarding anomaly detection accuracy, FBeM achieved $(85.64 \pm 3.69)\%$; eGNN reached $(96.17 \pm 0.78)\%$; eGFC obtained $(92.48 \pm 1.21)\%$; and eLP reached $(96.05 \pm 1.04)\%$. Besides being competitive, eLP particularly generates a log grammar, and presents a higher level of model interpretability.

Contents

1 Introduction **13**

1.1 Objectives . . . 16

1.2 Contributions . . . 16

1.3 Organisation . . . 17

2 Data Center and Maintenance Systems **19**

2.1 Grid Infrastructure . . . 20

2.2 State of the Art . . . 21

2.2.1 Reactive Maintenance . . . 22

2.2.2 Predictive Maintenance . . . 25

2.3 System Maintenance Metaphors . . . 26

2.3.1 Health Care . . . 26

2.3.2 Speech Translation . . . 27

2.4 Proposed Framework . . . 29

2.5 Summary . . . 31

3 Evolving Fuzzy Granular Computing **33**

3.1 Granular Computing as Paradigm . . . 33

3.2 Fundamentals . . . 34

3.2.1 Interval Analysis . . . 35

3.2.2 Fuzzy Sets . . . 38

3.2.3 Aggregation Functions . . . 40

3.3 Granulation . . . 42

3.3.1 Domains . . . 42

3.3.2 Information Granules . . . 43

3.3.3 Evolving Granular Systems . . . 43

3.4 Summary . . . 44

4 Evolving Neural and Granular Systems **45**

4.1 Log-based Anomaly Detection . . . 46

4.2 Evolving Granular Classifiers . . . 48

4.2.1 FBeM: Fuzzy Set-Based Evolving Modeling . . . 49
4.2.2 eGFC: Evolving Gaussian Fuzzy Classifier . . . 51
4.2.3 eGNN: Evolving Granular Classification Neural Network . . . 56
4.3 eGNN: Aggregation Operators as Neurons . . . 61
4.3.1 T-norms . . . 62
4.3.2 S-norms . . . 62
4.3.3 Averaging Operators . . . 63
4.3.4 Ordered Weighted Averaging . . . 63
4.3.5 Compensatory T-S Operators . . . 64
4.3.6 Aggregation Neuron Model . . . 64
4.4 Summary . . . 65

5 eLP: evolving Log Parsing 67
5.1 Preliminary Concepts . . . 68
5.1.1 Logs as a language . . . 68
5.1.2 The eLP granular framework . . . 70
5.1.3 Log parsing . . . 71
5.2 eLP: evolving Log Parsing . . . 72
5.2.1 Preliminaries . . . 73
5.2.2 Model structure . . . 73
5.3 Online Learning from Word Streams . . . 74
5.3.1 Creating granules . . . 74
5.3.2 Updating granules . . . 75
5.3.3 Merging granules . . . 76
5.3.4 Deleting Attributes and Granules . . . 76
5.4 Learning algorithm . . . 76
5.5 Summary . . . 78

6 Methodology 79
6.1 Log File . . . 79
6.1.1 StoRM Service Logs . . . 79
6.2 Anomaly Detection Problem . . . 80
6.2.1 About the AD Dataset . . . 81
6.2.2 Log Self-learning Methodology . . . 81
6.3 Log Parsing Problem . . . 82
6.4 Evaluation Indexes . . . 83
6.4.1 Interpretability . . . 83
6.4.2 Accuracy . . . 86
6.4.3 Compactness . . . 86
6.4.4 Execution Time . . . 87
6.5 Summary . . . 87

7 Experimental Results 89
7.1 Anomaly Detection . . . 89
7.1.1 Performance Comparison . . . 89
7.1.2 eGNN Aggregation Layer . . . 94
7.2 Log Parsing . . . 97
7.3 Summary . . . 100

8 Non-stationary System Maintenance 103
8.1 Health System Model . . . 106
8.2 Proposed Models . . . 108
8.2.1 Naive State Models . . . 108
8.2.2 Granular State Models . . . 111
8.3 Summary . . . 115

9 Conclusion 117
9.1 Summary . . . 117
9.2 Future Research . . . 119

A How to read the Multi-Class Confusion Matrix 127

B Tutorial of the Log-based evolving Maintenance Service 129
B.1 How to choose the log file . . . 130
B.2 About Implementation Decisions . . . 131
B.2.1 How can we approach big data problems? . . . 131
B.2.2 Why are fuzzy-set solutions a good idea to log-based predictive maintenance systems? . . . 131
B.2.3 Why is it a good idea to use granular computing in a data-driven problem? . . . 132
B.2.4 Why is it important to identify intervals of anomaly behavior of a service? . . . 132
B.2.5 Why is it a good idea to do service-oriented maintenance system? . . . 132
B.3 Software Description . . . 132
B.3.1 Anomaly detection . . . 132
B.3.2 Log Parsing . . . 137
B.3.3 Event Tracking . . . 137
B.3.4 Code Availability and Setting up . . . 138

C Non-granular Log-based approaches 139
C.1 Intelligent Diagnostic Maintenance Framework . . . 139
C.1.1 Log Preprocessing . . . 140
C.1.2 Volatility . . . 140
C.1.3 Anomaly Detection Clustering . . . 141
C.1.4 Anomaly Selection . . . 141
C.1.5 Algorithm . . . 142

C.2 Methodology . . . 143
C.2.1 About the Dataset . . . 143
C.3 Experimental Design . . . 144
C.4 Data Analysis . . . 144
C.5 Results . . . 145
C.6 Conclusion . . . 148

Bibliography **149**

Chapter 1

Introduction

These days, computing systems are embedded in the society, supporting the synergistic collaboration between the human and the machine. This tendency began in the middle of the last century with the development of the basis of the modern computer. Thenceforth, the digital revolution became a race of the best applications using series of paradigms, languages, methodologies, etc., to develop a favourable environment to turn the computing technology ubiquitous.

There are a set of important impacts of the digitisation process from robotics, social applications, passing through spacial race to climate change issues, among others. Beyond the economic factor, the energy consumption and the heat generation are strongly affected by the performance of computing systems. Although hardware efficiency is essential for computing performance, the software quality may invalidate any optimization attempt. In the ages of information, data-driven machine learning algorithms are the basis of the human-centered applications, in which the output interpretability become one of the most important hot topics.

In particular, the software development and maintenance are critical disciplines in the computer science. Both disciplines are interdependent and the basis of the automatising growth, supporting the evolution of the computing systems. In the 70's, Lehman [1] defines a set of laws related to how the software tends to evolve over time. The laws were mainly reviewed and extended until the begging of the 21^{st} century [2], creating the basis of the Software Engineering discipline, *Software Evolution*. According to Lehman, the development of programming methodology, high-level languages and associated concepts can not be neglected, being considered by far the most important step for successful computer usage by many experts, followed by the additional methodologies and tools to support the program maintenance.

To Lehman, software maintenance is generally used to describe all changes made after the first software release. A software application deteriorates since the operational environment changes over time, being corrected by repair or replacement. These changes are related to programming language or tools improvement because the program execution may be wrong, inappropriate or outdated.

The basis for the maintenance of a computing system is the program maintenance. According to the SPE-classification [1], there are 3 classes of programs:

- S-type: (Specified) programs that are derivable from a static and well-defined specification, in which it is possible formally being proved as correct or not.
- P-type: (Problem solving) programs that attempt to settle problems that can be formally defined, but can not directly be solved using computers, in which the theoretical problem must be approached using heuristics or approximations.
- E-type: (Evolutionary) programs that are the implementation of human processes or real-world problems.

The system maintenance applied to data centers[1] is based on a data-driven composition of the users and services behaviors on a network environment. Based in both human activity and real-world problem, the system maintenance is an E-type application.

According to the Software Evolution Theory (SET) [1, 2], programs are modified and adapted according to the changing environment. Based on empirical research, SET summarises the results initially in five laws extended posteriorly in more other three laws supported by invariant properties related to classes of software projects.

This research area is very controversial specially because the software development and maintenance are human-centered fields. Nevertheless, some statements can be made with certainty: (i) the software code is constantly in progress until to be more convenient economically to be replaced instead to update it (Law of Continuing Change), and (ii) even the change is an invariant property of the big software project, the way that the change occurs is not dependent just on the type of project but also of the team responsible of the software maintenance and development.

Since the technical team and the system itself are in constant evolution, a program maintenance needs an evolving model. Precisely, the data center maintenance is a composition of software-based maintenance systems and their synergy. Obviously, the data center maintenance is a harder problem to solve than a single program maintenance, but it can be modularized in program-based modules orchestrated by an omniscient manager to approach the decision-making role of the whole infrastructure.

With the increasing of the system of interest (Law of Continuing Growth), the software tends to be intrinsically more complex (Law of Increasing Complexity), and tends to be less satisfactory to the user (Law of Declining Quality). During the software story, composed by all produced releases, the code tends to increase its entropy specially in big and mismanaged projects. The high entropy blocks the implementation of new functionalities and the maintenance of the system.

Essentially, a data center maintenance systems are a software to manage a set of software. In addition, open and closed-source software cohabit in the same ecosystem, complicating and squishing even more the stabilising process of the system. Increasing the quantity of services, program installation and setup, libraries, programming environment, tools, etc; in which each software probably has a different owner with different code quality following a different development methodology; the complexity of the bug/anomaly conjunction has a propensity to increase until the data center inoperability.

In this sense, the data center is an evolving scenario that tends naturally to crash if nothing is done. In general, online machine learning approaches are used as auxiliary tools to uphold the sustenance

[1]The argument of this thesis.

of the log-based computing systems. Using logs to promote the system preservation is an inexpensive and smart choice as most software registers the inner events in logs.

The ease of obtaining logs is counterbalanced by the difficulty of handling them. Logs are characterised as Big Data assembled in large-flow streams of messy information. Being unstructured, heterogeneous, imprecise, and uncertain, to extract useful information from logs is not a simple task. As a data stream, logs expires over time, declining their relevance during the data treatment chain. In this sense, maintenance systems are essentially a real-time issue, requiring online data processing to dispose of computing resources continuously.

As logs are a deeply redundant data type, to identify which details are important is crucial to maintenance systems. To help in this task, the Granular Computing paradigm proposes to control the abstraction level of a problem through the granularization degree of the system information. The smaller the information granule, the larger the zoom of the system view, and vice-versa. This mechanism permits that the observer changes convenently the problem perspective. Information granules are clusters of similar data which together form an transitory painting as a system representation.

Furthermore, system maintenance is an ultimately decision-making problem, as such it concerns with the explainability potential of the taken decisions. The Explainable Artificial Intelligence (XAI) appeals to get transparent and intelligible the results of the machine learning approaches, understanding the impacts of the taken decisions as well as acknowledging the process mechanisms of the systems. Solutions must provide transparent decisions, i.e., term-by-term interpretations in comprehensive format in a known language, furnishing besides the correct inference, phenomenon explanations as event causes, and the reasons of the made choices. For such, models must be based on fair objective functions, using impartial data through systematic processes.

Being comprehensive must take in account the limited human capacity to process data and understand processes. Remembering that a decision is a good decision just in a given instant (probably), it is important to consider not just the accuracy of the solution but also if the achieved answer is still relevant. Beyond this hamper, uncertainty is an intrinsic feature of the data. Imprecise and uncertain, log processing is the basis of decision-making systems. A good compromise between these obstacles is to combine fuzzy and granular concepts into the same approach, coupling the best of logs can provide with the demand of system maintenance to generate interpretable analysis of the running software.

The log-based system maintenance is a hard multidisciplinary computing problem, leading with hot topics as automatising, efficiency, uncertain data, semantic, ontology, reliability, data mining, optimisation, machine learning, fuzzy logic, natural language processing (NLP), predictive maintenance, forecasting, diagnostics, monitoring, among others. The research area is a huge mosaic of problems, being arduous to be completely covered in all of its diversity in a single approach. The main objective is to keep the system working within a pre-established Quality of Service standard through the adoption of smart tools.

General-purpose solutions are preferable to ad-hoc algorithms in order to facilitate the infrastructure managing. However, to develop general-purpose algorithms is needed a greater problem domain than the used to generate ad-hoc solutions, being more recommended for cluster of systems commonly found in data centers. Besides the solutions' elegance and sophistication, this strategy increases the applicability and maintainability of complex systems, reflecting in cost savings.

This chapter is organized as follows. Section 1.1 describes the thesis' objectives. Section 1.2 details the thesis' contributions, and Section 1.3 shows how the thesis is organised by chapters.

1.1 Objectives

This research project aims to provide a data-stream driven solutions for system maintenance through human-centered approaches, focusing in the interpretability of the models and outputs. Many questions were addressed during the course of this thesis, and being surpassed to reach the main objective. To achieve that, it was necessary to answer the following topics:

- how to process a huge amount of logs in a real-time applications,
- how to effectively provide a better tooling support for maintenance' staff,
- how to transform a textual, unstructured, and unsupervised logs in numerical, structured, and well-behaved data,
- how to convert automatically unsupervised in supervised data in order to be used by predictive approaches,
- how to analyse the log content automatically,
- how to link different types of maintenance problems in a data processing pipeline,
- how to minimise the need of log processing in the composition of solutions,
- how to model a system state, and how to identify a state changing, and
- how to compress the logs database, minimising the storage usage.

Other details are addressed in the chapter descriptions through the thesis.

1.2 Contributions

The thesis contributions are clustered in (i) methodological, (ii) modelling, and (iii) computational. The main answered question is how to approach the system maintenance problem considering an automatic process that covers all types of systems. In (i), the methodology defines how to transform generic log datasets in an appropriate format to be directly processed by maintenance problems. Since the format must be applied in thousands of different software, the data pre-processing must to be cheap and generic, considering data transformation, feature extraction, and data classification. Therefore, the methodology provides a conversion of a textual, unstructured, and unsupervised logs in a supervised numerical vectors, maintaining the system behaviour's characteristics. The methodology generates a numerical vector stream that follows a Gaussian Distribution. This feature is taken advantage by the self-learning method to generate supervised datasets. Even the predictive maintenance is not approached in this thesis, the generated dataset is appropriated to be directly applied to solve this problem.

In (ii), the computing center maintenance is divided in the classical computing problems: self-learning classification, anomaly detection, log parsing, and anomaly prediction, being organized in a data processing pipeline with intermediary outputs. Two possible computing design approaches are proposed to correlate the computing problems: the health care, and the speech translation metaphors. In addition, four different models are proposed to address the system state considering the outputs of the fuzzy-granular anomaly detection approaches. In this sense, a complete system state machine can be structured using one of the proposed state models. Using event-oriented textual logs, the system modelling embraces the whole process, from log pre-processing to system state through a general-purpose methodology.

In (iii), the anomaly detection and log parsing problems are implemented using the fuzzy-granular paradigm. Regarding to the anomaly detection problem, it is implemented the feature extraction and self-learning method to proceed the experiments using the three classical fuzzy-granular algorithms: Fuzzy-Set-Based evolving Model (FBeM), evolving Granular Neural Network (eGNN), and evolving Gaussian Fuzzy Classifier (eFGC). Considering the log parsing problem, it is implemented the first fuzzy-granular algorithm - evolving Log Parsing (eLP) - to deal with textual input to provide parsing of a log language through a formal grammar generation. All the implementations are evaluate experimentally, in which the AD solutions are confronted related to accuracy, time execution, and number of rules.

1.3 Organisation

This thesis is organized into nine chapters:

- This chapter states the overview of the system maintenance problem, highlighting the difficulties related to the software development and evolution. The thesis' objectives, contributions, and organisation are outlined.
- Chapter 2 introduces the thesis' motivation and study case, as well as the literature review of system maintenance, providing metaphors to approach the problem statement, detailing the used framework.
- Chapter 3 summarises the aspects related to Granular Computing applied in this thesis, as the mathematical foundations related to interval arithmetic, fuzzy sets, aggregation functions, and granulation concepts.
- Chapter 4 provides a general-purpose modelling to the data-stream driven anomaly detection problem applied to computing center maintenance through a self-learning algorithm to categorise the log data using three different fuzzy-granular classifiers.
- Chapter 5 implements a general-purpose fuzzy-granular algorithm to the data-stream driven parsing problem applied to logs, providing the syntactic analysis through a formal grammar

generation, proposing a data compression scheme.

- Chapter 6 describes the methodology used in both problem, including how to build the used database from raw system logs, and the formalisation of the indexes used to evaluate the solutions.
- Chapter 7 describes the experiments and shows the results of the experiments executed to validate the aforesaid implementations, considering the indexes described in the previous chapter.
- Chapter 8 proposes four state models based on granular anomalies, with different levels of dimensional complexity. These models provide a screenshot of the system functioning, being pictorial (or data) representation of system states.
- Chapter 9 concludes this research, summarising the thesis content, and outlines the future research possibilities.

Chapter 2

Data Center and Maintenance Systems

Separating High-Energy Physics (HEP) developments and experiments from computational approaches to data analysis is currently an infeasible task. For instance, the Large Hadron Collider (LHC) at CERN in Geneva (Switzerland) produces several petabytes of data yearly from particle collision experiments and simulations. Exabytes of data are required to be processed, including metadata, and data from a posteriori analysis. Therefore, a huge amount of computing resources is needed for data storage, and to support a computing throughput of around 10^5 jobs per day followed by an increasing demand for efficient data sharing among computing centers through high-speed networks.

The Worldwide LHC Computing Grid (WLCG) has been created to support HEP experiments at CERN. The grid infrastructure is an essential asset to support the LHC discoveries. Nonetheless, grid resource requests tend to boom in the near future due to a scheduled LHC upgrade that aims to increase the experiment's luminosity by a factor of 10 over its current value, increasing the amount of data to process. In HEP scattering, luminosity ($\mathcal{L}$) is the ratio of events (ΔN) detected through a cross-section (σ) over a period of time (Δt), i.e.,

$$\mathcal{L} = \frac{\Delta N}{\Delta t}. \tag{2.1}$$

With the increasing of the luminosity, it is expected at least a proportional increasing of the rate of particle collisions, enlarging substantially the volume of data and experiments, and consequently, the amount of log data produced by the grid. A complex technological challenge is envisioned, namely, to keep the grid infrastructure working along the Run-3 and Run-4 stages of the High-Luminosity LHC project (HL-LHC) [3, 4].

The HEP Software Foundation (HSF) released a road-map document describing the actions needed to prepare the grid to support the HL-LHC upgrade [5]. As a result, the Operational Intelligence group (OpInt) was created as a task force to improve the WLCG quality of service (QoS). Through data analytics and log data mining, its main research line concerns the development and maturation of machine learning (ML) tools based on event-oriented maintenance systems. Many ad-hoc solutions have been promoted by the OpInt group, from log parsing to diagnostic systems, as real-time anomaly detection approaches developed to assist the computing center of the Italian Institute of Nuclear Physics (INFN-CNAF) [6]. The usage of ML algorithms tends to reduce system downtime and optimise the usage of resources.

The present thesis was funded by INFN-Bologna as an initiative to improve the WLCG QoS through the log data processing using alternative modelling of ML approaches. This chapter is organised as follows. Section 2.1 describes the thesis' motivation and background scenario. Section 2.2 shows the state of the art of the different types of system maintenance. Section 2.3 introduces two possible metaphor models to approach the system maintenance problem: (i) health care, and (ii) speech translation. Section 2.4 explains how to apply the health care model to system maintenance problem. Section 2.5 summarises the chapter.

2.1 Grid Infrastructure

WLCG is a worldwide grid infrastructure involving more than 170 computing centers over 42 countries organized in tiers according to their importance in the grid, being linked by the high-speedy Italian (GARR) and European (GÉANT) research networks with more than 200 Gbps. The main data center of WLCG is the Tier 0 at CERN. During Run 2, the data produced by the LHC experiments exceeded 6 gigabytes per second, being necessary to process these data, through Monte Carlo simulations, to generate the physics analysis. In total, the LHC experiments rely on more than 400 PB of disk storage, 700 PB of tape storage, almost 1 million CPU cores, and both dedicated and shared network infrastructures.

Providing a common middle-ware for HEP experiments, the infrastructure allows data and resource sharing, providing applications that run on cloud facilities. Such infrastructure is distributed across hundreds of computing centers worldwide under different administrative domains. WLCG has off-the-shelf resources with commodity hardware (a typical university computing cluster), and a growing number of non-standard resources, such as HPCs [1], opportunistic/volunteer resources, as well as commercial and scientific clouds.

To keep the WLCG infrastructure operational, various methodologies, techniques, and methods could be implemented at the data centers. Massive fault tolerant strategies can be applied to the grid in order to minimise the failure perception to the final user. In the other hand, some of these technologies could be used to prevent, predict and prescript the anomaly occurrences at the distributed system using ML maintenance approaches.

The main Italian WLCG Data Centre is located at Bologna (Italy) at the INFN-CNAF [7], supplying resources and services through remote access. INFN-CNAF has approximately 40,000 CPU cores, 40 PB of disk storage, and 90 PB of tape storage. Its services generate a huge amount of logs that can be used in monitoring, diagnostic and prognostic purposes. The amount of produced log data varies in time, but as illustrative criteria, during the months of May/June 2019, about 420 GB were produced.

[1] High Performance Computers

The future expectations are the continuous increasing of the data flow because of the beginning of the new phase of the HL-LHC project [8].

All of this information could be used to optimise the CNAF QoS, modernising the system maintenance, providing a better usage of the resources, preventing faults and unscheduled downtime. Nonetheless, until 2017 at this data center, all the fault-correction service was done manually by technical staff without a significant computing help. Because of the practical appeal, there are some ongoing projects that aim to characterise the complexity of the diagnostic data and to establish connections among them.

Because of that, the system maintenance problem applied to computing centres is a CERN hot topic. Many efforts are being done at WLCG Tier-1 Bologna in order to create predictive maintenance tools using the log data. A first work based on the Elastic Stack Suite catalogues the log records and anomalies using an embedded unsupervised ML tool [9]. Another initiative uses supervised ML approaches to predict anomalies of system behavior in an ad-hoc solution [10]. Another work, also focused on a content-processing strategy, provides a clustering method used to characterize log records using Levenshtein distance [11]. In particular, it was created a prototype to identify anomalous system behaviour, using a binary classification, considering the log data generation rate and an One-class SVM approach [12, 13].

Autonomous system maintenance is a promising approach for this scenario that demands real-time responses to keep an uninterruptible availability. Currently, there is no automatic log analysis at INFN-CNAF, considering a prospective scenario of data-production increasing. In these conditions, it is mandatory the development of smart tools to support the data center operability.

2.2 State of the Art

Emerging technologies contribute to the massive growing of data throughput on networks [14], boosting the usage of monitoring systems based on Big Data analysis in order to explain hidden patterns of the running processes [15, 16]. In general, the intention is to identify fault [17, 18] or threat [19, 20] occurrences through batch or real-time anomaly detection. Data centers are rich environments of linked technologies organized in a complex mosaic, in which the maintenance problems must to be approached in a holistic way. In this scenario, it is important to concentrate endeavours on log data analysis, in order to keep the availability, accessibility, and reliability levels in line with the QoS agreements [15, 18, 21].

Overall, one of the main data sources to construct the system status is the log analysis [22]. Logs are generated by multiple services running through the data center. The composition of multiple logs in order to identify an event is a high-dimensional anomaly detection problem. Due to exponential searching space, data snooping-bias, and irrelevant features, it is a challenging topic. However, it is important to note that (i) logging is still challenging not only in academy but also in industry, (ii) machine learning is an encouraging strategy that can provide a contextual analysis of code execution to recommendation systems; tool usability is an open-problem in practice, and (iii) to storage log data efficiently is still a problem [23].

Figure 2.1 shows the temporal relationship of different types of maintenance approaches that can be classified according to the used intervention strategy when an unexpected event occurs [24]. Reactive or Corrective maintenance is an answer to failure occurrences in order to recover the system stableness, working as a palliative or curative solution generated through a diagnostic methodology. In the other hand, preventive maintenance are procedures executed regularly to decrease the probability of a system failure, as predetermined Maintenance, or condition-based maintenance, in which the intervention is done as soon as an anomaly is detected. Finally, predictive maintenance involves failure forecasting methods using dataset of system metrics.

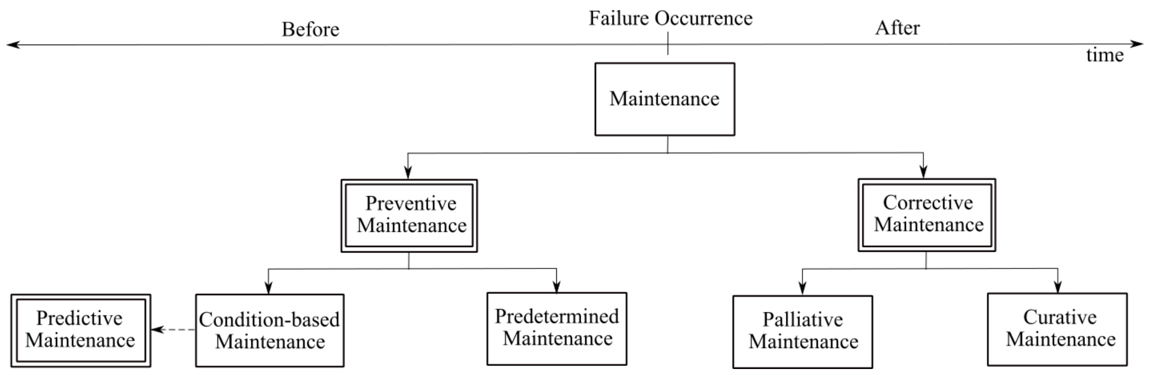


Figure 2.1: Maintenance System (MS) classification according with the MS intervention strategy related to the moment of the failure occurrence.

According with the previous classification, the system maintenance problem can be split in three main steps: (i) monitoring, (ii) diagnosis, and (iii) prediction [6]. Monitoring (i) log data can approach the apperception of the current system state, through symptoms and status changing (ii), avoiding, or forecasting (iii) system failures. This thesis concentrates on the monitoring (i) and diagnosis phases (ii), being classified as reactive maintenance solutions. The present research provides the necessary condition to the direct application of supervised prediction (iii) solutions to maintenance systems.

2.2.1 Reactive Maintenance

Reactive maintenance is based on a response to an event occurrence in a software detected by a monitoring system. After an anomaly to be individualised, a diagnostic task is triggered to identify its causes. Anomalous occurrences in large-scale systems can impact thousands or even millions of users, being crucial to be identified by real-time applications to keep the infrastructure operability.

As logs are omnipresent in computing systems, it is an important source of functioning details of running systems. Monitoring system based on anomaly detection can optimise the diagnostic process, selecting the most promising piece of logs to be processed. Monitoring systems can be both task-dedicated focusing to find a specific type of anomaly, and general-purpose, looking for generic functioning outliers. In both cases, anomaly detection problem[2] can be used as monitoring strategy as well as part of the diagnostic phase.

Monitoring System

Monitoring systems provide a constant surveillance of running software on a infrastructure, including services and applications. Modern software development and execution monitor the system behaviour in production [23]. Considering cloud computing and large-scale distributed systems, the modelling of a monitoring systems must to have scaling potential to be automatised [25]. Another important consideration is the exponential searching-space problem. To overcome the issue, a research combines an unsupervised Deep Belief Networks approach with an anomaly detection technique based on One-Class SVM [26].

Anomaly detection can be explored by classical ML approaches. In particular, unsupervised techniques can be applied, including methods as K-mean, and Expectation-Maximisation Clustering; passing through supervised ones as Classification-Tree, Fuzzy Logic, Neural Networks, and Support Vector Machine (SVM); statistical methods as Bayes Networks; and finally, hybrid possibilities as Cascading Supervised techniques, and combining of unsupervised and supervised approaches [19].

One important variation of this problem is the real-time anomaly detection (RTAD). An approach treats the problem considering streaming data with an online sequence-memory algorithm called Hierarchical Temporal Memory [27]. In another research, a log-based network monitoring system is designed to oversee the network security, gathering and counting device log to detect intruders using the traffic on the infrastructure [28]. Another project reviews RTAD also applied to network security, concluding that it is an open problem since the current approaches are not efficient enough [20].

Monitoring large-scale applications is a complex task since the nature of execution data is syntactic and unstructured. An approach formalises logs using formal semantics to combine their meaning. In order to automate monitoring and management tasks in the mentioned solution, a social-network analysis treats missing and incomplete data, being tested in a large-scale industrial use-case applications [29].

Another large-scale proposal, MoniLog, is a real-time distributed approach that detects sequential anomalies using a multi-source log stream. For this approach, it is necessary that an expert labels and evaluates anomalies according with their criticality [25]. Considering an information-security case study with 10,000 users with a high event rate, a research implements a Big Data ecosystem, in which the most important activity is the data cleaning phase using a comparative technique based on Fellegi-Sunter theory[3]. This task removes useless information and decrease the storage needs, concluding that some data could be safely ignored [30].

[2]More related works about anomaly detection in Chapter 4.

[3]The Fellegi and Sunter method is a probabilistic approach to solve record linkage problem based on decision model.

Diagnostic Systems

Diagnostic systems infer the possible causes of an event occurrence, being a critical component as information source to decision-making recommendation systems that work as prescriptive maintenance. Diagnosis is essentially a semantic problem, not only difficult to model but also expensive to process. In general, diagnostic systems are ad-hoc implementation of engineering applications. In these cases, the possible states are limited, known, and lightly dependent of user behaviour. On the other hand, diagnosis is a harder problem in computing centers, since the scenario can evolve potentially hundreds or millions of users in an infinite and evolving set of possible states.

A work shows how multi-process domain data can be extracted, and semantically transformed into appropriate formats to support the discovery, monitoring and enhancement of real-time applications. The solution individualises patterns and behaviors of the computing systems through further semantic analysis of the learning models produced by means of the Semantic Learning Process Mining formalisation, generating an event ontology basis. The algorithm is technically described as Semantic-Fuzzy Miner [31].

Diagnosing electronic systems are uncertain and ambiguous if based on symptoms description. In this sense, a traditional expert systems are not effective in providing reliable analysis. A work proposes a fuzzy logic-based neural network (FLBN) as an implementation to a diagnostic system based on symptoms of electronic systems using a real call-log database. FLBN is able to perform fuzzy logic rules learned from samples considering simple systems with a performance similar of a human expert [32].

In a research, industrial case studies illustrate the use of techniques to data pre-processing as wavelets and principal component analysis, multivariate statistical analysis, and unsupervised machine learning approaches. Also it is used inductive learning for conceptual clustering, and knowledge discovery for automatic analysis and interpretation of operational processes in order to improve the performance quality of applications [33]. In other research, a log-based tool is proposed to approach the minimisation of the impact of network convergence events, introducing the Route Convergence Visualiser used to diagnose the event and quantify the amount of impacted users during a network incident [34].

Regarding to fault diagnosis systems on aircraft, the state-of-the-art combines an expert-based model that associates faults and symptoms through a Naive Bayes reasoner. For complex systems with high-coupled components, this kind of modelling are often incomplete and inaccurate. In order to maintain the original model structure, a paper combines information from adverse log events with real flight data. The solution is extended from a Naive Bayes learning to a Tree Augmented Naive Bayesian algorithm to capture the component dependencies to the diagnostic system. After that, the learning algorithm is trained using real flight data. The limitation of this approach, as well as the majority of diagnostic system, is the dependence of a previous system study by a human expert in an ad-hoc project [35].

Usually, log-based diagnosis to computing systems are based on log parsing approaches as basis for further analyses [4]. Computing systems need a much more complex log analysis because of the large number of high-coupled computing components and their interaction with a big number of users. Differently of well-defined engineering applications, computing systems deal with a large amount of software from different origins and purposes, converting the fault diagnosis in a hard syntactic and semantic problem.

2.2.2 Predictive Maintenance

Predictive maintenance can be applied in a huge variety of computing and engineering applications, forecasting events and cutting operational costs. It is essentially based on the recognition of the system state using machine learning methods to diagnosis and to predict trends in the system behaviour, involving decision-making tasks to autonomously preserve the system stability.

In monitoring systems, predictive maintenance is the forward step from anomaly detection. Once its discovery is clear, the next matter is to forecast its occurrence. A related paper applies predictive maintenance in a cloud manufacturing case, providing a set of machine tools from an exhaustive combination of ML algorithms, including Principal and Independent Components Analyses, and 2 types of features selections [36].

Still in a cloud manufacturing issue, a research analyses the quality of a real-time predictive management cloud-based system, distributed by a cloud-storage provider, using the time-to-failure metric applied on hard disk drives (HDD) [37]. In another work, also in a HDD monitoring in a cloud scenario, it is presented a real-time predictive maintenance system based on Apache Spark to identify a forthcoming HDD failures in data centers [22]. Still in the prevent equipment downtime, a work approaches a multiple-instance learning method by mining equipment logs [38].

Considering data-driven modelling, the degradation of the model performance problem is modelled as a data-imbalanced distribution. The skew data distribution problem can generate irregular patterns and trends, harming their recognition. A paper proposes a hybrid machine learning approach that blends natural language processing techniques and ensemble learning to predict extremely rare aircraft component failures. The solution is tested using a real log-based maintenance system dataset considering imbalanced data [39]. Another paper deals with the problem of forecast rare faults in the aircraft real-word datasets. A solution approaches the predicting of extremely rare faults combining two deep learning techniques: (i) Auto-encoder, and (ii) Bidirectional Gated Recurrent Unit network. In (i), the component is trained to detect rare failures, and its result is used to feed the module in (ii), predicting the next fault occurrence [40].

Predictive maintenance can be used to forecast equipment failures, scheduling appropriately the needed corrective maintenance, avoiding unexpected equipment downtime. A paper proposes a data-driven approach for estimating the likelihood of machine breakdown in a future time interval in manufacturing systems. The implementation uses real-world datasets including machine log messages, event logs, and operational information, applying data-mining, feature-extraction, and machine learning methods to indicates machine failures up to 168 hours in advance [41]. Another work approaches

[4]See for related works of log parsing in Chapter 5

the predictive maintenance applied to equipment failure problem through a data-driven method based on multiple-instance learning. The application also uses real datasets of event logs related with medical equipment [42].

2.3 System Maintenance Metaphors

The system maintenance problem can be mapped in well-defined and widely explored research areas since it is possible to connect them metaphorically. This strategy is broadly used in computer science approaches, i.e., natural computing and bio-inspired algorithms are motivated on the nature observations, for example. These insights generate successful models that are used as basis of well-established research areas, as neural and deep networks, evolutionary and genetic algorithms, bioinformatics, among others. Gossip algorithms are used on virus spread in epidemiology. Swarm intelligence algorithms can look for the best solution in the search space of optimisation problems. Collective intelligence is used to approach decision-making systems. All these applications have in common the fact that they are based on the solution of unrelated problems. The power of the metaphor is to create an interpretable link among new and old issues towards the AI explainability.

In this section, two perspectives are proposed as system maintenance inspirations: (i) health care, and (ii) language translation. In (i), a system is seen as a patient in which it is important to keep the quality of life, monitoring his health and diagnosing diseases. On the other hand, in (ii), the pair system/system maintenance is modelled a conversation between a patient and a therapist that speak different languages. In both cases, the professional that tries to help the patient can not communicate directly to him for some reason. In the first case, the patient is unable to express what he is feeling, maybe because he is a baby or a dog, for example. Because of this, the patient's behavior is monitored in order to infer his quality of life. In second case, the patient can not speak the same language of the therapist, being necessary to translate the communication before to interpret the signs. In both, there is a semantic gap to fulfil in order to improve the quality of life of the patient or, in our case, to keep the QoS of the system.

2.3.1 Health Care

Aiming to keep the QoS, system maintenance is a health care approach. In this metaphor, a system is a person, and its functioning expresses the quality of his life. Patient care protocols [43] can be used as inspiration to this system maintenance analogy, as:

- *Anamnesis*: is the process to remember the past, in order to collect and store the patient's medical history. Particularly, the *catamnesis* is the medical history of a patient from the onset of a disease. Both registrations help in the diagnosis phase.
- *Prophylaxis*: is the set of techniques used to prevent or protect against diseases or control their possible spread.
- *Diagnosis*: is the analytical process to identify the disease or condition by scientific evaluation. The *diagnostic process* evaluates the factors that influence the patient's status, using the previous

information collected by the *anamnesis* phase, and medical exams. Each disease/condition has associated a *diagnostic sensitivity* and *specificity*, i.e., the probabilities of the ill person be diagnosed with a given disease, and a healthy person be not, respectively.

- *Prognosis*: is the prediction of the probable outcome of a disease based on the patient's health condition and on the usual development of the illness. It evolves issues as duration, evolution, and conclusion of the patient's health condition, using the prognostic indicators as monitoring metrics.

Figure. 2.2 shows the connection of medical areas, exploring the isolated results collaboratively to promote the patient health. There are two concomitant processes: (i) predetermined prophylaxis, and (ii) health monitoring. In (i), it is explored issues as health and sexual education, healthy eating and regular physical exercises promotion, vaccination, body and mouth hygiene, among others. The health monitoring is detailed in (ii) through anamnesis, medical tests and propaedeutics[5]. If any exam results are anomalous or important events happen, the health professional begin the diagnosis and prognosis steps, using all available information provides by the medical records or eventual extra exams. When the health professional is able to diagnose the patient, he can decide the suitable treatment according with his beliefs.

In general, prognosis is strongly connected to the diagnosis, being an obligatory step before to choose the best possibility among the feasible treatments. The information is stored and updated whenever something happens in the *medical records*, being available in whole process. The monitoring step is based on information-collection activities, as *anamnesis*, *catamnesis*, *medical exams*, *propaedeutics*, and *appointments*. The evaluation moments, in which a data interpretation is needed, are represented by diamonds in the flowchart. The dashed rectangles, the *monitoring* and the *medical decision-making*, assemble arguments in broad concepts. The diagonal arrows at predetermined prophylaxis and health monitoring indicate the concomitant continuous-flow activities, and the other arrows indicate the next task of the activity. The solid-line rectangles show a set of techniques summarised in medical topics.

2.3.2 Speech Translation

Log data are the registration of inner system processes. As thoughts in the mind, logs are imprecise, indirect and codified. In this sense, the hole of the system maintenance is to interpret the events, extrapolating towards the causes, providing a comprehensive message that summarises what is happening. In this metaphor, the under-observed system is a patient undergoing therapeutic treatment with a therapist, the system maintenance, that can not speak his mother tongue.

The communication between patient and therapist is given by sign exegesis. The core of this approach can be subdivided in (i) the quality of the translation between the patient-therapist languages, and (ii) the mapping between how well the patient can express himself and how clever is the therapist to interpret signs and infer traumas. In (i), the patient expresses himself through a set of signs learned

[5]propaedeutics is a set of techniques used to generate an orientation basis to the diagnosis phase.

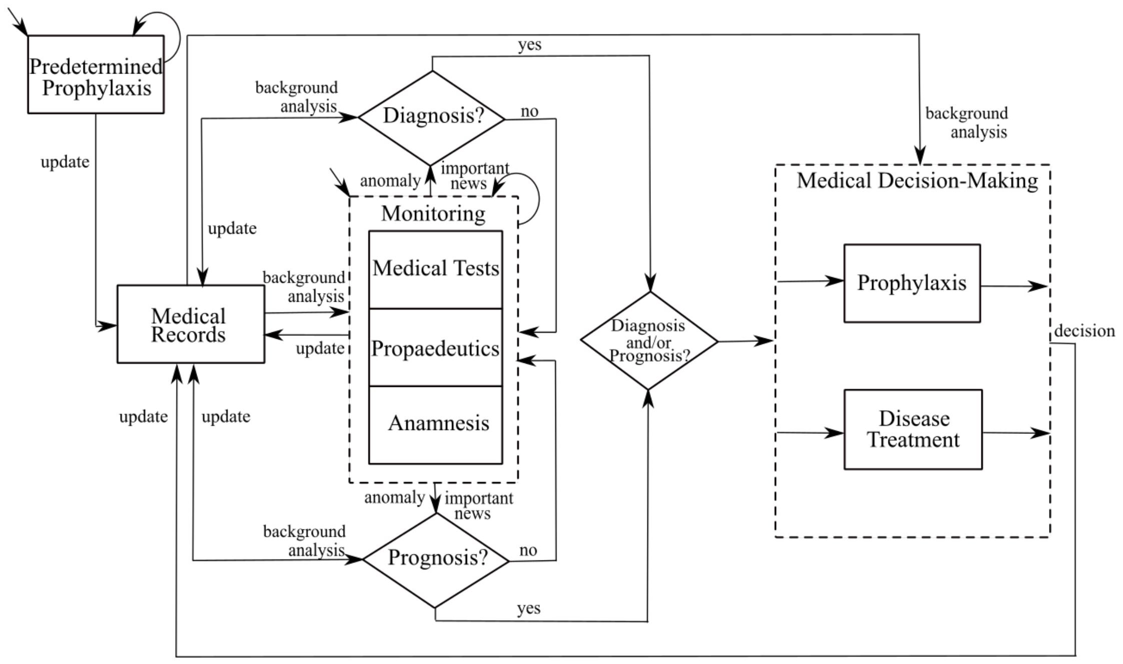


Figure 2.2: The flowchart represents the medical attendance with the main steps of the process.

through his life experience. Since the patient expresses his thoughts in a indirect speech style [6], the communication has not just one possible interpretation. The language is developed as a tool used to tell stories. The language complexity is related to the necessity to communicate the occurrence of events and the sophistication level of a language is related to necessity to tell elaborate stories. In a data center is not different: the services and systems must communicate the event occurrences, narrating the code execution story.

To achieve the root of trauma, it is necessary to proceed a second translation. In this turn, it is necessary to adjust the abstraction level between patient's thought and the therapist's comprehension ability. In the end, the conversation can be full of misunderstanding, as in every communication process. In (ii), the mapping between observable and cause events, based on inverse engineering, is the crux of the system maintenance. Each observable event is a patient behavior, i.e., an interpretation of the translation of the patient speech. Each patient behavior can be generated by one or more causes, i.e., traumas. In its turn, each trauma generates just one patient behavior, i.e., an observable event. Because of that, the observable event can be mapped to one or more causes, being not an unidirectional mapping of the patient's behavior and the trauma, because the observable behavior can be the result of a sum of behaviors.

Figure 2.3 shows a model of reasoning based on gossip, mapping event occurrences in linguistic expression. An event is defined as a set of logs. The dotted circles represent the capacity of the events or words change maintaining their meanings. The dashed-dotted rectangle are the occurrence and

[6]Since the therapist can not speak the patient's mother language.

language spaces. Event occurrences furnish a narrative to explain the system status. To comprehend this story is a matter of abstraction level changing, summarising extensive time intervals in words or phrases. Since the majority of languages are time-based, events can be distributed on the timeline in past-present-future variations as well as the system maintenance diversity.

The mapping between the occurrence and language spaces is the interpretation of happened events, increasing the abstraction level of the perceived information. The mapping defines the language, and the event occurrence specify the sophistication level of the language needed to communicate in a culture. Data centers provide the culture in which the events are inserted, and consequently, give the required degree of expression.

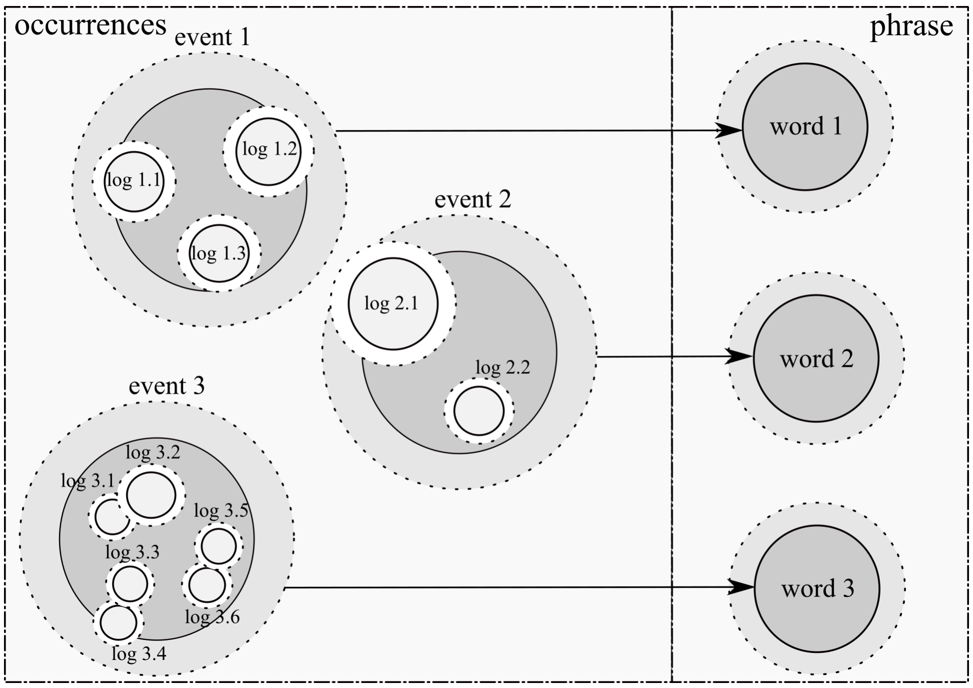


Figure 2.3: The mapping between the occurrence and language spaces is the interpretation of happened events, increasing the abstraction level of the perceived information.

2.4 Proposed Framework

In this work, the heath system metaphor is used to combine the maintenance approaches, being summarized in Fig. 2.4. The flowchart is divided in 3 concomitant processes: (i) prevention, (ii) identification, and (iii) intervention. In (i), the processes aim to avoid damages preventively during a predetermined time intervals as hardware replace and non-free and/or non-open source software update/upgrade. In (ii), it is presented the great part of the maintenance system, including the monitoring, the predictive maintenance, and the diagnostic system. Nowadays, the majority of the researches are concentrated in this module of the flowchart. The monitoring phase analyses the features extracted

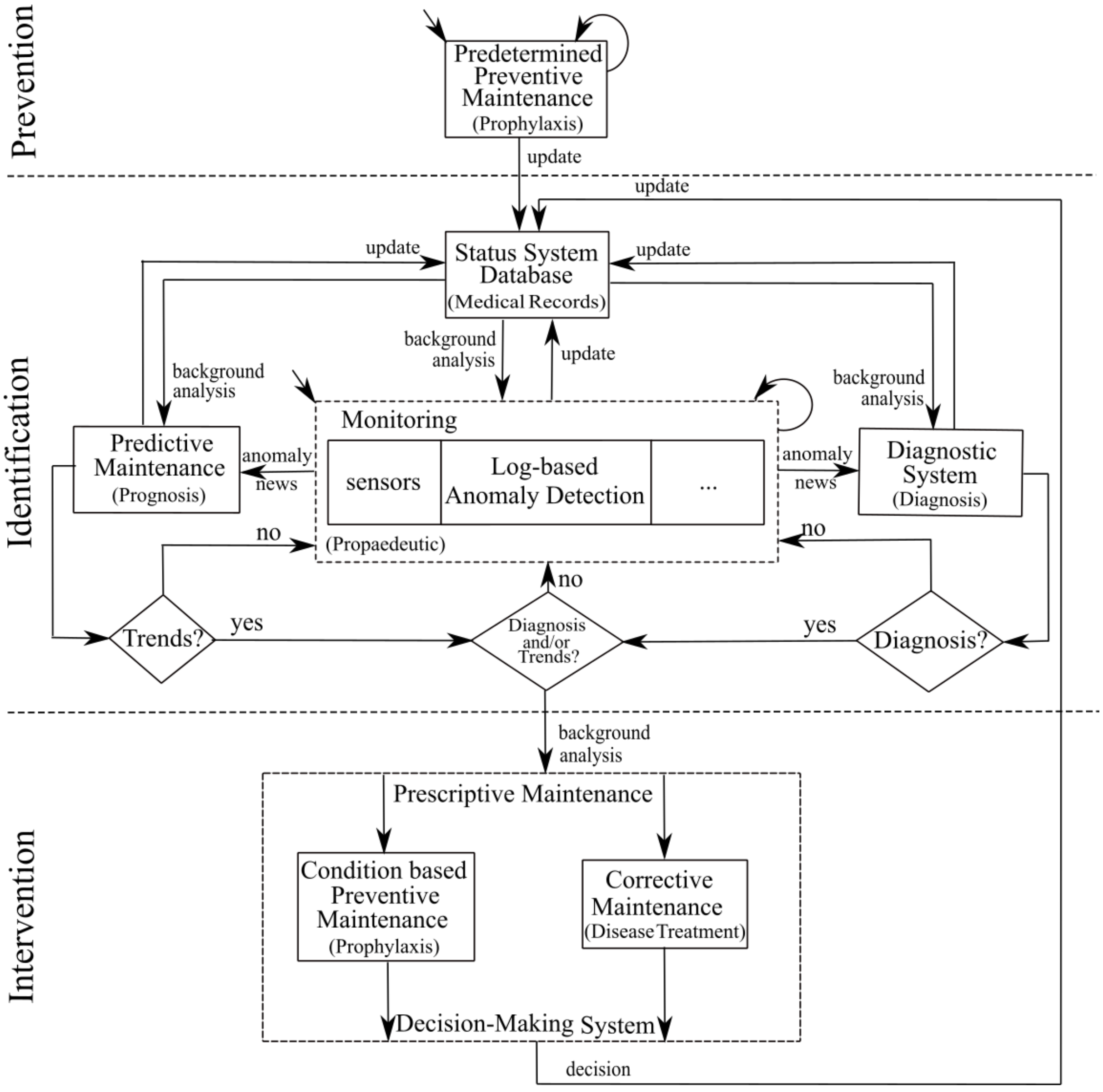


Figure 2.4: Maintenance framework based on an approach of Health Maintenance.

from the computing system, and uses this information to predict trends or anomalous event (Predictive Maintenance), and to diagnose the event occurrences (Diagnostic System). This phase produces the basis of the decision-making system in (iii), applying the appropriate prescriptive maintenance approach, correcting (Corrective Maintenance) or preventing (Condition-based Preventive Maintenance) damages.

A complete autonomous decision-making systems is an utopia currently. The state-of-the-art is far away from a maintenance system able to manager itself without any human interference to complex applications. A full-autonomous maintenance system is a hard problem leading with data transformations in different abstraction levels, uncertainties and inaccuracies in numerous processing steps. This work is mainly focused on approaching monitoring and reactive maintenance, the anomaly detection approaches in Chapter 4, and diagnostic system with the log parsing method in Chapter 5 (see Fig. 2.4).

The proposed modelling can be applied in a widely range of applications that can be mapped to the prevention-identification-intervention framework. The framework is a composition of monitoring and decision-making systems. Obviously, the first alternative application is health care systems with the framework directly applied. However uncorrelated problems can be approached by the framework as social context-based application, identifying event occurrences, and their impact on the society, for example, or in the industry, being applied in the production chain. Independently if the input data is textual, numerical, or pictorial, the framework identifies the needed steps to generate autonomous systems.

2.5 Summary

The chapter provides the background scenario of this thesis and its motivation, introducing a landscape of its topic - the system maintenance. It explains the state-of-the-art of the main types of variations of maintenance systems. It is formulated two different metaphors to approach the problem at a deeper level, organising the sub-problems in a framework.

Chapter 3

Evolving Fuzzy Granular Computing

This chapter introduces definitions and concepts of granular computing, including fundamentals of interval analysis and fuzzy sets from the granular computing perspective. Granular computing is mainly aligned with the fuzzy systems theory. Related concepts such as fuzzy sets, interval analysis, granular information, information granulation, aggregation function, evolving intelligence, and granular modelling are described and discussed. The main sources of information for this chapter are [44, 45, 46].

3.1 Granular Computing as Paradigm

Granular computing is an information processing paradigm that represents information through a set of entities, thus revealing multi-levels of data detailing to find useful abstractions to approach complex real-world problems. The meaning of the entity depends on the application. The entity defines a brick that forms the wall of a rule-based model. Its representation is not unique; its choice is essential to the quality of the solution and should conform to particular probability or possibility data distributions. The properties of a set of granules directly affect the accuracy of a model, viz., a classification or prediction model.

Evolving fuzzy granular solutions are online modelling approaches based on the human reasoning considering uncertain data to explain the trends that govern complex system. Dynamic and apparently chaotic environments provide information flows in which granular modelling can bring to light the mechanisms that modulate the system behaviour. Intrinsic details must be left behind in the name of a greater understanding of the whole problem.

Granular computing deals with big practical concerns as how to handle data and information from heterogeneous sources, how to obtain accurate and interpretable human-centered models from uncertain data, and how to interpret the results and the model. In particular, uncertainty is an information attribute since the human ability to deal with the reality is insufficient to capture all details. A granular computing method manages and learns from uncertain data streams. The resulting granular model should describe the essential aspects of the process or phenomenon that generates the data. A granular method aims to balance accuracy and model understandability by accepting a level of uncertainty regarding the elements that define the problem, such as the data and the model representation and parameters.

Currently, data-stream-driven modelling is usually a Big data problem. Many issues frequently arise on extracting information from large data sets through online methods. The veracity and validity of a wide variability of unstructured data is highly questionable. Related to the veracity and validity of the data, it is considered not just the data accuracy, i.e., how precise is the collected data, but also how valid and relevant the data are to a given application. Another important consideration is how long a data has to be considered as relevant to the current status of the system, i.e., how much volatile is the dataset. All these elements are extremely problem-driven, depending not only on the method used to collect the data and its respective data quality but even the approached problem class.

Aggregation functions are applied to the data to change the abstraction level of the dataset. The data aggregation works as magnifier, bringing the point of view closer and further away to answer distinct types of questions through the data. Different levels of data abstraction can intuit different insights about the system, thereat to choose the appropriate abstraction level is important to identify a good enough standpoint to the problem.

In a data-driven application, a higher output precision is often related to a higher meaningless result. In this sense, meaningful and precision are output features that can not be optimised in the same problem during the same time. It is the problem of the machine learning accuracy: obtaining a higher output accuracy is often associated to use black-box machine learning methods. You are able to predict (or classify) accurately the data, but not to explain the reasoning used to get there. In the other words, a precise answer is usually given by a higher-order non-linear model, and it is highly likely that you are not able to understand which terms are determinant to achieve that output. This is a big issue specially in applications coupled with human-centred decision-making systems.

The data granulation comes from the natural need to abstract and summarise the system information to guide the human comprehension and decision making processes. Following this strategy, fuzzy linguistic models can approach a data-driven problem that must to be linguistically understood. Extracting meaningful knowledge from non-ending data stream, the modelling dictates the way the humans deal with complexity, obscurity, and oblivion.

Commonly in real-world problems, data streams are non-stationary, non-linear, and heterogeneous, being subject to trend changing. In this scenario, a natural technique is to waive unnecessary details to assist the interpretability, transparency, applicability, and scalability of information systems. Specially, the behavior of non-ending and non-stationary data streams can be analysed by the evolving Granular Computing.

3.2 Fundamentals

Granular Computing is based on the conception of information granules generated by the granulation process, compounding granular worlds. The paradigm basis is found on mathematical concepts as fuzzy sets, intervals, and aggregation functions, being the main topics to the comprehension of this thesis.

3.2.1 Interval Analysis

Intervals are a branch of mathematics used as a problem-solving tool. The confidence interval and standard deviation are typical intervals used in scientific applications. In the granular computing context, intervals are granule instances that might appear in both formats, number or set; helping to interpret system analyses. The high interpretability of the granular models is an essential quality of applications that approach human-centered problems.

Intervals can be used as an expression of measure imprecision of a feature j, in which its real value is in the closed bounded set, $[\underline{l}, \overline{L}]$, in which $\underline{l}$ and $\overline{L}$ are the endpoints with $\underline{l} < \overline{L}$, and $\underline{l}, \overline{L} \in \mathbb{R}$.

Suppose a system state s represented by a data stream formed by a tuple of features, $s = (x_1, \ldots, x_j, \ldots, x_n)$. Such system representation is in a n-dimensional Cartesian product space, in which its dimension j is taken by the hyper-rectangle projection of the feature j, generating the j-axis. The s tuple is associated to an interval vector $I = \{I^1, \ldots, I^j, \ldots, I^n\}$, in which all I^j are fuzzy intervals. Each $x_j \in \mathbb{R}$ belongs to the respective I^j interval with α_j degree of membership with $\alpha_j > \alpha_{min}$.

Operations

Consider a point x as a degenerated interval I. Interval vectors are n-dimensional hyper-rectangles $I = \{I_1, \ldots, I_j, \ldots, I_n\}$, in which each interval dimension, I_j, is a special case of interval vector I with $j = 1$. Systems, defined by a feature set, can be modelled as a tuple composed by a n-dimensional interval vector, I, in which every $I_j = [\underline{l_j}, \overline{L_j}]$, and

$$[\underline{l_j}, \overline{L_j}] = \{x : \underline{l_j} \leq x \leq \overline{L_j}\}. \tag{3.1}$$

Operations from Set Theory, as intersection, $\cap$, and union, $\cup$, can be applied at intervals. Considers the intervals $I^1 = [\underline{l^1}, \overline{L^1}]$, and $I^2 = [\underline{l^2}, \overline{L^2}]$. If $\overline{L^1} < \underline{l^2}$ or $\underline{l^1} > \overline{L^2}$, then $I^1 \cap I^2 = \emptyset$, otherwise the result of $I^1 \cap I^2$ is the interval given by

$$I^1 \cap I^2 = [max(\underline{l^1}, \underline{l^2}), min(\overline{L^1}, \overline{L^2})]. \tag{3.2}$$

Now, considering I^1 and I^2 as interval vectors, their intersection is empty just if all intersection operations, taking the respective intervals two by two, are empty. Likewise, union of nonempty and non-disconnected intervals is

$$I^1 \cup I^2 = [min(\underline{l^1}, \underline{l^2}), max(\overline{L^1}, \overline{L^2})]. \tag{3.3}$$

The convex hull of two interval vectors, $ch(I^1, I^2)$, is the smallest interval vector

$$ch(I^1, I^2) = (ch(I_1^1, I_1^2), \ldots, ch(I_j^1, I_j^2), \ldots, ch(I_n^1, I_n^2)), \tag{3.4}$$

containing all elements of both and among them, considering also disconnected intervals, in which each ch_j, associated with the feature j, is given by

$$ch_j(I_j^1, I_j^2) = [min(\underline{l_j^1}, \underline{l_j^2}), max(\overline{L_j^1}, \overline{L_j^2})]. \tag{3.5}$$

This operation is used to connect independent sets, or disconnected intervals. The width of an interval vector I, $wdt(I)$, is defined as

$$wdt(I) = max(wdt(I_1), \ldots, wdt(I_j), \ldots, wdt(I_n)), \tag{3.6}$$

in which

$$wdt_j(I_j) = |\underline{l_j} - \overline{L_j}|. \tag{3.7}$$

In addiction, the midpoint of an interval vector I_j of I is defined as

$$mp_j(I_j) = \frac{\underline{l_j} + \overline{L_j}}{2}, \tag{3.8}$$

composing $mp(I) = (mp(I_1), \ldots, mp(I_j), \ldots, mp(I_n))$. The operations $ch(I^1, I^2)$, $wdt(I)$, and $mp(I)$ are explicitly applied in the fuzzy granular classifiers and predictors in this thesis.

Interval vectors, I^1 and I^2, can be manipulated by the operation $\star$ taken their elements two by two, $I_j^1 \star I_j^2$, in which $\star = \{+, -\}$, with

$$I^3 = I^1 \star I^2 = \{x^1 \star x^2 : x^1 \in I^1; x^2 \in I^2\}, \tag{3.9}$$

being I^3 the resulting interval vector. The product operation of the two independent interval vectors[1], I^1 and I^2, is defined as

$$I^3 = I^1 * I^2 = \{x^1 * x^2 : x^1 \in I^1; x^2 \in I^2\}. \tag{3.10}$$

The reciprocal operation is defined as

$$\frac{1}{I} = \left\{ \frac{1}{x} : x \in I \right\}, \tag{3.11}$$

if $0 \notin I$. So,

$$\frac{1}{I} = \left[\frac{1}{\overline{L}}, \frac{1}{\underline{l}} \right]. \tag{3.12}$$

The division of two independent interval vectors, $I^3 = \frac{I^1}{I^2}$, is defined as

$$\frac{I^1}{I^2} = I^1 * \frac{1}{I^2} = \left\{ \frac{x^1}{x^2} : x^1 \in I^1, x^2 \in I^2 \right\}. \tag{3.13}$$

$I^3 = [\underline{l^3}, \overline{L^3}]$ is given by

$$[\underline{l^3}, \overline{L^3}] = [min(\underline{l^1} \bullet \underline{l^2}, \underline{l^1} \bullet \overline{L^2}, \overline{L^1} \bullet \underline{l^2}, \overline{L^1} \bullet \overline{L^2}), max(\underline{l^1} \bullet \underline{l^2}, \underline{l^1} \bullet \overline{L^2}, \overline{L^1} \bullet \underline{l^2}, \overline{L^1} \bullet \overline{L^2})], \tag{3.14}$$

with $\bullet = \{+, -, *, /\}$.

Another important operation, that can be applied in the granular approaches, is the distance between the I^1 and I^2 interval vectors. The distance $d(I^1_j, I^2_j)$ between each correspondent element I^1_j and I^2_j is

$$d(I^1_j, I^2_j) = max(|\underline{l^1_j} - \underline{l^2_j}|, |\overline{L^1_j} - \overline{L^2_j}|). \tag{3.15}$$

In many applications, it is important to have a single number associated to the distance between intervals. Because of this, it is possible to consider distance measurements based in norms as the Euclidean or p-norm, for example. In this work, it is used a Maximum norm,

$$D(I^1, I^2) = max(d(I^1_1, I^2_1), \ldots, d(I^1_j, I^2_j), \ldots, d(I^1_n, I^2_n)), \tag{3.16}$$

[1]Two sets A and B are said independent if their intersection $A \cap B = \emptyset$. Extending the concept to interval analysis, two intervals I^1 and I^2 are independent if $I^1 \cap I^2 = 0$, and two interval vectors are independent if for all $I^1_j \cap I^2_j = 0$. In other words, I^1 and I^2 are independent if they are disconnected.

to summarized the $d(I_j^1, I_j^2)$ distances. All operations, except $wdt(I)$ and $D(I^1, I^2)$, can be expanded from intervals to interval vectors as in Eq. (3.4).

Interval Functions

A interval function $f(I)$ is an extension of $f(x)$, in which $x \in I$ given by

$$f(I) = \{f(x) : x \in I\}, \tag{3.17}$$

extendable to interval vector by

$$f(I_1, \ldots, I_j, \ldots, I_n) = \{f(x_1, \ldots, x_j, \ldots, x_n) : x_j \in I_j\}. \tag{3.18}$$

The f, defined in the $\mathbb{D}$ domain, can be segmented in "pieces" monotonically increasing or decreasing, given by a sequence of I_j intervals. Since usually $f(I_j)$ is not defined by a box or even by closed format, $f(I_j)$ can be approximated by an inclusion function $F(I_j)$ defined by a box enclosing the range of a continuous f, in which $f(I_j) \subseteq F(I_j)$.

The optimal $F^*(I_j)$ is when $F(I_j)$ is the interval hull of $f(I_j)$, i.e., the smallest box that contains $f(I_j)$. $F^*(I_j)$ is used to limited the search space in which f is defined considering the interval vector I_j. To define a F^* for all $\mathbb{D}$, it is needed a set of F_j^* defined in I_j that delimits the search space of f into a sequence of hyper-rectangles in $\mathbb{D}$.

3.2.2 Fuzzy Sets

In general, complex systems are defined by a non-linear relationship among a high number of variables. All these variables are associated to a given level of imprecision and uncertainty. Imprecision is related to the physical limitation of measurement of a variable value or the limitation of its numerical representation. On the other hand, uncertainty is an attribute of information since information is statistical in nature [47]. To provide more formalism to the variable treatment as error analysis, the Fuzzy Set Theory provide the needed mathematical formalism to the uncertainty model.

Related to the numerical precision in computing systems, the numerical handling is limited by the binary representation. In this sense, each number is an approximation of the real value, in which, sometimes, matches exactly it. Considering numerical intervals as an infinite set of real numbers, its approximation is an overlapping representation of the possible numbers in the range.

Introduced by Zadeh [48], fuzzy set is an approximation set taken as the extension of the classical concept of sets, in which its elements have a degree of membership. Using the interval terminology, fuzzy arithmetic is based on the extension of the arithmetic applied to fuzzy sets considering restrictively fuzzy sets as intervals. The type-1[2] fuzzy set combined with fuzzy intervals analysis help to deal with the uncertainty modelling.

Consider $I = [\underline{l}, \overline{L}]$ and the function $f(x)$ defined in $\mathcal{R}$

[2]The type-1 fuzzy set, differently of the type-2 fuzzy set, has an invariable membership function.

$$f(x) = \begin{cases} 1 : x \in I; \\ 0 : x \notin I. \end{cases} \tag{3.19}$$

The $f(x)$ function can degenerate to a singleton function $s(x)$ if $\underline{l} = \overline{L}$. Since intervals enclose a set of real numbers, its degenerated representation, i.e. $[b, b]$, contains a unique value with height 1, that is, a singleton. In this scenario, interval analysis can be considered as a subset of the fuzzy set theory.

The main difference between intervals and fuzzy sets are the possibility of elements of fuzzy sets have a partial membership, providing a smooth border of membership from the complete belongingness to a full exclusion, as

$$f(x) = \begin{cases} \phi_A : x \in [l, \lambda] \\ 1 : x \in [\lambda, \Lambda] \\ \iota_A : x \in [\Lambda, L] \\ 0 : otherwise \end{cases} . \tag{3.20}$$

This modelling merges the concepts of membership functions and intervals, permitting infinite solutions, considering the ϕ_A and ι_A choices. Among the possibilities, the trapezoidal family-like membership functions are a popular choice in fuzzy applications.

Fuzzy sets catch the essential information, generating granules to describe and identify phenomenons using basic concepts as linguistic variables, fuzzy rules, and fuzzy rules basis. The partial membership structure permits absorbing new information to an existent domain and discovery new domains through the data.

Fuzzy set A are defined by their $A : X \to [0, 1]$ membership functions. In this thesis, it is considered a trapezoidal (l, λ, Λ, L) and Gaussian (μ, σ) functions, Neural networks, and fuzzy intervals as strategies to identify the partial membership of new samples.

In the context of this thesis, it is important to introduce some concepts. A normal fuzzy set has at least one element x of the universe X with a membership function equal to 1, i.e.,

$$sup_{x \in X} A(x) = 1. \tag{3.21}$$

Other two important functions are support $supp(A)$, and core $core(A)$ of a membership function, and α-cut of a fuzzy set A, A_α, in which,

- $supp(A)$ is the set of elements of X with nonzero membership degrees in A,
- $core(A)$ is the set of elements of X with membership degrees equal to 1, and
- A_α is the set of the elements of X, in which the membership degree $A(x)$ is greater than the value α, i.e., $A_\alpha = \{x \in X | A(x) > \alpha\}$.

When $\alpha = 0$ (Support, until $\alpha \approx 0$ but $\alpha \neq 0$), and $\alpha = 1$ (core) are boundary cases of α-level sets. Another important concept for the application is the convex applied to fuzzy set. Consider $x^1, x^2 \neq 0$, in which $x^1, x^2 \in X$:

$$A(\kappa x^1 + (1-\kappa)x^2) \geq min(A(x^1), A(x^2)), \tag{3.22}$$

in which $\kappa = [0, 1]$. Concepts introduced to intervals, as $mp(A)$, $wdt(A)$, $ch(A^1, A^2)$, $A^1 \cup A^2$, and $A^1 \cap A^2$, are extended to a membership function A, i.e., trapezoidal function (FBeM), and applied to neural networks (eGNN). In eGFC, the Gaussian membership function is not explicitly used, but $mp(A)$ and $wdt(A)$ can be extended as μ and σ, respectively[3]. Additionally, considering the fuzzy sets A^1 and A^2, A^1 is a subset of A^2, i.e., $A^1(x) \leq A^2(x)$, if and only if, each element $x \in A^1$ is also an element of A^2.

Fuzzy Interval

Fuzzy intervals and fuzzy numbers are classified as fuzzy granular data defined as potentially inaccurate, hardly well-defined and quantified, in which it is often needed a pre-processing step that introduce more uncertainty in the model.

Consider an upper semi-continuous membership function $A : X \rightarrow [0, 1]$, in which $x \in X$ with $A(x) > \alpha$, i.e.; the α-cuts of A are closed intervals. Let its universe X be the set of real numbers and A satisfies the normality condition, $\exists x \in X$ in which $A(x) = 1$. The fuzzy set A is a fuzzy interval I if it is defined by

- a monotone increasing membership function ϕ_A,
- at least an element with a membership value equal to 1,
- a monotone decreasing membership function ι_A, and
- zero otherwise;

as described in the function 3.20. I satisfy the normality and convexity conditions as well as A. If there is just a single $x \in X$ with $A(x) = 1$, I degenerates to a fuzzy number[4], represented by a triangular or Gaussian membership functions, for example. Considering the definition, a trapezoidal family-like functions fit also with fuzzy interval.

3.2.3 Aggregation Functions

Consider aggregation operators defined as $\mathbb{C} : [0, 1]^n \rightarrow [0, 1]$, $n > 1$ in which input values, given by an unit hyper-cube $[0, 1]^n$, are combined into an output value in $[0, 1]$. The aggregation operators satisfy two conditions:

- monotonicity in all arguments, in which given $x^1 = (x^1_1, ..., x^1_n)$ and $x^2 = (x^2_1, ..., x^2_n)$, if $x^1_j \leq x^2_j$ $\forall j$ then $\mathcal{C}(x^1) \leq \mathcal{C}(x^2)$;

[3]For more details, see Chapter 4
[4]Fuzzy numbers are the extension of $\mathcal{R}$ considering multiple sources of uncertainty.

- boundary restrictions: $\mathcal{C}(0,0,...,0) = 0$ and $\mathcal{C}(1,1,...,1) = 1$.

The used aggregation operators $\mathbb{C}$ are T-norm, S-norm, Averaging and compensatory T-S aggregation operations.

T-Norm

T-norms (T) satisfies the boundary conditions $T(\alpha,\alpha,...,0) = 0$ and $T(\alpha,1,...,1) = \alpha$, $\alpha \in [0,1]$, and their neutral element is $e = 1$. T-norms are commutative, associative, and monotone operators on the unit hypercube, and can be presented as the minimum T-norm operator,

$$T_{min}(x) = \underset{j=1,...,n}{min}\, x_j. \tag{3.23}$$

It is the strongest T-norm because $\exists x \in [0,1]^n$, in which $T(x) \leq T_{min}(x)$, being idempotent, symmetric, and Lipschitz-continuous[5].

S-Norm

S-norm operators (S) have the boundary conditions $S(\alpha,\alpha,...,1) = 1$ and $S(\alpha,0,...,0) = \alpha$ on the unit hypercube, being as well commutative, associative, and monotone. Their neutral element is $e = 0$.

As T-norms, S-norms can be presented through many definitions. The maximum S-norm operator,

$$S_{max}(x) = \underset{j=1,...,n}{max}\, x_j, \tag{3.24}$$

shows the S-norms stronger than T-norms, in which $S_{max}(x)$ is the weakest S-norm, since $\exists x \in [0,1]^n$ in which $S(x) \geq S_{max}(x) \geq T_{min}(x) \geq T(x)$.

Averaging

If $\forall x \in [0,1]^n$, the aggregation operator C is bounded by $T_{min}(x) \leq C(x) \leq S_{max}(x)$, then it is averaging operator. Its main restriction is that the output cannot be lower or higher than any input value, being idempotent, strictly increasing, symmetric, homogeneous, and Lipschitz continuous. The C operator can be defined as the arithmetic mean,

$$M(x) = \frac{1}{n}\sum_{j=1}^{n} x_j. \tag{3.25}$$

Compensatory T-S

Compensatory T-S operators are a combination of T-norms and S-norms in order to balance their effects. T-S aggregation is uniform, not depending on parts of the underlying domain. It is composed

[5]It is a strong form of uniform continuity for functions, limiting how fast the it can change.

by the application of a weighted quasi-arithmetic mean to a T-norm and a S-norm outputs. The T-S operator, $L(x)$, can be defined as

$$L(x) = (1-v)T(x_1, \dots, x_n) + vS(x_1, \dots, x_n), \tag{3.26}$$

with $v \in [0,1]$. T-S operators satisfies $S(x) \geq L(x) \geq T(x), \forall x \in [0,1]^n$.

3.3 Granulation

The process of built information granules are called as information granulation. It can be applied in three domain, i.e., time, space, and feature. The space is the search space, and it is deeply correlated to the problem nature. In the system maintenance problem, the search space of interest is the state space, in which each state is time-correlated, being defined by a feature set extract from the system.

3.3.1 Domains

The human reasoning naturally works considering information granules. In the time domain, the humanity organized the time in different granular systems according with the environmental references as the granules day and year based on the sun rotation and translation respectively.

Nonetheless, as these two granules are not enough to organized all the human activities, other size of granules became necessaries, and with that, the derived multiples and parcels were emerging to make life easier. Weeks, seasons, decades, months, hours, minutes, seconds have been emerging to keep up with problem solving needs. Less precises definition also are used in everyday life as the time to finished some task, or to arrive some order or in some place, to take a meeting, and so on.

All these terminologies are evolved with the concept of time granulation. For us, the idea of the time imprecision is intuitive. Maybe it is not possible to precise the exact moment of a natural human birth, for example, but we know that it will be happen usually in some moment between the thirty-seventh and forty second week of the pregnancy, being most likely around the fortieth week. We are pretty sure that the human birth will happen in this time interval, although outliers may occur as a twenty-five-weeks baby.

Obviously, the idea of granulation can not be confused with modularization. In the problem description, it is possible to identify type of modules that compose the system. For example, in the case of predictive maintenance in computing centers, the respective log-based system is the computing center compounded by diverse modules as machines, peripherals, services, software, among others; all of them maintainable. Each module is a set of elements that relates to each other and to other modules that define the computing center. Each element has a set of features that represent its health status. In turn, each feature is granulated according with a linguistic meaning, for example, in our anomaly detection application, one of four levels of anomaly, from normal condition, passing through lighting anomalous, anomalous, and highly anomalous. A combination of the status of these features provides the state of the element that they describe. In turn, a combination of the states of the elements provides the module state, and for similarity, the state of the module set defines the state of the computing center. Since the system is previously modulated, each module provides a granulated piece that represents a local model of the state of computing center.

Modularization can provide the space granulation in a general sense. In the system maintenance, solutions look for the system status through its state space, that can be granulated in software, computing systems, or hardware elements. The state space is defined by features that are associated to one of the possible linguistic values, i.e., the feature granulation. Different combinations of elements status provide different abstraction levels of the system health.

3.3.2 Information Granules

The result of information granulation is named information granules, being a data summarization that clusters points according with a criterion as indistinguishability, similarity, functionality, or proximity [44].

The information granule is defined by the membership function and the learning algorithm. The membership functions already implemented to granular fuzzy algorithms are numerical interval (Interval-based evolving modelling, or IBeM), trapezoidal function (Interval-based evolving modelling, or FBeM), neural network (evolving Granular Neural Network, or eGNN), and Gaussian distribution (Evolving Gaussian Fuzzy Classifier, or eGFC). Usually by the membership function and the learning algorithm. The membership functions already implemented to granular fuzzy algorithms are numerical interval(Interval-based evolving modelling, or IBeM), trapezoidal function (Fuzzy Set-Based Evolving Modelling, or FBeM), neural network (evolving Granular Neural Network, or eGNN), and Gaussian function (Evolving Gaussian Fuzzy Classifier, or eGFC). Usually, learning methods includes addition, deletion, update, and merger of information granules considering the granularity ρ, the reference rate η, and quantity of remembered rounds h_r.

3.3.3 Evolving Granular Systems

The notion of evolving scenarios is based on online analysis, concept drift and shift in non-stationary systems. Evolving approaches are fed by data streams, processing each sample only once, absorbing their characteristics through a fuzzy rule set. Data streams define the system, providing the rules that govern its behaviour, modelled by probability distributions.

The most common type of distribution that govern real-world system is the non-stationary one, defined by parameters that change over time. Concept drift is given by this change in the distribution from which a model is learned and adapted to incorporate the new information [49].

The model decodes the system behaviour in order to generate a classified granular scenario or to predict trends. The classified scenario provides an screenshot image that enlighten the system status. To maintain the reliability of the model, evolving granular systems use a local model pool, i.e., the just-created rules may to be reinforced before being absorbed by the rule set as local model. During this pause, the local model matures its parameters to provide a more statistically relevant information granule to the model. The pool strategy is important approach to avoid the outlier incorporation, preserving the model continuity and integrity.

The new information granule is ready to be encompassed by the rule set when it is matured enough, i.e., if the granule absorbs at least a minimum pre-established number of samples being called stable [44]. Concept shift is the incorporation of new granules, being deeply related to concept drift. Concept shift changes class borders and function parameters in the classification and regression problems respectively. Concept shift permits a distribution learned by data-driven model to be applied to data drawn from another, i.e, a model learned in one region might be extrapolate to another region of the feature space [49].

3.4 Summary

In this chapter, it is presented the mathematical fundamentals of this thesis, i.e., interval analysis, fuzzy set, and the arithmetic used in the fuzzy-granular classifiers and predictors. It is described the most popular aggregation functions used to summarized the data stream into information granules. The mathematical basis, Granular Computing and system maintenance are linked to provide the needed concepts to structure the fuzzy-granular methods, not just related to a single application but also related to an overview solution of the proposed problem.

Chapter 4

Evolving Neural and Granular Systems

Maintenance systems are usually based on offline statistical analysis of log records in constant time intervals. Recently, online computational-intelligence-based structures, namely evolving fuzzy and neuro-fuzzy frameworks [50, 51, 52, 53, 54], combined with incremental machine-learning algorithms have been considered for on-demand anomaly detection, autonomous data classification, and predictive maintenance of an array of systems [55, 56, 57, 58].

The background of the present study, the Tier-1 Bologna data center [1], produces a huge amount of service-oriented unstructured log data [2]. In this context, logging activity[3] means the rate of lines written in a log file. The logging activity depicts the overall computing center behaviour in terms of service utilisation. The rate may be used to monitor and detect service behavior anomalies, assisting in, besides their diagnosis, predictive and preventive maintenance. Certainly, the quality of logs is important to generate relevant information about the system, but it is out of the scope of this work.

Being log data processing a highly time and resource-consuming application, an important issue concerns to identify the most promising pieces of log files using an online machine learning strategy. In this way, log fragments can be processed by priority, maximising the likelihood of finding useful information to the system maintenance.

Many attributes should be extracted using log timestamps, i.e., the moment of log record writing. These attributes establish a sequence of encoded information that summarises the system use in a time interval. In this work, the encoded information are the mean, maximum difference between two consecutive means, variance, and minimum and maximum number of lines writing per time window in a given log file. It is considered the hypothesis that the system activity is directly proportional to the logging activity. Moreover, it is proposed a control-chart-based approach to self-label the online data instances autonomously.

This chapter addresses the dynamic log-based anomaly detection problem in data center context as an unbalanced multi-class classification problem in which the classes refer to the severity of anomalies

[1]See Chapter 1 to know more about the data center focus of this work.
[2]See Section 6.1 to more details about log files.
[3]See Chapter 6 to see the logging activity definition.

and the usual system behaviour. A Fuzzy-Set-Based evolving Model (FBeM) [59], an evolving Granular Neural Network (eGNN) [60], and an evolving Gaussian Fuzzy Classifier(eGFC) [61, 62] are developed from a stream of data dynamically extracted from time windows. FBeM, eGNN, and eGFC are supported by incremental learning algorithms able to keep their parameters and structure updated to reflect the current environment. As time windows are classified according to the severity of their anomalies, system maintenance can focus on that specific log information – a small percentage of the whole. FBeM, eGNN and eGFC are compared in classification accuracy, model compactness, and processing time. The eGNN implementation, the best achieve performance among eFGCs, is deeply evaluated, varying the neuron types.

The remainder of this chapter is organised as follows. Section 4.1 shows the related works of anomaly detection. Section 4.2 describes the evolving fuzzy and neuro-fuzzy approaches to anomaly classification. Section 4.3 presents aggregation functions used to model the neuron on the neural network of eGNN. The chapter' summary is outlined in Section 4.4.

4.1 Log-based Anomaly Detection

Considering events as rare occurrences, their recognition is a high-dimensional anomaly detection problem based on imbalanced classes used to diagnose the system state. Because of the exponential cost of to traverse the search space to identify the system state, data snooping-bias, and feature extraction are challenging related topics. In order to map the search space, log data can be processed to extract evidences about the status changing, minimising the rising cost with extra hardware to monitor systems [16]. For traditional standalone systems, specialised personnel could manually analyses logs to find anomalies based on their domain expertise, however this approach tends to be catastrophic for distributed infrastructures or even for more complex systems.

Anomaly detection problem can be approached using the classical supervised and non-supervised methods [63]. Usually, computing centers have available a large amount of log data from hundreds of running sub-systems on the infrastructure. Since log data are typically a big data problem, they often are heterogeneous, unstructured, and textual, varying significantly their format and semantics from system to system. For this reason, to create a general-purpose solution using logs is extremely challenging to the system maintenance research. Because of their features, these data are in general neither classified nor pre-processed. In these cases, the first step is to apply an unsupervised method to generate comparison parameters to describe the scenario, converting logs in a well-behaved supervised data.

Regarding to supervised methods, the anomaly detection problem can be addressed using a logistic regression approach implemented to estimate the probability of the anomaly occurrence [64]. A similar work proposes a decision-tree to predict the state for each instance based on a state diagram [65]. In addition, a regression-based approach correlates log events and resource uses, identifying how anomalous behaviours impact on the resource consumption [66].

Another work approaches anomaly detection of log events considering unknown data delays with a Long Short Term Memory (LSTM) recurrent neural network [67]. Still using a LSTM methods, the anomaly detection framework, DeepLog, detects outliers in generic log files, using keywords and log metric values to filter the log inputs, detecting nonlinear and high-dimensional dependencies among events with a high probability [68]. A bidirectional LSTM network improves the efficiency of the model using an evolving training set that considers the change of source code and the processing of the messages not available during the training phase [69]. Furthermore, a LSTM-based Generative Adversarial Network framework mitigates the impact of class imbalance regarding to anomalous and regular events, improving the detection performance [70].

A Support Vector Machine (SVM) implementation recognises anomalous scenarios, classifying instances in a high-dimension space [71]. In other work [72], the authors compare rule-based SVM with a Nearest Neighbours ad-hoc classifiers. As expected, the authors show that the custom-made predictor outperforms the general-purpose solution by a wide margin. Nonetheless, the proposed solution requires a large and labelled training set, decreasing its applicability in the real-world scenarios. Generally, besides being unstructured, the available data are unlabelled and class imbalanced [73]. For these reasons, unsupervised methods are more suitable to provide a general-purpose solution to the log-based anomaly detection problem.

Related with unsupervised methods, an One-class Support Vector Machine (One-class SVM) classifier distinguishes anomalous time-windows, using logging activity and volatility extraction from log data [12, 13]. Another work approaches the anomaly detection problem using an unsupervised Deep Belief Networks combined to One-class SVM [74].

Clustering methods can classify log events based on distance calculus among the samples, without knowing any data property [75]. The K-Nearest Neighbours strategy can provide an effective outlier detection, using feature analysis to overcome the problems caused by different sample sizes in the training data [73]. Upgrading the previous results, MinHash Multi-vantage-point Tree improves the selection of the k-group of similar logs [76].

Unsupervised methods can also be used as self-learning methods, performing the classification of data samples that could be afterwards used as training set of supervised algorithms. For example, a k-Mean clustering implementation can separate anomalies from normal events, using input data labeled by a Gradient Tree-Boosting method [77].

Besides, Isolation Forest (IF) can be used to approach anomaly detection applications, isolating unlabelled anomalous samples [78]. Extending this concept, a novel log-based IF anomaly detection method identifies normal log scenarios, using auto-encoder networks for feature extraction [79].

The evolving Fuzzy Granular Computing (eFGC) approaches that efficiently address the log-based anomaly detection problem, overcoming common data restrictions as lack of structure and labels of imbalanced classes data-set [80]. The supervised eFGC implementations have been applied to 4-class anomaly detection problem associated to a self-learning solution. In this approach, it is used a control chart strategy, converting the unsupervised log data to supervised weak-labeled data [81, 82].

Control chart[4] is a widely known monitoring chart often used on engineering industry. This strategy is the basis of FBeM [83], eGNN [84], and eGFC [85] [5] classifiers applied to log-based anomaly detection

[4]See Chapter 6 about Control Chart and the self-learning methodology.

[5]These works are presented in this chapter.

problem in data centers. The state of the art using this paradigm is the eGNN classifier. All eFGC implementations are evolving, online, and fuzzy-set-based methods, differentiating themselves in how the fuzzy-rule set are created.

4.2 Evolving Granular Classifiers

A diversity of evolving classifiers have been developed based on typicality and eccentricity concepts [86], local strategies to smooth parameter changes [87], self-organization of fuzzy models [88], ensembles of models [89], scaffolding fuzzy type-2 models [90], double-boundary granulation [91], semi-supervision concepts [92], interval analysis [45], and uncertain data and multi-criteria optimization [51, 93].

In this chapter, it is compared FBeM, eGFC, and eGNN classifiers in an non-stationary scenario for 4-classes anomaly detection problem. In particular, the classifiers (i) evolve over time; (ii) are based on fuzzy rules, codified in a neural network or not; and (iii) granulate the data domain incrementally, adapting their knowledge basis to new scenarios. In other words, their parameters and structure are learned on the fly, according to the data stream. Prototype rules initiation is not needed, tending to increase their accuracy over time until the convergence by means of recursive machine learning mechanisms [60].

Data instances are classified using a set of fuzzy rules extracted from previous data. Each fuzzy rule is associated to a granule, and each class is represented by at least one granule, and consequently, at least a rule. Data granulation, granularity, and granular models come from the Granular Computing (GrC) theory. GrC takes advantage of the premise that precision is expensive and usually unnecessary to model complex systems [60]. GrC theory aims to express information in a higher level of abstraction. Granular learning, in this chapter, means to map data stream in granular models. Fuzzy granules can be added, updated, removed and combined along the learning steps. Therefore, FBeM, eGFC, and eGNN capture new information from a data stream, self-adapting the model to the current scenario.

In the classifiers, each granule i is modelled by the i-th fuzzy rule R^i and, in its turn, by membership functions A^i_j on the j-th attribute domain, being shape-evolving within a maximum extension region, E^i_j. A datum x^j, the j-th feature of the current element of the data stream, may have partial membership on A^i_j. The fuzzy rules associate granules to class labels as consequent, being defined as

$$R^i(x): \text{ if}(x_1 \text{ is } A^i_1) \text{ and} \ldots \text{and } (x_n \text{ is } A^i_n) \text{ then } (\bar{y} \text{ is } \hat{C}^i). \tag{4.1}$$

Time-varying rules R^i compose the model $R = \{R^1, \ldots, R^i, \ldots, R^c\}$ with $c = 1, 2, \ldots,$ holding within the limits of the granules. An input instance is denoted by $\mathbf{x} = (x_1, \ldots, x_j, \ldots, x_n)$[6]; and $\hat{C}$ is the estimated class given by the Control Chart methodology[7]. C^i is the class label of the i-th rule R^i, in which

[6] $n = 5$.

[7] The estimated class is in $\{0,\ 1,\ 2,\ 3\}$.

the rules R^i $\forall i$ form the rule basis. The number of rules, c, is variable, depending on the data stream. A time-indexed pair $(\mathbf{x}, \hat{C})^{[h]}$ is an entry of the data stream. The main difference among the methods is the strategy to deal with granules and rules, creating different decision boundaries. The methods differ by the definition of A_j^i membership function, being a trapezoidal function $A_j^i = (l_j^i, \lambda_j^i, \Lambda_j^i, L_j^i)$, a normalised Gaussian function $A_j^i = \mathcal{G}(\mu_j^i, (\sigma_j^i)^2)$, and a neural network model associated with a given aggregation function as neuron to FBeM, eGFC, and eGNN, respectively. The new data $\mathbf{x}$ activates a set of granules according to the similarity between the new information and the granule. The rule R^{i^*} of most activate granule absorbs the new information, keeping the model updated.

The used data is extensively described in the Chapter 6. In short, logs are processed to extract 5-features vector that summarises the system behavior over time, generating a data vector stream. As the classified log data are associated with an anomaly class and each class is described through at least a fuzzy rule, it is possible to identify the diversity of log events with the same anomaly level, and ideally, each granule can describe a specific event.

4.2.1 FBeM: Fuzzy Set-Based Evolving Modeling

FBeM is an online classifier based on fuzzy rules supported by an incremental learning algorithm. The method provides nonlinear class boundaries summarised in adaptive models [60]. The generated model is a time-varying amount of elements (granules) of granularity ρ, created by the intense data streams processing.

The trapezoidal membership function $A_j^i = (l_j^i, \lambda_j^i, \Lambda_j^i, L_j^i)$ compose the FBeM fuzzy rule as described in (4.1). The most active rule of $(\mathbf{x}, y)^{[h]}$ is given by

$$\alpha^* = S(\alpha^1, \ldots, \alpha^i, \ldots, \alpha^c) \tag{4.2}$$

with

$$\alpha^i = T(A_1^i, \ldots, A_j^i, \ldots, A_n^i), \tag{4.3}$$

in which S and T are the *min* and *max* operators, respectively. The width of A_j^i is

$$wdt(A_j^i) = L_j^i - l_j^i, \quad \text{then} \quad wdt(A_j^i) \leq \rho_j, \tag{4.4}$$

in which ρ_j is the j-th model granularity. Changing the initial ρ, it is produced different classifiers. In addition, ρ evolves iteratively over time through

$$\rho^{[h]} = \begin{cases} (1 + \frac{r}{h_r})\rho^{[h-hr]}, & r > \eta \\ (1 - \frac{\eta - r}{h_r})\rho^{[h-hr]}, & r \leqslant \eta \end{cases} \tag{4.5}$$

in which η is the growth rate; $r = c^{[h]} - c^{[h-h_r]}$, and $c^{[h]}$ is the amount of granules at the h-th time step; $h > h_r$. The model is able to remember the last h_r learning algorithm rounds. The fuzzy set A_j^i can be expanded within the region E_j^i defined by

$$E_j^i = [mp(A_j^i) - \frac{\rho_j}{2}, mp(A_j^i) + \frac{\rho_j}{2}], \tag{4.6}$$

in which

$$mp(A_j^i) = \frac{\lambda_j^i + \Lambda_j^i}{2} \tag{4.7}$$

is the midpoint of A_j^i. If at least one feature, x_j, of $\mathbf{x}$ does not belong to any granule including E_j^i $\forall i$, then a new rule is created. The new membership functions are

$$A_j^{c+1} = (l_j^{c+1}, \lambda_j^{c+1}, \Lambda_j^{c+1}, L_j^{c+1}) = (x_j, x_j, x_j, x_j). \tag{4.8}$$

Otherwise, if a new $\mathbf{x}$ in an E^i, then the i-th rule is updated following one of the options:

$$\begin{array}{ll} \text{if } x^{[h]} \in [mp(A_j^i) - \frac{\rho_j}{2}, l_j^i]) & \text{then } l_j^i(new) = x^{[h]} \\ \text{if } x^{[h]} \in [l_j^i, \lambda_j^i] & \text{then } \lambda_j^i(new) = x^{[h]} \\ \text{if } x^{[h]} \in [\lambda_j^i, mp(A_j^i)] & \text{then } \lambda_j^i(new) = x^{[h]} \\ \text{if } x^{[h]} \in [mp(A_j^i), \Lambda_j^i] & \text{then } \Lambda_j^i(new) = x^{[h]} \\ \text{if } x^{[h]} \in [\Lambda_j^i, L_j^i] & \text{then } \Lambda_j^i(new) = x^{[h]} \\ \text{if } x^{[h]} \in [L_j^i, mp(A_j^i) + \frac{\rho_j}{2}] & \text{then } L_j^i(new) = x^{[h]} \end{array} \tag{4.9}$$

For a complete description about the updating, removing, conflict resolution, and merging procedures, see [59]. The FBeM algorithm is shown below, adapted by the introduction of the weak label to instances by means of a control chart methodology, as described in Subsec. 6.2.2.

FBeM Learning: Fuzzy Set-Based Evolving Modeling

```
1:  set $\rho^{[0]}$, $h_r$, $\eta$;
2:  for $h = 1 \dots$
3:    read $\mathbf{x}^{[h]}$;
4:    use control chart to label $\mathbf{x}^{[h]}$ with $C^{[h]}$;
5:    provide estimated class $\hat{C}^{[h]}$;
6:    compute estimation error $\epsilon^{[h]} = C^{[h]} - \hat{C}^{[h]}$;
7:    if $\mathbf{x}^{[h]} \notin E^i \; \forall i$ OR $C^{[h]}$ is new then
8:      create new granule $\gamma^{c+1}$, with class $C^{[h]}$;
9:    else
10:     update the most active granule $\gamma^{i*}$ whose class is $C^{[h]}$;
11:   end if
12:   delete $\mathbf{x}^{[h]}$;
13:   if $h = \beta h_r$, $\beta = 1, \dots$ then
14:     merge similar granules if needed;
15:     update granularity $\rho$;
16:     delete inactive granules if needed;
17:   end if
18: end for
```

4.2.2 eGFC: Evolving Gaussian Fuzzy Classifier

The eGFC algorithm is a semi-supervised evolving classifier derived from an online GrC framework [59, 62]. Even if eGFC deals with partially labeled data, a fully-labeled data-set is employed to the classifier in this work, using the proposal self-learning method to give a weak tag to the data. The eGFC method is based on Gaussian membership functions to define the rules of fuzzy granules, labelling new samples through the data space.

Granules are spread in the data space to model the knowledge, being detailed in local information whenever necessary. Its global response is derived from the aggregation of local models. An iterative algorithm builds a rule base, updating local models according to trend changing. The method supports unlimited amounts of data, maintaining the scalability [50, 94].

Local models are generated if the newest datum is sufficiently different from the granular basis, not being possible to be absorbed by the current knowledge. In order to adjust the knowledge model to the news, the learning algorithm can expand, reduce, delete, and merge information granules, reviewing the rules according to inter-granular relations. The eGFC approach creates nonlinear, non-stationary, and fuzzy discrimination boundaries among classes [50, 59].

Formally, an input-output pair $(\mathbf{x}, y)$ is related by the function $y = f(\mathbf{x})$. It is necessary to find an approximation function $\hat{f}$ that estimates the value of y given $\mathbf{x}$. In a classification problem, y is a class label, in which $y \in \{C_1, \; \dots, \; C_m\}$, and $\hat{f}$ defines the class boundaries. In the never-ending data streams problems, the algorithm deals with the classification of $(\mathbf{x}, y)^{[h]}$ time-indexed data pairs.

Particularly, the present work attends to a never-ending non-stationarity application, being required evolving classifiers to identify time-varying relations $\hat{f}^{[h]}$ in the data stream.

Gaussian Functions and Rule Structure

The eGFC membership functions, A^i_j $\forall j$ and $i = 1, ..., c$, are Gaussian membership functions built from the data stream. It is interesting to note that this approach does not need to guess how many data partitions exist, being a great advantage of the method applicability [50].

Normalised Gaussian membership functions $A^i_j = \mathcal{G}(\mu^i_j, (\sigma^i_j)^2)$, have height 1, with the modal value μ^i_j and dispersion σ^i_j [8] [95]. The Gaussian function is appropriate because it is its ability of learning and changing, since the modal values and dispersions can accurately be modelled by a data stream. Considering that the data are priorly unknown, a Gaussian function gives infinite support through its domain. Moreover, Gaussian functions provides smooth surface to fuzzy granules, $\gamma^i = A^i_1 \times ... \times A^i_j \times ... \times A^i_n$, in the n-dimensional Cartesian space. These surface can be obtained by the cylindrical extension of uni-dimensional Gaussian, through the application of the minimum T-norm aggregation [94, 95].

Adding Rules

Initially, rules may not exist, being created and evolved as data are available. If none of the existing rules $\{R^1, ..., R^c\}$ are sufficiently activated by a sample, $\mathbf{x}^{[h]}$, a new granule γ^{c+1} and its respective rule R^{c+1} are created. The learning algorithm absorbs the new information of $\mathbf{x}^{[h]}$. The granularity $\rho^{[h]} \in [0, 1]$ is an adaptive threshold that defines if a new rule is required. If

$$T\left(A^i_1(x^{[h]}_1), ..., A^i_n(x^{[h]}_n)\right) \leq \rho^{[h]},\ \forall i,\ i = 1, ..., c, \tag{4.10}$$

then the eGFC structure must to be expanded, in which T is any triangular norm. Particularly, this work uses the minimum (Gödel) T-norm, but there are other possibilities. The model is structurally stable and unable to capture concept shifts if $\rho^{[h]}$ is equal to 0. In the other hand, eGFC creates a rule for each new sample if $\rho^{[h]}$ is equal to 1, being impractical to classification problems. Wherefore, it is important to balance structural and parametric adaptability for intermediate values of $\rho^{[h]}$, to achieve the appropriate stability-plasticity trade-off [96].

The $\rho^{[h]}$ value regulates the granules size and their growing. The initial settings determine the converged model, impacting on its accuracy and compactness, generating different granular representations of the same problem.

Initially, a new granule γ^{c+1} is given by the membership functions, A^{c+1}_j, $j = 1, ..., n$, in which

$$\mu^{c+1}_j = x^{[h]}_j, \tag{4.11}$$

[8]In Gaussian distribution, the mean, median and modal values coincide. The dispersion metric is the standard deviation.

and

$$\sigma_j^{c+1} = 1/2\pi. \tag{4.12}$$

Eq. (4.12) is called by Stiegler approach to standard Gaussian functions, or *maximum approach* [94]. The strategy is to start with bigger granules that tend to converge gradually with the dispersion constriction whenever new samples activate the same granule, appealing for a compact model structure.

The class C^{c+1} of the rule R^{c+1} is generally undefined, i.e., the $(c+1)$-th rule continues unlabelled until it is confirmed. When the label $y^{[h]}$ of $\mathbf{x}^{[h]}$ is available, then

$$C^{c+1} = y^{[h]}, \tag{4.13}$$

else the first labeled sample that activates the R^{c+1} rule according to (4.10), is used to define its class, C^{c+1}. If a labeled sample activates a rule differently tagged, then a new rule must be created to represent this information as a partially overlapped granule. Partially overlapped Gaussian granules tagged with different labels tend to have their dispersions slashed progressively by the parameter adaptation procedure.The modal values of the Gaussian granules may also drift to improve the decision boundary.

The algorithm design gives priority to granules with balanced dimensions. The eGFC method has a trade-off between the principle of the balanced information granularity [80], and the Gaussian functions flexibility, improving the granules structure through adaptation mechanisms.

Parameter Adaptation

To keep the representativeness of the granule structure, the eGFC model continuously reduces or expands the Gaussian functions $A_j^{i^*}$, $\forall j$, of the most active granule, γ^{i^*}. In addition, its center must move toward regions of relatively dense population whenever necessary, and the associated rule is tagged as soon as the data are available. Adaptation develops more specific local models in the sense of Yager [97], promoting the coverage of the newest data.

If a rule R^i is sufficiently activated by an unlabeled sample (or with the same label of the rule), $\mathbf{x}^{[h]}$, it is candidate to be updated, absorbing the new data according to

$$min\left(A_1^i(x_1^{[h]}), ..., A_n^i(x_n^{[h]})\right) > \rho^{[h]}. \tag{4.14}$$

Geometrically, $\mathbf{x}^{[h]}$ belongs to a region highly influenced by the granule γ^i. Even if more than two rules reach the $\rho^{[h]}$ level, just the most active rule R^{i^*} is chosen to be adapt for the pair $(\mathbf{x}, y)^{[h]}$, considering that $y^{[h]}$ must match to the same class of the chosen rule R^{i^*}. Otherwise, if the classes don't match, the second most active rule of the active rule set is chosen for adaptation, and so on. If there is no rule candidate to absorb $(\mathbf{x}, y)^{[h]}$, then a new one is created.

The learning algorithm updates the modal values and dispersions of the corresponding membership functions $A_j^{i^*}$, $j = 1, ..., n$ to include $\mathbf{x}^{[h]}$ in R^{i^*} through

$$\mu_j^{i^*}(new) = \frac{(\varpi^{i^*} - 1)\mu_j^{i^*}(old) + x_j^{[h]}}{\varpi^{i^*}}, \tag{4.15}$$

and

$$\sigma_j^{i^*}(new) \quad = \quad \left(\frac{(\varpi^{i^*} - 1)}{\varpi^{i^*}} \left(\sigma_j^{i^*}(old)\right)^2 + \frac{1}{\varpi^{i^*}} \left(x_j^{[h]} - \mu_j^{i^*}(old)\right)^2 \right)^{1/2}, \tag{4.16}$$

in which ϖ^{i^*} is the number of times the $i^* - th$ rule was chosen to be adapted. Notice that the equations (4.14)-(4.16) are recursive, not requiring data storage.

As σ^{i^*} defines a convex region of influence around μ^{i^*}, defining the amount of granules in a given region, i.e., a very large or small σ^{i^*} values may induce, respectively, a unique or too many information granules per class. The approach keeps $\sigma_j^{i^*}$ between $1/4\pi$ and $1/2\pi$ (Stiegler limit).

Adapting ρ-Level

The activation threshold, $\rho^{[h]} \in [0, 1]$, is time-varying [59, 52]. The threshold assumes values according to the overall average dispersion

$$\sigma_{avg}^{[h]} \quad = \quad \frac{1}{cn} \sum_{i=1}^{c} \sum_{j=1}^{n} \sigma_j^{i[h]}, \tag{4.17}$$

in which c and n are the number of rules and attributes, therefore

$$\rho(new) \quad = \quad \frac{\sigma_{avg}^{[h]}}{\sigma_{avg}^{[h-1]}} \rho(old). \tag{4.18}$$

Rules' activation levels for an input $\mathbf{x}^{[h]}$ are confronted to $\rho^{[h]}$ to select between parametric or structural adaptation of the model. Commonly, eGFC initiates the learning process from an empty rule base, building progressively the knowledge about the data properties. Practice suggests $\rho^{[0]} = 0.1$ as the initial setting value. Gradually the classifier structure and parameters achieve a level of maturity and stability, converging the threshold to a proper value after some time steps. Non-stationarity and new classes govern $\rho^{[h]}$ to values that better represent the demand of the current environment. A time-varying $\rho^{[h]}$ prevents guesses about how often the data stream changes.

Merging Similar Granules

Two granules may be combined to form a unique granule that inherits the essential information of the merged granules, if they are alike enough and have the same class label. The distance measure

$d(\gamma^{i_1}, \gamma^{i_2})$ between Gaussian objects analyses the inter-granular relations, being

$$d(\gamma^{i_1}, \gamma^{i_2}) \quad = \quad \frac{1}{n}\Bigg(\sum_{j=1}^{n} |\mu_j^{i_1} - \mu_j^{i_2}| + \sigma_j^{i_1} + \sigma_j^{i_2} - 2\sqrt{\sigma_j^{i_1}\sigma_j^{i_2}} \Bigg)$$

the distance between the granules γ^{i_1} and γ^{i_2}. This measure deal with Gaussian objects and the specificity of information, in which, in turn, inversely related to the Gaussian's dispersion [98]. The further different the dispersion values $\sigma_j^{i_1}$ and $\sigma_j^{i_2}$, the greater the distance between the underlying Gaussian objects.

The eGFC method may fuse the pair of granules with the smallest value of $d(.)$ for all pairs of granules, in which both granules must be either unlabeled or tagged with the same class label. The merging decision is based on a threshold value, Δ, or specialised expertise related to how to blend granules to increase the compactness of the model. We suggest $\Delta = 0.1$ as default for data within the unit hyper-cube, i.e., the candidate granules should be similar enough, carrying the same information indeed.

The new granule, γ^i, results from γ^{i_1} and γ^{i_2} combination, it is defined by Gaussian functions with modal values

$$\mu_j^i = \frac{\frac{\sigma_j^{i_1}}{\sigma_j^{i_2}}\mu_j^{i_1} + \frac{\sigma_j^{i_2}}{\sigma_j^{i_1}}\mu_j^{i_2}}{\frac{\sigma_j^{i_1}}{\sigma_j^{i_2}} + \frac{\sigma_j^{i_2}}{\sigma_j^{i_1}}}, \; j = 1, ..., n, \tag{4.19}$$

and dispersion

$$\sigma_j^i = \sigma_j^{i_1} + \sigma_j^{i_2}, \; j = 1, ..., n. \tag{4.20}$$

The uncertainty frontier of the original granules is take in account to optimise the location and size of the fused granule. The model redundancy is bounded by the granules merging, limiting the increasing of the quantity of rules [59, 98].

Deleting Rules

If a rule is discrepant with the current environment, it is delete from the model, i.e., if a rule is not activated in the last h_r iterations, then it must be removed from the knowledge basis. In a special case in which a class is rare, h_r may be set to infinity to keep the inactive rules. Removing rules periodically helps to keep the knowledge basis updated and the model compact.

For a complete description about the updating, removing, and merging procedures, see [62]. The eGFC algorithm is shown below, adapted by the introduction of the weak label to instances by means of a control chart methodology, as described in Chapter 6.

eGFC: Online Semi-Supervised Learning

```
1: Initial number of rules, $c = 0$;
2: Initial meta-parameters, $\rho^{[0]} = \Delta = 0.1$, $h_r = 200$;
3: Read input data sample $\mathbf{x}^{[h]}, h = 1$;
4: Create granule $\gamma^{c+1}$ (Eqs. (4.17)-(4.12)), unknown class $C^{c+1}$;
5: FOR $h = 2, ...$ DO
6:   Read $\mathbf{x}^{[h]}$;
7:   Calculate rules' activation degree (Eq. (4.10));
8:   Determine the most active rule $R^{i^*}$;
9:   Provide estimated class $C^{i^*}$;
10:  // Model adaptation
11:  IF $T(A_1^i(x_1^{[h]}), ..., A_n^i(x_n^{[h]})) \leq \rho^{[h]}\ \forall i,\ i = 1, ..., c$
12:    IF actual label $y^{[h]}$ is available
13:      Create labeled granule $\gamma^{c+1}$ (Eqs. (4.17)-(4.13));
14:    ELSE
15:      Create unlabeled granule $\gamma^{c+1}$ (Eqs. (4.17)-(4.18));
16:    END
17:  ELSE
18:    IF actual label $y^{[h]}$ is available
19:      Update the most active granule $\gamma^{i^*}$ whose class
20:      $C^{i^*}$ is equal to $y^{[h]}$ (Eqs. (4.14)-(4.16));
21:      Tag unlabeled active granules;
22:    ELSE
23:      Update the most active $\gamma^{i^*}$ (Eqs. (4.14)-(4.16));
24:    END
25:  END
26:  Update the $\rho$-level (Eqs. (4.17)-(4.18));
27:  Delete inactive rules based on $h_r$;
28:  Merge granules based on $\Delta$ (Eqs. (4.19)-(4.20));
29: END
```

4.2.3 eGNN: Evolving Granular Classification Neural Network

The eGNN method is a neuro-fuzzy granular network built incrementally from an online data stream [60]. Its processing units are fuzzy neurons and granules that codify an evolving set of fuzzy rules derived from the data stream, according to a fuzzy inference system. The network architecture is constructed by a progressive process. The consequent part of an eGNN rule is a class label in this work.

Neural Network

Consider that the data stream $(\mathbf{x}, y)^{[h]}$, $h = 1, ...,$ is measured from an unknown function f. Inputs x_j and output y are numerical data and a class. Figure 4.1 shows a four-layer eGNN model. The input layer receives $\mathbf{x}^{[h]}$. The granular layer is a set of granules G_j^i, $i = 1, \ldots, c$, stratified from input data, forming a fuzzy partition of the j-th input domain. A granule $G^i = G_1^i \times \cdots \times G_n^i$ is a fuzzy relation, i.e., a multidimensional fuzzy set in $X_1 \times \cdots \times X_n$. Thus, G^i has membership function $G^i(x) = min\{G_1^i(x_1), \ldots, G_n^i(x_n)\}$ in $X_1 \times \cdots \times X_n$.

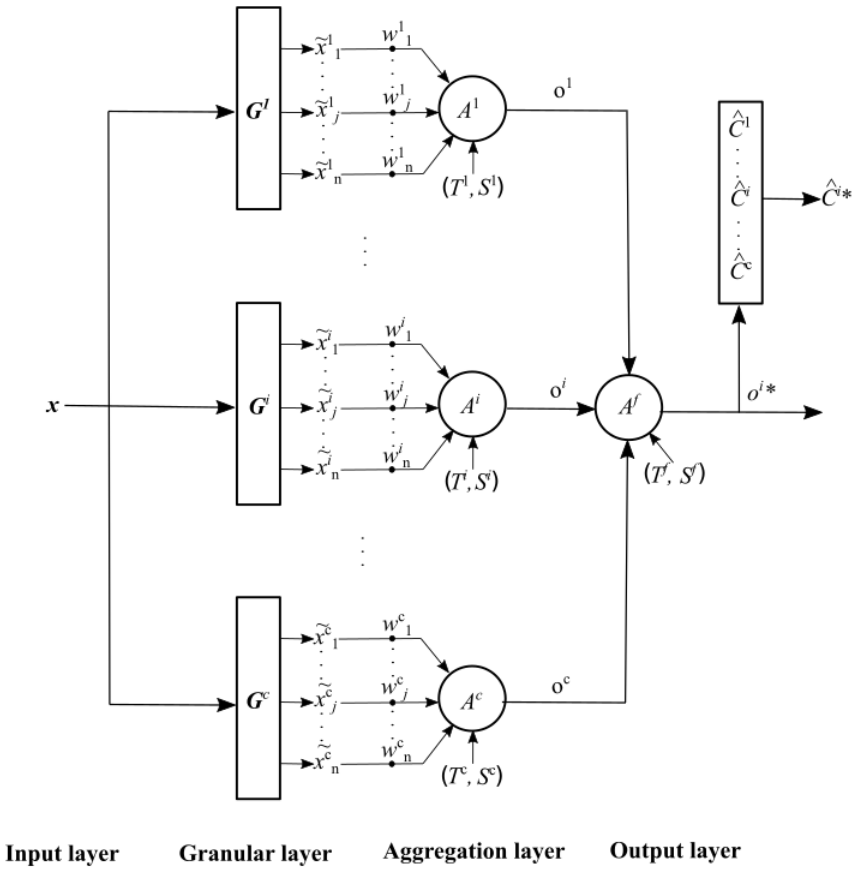


Figure 4.1: eGNN: Evolving neuro-fuzzy network architecture for classification

Similarity degrees $\widetilde{\mathbf{x}}^i = (\widetilde{x}_1^i, \ldots, \widetilde{x}_n^i)$ is the result of matching between $\mathbf{x} = (x_1, \ldots, x_n)$ and trapezoidal fuzzy sets $G^i = (G_1^i, \ldots, G_n^i)$, with $G_j^i = (\underline{\underline{g}}_j^i, \underline{g}_j^i, \overline{g}_j^i, \overline{\overline{g}}_j^i)$. In general, data and granules can be trapezoidal objects. A similarity measure to quantify the match between a numerical instance (the case of this paper) and the current knowledge is [60]:

$$\widetilde{x}_j^i = 1 - \frac{|\underline{\underline{g}}_j^i - x_j| + |\underline{g}_j^i - x_j| + |\overline{g}_j^i - x_j| + |\overline{\overline{g}}_j^i - x_j|}{4(\max(\overline{\overline{g}}_j^i, x_j) - \min(\underline{\underline{g}}_j^i, x_j))}. \tag{4.21}$$

The aggregation layer is composed by neurons A^i. A fuzzy neuron A^i combines weighted similarity degrees $(\widetilde{x}_1^i w_1^i, \ldots, \widetilde{x}_n^i w_n^i)$ into a single value o^i, which refers to the level of activation of R^i. The output layer processes $(o^1, \ldots, o^c)$ using a neuron A^f that performs the maximum S-norm. The class C^{i*} of the most active rule R^{i*} is the output.

Under assumption on specific weights and neurons, fuzzy rules extracted from eGNN are of the type

$$R^i(x) : \text{if } (x_1 \text{ is } G_1^i) \text{ and } \ldots \text{ and } (x_n \text{ is } G_n^i) \text{ then } (\hat{y} \text{ is } \hat{C}^i).$$

Neuron Model

Fuzzy neurons are neuron models based on aggregation operators. Aggregation operators $A : [0,1]^n \rightarrow [0,1]$, $n > 1$, combine input values in the unit hyper-cube $[0,1]^n$ into a single value in $[0,1]$. They must satisfy the following: monotonicity in all arguments and boundary conditions [60]. This study uses the minimum and maximum operators only [95, 99]. Figure 4.2 shows an example of fuzzy neuron in which synaptic processing is given by the T-norm product, and the aggregation operator A^i is used to combine individual inputs. The output o^i is $A^i(\widetilde{x}_1^i w_1^i, \ldots, \widetilde{x}_n^i w_n^i)$.

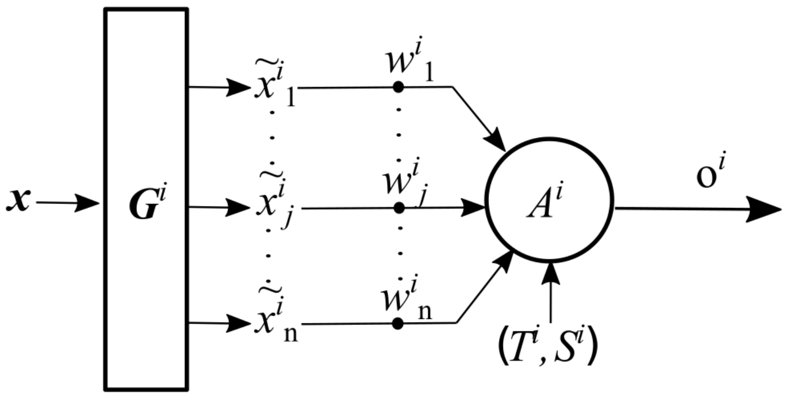


Figure 4.2: Fuzzy neuron model

Granular Region

As G_j^i is a trapezoidal membership function, its support, core, midpoint, and width are

$$\text{supp}(G_j^i) = [\underline{\underline{g}}_j^i, \overline{\overline{g}}_j^i], \quad \text{core}(G_j^i) = [\underline{g}_j^i, \overline{g}_j^i] \tag{4.22}$$

$$\text{mp}(G_j^i) = \frac{\underline{g}_j^i + \overline{g}_j^i}{2}, \quad \text{wdt}(G_j^i) = \overline{\overline{g}}_j^i - \underline{\underline{g}}_j^i. \tag{4.23}$$

It is possible to expand the width of fuzzy sets G_j^i within the area E_j^i delimited by ρ, i.e., $\text{wdt}(G_j^i) \leq \rho$. E_j^i is given by $[\text{mp}(G_j^i) - \frac{\rho}{2}, \text{mp}(G_j^i) + \frac{\rho}{2}]$. Clearly, $\text{wdt}(G_j^i) \leq \text{wdt}(E_j^i)$.

ρ Update

The value of ρ affects the information granularity, and consequently the model accuracy. $\rho \in [0, 1]$ is used to control the size of expansion regions.

eGNN starts with an empty rule base, and $\rho^{[0]} = 0.5$ is used as default. Let r be the number of rules created in h_r steps, and η be a reference rate. If the number of rules grows faster than the rate η, then ρ is increased, otherwise ρ is reduced, as in Eq. (4.5). Appropriate values for ρ are found autonomously. If $\rho = 1$, then eGNN is structurally stable, but unable to capture abrupt changes. Conversely, if $\rho = 0$, then eGNN overfits the data causing excessive model complexity. Adaptability is reached from intermediate values.

Reducing ρ requires reduction of big granules according to

$$\begin{array}{ll} \text{if } \text{mp}(G_j^i) - \frac{\rho(\text{new})}{2} > \underline{\underline{g}}_j^i & \text{then } \underline{\underline{g}}_j^i(\text{new}) = \text{mp}(G_j^i) - \frac{\rho(\text{new})}{2} \\ \text{if } \text{mp}(G_j^i) + \frac{\rho(\text{new})}{2} < \overline{\overline{g}}_j^i & \text{then } \overline{\overline{g}}_j^i(\text{new}) = \text{mp}(G_j^i) + \frac{\rho(\text{new})}{2} \end{array}$$

Cores $[\underline{g}_j^i, \overline{g}_j^i]$ are handled in a similar way.

Developing Granules

If the support of at least one entry of $\mathbf{x}$ is not enclosed by expansion regions $(E_1^i, \ldots, E_n^i)$, eGNN generates a new granule, G^{c+1}. This new granule is constituted by fuzzy sets whose parameters are

$$G_j^{c+1} = (\underline{\underline{g}}_j^{c+1}, \underline{g}_j^{c+1}, \overline{g}_j^{c+1}, \overline{\overline{g}}_j^{c+1}) = (x_j, x_j, x_j, x_j). \tag{4.24}$$

Updating granules consists in expanding or contracting the support and the core of fuzzy sets G_j^i. In particular, G^i is chosen from $\mathit{arg\ max}(o^1, \ldots, o^c)$.

Adaptation proceeds depending on where x_j in placed in relation to G_j^i.

$$\begin{array}{llll} \text{if} & x_j \in [\text{mp}(G_j^i) - \frac{\rho}{2}, \underline{\underline{g}}_j^i] & \text{then} & \underline{\underline{g}}_j^i(\text{new}) = x_j \\ \text{if} & x_j \in [\text{mp}(G_j^i) - \frac{\rho}{2}, \text{mp}(G_j^i)] & \text{then} & \underline{g}_j^i(\text{new}) = x_j \\ \text{if} & x_j \in [\text{mp}(G_j^i), \text{mp}(G_j^i) + \frac{\rho}{2}] & \text{then} & \underline{g}_j^i(\text{new}) = \text{mp}(G_j^i) \\ \text{if} & x_j \in [\text{mp}(G_j^i) - \frac{\rho}{2}, \text{mp}(G_j^i)] & \text{then} & \overline{g}_j^i(\text{new}) = \text{mp}(G_j^i) \\ \text{if} & x_j \in [\text{mp}(G_j^i), \text{mp}(G_j^i) + \frac{\rho}{2}] & \text{then} & \overline{g}_j^i(\text{new}) = x_j \\ \text{if} & x_j \in [\overline{\overline{g}}_j^i, \text{mp}(G_j^i) + \frac{\rho}{2}] & \text{then} & \overline{\overline{g}}_j^i(\text{new}) = x_j \end{array}$$

Operations on core parameters, $\underline{g}_j^i$ and $\overline{g}_j^i$, require additional adaptation of the midpoint

$$\text{mp}(G_j^i)(\text{new}) = \frac{\underline{g}_j^i(\text{new}) + \overline{g}_j^i(\text{new})}{2}. \tag{4.25}$$

Therefore, support contractions may be needed:

$$\begin{aligned}&\text{if } \text{mp}(G_j^i)(\text{new}) - \tfrac{\rho}{2} > \underline{\underline{g}}_j^i \text{ then } \underline{\underline{g}}_j^i(\text{new}) = \text{mp}(G_j^i)(\text{new}) - \tfrac{\rho}{2}\\ &\text{if } \text{mp}(G_j^i)(\text{new}) + \tfrac{\rho}{2} < \overline{\overline{g}}_j^i \text{ then } \overline{\overline{g}}_j^i(\text{new}) = \text{mp}(G_j^i)(\text{new}) + \tfrac{\rho}{2}.\end{aligned}$$

Updating Neural Network Weights

$w_j^i \in [0,1]$ is proportional to the importance of the j-th attribute of G_j^i to the neural network output. When a new granule G^{c+1} is generated, weights are set as $w_j^{c+1} = 1$, $\forall j$.

The updated w_j^i, associated to the most active granule G^i, $i = arg\ max(o^1, \ldots, o^c)$, are

$$w_j^i(\text{new}) = w_j^i(\text{old}) - \beta^i \widetilde{x}_j^i |\epsilon|. \tag{4.26}$$

in which $\widetilde{x}_j^i$ is the similarity to G_j^i; β^i depends on the number of right ($Right^i$) and wrong ($Wrong^i$) classifications

$$\beta^i = \frac{Wrong^i}{Right^i + Wrong^i} \qquad and \qquad \epsilon^{[h]} = C^{[h]} - \hat{C}^{[h]}$$

in which $\epsilon^{[h]}$ is the current estimation error, and $C^{[h]}$ is a weak label provided by the control chart approach described in the methodology section.

Learning Algorithm

The learning algorithm to evolve eGNN classifiers is given below.

eGNN Learning: Evolving Granular Neural Network

```
1: select a type of neuron for the aggregation layer
2: set parameters $\rho^{[0]}$, $h_r$, $\eta$;
3: read instance $\mathbf{x}^{[h]}$, $h = 1$;
4: use control chart to label $\mathbf{x}^{[h]}$ with $C^{[h]}$;
5: create granule $G^{c+1}$, neurons $A^{c+1}$, $A^f$, and connections;
6: for $h = 2, \ldots$ do
7:     read and feed-forward $\mathbf{x}^{[h]}$ through the network;
8:     compute rule activation levels $(o^1, \ldots, o^c)$;
9:     aggregate activation using $A^f$ to get estimate $\hat{C}^{[h]}$;
10:     // the class $C^{[h]}$ becomes available;
11:     compute output error $\epsilon^{[h]} = C^{[h]} - \hat{C}^{[h]}$;
12:     if $\mathbf{x}^{[h]}$ is not $E^i$ $\forall i$ or $\epsilon^{[h]} \neq 0$ then
13:         create granule $G^{c+1}$, neuron $A^{c+1}$, connections;
14:         associate $G^{c+1}$ to $C^{[h]}$;
15:     else
16:         update $G^{i*}$, $i* = arg\ max(o^1, \ldots, o^c)$;
17:         adapt weights $w_j^{i*}$ $\forall j$;
18:     end if
19:     if $h = \beta h_r$, $\beta = 1, \ldots$ then
20:         adapt model granularity $\rho$;
21:     end if
22: end for
```

4.3 eGNN: Aggregation Operators as Neurons

Aggregation neurons are artificial neuron models based on aggregation operators. eGNN may use different types of aggregation neurons to perform information fusion. There are no guideline to choose a particular operator to construct an aggregation neuron. The choice depends on the application environment and domain knowledge; it usually conforms to simplicity, and understandability of granular rules. An operator $A : [0, 1]^n \to [0, 1]$, $n > 1$ maps input data in the unit hypercube $[0, 1]^n$ into an output datum in $[0, 1]$. The operator satisfies two properties: (***i***) monotonicity in all arguments, i.e., given $x^1 = (x_1^1, ..., x_n^1)$ and $x^2 = (x_1^2, ..., x_n^2)$, if $x_j^1 \leq x_j^2$ $\forall j$ then $A(x^1) \leq A(x^2)$; (***ii***) boundary conditions: $A(0, 0, ..., 0) = 0$ and $A(1, 1, ..., 1) = 1$. We have particularly analyzed some families of operators within the eGNN framework, as summarized below.

4.3.1 T-norms

T-norms (T) are commutative, associative and monotone operators on the unit hypercube whose boundary conditions are $T(\alpha, \alpha, ..., 0) = 0$ and $T(\alpha, 1, ..., 1) = \alpha$, $\alpha \in [0, 1]$. The neutral element of T-norms is $e = 1$. An example is the minimum operator

$$T_{min}(x) = \min_{j=1,\ldots,n} x_j, \tag{4.27}$$

which is the strongest T-norm because

$$T(x) \leq T_{min}(x) \text{ for any } x \in [0, 1]^n. \tag{4.28}$$

The minimum is also idempotent, symmetric and Lipschitz-continuous. Other examples of T-norms are the product,

$$T_{prod}(x) = \prod_{j=1}^{n} x_j, \tag{4.29}$$

and the Lukasiewicz T-norm,

$$T_L(x) = \max \left(0, \sum_{j=1}^{n} x_j - (n - 1) \right). \tag{4.30}$$

4.3.2 S-norms

S-norms (S) are operators on the unit hypercube which are commutative, associative and monotone. $S(\alpha, \alpha, ..., 1) = 1$ and $S(\alpha, 0, ..., 0) = \alpha$ are the boundary conditions of S-norms; and $e = 0$ is their neutral element.

S-norms are stronger than T-norms. The maximum operator,

$$S_{max}(x) = \max_{j=1,\ldots,n} x_j, \tag{4.31}$$

is the weakest S-norm, i.e.,

$$S(x) \geq S_{max}(x) \geq T(x), \text{ for any } x \in [0, 1]^n. \tag{4.32}$$

Other examples of S-norms are the probabilistic sum,

$$S_{prob}(x) = 1 - \prod_{j=1}^{n}(1 - x_j), \tag{4.33}$$

and the Lukasiewicz S-norm,

$$S_L(x) = \min\left(1, \sum_{j=1}^{n} x_j\right). \tag{4.34}$$

4.3.3 Averaging Operators

An aggregation operator A is averaging if for every $x \in [0,1]^n$ it is bounded by

$$T_{min}(x) \leq A(x) \leq S_{max}(x). \tag{4.35}$$

The basic rule is that the output value cannot be lower or higher than any input value. An example of averaging operator is the arithmetic mean,

$$M(x) = \frac{1}{n}\sum_{j=1}^{n} x_j. \tag{4.36}$$

Averaging operators are idempotent, strictly increasing, symmetric, homogeneous, and Lipschitz continuous.

4.3.4 Ordered Weighted Averaging

Ordered Weighted Averaging (OWA) is a specialisation of averaging aggregation functions associated with a weighting vector $\mathbf{w}$. This class includes the max, min, median, and arithmetic average operators as special cases distinguished by the choice of a different $\mathbf{w}$. OWA is a mapping of dimension n, $F_w : R_n \to R$ in which $\mathbf{w}$ with $w_i \geqslant 0$, $\sum w_i = 1$,

$$OWA_w(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i < \mathbf{w}, \mathbf{x} \searrow >, \tag{4.37}$$

in which the notation $x \searrow$ denotes the vector $\mathbf{x}$ sorted in *non-increasing* order ($x_{(1)} \geqslant x_{(2)} \geqslant \cdots \geqslant x_{(n)}$).

An appealing property of OWA operator, called *orness*, it is introduced as a measure to give a numerical quantification of the degree of disjunctive behaviour of an operator, being also saw as the mode of decision making related to the aggregation process[100, 101]. A min and max aggregators are considered as a pure "or" and "and" respectively, and the $orness^y(OWA)$ provides a measure of the location between these operators to an OWA function:

$$orness^y_{(\mathbf{w})} = \frac{1}{n-1}\sum_{i=1}^{n}(n-i)w_i. \tag{4.38}$$

in which the superscript y denotes Yager's definition. Related to the $\mathbf{w}$ calculus, some OWA cases are

- Arithmetic mean (am): $w_i = \frac{1}{n}$ and $orness^y(am) = \frac{1}{2}$.
- Min: if $w = (1, 0, \ldots, 0)$ then $OWA_w = max(\mathbf{x})$ and $orness^y(min) = 0$.
- Max: if $w = (0, 0, \ldots, 1)$ then $OWA_w = min(\mathbf{x})$ and $orness^y(max) = 1$.
- Hurwicz aggregation (ha): if $w = (\alpha, 0, \ldots, 1-\alpha)$ then $OWA_w(ha) = \alpha max(\mathbf{x} + (1-\alpha)min(\mathbf{x})$
- OWA_{w1} with $w_i = \frac{1}{n}\sum\limits_{j=i}^{n}\frac{1}{j}$ and $orness^y(OWA_{w1}) = \frac{3}{4}$
- OWA_{w2} with $w_i = \frac{2(n+1-i)}{n(n-1)}$ and $orness^y(OWA_{w2}) = \frac{2}{3}$
- Neat OWA: $w_i = \frac{x^p_{(i)}}{\sum\limits_{i=1}^{n} x^p_{(i)}}$ with $p \in]-\inf, \inf[$.

4.3.5 Compensatory T-S Operators

Compensatory T-S operators balance the opposite effects of T- and S-norms. T-S aggregation is uniform in the sense that it does not depend on parts of the underlying domain. T-S operators averages two values – obtained from a T-norm and a S-norm – by means of weighted quasi-arithmetic mean. The linear convex operator

$$L(x) = (1-v)T(x_1, ..., x_n) + vS(x_1, ..., x_n), \tag{4.39}$$

in which $v \in [0, 1]$, is an example of T-S operator of the family of weighted quasi-arithmetic means. T-S operators need not to be dual in terms of T and S. It follows that:

$$S(x) \geq L(x) \geq T(x), \text{ for any } x \in [0, 1]^n. \tag{4.40}$$

4.3.6 Aggregation Neuron Model

Let $\widetilde{x} = (\widetilde{x}_1, ..., \widetilde{x}_n)$ be a vector of membership degrees of a sample $x = (x_1, ..., x_n)$ in the fuzzy sets G_j of $G = (G_1, ..., G_n)$. Let $w = (w_1, ..., w_n)$ be a weighting vector such that

$$w_j \in [0, 1], \; j = 1, ..., n. \tag{4.41}$$

Aggregation neurons use product T-norm to perform synaptic processing and an operator A to fuse the individual results of synaptic processing in the neuron body. The output of the neuron is

$$o = A(\widetilde{x}_1 w_1, ..., \widetilde{x}_n w_n). \tag{4.42}$$

An aggregation neuron performs a diversity of nonlinear input-output maps depending on the choice of weights w, and triangular norms T and S. The structure of a neuron is shown in Fig. 4.2.

4.4 Summary

We present the evolving fuzzy and neuro-fuzzy granular classifiers, FBeM, eGFC, and eGNN, in a real-time anomaly detection problem considering online log data streams from a computing center. Logging activity rate is a standard behavioural metric obtained from online log data, used as input to the methods. All the adaptive mechanisms, i.e., creating new granules, merging similar granules, deleting the inactive granules, are deeply described, highlighting the granularity and model evolving, as well as the aggregation neuron models to eGNN.

Chapter 5

eLP: evolving Log Parsing

Logs are one of the most valued sources of information for large-scale service maintenance [102]. Log processing for model development is a branch of *Natural Language Processing* (NLP) with intersection to *Data Mining* [103]. Log files are, in general, time-indexed unstructured textual data. They can be seen as written expressions, with poor syntax. In fact, log files are created by using an austere and restricted idiom. Most learning and monitoring approaches handles logs as strings. The approaches perform string matching based on strings distance measures [104].

Many studies link the importance of addressing anomaly detection together with log parsing. The data-parsing task tends to increase the ability of a model to recognise anomalous behaviors through data structuring. Optimal log parsing is defined theoretically in [105]. This work considers log parsing as an abstraction function that generates information in a structured way, thus summarising multiple log data.

A comparative study, [106], evaluates 13 automated log parsing methods based on benchmark datasets. The evaluation considers accuracy, robustness, and efficiency. The methods are grouped by the technique used to parse the logs, i.e., frequent pattern mining, data clustering, and heuristics based on token identification and partitioning strategies. Other techniques include the longest common sub-sequence method, and multi-objective optimization by means of an evolutionary algorithm. The parsing accuracy of the 13 methods varies from 48% to 86%. Drain, the best of the methods evaluated, is an online log parser based on a fixed depth parse tree; a model encodes rules for log parsing [107].

Natural Language Processing (NLP) methods stand out as log parsing methods using off-the-shelf NLP algorithms [103]. The well-known Word2vec methods can identify words in low dimensional vectors efficiently. Ad-hoc Word2vec algorithms improve log parsers by extracting events [108]. Log Parsers based on Vectorization (LPV) convert logs into vectors. LPV clusters the logs using similarity or distance measures, and extracts templates from the resulting clusters [104].

The feature-based method in [109] extracts system events from unstructured free-text logs based on an analytic solution that groups logs by similarity considering a dataset preprocessed by regular expressions. The LogParse method [102] is proposed as an adaptive framework based on word classification. It learns template features using an open-source toolkit. The Paddy method [110] uses a dynamic dictionary structure with a scheme of inverted index to efficiently search template candidates using the Jaccard similarity. Prefix-Graph [111] is an online approach based on a probabilistic graph structure derived from a prefixed tree. The method merges branches with highly similar probability distributions. The resulting model represents templates as the combination of cut-edges in root-to-leaf paths of the graph. The Clustering based on Length and First token (CLF) method [69] extracts templates in three steps: by (i) clustering unstructured logs using heuristic rules; (ii) clustering according to specific separation rules; and (iii) identifying the associated events.

This chapter introduces a granular-computing method for log parsing. The method, called eLP, extracts a formal grammar from textual data streams, and stores local grammars into information granules. Granules are described by word vectors and an interval rule base, which is evolved on the fly in a textual scenario. In particular, while the vast majority of evolving intelligent methods are focused on numerical point-wise data processing, interval data streams are considered by the granular methods in [112] [113], and fuzzy data streams are evaluated in [60] [114] [62]. Nevertheless, for the first time in the evolving intelligence literature, the eLP method is able to process streams of words and sentences.

The rest of this chapter is structured as follows. Section 5.1 defines logs as a formal language, explaining the eLP parsing modelling framework, and exemplifying it. Section 5.2 addresses the basics of the eLP model. Section 5.3 provides the updating mechanisms. Section 5.4 presents the eLP learning algorithm from textual data streams.

5.1 Preliminary Concepts

Computing center is a set of software working collaboratively to achieve a common purpose through resources sharing among users/customers/software. This ecosystem is propitious to generate resource conflicts and data inconsistencies inwardly or among systems. As a community of computing elements, it is interesting that each member has the possibility to tell about the occurrence of events. Nowadays, this communication is unidirectional and inefficient, but anyway, expressed by a language – the system logs.

5.1.1 Logs as a language

A natural or programming language is formed by syntactic rules. A language is generated by a grammar $\mathcal{G}$ defined as a set of sentences [115]. Precisely, mental operations are the basis of all possible grammars [116]. Logs can be understood as expressions of a rudimentary language, and, as such, a language $\mathcal{L}$ can be defined by a set of syntactic rules.

The language $\mathcal{L}$ of software provides indirect and ineffective machine-to-human communication since the product of its expression – the logs or log files – are excessive, redundant, and hard to be understood by humans.

Let logs be generated by $\mathcal{L}(\mathcal{G})$. $\mathcal{G} =< \Sigma, \Phi, S, R >$ is a formal grammar. The elements of $\mathcal{G}$ mean:

- Σ is alphabet of terminal terms;

- Φ is alphabet of non-terminal terms, such that $\Sigma \cap \Phi = \emptyset$;
- S is the initial symbol; and
- R is a finite set of production rules that maps α in β: $\alpha \rightarrow \beta$; being $R \subseteq \Gamma^* \times \Gamma^*$ with $\Gamma = \Sigma \cup \Phi$; α is non-empty, i.e., $\alpha \neq \varepsilon$; and $\alpha \notin \Sigma^*$ [115].

Different from terminal terms Σ, the non-terminal terms Φ can apply a production rule, $\alpha \rightarrow \beta$.

A formal grammar is a concept from linguistics, applied in compilers to recognise linguistic expressions of programming languages. Syntactic analysis verify if code production follows the syntactic rules of a programming language. A production rule $\alpha \rightarrow \beta$ replaces the antecedent α by a product β, recognising part of β composed by terminals. The non-terminal terms in the consequent part of a rule are replaced by other production rules, until only non-terminal terms remain. Consider the following grammar $\mathcal{G}$:

$$\begin{aligned}
S \quad &\rightarrow A^{0'} \\
A^{0'} &\rightarrow a'A^{0'} \mid w_1 A^{1'} \mid \ldots \mid w_k A^{k'} \mid \ldots \mid w_n \mid \varepsilon \\
A^{1'} &\rightarrow a'A^{1'} \mid w_2 A^{2'} \mid \ldots \mid w_k A^{k'} \mid \ldots \mid w_n \mid \varepsilon \\
A^{2'} &\rightarrow a'A^{2'} \mid w_3 A^{3'} \mid \ldots \mid w_k A^{k'} \mid \ldots \mid w_n \mid \varepsilon \\
&\vdots \\
A^{(n-1)'} &\rightarrow \quad a'A^{(n-1)'} \mid w_n \mid \varepsilon.
\end{aligned} \tag{5.1}$$

Grammar $\mathcal{G}$ assists on the recognition of a type of log, that is, $\mathcal{G}$ identifies the typical words in consecutive $x^{[h]}$, $h = 1, ...$ In $\mathcal{G}$, ε is the empty set; and $\Phi = \{S, A^{1'}, A^{2'}, \ldots, A^{(n-1)'}\}$ is the alphabet of non-terminal terms. Any local template of a granular model (as defined in Section II-B) realises the same grammar $\mathcal{G}$ (5.1), but the terminal terms of different templates associated to different information granules γ^i, $i = 1, ..., c$, are distinctive, i.e., $\Sigma = \mathbf{w}^i$. Granules, granular model, and templates will be defined in the next sections.

Fundamentally, Equation (5.1) says that the recognition of a message in a log stream – in which the message is expressed by an n-word input vector $\mathbf{x}^{[h]} = [x_1^{[h]}, ..., x_n^{[h]}]$, with $h = 1, ...$ being the time index – is based on $\mathcal{G}$. First, S is trigged so that $S \rightarrow A^{0'}$. Next, a rule of the set of n production rules, $A^{0'} \rightarrow a'A^{0'} \mid w_1 A^{1'} \mid \ldots \mid w_k A^{k'} \mid \ldots \mid w_n$, is trigged, e.g., if the first word to be read in $\mathbf{x}^{[h]}$ is equal to the word w_1 of the grammar $\mathcal{G}$, i.e., $\mathbf{x}_1^{[h]} = w_1$, then the word is recognised by the rule $A^{0'} \rightarrow w_1 A^{1'}$, and rule $A^{1'}$ is trigged. At the $A^{0'}$ level (second row of (5.1)), any w_k of $\mathbf{w}^i$ can be found in $\mathbf{x}^{[h]}$. For example, if $x_1^{[h]} = w_2$, then $x_1^{[h]}$ is recognised by the rule $A^{0'} \rightarrow w_2 A^{2'}$. However, if $x_j^{[h]}$, $j > 1$, is equal to w_1, then $x_j^{[h]}$ can not be recognised as a word template, since w_1 must to be recognised before w_2. In other words, word ordering matters for template recognition. The same procedure is performed until either the last word of the m-word template or the last word of the n-word instance $\mathbf{x}^{[h]}$.

The words that characterise a template may match, fully or partially, a textual data instance. The boundary cases happen when a template does not match any word of an instance, or when all of its words are found in $\mathbf{x}^{[h]}$. Production rules $A^{k'} \rightarrow a' A^{k'}$, $k = 0, 1, \dots, n-1$, process the attributes $x_j^{[h]}$, $j = 1, ..., n$, indefinitely, no matter the amount of attributes n. The grammar $\mathcal{G}$, (5.1), operates continuously until a particular template (an information granule) is found. Such granule may be constructed and represented within a granular model if: (i) the m production rules of its associated template are processed for a single $\mathbf{x}^{[h]}$; and (ii) the granule is more active than other granules of the granular model.

Notice that the grammar $\mathcal{G}^i$ of the granule γ^i is capable of recognising instances that include non-expected words between the w_k's, i.e., some additional words may appear as attributes of $\mathbf{x}^{[h]}$. This is a result of the continuous search for attributes that conform to the template of a granule. Not all production rules are trigged during the recognition process. Additionally, the property of accepting incomplete (partial) match is of utmost importance in log stream processing since some attributes of log messages $\mathbf{x}^{[h]}$ are very dynamic in terms of the value itself, and amount (variable size, n).

A language $\mathcal{L}$ defined by $\mathcal{G} = \{\mathcal{G}^1, \dots, \mathcal{G}^c\}$ considers c varieties of linguistic expressions. A granule of a granular model $\gamma = \{\gamma^1, \dots, \gamma^c\}$ has its own grammar. The interval granular learning method introduced in the following sections, called eLP, evolving Log Parsing, is highly convenient to log parsing because, different from the vast majority of NLP methods based on formal grammars, the eLP method deals with time-varying amounts of atypical words arising in any position of streaming sentences. Moreover, granules and templates are evolved from scratch as word patterns are found in a textual data stream.

5.1.2 The eLP granular framework

Let $\gamma = \{\gamma^1, \dots, \gamma^i, \dots, \gamma^c\}$ be the collection of granules of a granular model developed from an online data stream. We define a granule γ^i as associated to a template. The i-th template describes the i-th granule. A template contains two components basically: (i) a set of m words, i.e.,

$$\mathbf{w}^i = [w_1^i, \dots, w_k^i, \dots, w_m^i], \tag{5.2}$$

being w_k^i the k-th word of the i-th granule; and (ii) a set of numeric intervals,

$$\mathbf{I}^i = [[l_1^i, L_1^i], \dots, [l_k^i, L_k^i], \dots, [l_m^i, L_m^i]]. \tag{5.3}$$

The interval endpoints, l_k^i and L_k^i, $\forall k$, are positive integers in the set $\{1, 2, ..., n\}$; and $l_k^i \leq L_k^i$. Section IV describes a procedure to translate words w_k^i into corresponding intervals I_k^i. Essentially, l_k^i and L_k^i are indices in which the k-th word of the i-th template are found in a streaming word vector $\mathbf{x}^{[h]}$,

$$\mathbf{x}^{[h]} = [x_1^{[h]}, \dots, x_j^{[h]}, \dots, x_n^{[h]}]; \tag{5.4}$$

$h = 1, 2, \dots$ is time index.

The amount of words and intervals, m, may be different for different templates i, $i = 1, ..., c$. We use m locally (with no additional index) for any granule of the granular model to avoid abuse of notation. From a geometric perspective, m is the number of dimensions of a particular granule. The same reasoning holds to n, the length of $\mathbf{x}^{[h]}$, which may be different in different time steps h.

In this work, an $\mathbf{x}^{[h]}$ is a log message. This means that some random attributes of the input word vector $\mathbf{x}^{[h]}$ are typical in many consecutive instances. Such repetitive words describe a message type, and are usually written by programmers in "print-file function" style. Their call points in an algorithm are chosen conveniently by the programmers. On the other hand, some attributes of the input vector $\mathbf{x}^{[h]}$ are in fact related to the current state of the services in a computer network.

From a linguistic perspective, the recognition that an $\mathbf{x}^{[h]}$ has full or partial membership in a granule γ^i is based on word comparison and on their relative order in a sentence.

5.1.3 Log parsing

Log parsing is the first procedure in textual data processing. Log parsing aims to separate the typical words in consecutive messages $\mathbf{x}^{[h]}$, $\mathbf{x}^{[h+1]}$, ..., from the attributes that describe the current system state. While the typical part of a streaming instance $\mathbf{x}^{[h]}$ suggests global patterns (events) in a large computer network, the variable state part of the instance contains details of a particular pattern.

While the textual data stream $\mathbf{x}^{[h]}, h = 1, ...$, gives a track of system events, an evolving granular model γ that deals with words and word frequency can be used to infer the system state and predict the evolution of its global operating dynamics. The meaning and amount of state attributes that describe a running system change with the associated message type, that is, change for different typical words.

Example

Given the sentence: 'My house is lovely and full of joy'. The n-word input vector ($n = 8$) at the time step h is:

$\mathbf{x}^{[h]}$ = ['My', 'house', 'is', 'lovely', 'and', 'full', 'of', 'joy'].

Let γ be a model with two m-word granules ($m = 4$):

$\mathbf{w}^1$ = ['My', 'house', 'is', 'and'],

and

$\mathbf{w}^2$ = ['My', 'heart', 'is', 'and'].

Thus, $\mathbf{w}^1$ matches $\mathbf{x}^{[h]}$ (4 out of 4), and $\mathbf{w}^2$ partially matches $\mathbf{x}^{[h]}$ (3 out of 4). Concerning the viewpoint of $\mathbf{w}^2$, five of the eight attributes of $\mathbf{x}^{[h]}$ are atypical (state attributes of $\mathbf{w}^2$). They are: $x_2^{[h]}$ = 'house', $x_4^{[h]}$ = 'lovely', $x_6^{[h]}$ = 'full', $x_7^{[h]}$ = 'of', and $x_8^{[h]}$ = 'joy'.

In a further time step, granules γ^1 and γ^2 are evaluated by a learning algorithm as being similar to each other. Then, they are merged into γ^3 to form

$\mathbf{w}^3$ = ['My', 'is', 'and'].

The words ‘house’ and ‘heart’ were removed from the typical set of words to form $\mathbf{w}^3$. The resulting γ^3 contains a more generic template than the templates of the merged granules.

Consider the translation of the entries of $\mathbf{w}^3$ into intervals $\mathbf{I}^3$. Thus,

$\gamma^3 = [$‘My’$\rightarrow[1, 1]$, ‘is’$\rightarrow[2, 5]$, ‘and’$\rightarrow[7, 9]]$.

Notice that, while the word ‘is’ may appear from the second to the fifth position of a streaming instance, i.e., $[l_2^3, L_2^3] = [2, 5]$; the word ‘My’ appears as the first word of a sentence, i.e., $[l_1^3, L_1^3] = [1, 1]$ (a degenerated interval), and so on.

Suppose the same sentence $\mathbf{x}^{[h]}$ arises in a further time step, $h + \theta$, $\mathbf{x}^{[h+\theta]}$, after the existence of γ^3. Then,

$\mathbf{x}^{[h+\theta]} = [\text{‘My’}, \text{‘house’}, \text{‘is’}, \text{‘lovely’}, \text{‘and’}, \text{‘full’}, \text{‘of’}, \text{‘joy’}]$.

Therefore, the word set {‘house’, ‘lovely’, ‘full’, ‘of’, ‘joy’} is atypical to $\mathbf{w}^3$.

Define

$$z_k^i = \arg_{j(w_k^i \equiv x_j^{[h]})}(\mathbf{x}^{[h]}), \; k = 1, ..., m, \tag{5.5}$$

z_k^i is an integer in the set $\{0, 1, \ldots, n\}$, in which $n = 8$ is the length of $\mathbf{x}^{[h]}$. Precisely stated, z_k^i is equal to the argument j that indicates match between a word w_k^i of $\mathbf{w}^i$ and a word $x_j^{[h]}$ of $\mathbf{x}^{[h]}$. If there is no correspondence between a w_k^i and all words of $\mathbf{x}^{[h]}$, then $z_k^i = \#$. We shall formally and generically define z_k^i later, in Section III-B.

Back to the example, from $\mathbf{x}^{[h+\theta]}$ we get

$\mathbf{z}^3 = [$‘My’ $\rightarrow 1$, #, ‘is’ $\rightarrow 3$, #, ‘and’$\rightarrow 5$, #, #, #].

The number of matches between γ^3 and $\mathbf{x}^{[h+\theta]}$ depends on $\mathbf{I}^3$. The words ‘My’ and ‘is’ are found in $j = 1$ and $j = 3$. Since $1 \in [1, 1]$, and $3 \in [1, 5]$, we have matches. However, the word ‘and’ is in $j = 5$ and, therefore, out of the bounds $[7, 9]$. There are 2 matches between $\mathbf{x}^{[h+\theta]}$ and γ^3.

Assume a fourth m-word granule ($m = 3$),

$\gamma^4 = [$‘My’$\rightarrow[1, 3]$, ‘is’$\rightarrow[2, 4]$, ‘and’$\rightarrow[3, 6]]$.

In this case, γ^4 and $\mathbf{x}^{[h+\theta]}$ matches 3 times, since $1 \in [1, 3]$; $3 \in [2, 4]$; and $5 \in [3, 6]$. Although $\mathbf{x}^{[h+\theta]}$ finds correspondence with both γ^3 and γ^4, γ^4 is chosen to accommodate $\mathbf{x}^{[h+\theta]}$ since it provides a larger amount of matches.

5.2 eLP: evolving Log Parsing

This section outlines eLP, an unsupervised evolving classifier to handle the online log parsing problem. Log parsing is a computationally expensive task as it deals with big textual data processing. eLP is a general-purpose classifier, and a member of the evolving granular classifier (eGC) family [50][62]. It is inspired on the method called Interval-Based Evolving Modeling (IBeM) [112]. While the latter is useful for interval vectors only, the proposed method is the first in the evolving systems literature that is suitable for textual data processing. eLP focuses on extracting templates and features from log data by means of interval-based granules centered on words as basic processing units. Additionally, eLP assists on the identification of context-oriented event occurrences and therefore is useful to support decision making.

5.2.1 Preliminaries

Let a log stream, $\tilde{L}$, given by time-indexed word instances $\mathbf{x}^{[h]}$, $h = 1, ...,$ be written in a file continuously. $\mathbf{x}^{[h]}$ is a usual log message containing typical repetitive attributes and some variable system-state attributes. The length n of distinct $\mathbf{x}^{[h]}$ may be different in different time steps h.

A word w_k^i of the i-th granule of the current collection of granules $\{\gamma^1, ..., \gamma^c\}$ of a granular model γ, as well as a word $x_j^{[h]}$ of $\mathbf{x}^{[h]}$, is formed by one or more ('+') characters,

$$word = char^+, \tag{5.6}$$

in which

$$char = \{\text{a, ..., z, A, ..., Z, 0, ..., 9, `specialChar'}\}, \tag{5.7}$$

being 'specialChar' a UNICODE character.

The length m of different $\mathbf{w}^i$, $i = 1, ..., c$; and the length n of $\mathbf{x}^{[h]}$, $h = 1, ...,$ may change at each time step. A message type (a template) describes a granule and determines a class C^i in a classification problem. Granules are created whenever a message $\mathbf{x}^{[h]}$ is significantly different from the words within the template of the existing granules. eLP learning expands, reduces, merges, and deletes intervals derived from the words in a granule. The word w_k^i of an existing granule γ^i is deleted if it becomes unusual in the data stream.

5.2.2 Model structure

An eLP classifier is evolved on the fly. Interval-based rules, R^i, $i = 1, ..., c$, are created and updated according to the stream of words. A rule is given by

$$R^i \quad : \text{If } ((l_1^i \leq z_1 \leq L_1^i) \text{ and } \ldots (l_k^i \leq z_k \leq L_k^i) \text{ and } \ldots (l_m^i \leq z_m \leq L_m^i)) \text{ then } (y^i \text{ is } C^i) \tag{5.8}$$

in which

$$z_k^i = \arg_{j(w_k^i \equiv x_j^{[h]})}(\mathbf{x}^{[h]}), \;\; k = 1, ..., m. \tag{5.9}$$

We repeated Eq. (5.5) as Eq. (5.9) for reading convenience. $z_k^i \in \{0, 1, \ldots, n\}$; n is the length of $\mathbf{x}^{[h]}$. In words, z_k^i is the argument j that indicates match between w_k^i of $\mathbf{w}^i$ and $x_j^{[h]}$ of $\mathbf{x}^{[h]}$. If there is no correspondence between a specific w_k^i and all words of $\mathbf{x}^{[h]}$, then $z_k^i = 0$. Each word w_k^i of $\mathbf{w}^i$ receives at least one corresponding value z_k^i, which is inferred from the indices of the words of the input vector $\mathbf{x}^{[h]}$. The process of searching for a w_k^i in $\mathbf{x}^{[h]}$ is carried out m times in a single time step h.

The interval $[l_k^i, L_k^i]$ is formed by a beginning point l_k^i, which is the argument j of $\mathbf{x}^{[h]}$ that reports the first possible position for the word w_k^i in the message $\mathbf{x}^{[h]}$. Similarly, L_k^i is the argument j of $\mathbf{x}^{[h]}$ that reports the last possible position for w_k^i in $\mathbf{x}^{[h]}$. Each word w_k^i is assigned to a numeric interval $[l_k^i, L_k^i]$ so that $l_k^i, L_k^i \in \{1, 2, \ldots, n\}$, and $l_k^i \leq L_k^i$. A special case is when the word w_k^i can only be found in a unique (j-th) position of $\mathbf{x}^{[h]}$. In this case, $l_k^i = L_k^i = j$.

Consider that all words w_k^i, $k = 1, ..., m$, of γ^i were found in $\mathbf{x}^{[h]}$. Then, γ^i is a candidate to accommodate $\mathbf{x}^{[h]}$. Consider also a binary number, s_k^i, so that $s_k^i = 1$ if $z_k^i \in [l_k^i, L_k^i]$; and $s_k^i = 0$ if $z_k^i \notin [l_k^i, L_k^i]$. Moreover, let

$$\mathfrak{z}^i = \frac{1}{m} \sum_{k=1}^{m} s_k^i; \tag{5.10}$$

$\mathfrak{z}^i \in [0, 1]$. The most active rule for $\mathbf{x}^{[h]}$ is R^{i*}, in which

$$i^* = \arg_i(\max\{\mathfrak{z}^1, ..., \mathfrak{z}^c\}). \tag{5.11}$$

Only the most active rule R^{i*} is useful to provide an estimated class C^{i*}, and for a further adaptation step.

5.3 Online Learning from Word Streams

An eLP rule-based model is updated if: (i) a new granule γ^{c+1} is created; (ii) an existing granule γ^i is activated by an input instance; (iii) similar granules are merged; (iv) words w_k^i match input instances and becomes a feature; and (v) inactive granules are deleted.

5.3.1 Creating granules

In case none of the existing granules is sufficiently activated by an instance $\mathbf{x}^{[h]}$, a new granule γ^{c+1} is generated. The condition to be verified for granule creation is

$$(\max\{\mathfrak{z}^1, ..., \mathfrak{z}^c\}) \leq \sigma. \tag{5.12}$$

in which $\sigma \in [1, n]$ is a threshold. The template of the new granule is formed by a vector of words,

$$\mathbf{w}^{c+1} = [x_1^{[h]}, x_2^{[h]}, \ldots, x_n^{[h]}]; \tag{5.13}$$

and a vector of intervals,

$$I^{[c+1]} = [[z_1^{[c+1]}, z_1^{[c+1]}], ..., [z_n^{[c+1]}, z_n^{[c+1]}]], \tag{5.14}$$

with $n = m$ initially. Thus, $w_k^{c+1} = x_k^{[h]}$, and $[l_k^{[c+1]}, L_k^{[c+1]}] = [z_k^{[c+1]}, z_k^{[c+1]}]$, $\forall k$, as in (5.9). The components of $I^{[c+1]}$ are initially pointwise (degenerated intervals).

5.3.2 Updating granules

In case the most active eLP granule, γ^{i*}, for an $\mathbf{x}^{[h]}$ satisfies Eq. (5.12). Thus, five updating possibilities take place. Figure 5.1 exemplifies the k-th entry of the interval vector of the template of γ^{i*}, namely, $I_k^{i*} = [l_k^{i*}, L_k^{i*}]$. The expansion region E^i of an interval vector I^i, based on the granularity ρ, has components

$$E_k^i = [L_k^i - \rho, l_k^i + \rho], k = 1, ..., m. \tag{5.15}$$

Consequently, Figure 5.1 shows five regions: inside the interval $[l_k^i, L_k^i]$ (region 3); and inside (regions 2 and 4) and outside (regions 1 and 5) the expansion region E_k^i defined in $\{1, 2, \ldots, n\}$.

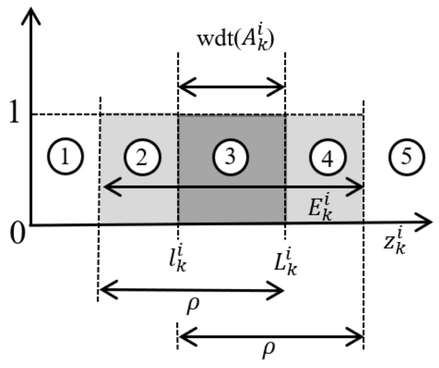


Figure 5.1: Example of interval $I_k^i = [l_k^i, L_k^i]$, and expansion region E_k^i

If a word of the granule γ^i, say w_k^i, is found on the current streaming instance $\mathbf{x}^{[h]}$, e.g., the j-th word $x_j^{[h]}$ matches w_k^i, then the index j (the position of the word $x_j^{[h]}$ in the sentence $\mathbf{x}^{[h]}$) is comprised in one of the five regions shown in Figure 5.1. We called such index z_k^i in (5.9). For j, or equivalently z_k^i, outside E_k^i (regions 1 and 5), a new granule γ^{c+1} is created (see Section 5.3.1). For z_k^i inside E_k^i (regions 2 and 4), an endpoint of I_k^i is updated (expanded) to include z_k^i, i.e.,

$$[l_k^i, L_k^i] = [z_k^i, L_k^i], \tag{5.16}$$

or

$$[l_k^i, L_k^i] = [l_k^i, z_k^i]. \tag{5.17}$$

If $z_k^i \subset [l_k^i, L_k^i]$ (region 3), no updated is needed.

5.3.3 Merging granules

Two granules, say γ^r and γ^s, with $r, s \in \{1, 2, \ldots, c\}$ are merged if $S(\gamma^r, \gamma^s) \geq \rho$. $S(.)$ is a similarity measure,

$$S(\gamma^s, \gamma^r) = \frac{\mathfrak{z}^r + \mathfrak{z}^s}{2}, \tag{5.18}$$

in which $\mathfrak{z}^r$ and $\mathfrak{z}^s$ are calculated using Eq. (5.10). $\mathfrak{z}^r$ gives the number of common words between $\mathbf{w}^r$ and $\mathbf{w}^s$ divided by $\mathbf{w}^r$. $\mathfrak{z}^s$ is obtained analogously. $S(.) \in [0, 1]$. The higher the value of $S(.)$, the more similar the pair of granules.

After merging, γ^r and γ^s are deleted. A new granule γ^{c+1} is generated. The latter is formed by the common words of γ^r and γ^s,

$$\mathbf{w}^{c+1} = \{w_k^r \mid w_k^r = w_k^s, \ \forall k \in \mathbf{w}^r\}, \tag{5.19}$$

and

$$\mathbf{I}_k^{c+1} = [min(l_k^r, l_k^s), max(L_k^r, L_k^s)], \quad k = 1, ..., m. \tag{5.20}$$

5.3.4 Deleting Attributes and Granules

An inactive granule, say γ^i, does not win competitions (5.11) in a number of time steps, say h_r. After h_r steps, the granule is deleted from the granular model. While h_r is a hyperparameter helpful to decide for deleting inactive granules γ^i – and therefore to reduce the amount of rules c of a granular model – attributes k of word $\mathbf{w}^i$ and interval $\mathbf{I}^i$ vectors can also be deleted along the learning process for a more compact antecedent part of rules.

To verify if w_k^i is either a typical word of γ^i or not we use the hit rate, v_k^i. Initially, a v_k^i is set to zero in the hit-rate vector $\mathbf{v}^i$. Whenever w_k^i is found in some $\mathbf{x}^{[h]}$, then one is added to the hit counter $\mathfrak{v}_k^i$. If v_k^i – which is given by $\mathfrak{v}_k^i$ divided by the the total number of matches w_k^i had in the past – exceeds a threshold value $\mathfrak{r}$, then the components w_k^i and I_k^i are respectively deleted from $\mathbf{w}^i$ and $\mathbf{I}^i$. Deleting attributes from templates helps to improve the matching procedure.

5.4 Learning algorithm

The eLP Learning algorithm summarises granular learning processing. The algorithm's parameter are initialised in Line 1. From the lines 2 to 34, the $\mathbf{x}^{[h]}$ message is read and absorbed by the knowledge basis. The first information granule is created in line 5. If it is not the first read message, the most active granule γ^* is identified in line 7. If this granule is enough activated, it is verified which case is the most appropriate according with Fig. 5.1 in lines 10-19 $\forall k$ words of the template $\mathbf{m}^i$. If γ^* is not enough activated, a new granule γ^{c+1} using $\mathbf{x}^{[h]}$ is created in line 21. From the lines 24-28, it is decided if it is necessary to create a new granule using the calculated intervals or adjust the most active granule to accommodate the new sample. The knowledge basis is updated through merging similar granules, updating the $\mathbf{m}^i\ \forall i$ and the granularity ρ, and deleting the inactive granules if needed in the lines 29-33.

Online Learning: evolving Log Parsing

```
1: Set initial granularity $\rho^{[0]}$; and deleting threshold $h_r$, $\mathfrak{r}$; Initialise $active = []$, $c = 0$;
2: For $h = 1, \ldots$
3:   Read $\mathbf{x}^{[h]}$;
4:   If $h = 1$ then
5:     Create granule $\gamma^1$: $\mathbf{w}^1$, $\mathbf{I}^1$; $c++$;
6:   Else   // $h = 2, \ldots$
7:     Identify $i^*$, Eq. (5.11);
8:     If $\mathfrak{z}^* > \rho$ then // Eq. (5.10)
9:       For $k = 1, \ldots, n$ do
10:        If $z_k^i < L_k^i - \rho$ then $[l_k^i, L_k^i] = [z_k^i, z_k^i]$   %case 1
11:        Else if $z_k^i \in [L_k^i - \rho, l_k^i]$ then
12:          $[l_k^i, L_k^i] = [z_k^i, L_k^i]$                      %case 2
13:        Else if $z_k^i \in [l_k^i, L_k^i]$      then
14:          No update needed                        %case 3
15:        Else if $z_k^i \in [L_k^i, l_k^i + \rho]$ then
16:          $[l_k^i, L_k^i] = [l_k^i, z_k^i]$                      %case 4
17:        Else $[l_k^i, L_k^i] = [z_k^i, z_k^i]$                  %case 5
18:        End if
19:      End for
20:    Else
21:      Create granule $\gamma^{c+1}$: $\mathbf{w}^{c+1}$, $\mathbf{I}^{c+1}$; $c++$;
22:    End if
23:  End if
24:  If $case1$ or $case5$, for any $k$ then
25:    Create granule $\gamma^{c+1}$: $\mathbf{w}^{c+1}$, $\mathbf{I}^{c+1}$; $c++$;
26:  Else if $case$ 2 or $case$ 4, for any $k$ then
27:    Update granule $\gamma^*$ to accommodate $\mathbf{x}^{[h]}$;
28:  End if
29:  If $h = \beta h_r$, $\beta = 1, \ldots$ then
30:    Merge similar $\gamma^r$ and $\gamma^s$ $\forall r, s \in \{1, \ldots, c\}$;
31:    Update $\mathbf{w}^i$ $\forall i$ if needed; Update granularity $\rho$;
32:    Delete inactive granules based on $h_r$;Delete local inactive attributes based on $\mathfrak{r}$;
33:  end if
34: end for
```

5.5 Summary

System logs are elementary expressions of language that are used by computational systems to communicate with human experts unidirectionally. The logs tell stories based on event occurrences. Any software expresses itself through a unique log language. The language can be explored by a general purpose syntax generator. This chapter introduces the evolving Log Parsing (eLP) method to extract interval granular rules from streams of words encountered in log files. eLP has identified templates (patterns in textual data) in an unsupervised one-shot incremental way.

Grammars evolve over time. The classes and classification model granularity are updated if it is needed. eLP is the first granular algorithm for log parsing based on words. In the context of system maintenance, eLP makes the syntactic analysis of the information as the first step to identify the semantic context of the communication.

Logs as a language concerns a viewpoint that allows extrapolation of the spontaneous communication between systems. The language provides meaning to syntactic analyses. As a grammar is not enough for communication, event occurrences enrich the vocabulary, thus providing the possibility of story-telling communication between systems in the future. Online pattern classification is achieved with an efficiency/accuracy of $(96.05 \pm 1.04)\%$. Additionally, we present an incremental interpretability index to evaluate the interval granular classifier on the fly.

Chapter 6

Methodology

This chapter describes the methodology used to evaluate the approached problems: (i) anomaly detection on the computing center functioning, and (ii) the parsing of the log language. Initially, the data used as input to Maintenance Systems is described in Sec. 6.1. Sec. 6.2 and 6.3 describe the dataset used to the AD and LP problems. Sec. 6.4 details the indexes and metrics used to evaluate the proposed methods. The chapter summary is shown in Sec. 6.5.

6.1 Log File

Logs consist of a sequence of lines formed by the timestamp and the log message content. Timestamps record the instant in which the log message was written. Each log file has a set of possible messages. The log production depends on the events occurred during a time window, reaching up to millions of lines per day for a single log file of a single service.

Usually, computing centers have a large amount of logs available from hundreds of running subsystems on their infrastructure. Since logs are often heterogeneous, unstructured, and textual, they significantly vary in format and semantics from system to system. For this reason, developing a general-purpose solution using logs is an extremely challenging and important task. The proposed solutions are designed to support any kind of logs.

6.1.1 StoRM Service Logs

Storage Resource Manager (SRM) is a service that provides the storage system used by INFN-CNAF. SRM aims to provide high performance to parallel file systems, such as the GPFS (the IBM General Parallel File System) and POSIX (Portable Operating System Interface), through a graphic interface to INFN-CNAF infrastructure users. SRM has modular architecture constitute by StoRM Front-end (FE), StoRM Back-end (BE), and databases (DB) [10]. The FE module manages user authentication, and store and load requests. The BE module is the main StoRM component regarding functionality. It executes all synchronous and asynchronous SRM operations, which allows interaction among grid elements. The SRM modules and their relations are shown in Fig. 6.1.

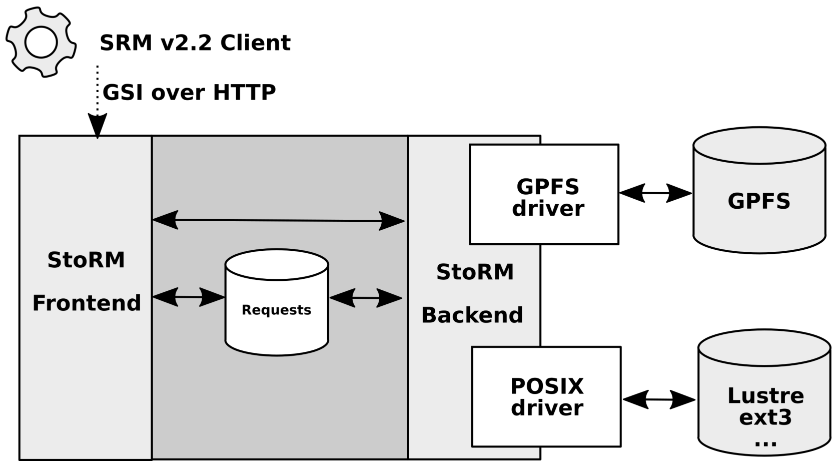


Figure 6.1: SRM architecture and relations among its main modules: Front-end, Back-end, and databases

All modules handle unstructured log data and two or more types of log writing mechanisms, such as scheduled and event-oriented mechanisms. There is, however, a common information to all log files is the timestamp of the log record writing. Based on such timestamps, it is possible to group log files in time windows to monitor the logging activity rate. As scheduled log data follows a regular logging activity, we focus on event-oriented data.

6.2 Anomaly Detection Problem

Logs are time series of textual sentences. They register event occurrences in the software code executed during the system runtime. Log Preprocessing (LP) involves three streams of data: (i) raw log stream, (ii) log activity stream extracted from (i), and (iii) the mean of logging activity stream using a sliding window. The computational complexity to process the log stream is $\mathcal{O}(n)$, being n the number of lines in a log file; the memory complexity is $\mathcal{O}(1)$, since each element of the stream is scanned only once.

Let $\tilde{\mathbf{u}}_j = [u_1 \ \dots \ u_i \ \dots \ u_n]$, $j \geqslant 1$ and $\tilde{\mathbf{u}}_j \in \mathbb{N}^n$, be a sequence of values that represent the log activity rate calculated using a sliding window $w_i = [\underline{w}_i \ \overline{w}_i]$ to u_i, with $|w_i| = k_1$, and $w_j = [\underline{w}_j \ \overline{w}_j]$ to $\tilde{\mathbf{u}}_j$, with $|w_j| = k_2$ with $k_2 = nk_1$. The logging activity is given as follows

$$u_i = \sum_{[\underline{w}_i, \overline{w}_i]} 1, \tag{6.1}$$

being u_i the number of log lines within the time window. The bounds of the time window, $\underline{w}_j$ and $\overline{w}_j$, are equal to u_1 and u_n. In addition, let μ_j be the mean of $\tilde{\mathbf{u}}_j$, thus

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} u_i,\ \ u_i \in [\underline{w}_i\ \ \overline{w}_i]. \tag{6.2}$$

Then, $[\mu_1\ \ \ldots\ \ \mu_j\ \ \ldots\ \ \mu_m]$ is a stream of means.

This measure gives an estimation of the intensity of the task execution of a running service. Any error, fault, or system failure is assumed to be followed by increased or decreased log productions. Other phenomena responsible to change log productions exist, but they are out of the scope of this thesis.

6.2.1 About the AD Dataset

It was monitored 20 different log files generated by StoRM service from May to August 2019. The data used in the experiments were sample during the data collection phase.

StoRM service generates a stream of time-indexed log records. Each log entry is composed by timestamp related to the writing moment, and the message itself. In this problem approach, we are focused on the logging activity, and the analysis of the message content is approached in the log parsing method.

We extract 5-metrics from the vector of logging activities, $\mathbf{x} = [x_1\ \ x_2\ \ x_3\ \ x_4\ \ x_5]$, which elements are the mean, $\overline{\mu}$, the standard deviation, $\sigma(\mu_j \forall j)$, the minimum value, $min(\mu_j \forall j)$, the maximum value, $max(\mu_j \forall j)$, and the maximum difference of the amplitude of two consecutive μ_j, in which $\mu_j \subset w_j$.

A vector $\mathbf{x}^{[h]}$ is associated to a class label $C = \{1, 2, 3, 4\}$ that, in turn, indicates the system behaviour. The true label, C, is available after an estimation, $\hat{C}$, is provided by the FBeM, eGNN or eGFC model. When available, the pair $(\mathbf{x}, C)^{[h]}$ is used by the learning algorithms for an updating step. The true label C is given by the Control Chart Self-learning method approached in next sub-section.

6.2.2 Log Self-learning Methodology

A control-chart-based approach is a self-learning strategy to label the log stream. Time-windows are classified according to the anomaly severity and standard deviations from the usual system behaviour [117]. A control chart is a time-series graphic used to monitor a process behavior, phenomenon or variable using the Central Limit Theorem. It is based on the mean $\mu(u)$ of a random variable u that follows a normal distribution [118].

Tagging Strategy

Let an instance μ_j be labelled using a control chart. The mean of μ is

$$\overline{\mu} = \frac{1}{m}\sum_{j=1}^{m} \mu_j. \tag{6.3}$$

The k-th upper and lower horizontal lines in relation to $\overline{\mu}$ refer to the k-th standard deviation,

$$\sigma_k(\mu) = k * \sqrt{\frac{1}{m}\sum_{j=1}^{m}(\overline{\mu} - \mu_j)^2}. \tag{6.4}$$

If $\mu_j \subset [\overline{\mu} - k\ \sigma(\mu_j \forall j),\ \overline{\mu} + k\ \sigma(\mu_j \forall j)]$ and $\mu_j \not\subset [\overline{\mu} - (k-1)\ \sigma(\mu_j \forall j),\ \overline{\mu} + (k-1)\ \sigma(\mu_j \forall j)]$, then, for $k = 1$, the instance is tagged as 'Class 1', which means normal system operation. If $k = 2, 3, 4$, then μ_j is tagged as 'Class 2', 'Class 3' and 'Class 4', respectively, see Fig. 6.2. These mean low, medium, and high-severity anomaly. The greater the value of k, the greater the severity of the anomalous behavior. The probability that μ_j is within each class is 67%, 28%, 4.7%, and 0.3%, respectively. The described online anomaly detection problem is unbalanced, and the labels are weak.

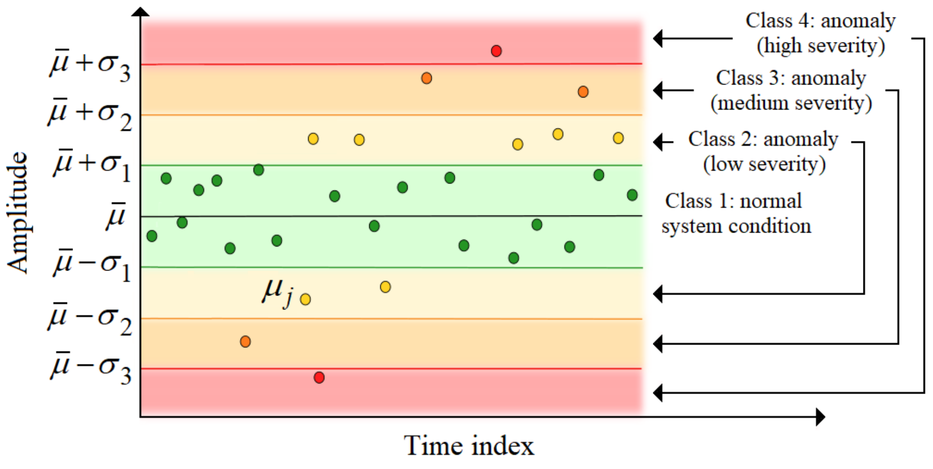

$\overline{\mu}+\sigma_3$
$\overline{\mu}+\sigma_2$
$\overline{\mu}+\sigma_1$
$\overline{\mu}$
$\overline{\mu}-\sigma_1$
$\overline{\mu}-\sigma_2$
$\overline{\mu}-\sigma_3$
Amplitude
μ_j
Time index
Class 4: anomaly (high severity)
Class 3: anomaly (medium severity)
Class 2: anomaly (low severity)
Class 1: normal system condition

Figure 6.2: Control chart procedure to tag log data

6.3 Log Parsing Problem

eLP is an evolving unsupervised solution based on a granular fuzzy-set classifier. The method analyses continuously the log data stream in a non-stationary environment without any previous knowledge. In this scenario, it is necessary to be able to extract the evolving structure that represents the input textual data. In addiction, fuzzy-set methods are capable to deal with the stochastic nature of logs.

The proposed solution not just extracts the syntactic rules $\mathcal{G}$ of $\mathcal{L}(\mathcal{G})$, but also evolves them over time, admitting the natural evolution of the language that can be both intentional, with the software upgrade, and occasional, with non-frequent code execution. To improve the accuracy of the experiments, the granular paradigm improve the non-linearity of class boundaries, in which templates can have one or more granules, and each granules can be associated to event occurrences of the system posteriorly. The data input is a textual phrase without any previous classification.

The data-set is formed by logs periods generated by the StorM Service randomly sampled. No privileged information are used to improve the quality of the results, being this solution entirely general-purpose. The log data is directly used by eLP, and no data pre-processing is needed.

6.4 Evaluation Indexes

In order to evaluate the performance of the classifiers, it is used four different categories of indexes, approaching the main aspects of the problem: (i) model interpretability, (ii) classification accuracy, (iii) model compactness, and (iv) execution time. The former are calculated using recursive relations. The indices are formulated to approach evolving systems fed by a data input stream $(x, \hat{y})^{[h]}$, with $h = 1, 2, 3, \ldots$, in which x is a vector of features and $\hat{y}$, if exists, is the related class label given by a self-learning method. In general, the indices are evolving, produced in each round of the algorithm.

6.4.1 Interpretability

Since the model interpretability is one of the main advantage of fuzzy systems, the modelling can be made considering this issue. Generally, a problem modelling concerns in to formulate the most accurate model given a data-set. However, to achieve higher accuracy, the learning algorithm increases the model complexity penalising its interpretability. For this reason, the focus has to change from an accuracy maximisation to a max-min optimization problem, i.e., maximising the accuracy taking in account the minimisation of the increasing of the model complexity. The interpretability-accuracy trade-off is the focus of the Explainable AI and the fuzzy logic system (FLS) is the best paradigm to confront it.

The interpretability concept is often merged into similar terms as readability, comprehensibility, understandability, intelligibility, transparency, among others. The main problem of the term definition is the intrinsic difficulty to identify what makes something understandable. Considering knowledge-based systems, more specifically fuzzy rule-based systems, the interpretability indices can evaluate the ability of the model to be understandable by human beings through a numerical value, helping to ranking the rules according to their understandability internally in a method, or externally comparing different approaches.

Since this concept is subjective, a huge variety of indices were described in [119]. This sub-section proposes a version of the Nauck's index to measure the interpretability of the evolving granular fuzzy systems (eGFS) method family [120]. The index was originally designed to approach general FLS, but here it is reinterpreted to contemplate eGFS. Differently of the FLS approaches, eGFS have an evolving fuzzy rule set, in which all rules have the exact same format and type of antecedents. In this sense, the adapted Nauck's index is defined as

$$I_{Nauck} = comp \ . \ \overline{cov} \ . \ part \tag{6.5}$$

in which $comp$ is the model complexity, $\overline{cov}$ is the averaged coverage of the fuzzy granules, and $\overline{part}$

is a penalty component related to the number of active granules of the current entry $x^{[h]}$ of the input data stream, with $h = 1,\ 2,\ \ldots$. Considering the data stream domain X that is partitioned in c sub-domains, one for each created granule, in which i-th rule R^i is defined into the X_i domain, and $X = \{X_1\ \cup\ X_2\ \cup \cdots \cup\ X_c\}$ corresponding to $R = \{R^1,\ R^2,\ \ldots\ R^c\}$. The *comp* component is given by

$$comp = \frac{m}{n.c},$$

in which m is the number of possible classes (m=4) and n is the sum of the number of variables of the rule $i\ \forall i$. In this work, it is the same number of features in the input data vector (n=5 in AD problem, n is variable in LP problem). The model complexity for all evolving granular methods to AD problem is

$$comp(new) = \frac{4}{5c^{[h]}}. \tag{6.6}$$

The term $c^{[h]}$ is the number of rules at the time-instant h. Considering the modelling, the AD method complexity is in $(0,\ \frac{4}{5}]$, being inversely dependent of the current number of granules.

The evolving method coverage is given by the average of all rule coverage after the update of the knowledge basis

$$\overline{cov}(new) = \frac{\sum_{i=1}^{c^{[h]}} cov_i}{c^{[h]}}. \tag{6.7}$$

The $\overline{cov}$ is in $[0, 1]$, in which the complete coverage is indicated by $\overline{cov} = 1$. The evolving coverage cov_i of R^i is defined as

$$cov_i(new) = \frac{h-1}{h} cov_i(old) + \frac{1}{hN_i} \hat{h_i}(x^{[h]}), \tag{6.8}$$

in which N_i is the number of the data stream entries absorbed by R^i, i.e., $|X_i|$. The $\hat{h_i}(x^{[h]})$ factor is defined as

$$\hat{h_i}(x^{[h]}) = \begin{cases} h_i(x^{[h]}), & \text{if } 0 \leq h_i(x^{[h]}) \leq 1 \\ \frac{p^{[h]} - h_i(x^{[h]})}{p^{[h]} - 1}, & \text{otherwise.} \end{cases}$$

The term $h_i(x^{[h]})$ is normalised if its value is not well-behaved, being given by

$$h_i(x^{[h]}) = \sum_{k=1}^{p^{[h]}} \mu_k(x^{[h]}), \tag{6.9}$$

in which $p^{[h]}$ is the number of all activated rules at h, $1 \geq p^{[h]} \geq c$, c is the total of granules, and $\mu_k(x^{[h]})$ the degree of membership of $x^{[h]}$ in regard with the k-th activated rule.

The *part* of $x^{[h]}$ is defined as

$$part(new) = \frac{1}{(p^{[h]} - 1)}, \tag{6.10}$$

with $p^{[h]} \geq 2$ or the entry $x^{[h]}$ do not produce a penalty term. The shorter the *part* term, the greater the penalty since it is a multiplicative term. Originally, $p^{[h]}$ is p_j, the number of possible values of each feature j of the input data x. In this work, the feature space is not granular, and because that, we reinterpret this metric to $p^{[h]}$ as the number of activated granules considering $x^{[h]}$.

All components of I_{Nauck} are evolving as well as the index. The index approaches the main issues correlated to eGFS classifier family as number of granules and possible classes, variables in the antecedents, rule coverage, and number of activated granules by each entry. Indirectly, questions as non-linearity of the class frontier are approached in the *comp* term.

Granular Interpretability Index

A simplified instantaneous system's interpretability of an information granulation is

$$I^{[h]} = \frac{1}{n^{[h]} * c^{[h]} * \mathcal{E}^{[h]}}, \tag{6.11}$$

in which n and c are respectively the number of the used features to define a granule (the granule dimensionality), and the number of granules in the rules set at the h instant. The $\mathcal{E}$ equilibrium, $\mathcal{E} \in [0, 1]$, is

$$\mathcal{E}^{[h]} = \frac{1}{n} \sum_{f=1}^{n} W_{(max),f} - W_{(min),f}, \tag{6.12}$$

in which

$$W_{(max),f} = max(wdt([l_f^1, L_f^1]), \ldots, wdt([l_f^c, L_f^c])), \tag{6.13}$$
$$W_{(min),f} = min(wdt([l_f^1, L_f^1]), \ldots, wdt([l_f^c, L_f^c])). \tag{6.14}$$

$W_{(min),f}$ and $W_{(max),f}$ changes according with the eGFS method. The present definition is based on the IBeM method. Higher the value of $\mathcal{I}$, the bigger is the interpretability. A high readability and understandability is related with the system association with a small amount of concise rules balanced along all their dimensions.

To apply the $I^{[h]}$ index, it is needed to have granules with the same dimensionality as in Chapter 4. An $\mathcal{E}^{[h]}$ adaptation is possible to problems that deal with set of granules with different dimensionality:

$$\mathcal{E}^{[h]} = \frac{1}{c}\sum_{i=1}^{c} W_{(max),i} - W_{(min),i}, \tag{6.15}$$

in which

$$W_{(max),i} = max(wdt([l_1^i, L_1^i]), \dots, wdt([l_n^i, L_n^i])), \tag{6.16}$$

$$W_{(min),i} = min(wdt([l_1^i, L_1^i]), \dots, wdt([l_n^i, L_n^i])). \tag{6.17}$$

In this case, $\mathcal{E}^{[h]}$ is balanced along the granules set instead of the granule dimensions. This version is used to evaluate the log parsing problem in Chapter 5.

6.4.2 Accuracy

Classification accuracy, $Acc \in [0, 1]$, is obtained from

$$Acc(new) = \frac{h-1}{h}\, Acc(old) + \frac{1}{h}\, \tau, \tag{6.18}$$

in which $\tau = 1$ if $\hat{C}^{[h]} = C^{[h]}$ (right estimation); $\hat{C}^{[h]}$ and $C^{[h]}$ are the estimate and actual classes. Otherwise, $\tau = 0$ (wrong class estimation). Considering the unsupervised log parsing problem, there is no comparative value to identify if the classification is correct, the verification is the comparison between the complete match between the sample and the granule during absorbing process, and the sample-knowledge basis match at h, calculated by Eq. (5.10), and it is referenced as template effectiveness.

6.4.3 Compactness

The average number of granules or rules over time, c_{avg}, is a measure of model concision. It is computed as

$$c_{avg}(new) = \frac{h-1}{h}\, c_{avg}(old) + \frac{1}{h}\, c^{[h]}. \tag{6.19}$$

6.4.4 Execution Time

The execution time is also given using a DELL Latitude 5490 64-bit quad-core (Intel Core i58250U, 8GB RAM).

6.5 Summary

This chapter presents the dataset design and how to generate it from a raw log dataset in order to repeat the experiments of both approached problems. The metrics describe how to evaluate and compare the methods, detailing the used metrics as model interpretability, model accuracy, model compactness, and time execution. It is proposed two interpretability indexes to evaluate the online fuzzy-granular methods, monitoring the interpretability performance of the granular method on-the-fly, considering two granularity scenario, i.e., when (i) the granule set has only one granule dimensionality, and (ii) multiple dimensionality. Chapter 4 approaches the (i) scenario, and Chapter 5, the (ii) scenario. The model interpretability is used to evaluate LP problem.

Chapter 7

Experimental Results

This chapter presents the computational experiments to evaluate the evolving methods discussed in this thesis. First we give a comparative analyses regarding the fuzzy-granular methods FBeM, eGNN and eGFC toward the supervised anomaly detection problem of Chapter 4. Second, we present the particular performance of the proposed solution – eLP – to the log parsing problem described in Chapter 5. eLP approaches the problem in an unsupervised way. The methodology described in Chapter 6 supports the experiments.

7.1 Anomaly Detection

We compare FBeM, eGNN and eGFC for system-driven anomaly classification in a computing center using attributes extracted from its logging activity. In this approach, we do not assume any prior knowledge about the data to conduct the experiments. Classification models are evolved from scratch based on the data stream.

7.1.1 Performance Comparison

We compare the evolving fuzzy and neuro-fuzzy granular classifiers in terms of accuracy, model compactness, and processing time.

Experiment 1

In the first experiment, we set h_r within 75 and 125, $\eta = 3$, and the granularity $\rho^{[0]}$ around 0.3 and 0.7. The ρ level evolves during the learning process. These meta-parameters allowed the generation of eGFC, FBeM and eGNN models with about 10 to 18 rules – a reasonably compact structure.

Table 7.1 shows the comparison among eGFC, FBeM, and eGNN. The results are averaged over 5 runs of each method for each window length, and shuffled datasets, as described in Chapter 6. In other words, for a single window length, a unique dataset is produced, but its instances are shuffled differently 5 times. The window lengths are $w = \{60, 30, 15, 5\}$-minutes, in which 60 minutes is the best settings for all tested methods. Each dataset consists of 1,436 instances with 5 attributes each one plus the associated class label. The data supervision was automatically generated by the self-learning control chart approach. Four classes are possible: classes 1 to 4 mean 'normal system operation', 'low severity', 'medium severity', and 'high severity' anomaly, respectively.

Table 7.1 shows that the classification performance of all algorithms, in which all method accuracies increase progressively with the increasing of the time windows, keeping a similar model compactness, i.e., $10 - 18$ rules. The low-pass filter effect strongly impacts the FBeM accuracy, which improves about 18% from the worst to the best case. The best FBeM scenario ($85.64\% \pm 3.69$) is, however, comparable to the worst eGNN and eGFC scenarios (85.00 ± 2.39 and 81.97 ± 5.02).

Table 7.1: eGNN Performance in Multi-class Classification of System Anomalies with 99% of confidence

FBeM			
Time Window (min)	*Acc*(%)	# Rules	Time (s)
60	85.64 ± 3.69	12.63 ± 3.44	0.18 ± 0.02
30	75.58 ± 5.91	10.64 ± 1.89	0.19 ± 0.06
15	66.99 ± 3.63	9.94 ± 1.50	0.18 ± 0.02
5	67.27 ± 4.26	13.88 ± 1.17	0.19 ± 0.02
eGNN			
Time Window (min)	*Acc*(%)	# Rules	Time (s)
60	96.17 ± 0.78	10.35 ± 1.32	0.18 ± 0.02
30	90.56 ± 2.70	13.79 ± 2.26	0.22 ± 0.04
15	86.28 ± 5.29	13.68 ± 1.31	0.21 ± 0.02
5	85.00 ± 2.39	12.61 ± 0.64	0.22 ± 0.02
eGFC			
Lenght (min)	*Acc*(%)	# Rules	Time (s)
60	92.48 ± 1.21	13.42 ± 4.32	0.36 ± 0.10
30	88.01 ± 4.96	17.22 ± 2.59	0.45 ± 0.04
15	82.57 ± 5.64	18.13 ± 4.79	0.49 ± 0.10
5	81.97 ± 5.02	16.09 ± 2.51	0.41 ± 0.06

The neuro-fuzzy classifier reached an accuracy of 96.17 ± 0.78 using 60-minute sliding time windows, and about 10 rules. Compared to FBeM, eGNN uses a higher number of parameters per local model and, therefore, the decision boundaries among classes are more flexible to non-linearities. Nevertheless, eGNN requires a greater number of instances to update a larger number of parameters.

Table 7.1 also shows that analysis of larger 60-minute windows facilitates the eGFC learning algorithm to detect and classify spatial-temporal patterns, which represent the anomaly classes. Notice that using the compact model structure (13.42 fuzzy rules on average along the learning steps), the eGFC model produced an average accuracy of $92.48\% \pm 1.21$.

The results using $\{30, 15, 5\}$-minutes dataset are not clearly statistically different, since there is an intersection of confidence intervals. However, the best result achieved by all methods are associated to the 60-minutes time-windows, in which the method accuracies vary in $[81.95, 89.33]$, $[91.27, 93.64]$, and $[95.31, 96.95]$ for FBeM, eGFC, and eGNN, respectively. In this way, the ranking of the best performance is obvious. In this scenario, the methods are clearly statistically different, since the confidence intervals do not have any intersection. From the worst to the best method's performance, the raking is FBeM, eGFC, and eGNN as shown in Table 7.2.

Table 7.2: Performance of the Algorithms in Multi-class Classification of System Anomalies with 99% of confidence, ranked in ascending order.

Algorithm	$Acc(\%)$	# Rules	Time (s)
FBeM	85.64 ± 3.69	12.63 ± 3.44	0.18 ± 0.02
eGFC	92.48 ± 1.21	13.42 ± 4.32	0.36 ± 0.10
eGNN	96.17 ± 0.78	10.35 ± 1.32	0.18 ± 0.02

The CPU time was measured in a quad-core i7-8550U with 1.80GHz and 8GB of RAM, being similar to FBeM and eGNN scenarios. As we can see, the execution time is twice bigger to eGFC than to the other methods, and eGNN has the best time execution and the most compact rules set in addition to the best accuracy.

Experiment 2

The second experiment consists in seeking the best meta-parameters, ρ and h_r, of FBeM and eGNN, and, hence, their best performance without upper constraint on model structure. The two methods were chosen because FBeM is the simplest method implementation (and the worst method performance), and eGNN is the best method performance in the present comparison. Figures 7.1 and 7.2 show the classification accuracy versus number of FBeM and eGNN rules, respectively, for a variety of combinations of the ρ and h_r meta-parameters[1]. Each point within the gray zone (area of possibility) of the graphics is the average of 5 runs of the respective algorithm on shuffled dataset considering a meta-parameter settings.

Notice in Fig. 7.1 that FBeM improves its performance if we allow its rule structure to grow. FBeM reaches a 96.1% anomaly classification accuracy using small initial granularities, such as $\rho = 0.1$, and about 40 rules, whereas the best average performance of eGNN is, in fact, 96.2% (as shown in the first experiment) using $\rho = 0.7$ and about 10 rules. Scenarios with few rules may mean that the dynamic of model adaptation is too fast in relation to the dynamic behaviour of the target system. In other words, a small amount of granules are dragged to the current scenario, which may be detrimental to the memory of past episodes.

Comparing Figs. 7.1 and 7.2, the relative higher slope of the gray zone of FBeM in relation to eGNN is explained by the higher number of parameters per eGNN local model. Fuzzy neurons and synaptic weights provide eGNN granules with a higher level of local nonlinear ability such that it requires less local models to represent the whole. However, if the eGNN structure grows rapidly, an additional number of parameters is added to the model such that their convergence/maturation to realistic values demand more data instances.

[1] $\rho^{[0]} = [0.1, 0.3, 0.5, 0.7, 0.9]$ and $h_r = [50, 75, 100, 125]$, with 60-minutes time-windows.

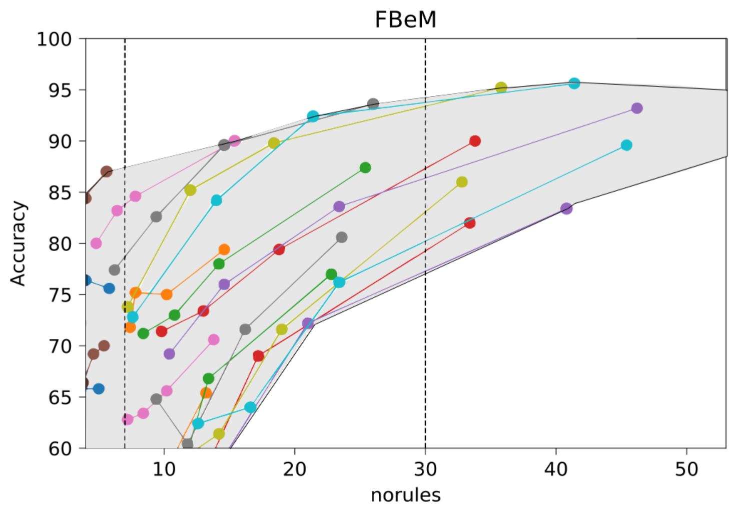


Figure 7.1: FBeM Classification accuracy versus amount of rules for an array of meta-parameters

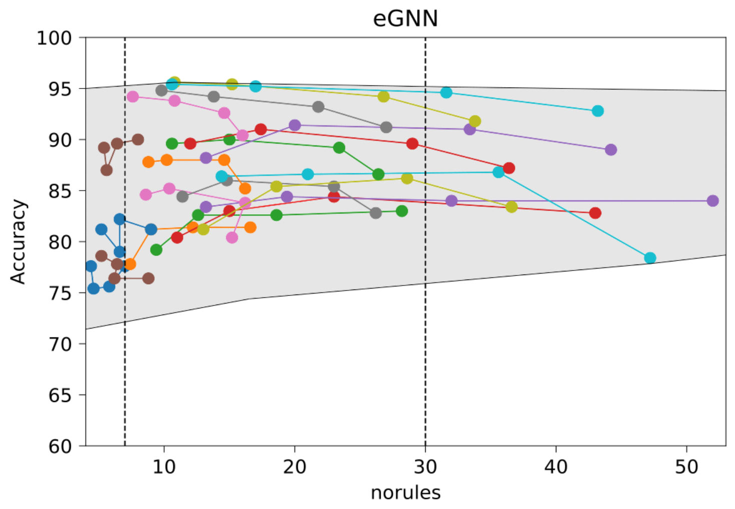
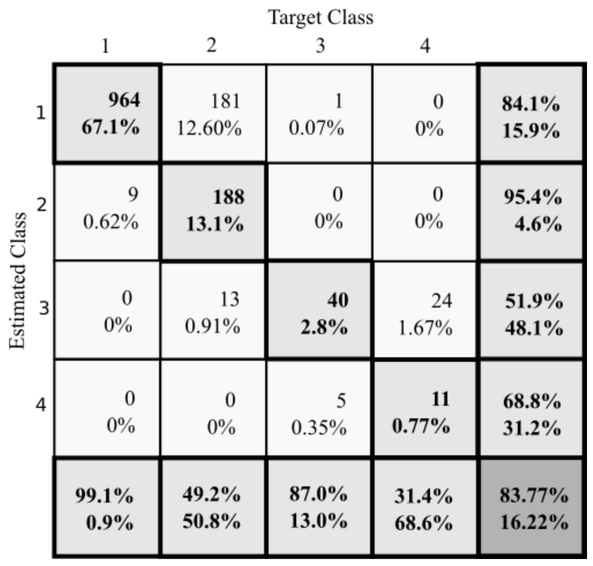


Figure 7.2: eGNN classification accuracy versus amount of rules for a variety of meta-parameters

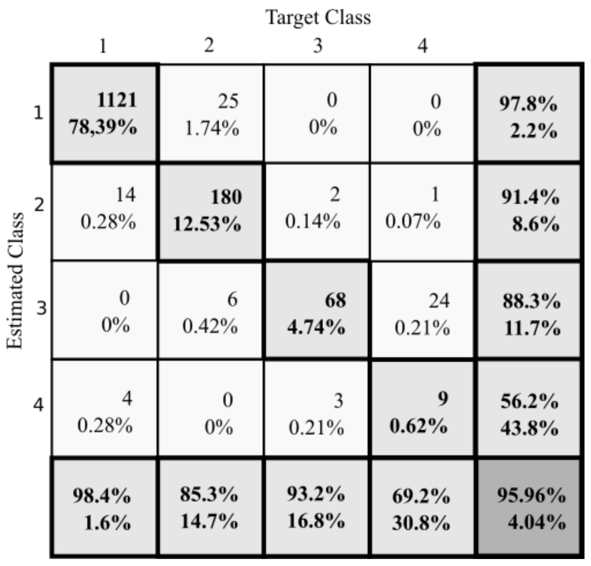
Figures 7.3, 7.4 and 7.5[2] show an example of confusion matrix for a 83.77%-accuracy of FBeM, a 95.96%-accuracy of eGNN, and 94.2%-accuracy of eGFC. 'Classes 3-4' (medium and high-severity anomalies) are relatively harder to be identified. This difficult is associated with the imbalanced-class system's characteristic, in which the 'Classes 3-4' are the rarest ones, representing less than 5% of the samples in the dataset. Notice that confusion happens in FBeM, in general, in the neighbourhood of a target class, whereas often eGNN estimates an instance as being 'Class 4' when it is 'Class 1'. This indicates that a local model may still be learning values for itself. FBeM slightly outperformed eGNN regarding detection of the most critical 'high severity' class. eGNN's overall accuracy is superior. In Figure 7.5, it is possible to notice the eGFC's misclassification happens strongly also in the neighbourhood of a target class, which means that if a higher number of streaming samples are further available, the eGFC model may improve its accuracy by fine tuning its decision boundaries. 'Class 1' (normal operation) and 'Class 2' (low severity) are those responsible for a larger reduction of the overall accuracy for the eGFC method.

Figure 7.3: Example of FBeM confusion matrix in anomaly classification.



Estimated Class \ Target Class	1	2	3	4	
1	**964** **67.1%**	181 12.60%	1 0.07%	0 0%	**84.1%** **15.9%**
2	9 0.62%	**188** **13.1%**	0 0%	0 0%	**95.4%** **4.6%**
3	0 0%	13 0.91%	**40** **2.8%**	24 1.67%	**51.9%** **48.1%**
4	0 0%	0 0%	5 0.35%	**11** **0.77%**	**68.8%** **31.2%**
	99.1% **0.9%**	**49.2%** **50.8%**	**87.0%** **13.0%**	**31.4%** **68.6%**	**83.77%** **16.22%**

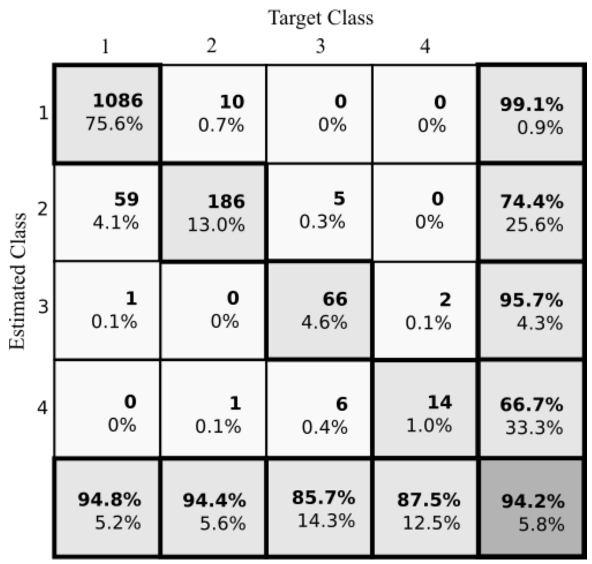
Figure 7.4: Example of eGNN confusion matrix in anomaly classification.



Estimated Class \ Target Class	1	2	3	4	
1	**1121** **78,39%**	25 1.74%	0 0%	0 0%	**97.8%** **2.2%**
2	14 0.28%	**180** **12.53%**	2 0.14%	1 0.07%	**91.4%** **8.6%**
3	0 0%	6 0.42%	**68** **4.74%**	24 0.21%	**88.3%** **11.7%**
4	4 0.28%	0 0%	3 0.21%	**9** **0.62%**	**56.2%** **43.8%**
	98.4% **1.6%**	**85.3%** **14.7%**	**93.2%** **16.8%**	**69.2%** **30.8%**	**95.96%** **4.04%**

Figure 7.6 gives a typical example of evolution of the ρ-level, accuracy, and number of eGFC rules. Four dimensions of the final Gaussian granules, at $h = 1,436$, are also shown. Notice that data from 'Class 2' and 'Class 3' (low and medium-severity anomalies) spread in a nonlinear way over the data space. These classes require more than one granule and rule to be represented, whereas the remaining classes are generally given in a common region. 'Class-4' data (high-severity anomaly) belong to a more compact region than the data of other classes and, therefore, are represented by a single granule. A higher number of granules to represent a class, in general, provides larger non-linearity of decision boundaries, which improves classification accuracy. Figure 7.7 emphasise the multi-dimensional ellipsoidal geometry of eGFC granules. This contour lines representation confirm

[2]To understand how to read the confusion matrix, see the appendix A



Target Class / Estimated Class

	1	2	3	4	
1	**1086** 75.6%	**10** 0.7%	**0** 0%	**0** 0%	**99.1%** 0.9%
2	**59** 4.1%	**186** 13.0%	**5** 0.3%	**0** 0%	**74.4%** 25.6%
3	**1** 0.1%	**0** 0%	**66** 4.6%	**2** 0.1%	**95.7%** 4.3%
4	**0** 0%	**1** 0.1%	**6** 0.4%	**14** 1.0%	**66.7%** 33.3%
	94.8% 5.2%	**94.4%** 5.6%	**85.7%** 14.3%	**87.5%** 12.5%	**94.2%** 5.8%

Figure 7.5: Example of eFGC confusion matrix in anomaly classification

the spreading characteristic specially related to 'Class-2' data, showing large overlapping regions of 'Class-1' and 'Class-2'.

To sum up, using the evolving fuzzy classification methodology and the sliding window control-chart-based approach, a system maintenance can accurately identify time windows that require further analysis in terms of text content[3]. The evolving methodology supports data and information mining to assist predictive maintenance. Overall system status can be modelled as Gaussian granules of the log activity rate, and status changing can be noticed visually from the control charts. In addition, the stream of system status can be used to diagnose the context of the current log status, and to predict the next status. Since methods preserve their accuracies in non-stationary environment, the approaches have shown to be reliable solutions to the system maintenance problem.

7.1.2 eGNN Aggregation Layer

In this subsection, we present the results of the extension of the eGNN experiments, considering of the variation of the aggregation function used as Aggregation Layer in the neuron, keeping the meta-parameters of the best result of the previous eGNN experiments. The meta-parameters are described in Table 7.4, using the aggregation function {T-norms, S-norms, Averaging Aggregation, T-S Min-Max Aggregation}, in which T-norms = {Min, Product, Lukasiewicz}, S-norms ={Probabilistic Sum, Lukasiewicz } in a full combinatorial experiment.

Table 7.4 shows that the best results are associated to the T-norm aggregation functions. The used T-norms apply *min*, *product*, and the maximum value between 0 and a value derived from the sample to aggregate the information. This type of aggregation function is conservative, dealing with the

[3]Approached in the Section 7.2

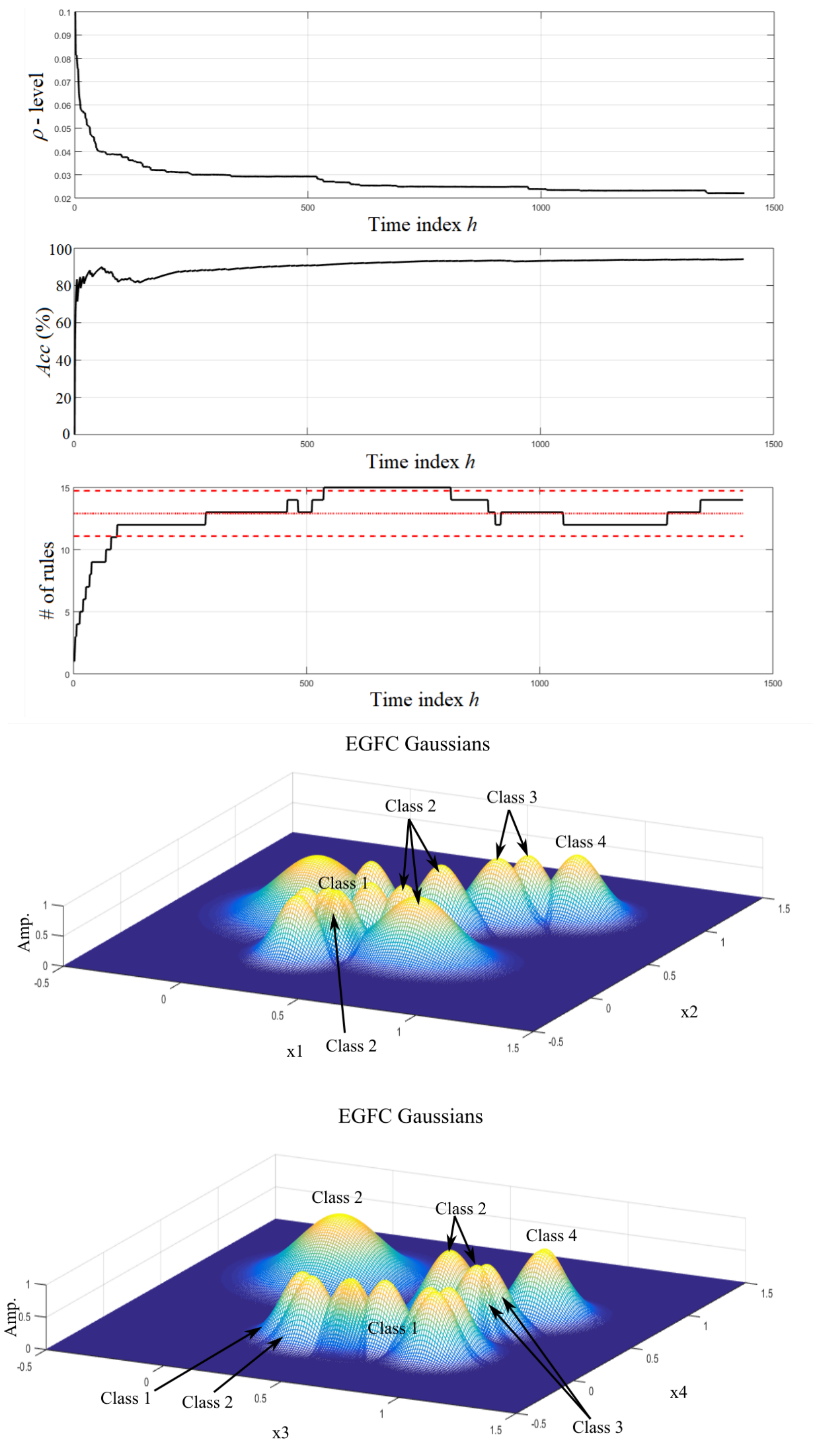


Figure 7.6: The time evolution of the evolving factors: granulation ρ and number of rules, and the model accuracy until the convergence at the 3 first graphics. At the last 2 graphics the eGFC Gaussian classes.

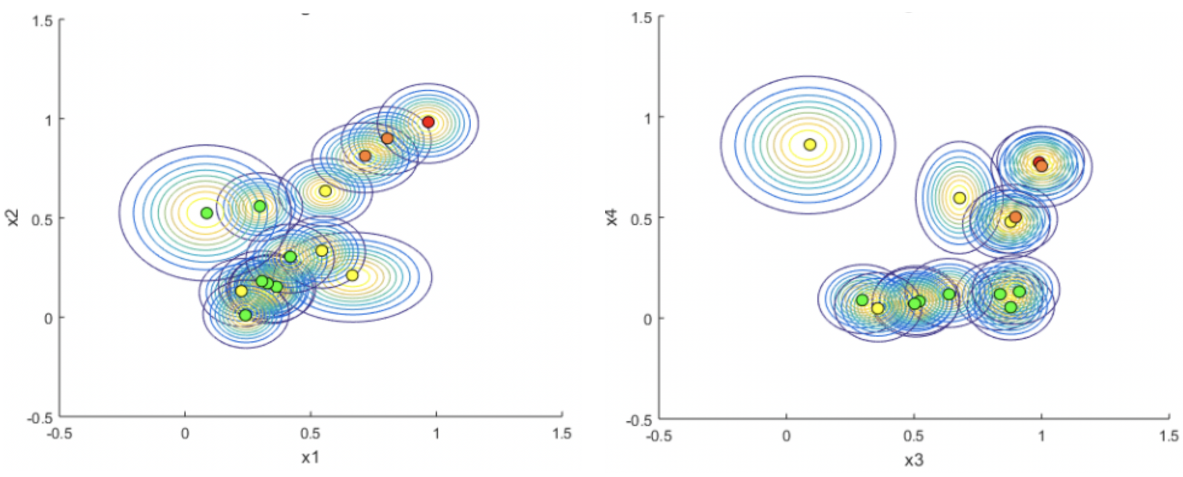


Figure 7.7: The multi-dimensional ellipsoidal geometry of eGFC granules using the first four attributes of the log stream. The colours of the centers refer to the control chat of Fig. 6.2, i.e., green: normal system condition; yellow, orange and red: low, medium and high anomaly severity

inferior limit as the information summarization. The opposite strategy is used by the S-norm functions, that priories the superior limit, taking a bigger value to represent the information summarization in a progressive vision of the scenario. Since AD problem is extremely time-dependent, data-stream driven system's analysis is favoured by T-norm. The other functions abstract the information using a value between a T-norm and S-norm values. The best aggregation function choice is intimately related to the characteristics of the data behaviour, specially how fast the scenario can change.

Table 7.3: eGNN Performance in Multi-class Classification of System Anomalies with 99% of confidence

	Neuron	ρ	h_r	$Acc(\%)$	# Rules
	min	0.3	100	96.06 ± 0.426	11.09 ± 1.15
T-Norm	product	0.5	125	96.45 ± 0.58	11.18 ± 1.03
	Lukasiewiscz	0.5	125	96.55 ± 0.80	11.02 ± 0.80
	max	0.5	75	87.34 ± 3.33	3.65 ± 0.18
S-Norm	probabilistic sum	0.7	100	77.97 ± 7.55	3.67 ± 0.08
	Lukasiewiscz	0.7	125	76.57 ± 30.60	3.73 ± 0.09
	Averaging Aggr.	0.3	100	77.63 ± 8.03	5.74 ± 0.65
	T-S Max-Min Aggr.	0.7	125	88.23 ± 5.75	3.64 ± 0.07

Table 7.3 shows the best result of the experiment done varying the aggregation function used as neuron in eGNN, considering the used h_r and ρ. All the T-norm function had a widely superior performance than the other aggregation functions. The experiments 1 and 2 were done using the T-norm min. All these aggregation functions are described in Chapter 4.

7.2 Log Parsing

We evaluate the eLP performance for system-driven parsing using textual logs to extract templates using the formal grammar concept defined in Chapter 5 through a fuzzy granular algorithm. The evolving parsing was evaluated in terms of effectiveness[4], model compactness[5], model interpretability, and the processing time. The experiments were done using 5 different files of $3,000$-log messages, and a file with $15,000$-log messages as input, shuffled 5 times each one.

We vary the experimental meta-parameters according with Table 7.4, considering 64 different settings, in a total of 320 executed experiments[6] for each used dataset. The ρ level and the rules set evolves during the learning process. The meta-parameters are (i) η - the growth rate[7], (ii) σ - the minimum value to activate a granule[8], (iii) $\rho^{[0]}$ - the initial value of ρ, (iv) h_r - the time interval between two executions of the basic operations[9], and (v) hit_{min} - the minimum hit rate to a word to be kept in a template. The meta-parameters allow the generation of eLP models with about 9 to 13 rules considering Table 7.5 – a reasonably compact structure to the parsing problem within the tested h_r.

Table 7.4: The eLP Experimental Meta-parameters' range

	Standard Settings
η	18
σ	$[0.5, 0.7]$
$\rho^{[0]}$	$[0.3, 0.5, 0.7, 0.9]$
h_r	$[75, 100, 125, 150]$
hit_{min}	$[0.5, 0.7]$

The eLP analysis permits a double application: (i) to provide a system snapshot based on the log content that can be used as a representation of the system state, and (ii) to extract progressively the formal grammar that defines a log language. In (ii), it is needed to store the deleted granules with high effectiveness level in order to increment continuously the known grammar. The eLP application improves the system knowledge basis, being used to identify its current state. The state study can generate a state machine of a system[10].

Table 7.5 summarises the results of the 320 experiments through the average and the confidence interval considering 99% of all experiments, evaluating the template effectiveness (Eff. %)[11], the model interpretability (Interp.)[12], the model compactness (# Rules)[13], the average of the number of tem-

[4]The effectiveness was used instead of accuracy because it is an unsupervised solution.
[5]The average of number of rules.
[6]Considering the 5 shuffled-executed experiment of each settings.
[7]Defined in Eq. 3.19
[8]It is used to choose the granule candidates to absorb a new sample.
[9]As to update ρ, to delete inactive granules, to merge similar granules, and to clean the templates.
[10]The implementation is out of the scope of this thesis.
[11]Given by f_D in Eq. (5.10).
[12]Given by $I^{[h]}$ in Eq. (6.11).
[13]Given by c_avg in Eq. (6.19).

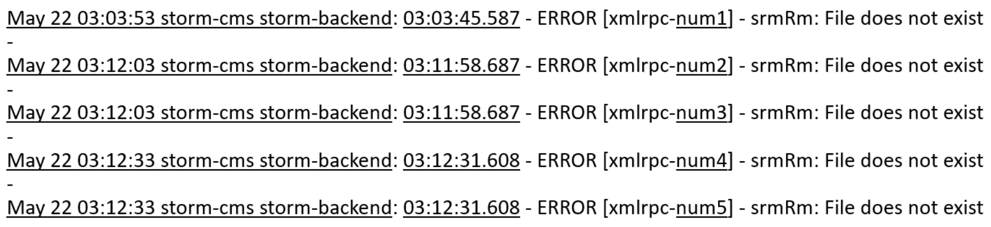
plate's words in the rules set (# Features), and the processing time in seconds.

Table 7.5: The eLP Experimental Results

3000-samples					
Dataset	Eff. (%)	Interp.	# Rules	# Features	Time(s)
1	0.94 ± 0.02	0.07 ± 0.05	9.54 ± 5.48	7.29 ± 1.70	2.68 ± 1.19
2	0.93 ± 0.02	0.04 ± 0.03	11.07 ± 5.36	8.55 ± 1.36	2.87 ± 1.13
3	0.93 ± 0.02	0.03 ± 0.02	12.45 ± 5.76	8.91 ± 1.65	3.48 ± 1.37
4	0.92 ± 0.02	0.04 ± 0.03	11.60 ± 5.41	8.81 ± 1.56	3.46 ± 1.25
5	0.94 ± 0.02	0.03 ± 0.02	11.38 ± 5.44	8.33 ± 1.52	2.94 ± 1.16
15000-samples					
6	0.96 ± 0.01	0.04 ± 0.03	7.31 ± 2.37	7.69 ± 1.48	53.46 ± 22.09

In Table 7.5, the results are coherent for all dataset, including the 15,000-dataset. As it is possible to see, the processing time it is not linear, since each h_r messages, it is necessary to update the rules set. If it is considered a larger h_r, the rules set has the possibility to grow until the h_r value, increasing the research time of the best rule candidate with the rules set's size. In addiction, the growth of the rules set implies a decrease in the model interpretability by definition[14].

Before to continue the analysis, it is important to make some considerations about logs. There are different levels of difficulties to identify logs. By way of illustration, considers the following logs set in Fig. 7.8. The samples are well behaved, in which the parameters[15] have a fixed size, being easily associated with the correct granule, if it is in the rules set.



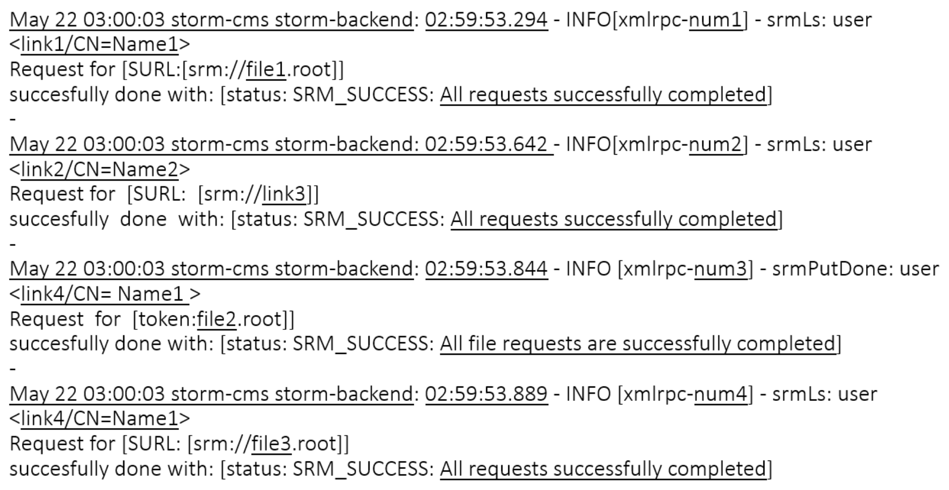
```
May 22 03:03:53 storm-cms storm-backend: 03:03:45.587 - ERROR [xmlrpc-num1] - srmRm: File does not exist
-
May 22 03:12:03 storm-cms storm-backend: 03:11:58.687 - ERROR [xmlrpc-num2] - srmRm: File does not exist
-
May 22 03:12:03 storm-cms storm-backend: 03:11:58.687 - ERROR [xmlrpc-num3] - srmRm: File does not exist
-
May 22 03:12:33 storm-cms storm-backend: 03:12:31.608 - ERROR [xmlrpc-num4] - srmRm: File does not exist
-
May 22 03:12:33 storm-cms storm-backend: 03:12:31.608 - ERROR [xmlrpc-num5] - srmRm: File does not exist
```

Figure 7.8: Example of logs in which the template is extracted easily.

On the other hand, Figures 7.9 and 7.10 show a different style of samples, in which the parameters have a variable size. If we consider only the logs in Fig. 7.9, we could conclude that the expression 'SRM_SUCCESS' is part of template as well as 'successfully done with'. But if if we consider that the sample in Fig. 7.10 was generated by the same previous template, these expressions would be converted to parameters.

In general, it is not possible to be sure about which template generated a log sample, because we hardly have access to classified logs. To consider all these samples as made for the same template or

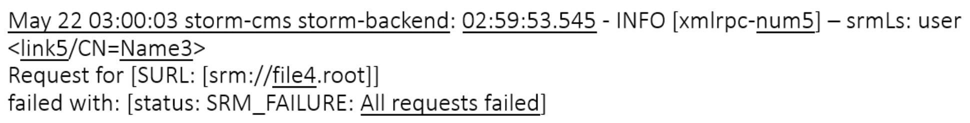
[14]See Chapter 6.
[15]Parameters are the underlined words.



```
May 22 03:00:03 storm-cms storm-backend: 02:59:53.294 - INFO[xmlrpc-num1] - srmLs: user
<link1/CN=Name1>
Request for [SURL:[srm://file1.root]]
succesfully done with: [status: SRM_SUCCESS: All requests successfully completed]
-
May 22 03:00:03 storm-cms storm-backend: 02:59:53.642 - INFO[xmlrpc-num2] - srmLs: user
<link2/CN=Name2>
Request for  [SURL:  [srm://link3]]
succesfully  done  with: [status: SRM_SUCCESS: All requests successfully completed]
-
May 22 03:00:03 storm-cms storm-backend: 02:59:53.844 - INFO [xmlrpc-num3] - srmPutDone: user
<link4/CN= Name1 >
Request  for  [token:file2.root]]
succesfully done with: [status: SRM_SUCCESS: All file requests are successfully completed]
-
May 22 03:00:03 storm-cms storm-backend: 02:59:53.889 - INFO [xmlrpc-num4] - srmLs: user
<link4/CN=Name1>
Request for [SURL: [srm://file3.root]]
succesfully done with: [status: SRM_SUCCESS: All requests successfully completed]
```

Figure 7.9: Example of logs in which extracting the template can be tricky.

not is a settings choice. The greater the information granule, the more chances of mixing logs from different templates. However, the smaller the granule, the greater the chance of duplicating identical templates, decreasing the model interpretability. For each log file, there is the best choice of the method settings in order to optimise the log clustering, and this choice probably evolves over time because of the non-stationary nature of the problem.



```
May 22 03:00:03 storm-cms storm-backend: 02:59:53.545 - INFO [xmlrpc-num5] – srmLs: user
<link5/CN=Name3>
Request for [SURL: [srm://file4.root]]
failed with: [status: SRM_FAILURE: All requests failed]
```

Figure 7.10: Log message that could harm the template extraction in Fig. 7.9.

Figure 7.11 (a)-(c) shows the template effectiveness in function of the size of the rule base and the parameters h_r and ρ. Similarly, Figure 7.11 (d)-(f) shows the model interpretability. In Fig. 7.11(a), we notice that the template effectiveness increases when more rules are developed and maintained in the model. Considering the pigeonhole principle[16], each rule (hole) must receive data samples (pigeons) to exist. Thus, the higher the number of rules associated to different classes, the greater the precision of the model in providing an adequate class to the input data. For a fewer number of rules to accommodate data, the corresponding granules are larger and more generic. The length of templates shrinks to fit in the granules, see Fig. 7.12 (c).

Figures 7.11-a and 7.11-d expound this relationship between template effectiveness and model interpretability, in which the growth of the template effectiveness impacts negatively in the model interpretability. As we see in Fig. 7.12-c, the template size tends to be more generic with the reduction of the rules set, and more specific with its rise. The same reasoning is seen is seen in Fig. 7.11-d with the

[16]If n items are put into m containers, with $n > m$, then at least one container must contain more than one item.

decreasing of the interpretability, since it is inversely proportional to the template size[17]. Figure 7.11-b reveals that the effectiveness decreases with the increasing of h_r. Since h_r is the interval between rules set updating, the greater the time interval, the greater the probability of classification considering more specific templates and, consequently, the probability of duplication of granules increases. Templates more specific implies in bigger templates (Fig. 7.12-b), and consequently in less interpretable models, as shown in Fig. 7.11-e, compatible with the observable increasing rules with the h_r growth in Fig. 7.12-e.

Fig. 7.11-c shows the relationship between effectiveness and interpretability considering the granularity ρ, in which it is possible to note a local maximum around $\rho^{[0]} = 0.7$. A completely mirrored behavior is seen in Fig. 7.11-f and 7.11-c, highlighting the conceptual trade-off between effectiveness and interpretability, reinforced by Fig. 7.12-f, in which is observed the inversely proportional relationship between them. A twin behavior between effectiveness and template's size is illustrated in Fig. 7.12-a, and number of rules in Fig. 7.12-d, more rules mean a more specific set of rules and, consequently, bigger templates.

7.3 Summary

In this chapter, we detailed the experimental results of the approached problems. The AD problem was evaluated using 3 different methods (FBeM, eGNN, and eGFC), considering ρ, h_r, and time-window as meta-parameters in two full-design experiments. The first experiment evaluates the three methods, and the second deepened the analysis for two of them (FBeM and eGNN). The best method was eGNN in this analysis, 96.17% of accuracy in anomaly detection to the system maintenance problem. Extra experiments were done to optimise the neuron choice of eGNN, in which the best neuron class was the T-norm aggregation functions (min, product and Lukasiewiscz T-norms).

The eLP method is proposed to approach the parsing problem in system maintenance approaches, being evaluated using 6 datasets in total, 5 of $3,000$-samples, and 1 of $15,000$-samples, considering ρ, h_r, σ, and hit_{min} as meta-parameters in a full-design experiment. The eLP method is capable to extract templates with $0.92 - 0.94\%$ of effectiveness without prior knowledge about the system or its logs. It was possible to get insights about the problem structure and its relationship in regard with effectiveness-interpretability, and their dynamics with the average template's size and the size of the rules set. At each time instant, the current rules set provides an abstract of the h_r interval related to the logs diversity and how they are distributed in the identified templates. As h_r is a meta-parameter, eLP could be used as a diagnostic tool for anomalous event that lasts h_r-samples.

[17]Number of Features.

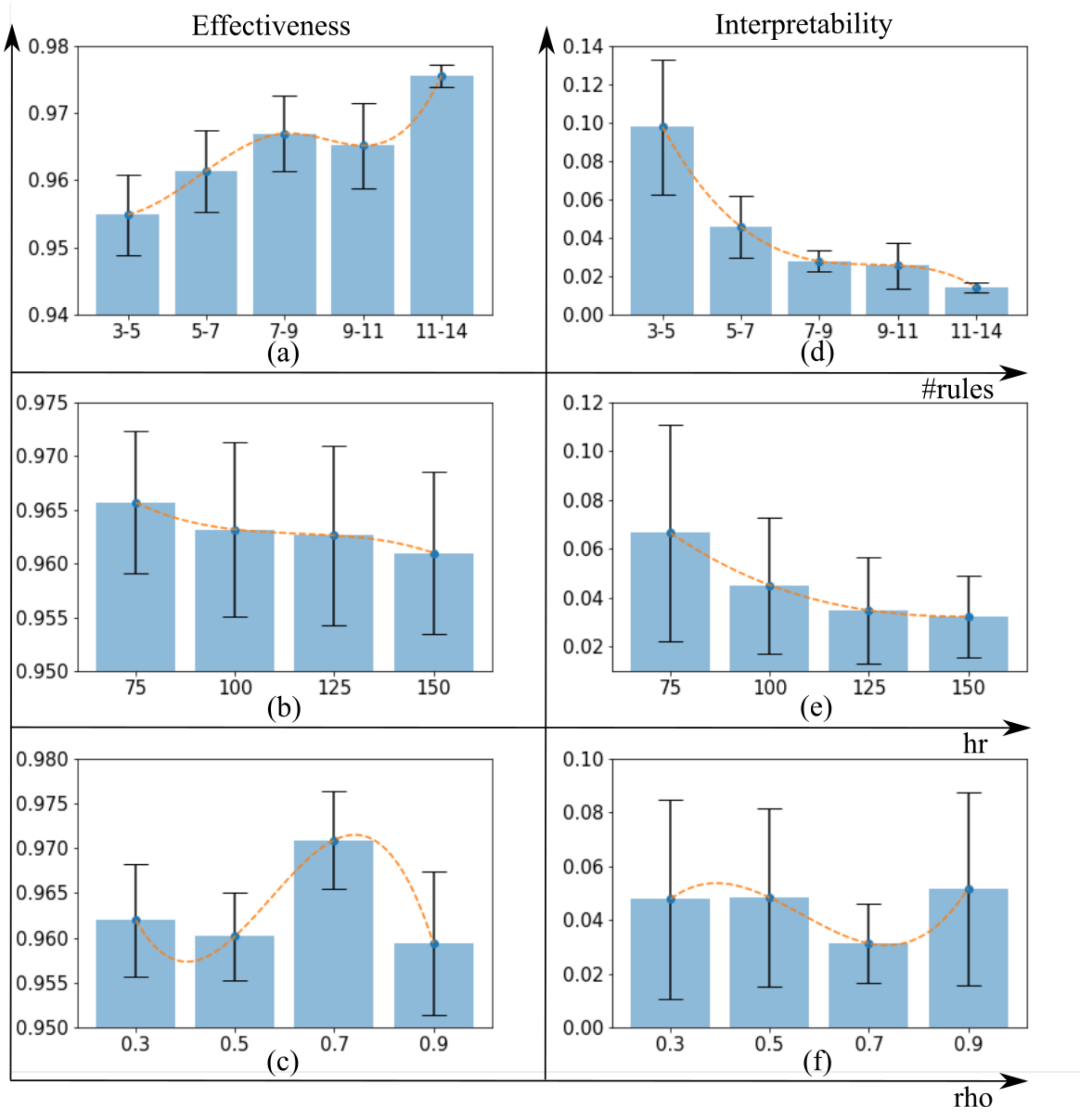


Figure 7.11: The template effectiveness (a-b-c) and the model interpretability (d-e-f) are deeply compare in regards to the number of rules (a-d), h_r (b-e), and ρ (c-f).

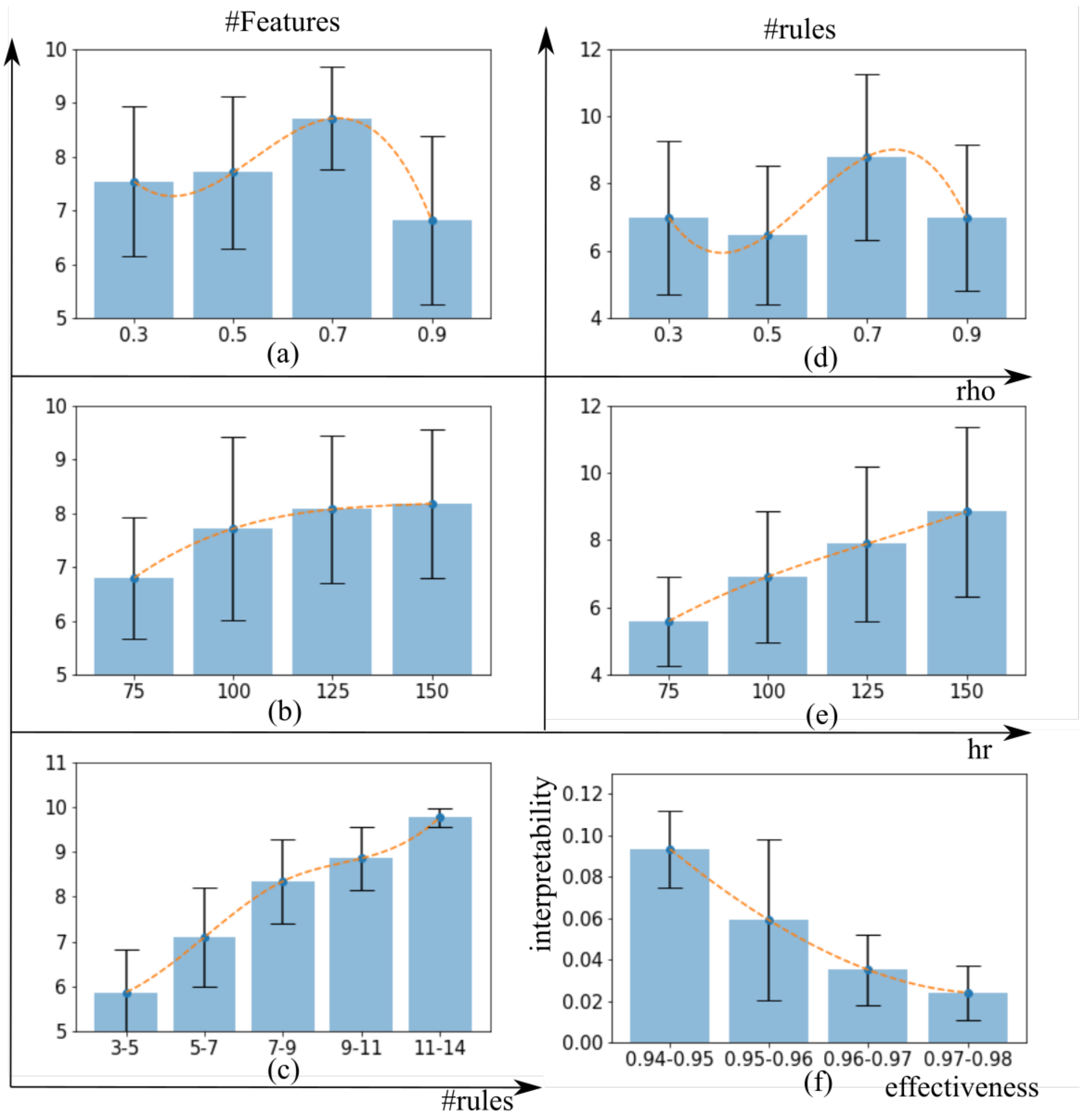


Figure 7.12: The template's size (a-b-c) and the number of rules (d-e) are deeply compare in regards to the number of ρ (a-d), h_r (b-e), and number of rules (c). In (f), it is highlighted the relationship between the model interpretability-template effectiveness.

Chapter 8

Non-stationary System Maintenance

System maintenance is a semantics problem by nature[1]. The communication between computing systems and humans is still lacking, however it tends to move towards automation. Current computing systems have a limited capacity to report anomalous occurrences before failure incidents compromise the system functioning. Maintenance solutions are based on the translation of the detailed system speech (smaller information granules) to a higher level of reasoning or abstraction (larger information granules), which are appropriate to human understanding. It is unlikely that an expert can easily understand system logs (a lengthy, unstructured, and complicated sequence of numbers and words) just reading log files. Logs are usually used as feedstock to deductive and investigative studies that infer how the detected problems happen.

Logs are codified data in which information about the existence of an anomaly is not explicit, or easily noticed by monitoring systems. The advantage of using and storing logs is related to their availability, abundance and low cost – in spite of their lack of clarity, i.e., the relevant information is indirect, redundant, uncertain, and imprecise. Additionally, labels are not associated to a large amount of logs for supervision and machine learning. Labelling logs manually is infeasible. Maintenance systems are strongly dependent on the ability of the system programmers to identify a sequence of execution points that may entail a failure in a real scenario.

Being fluent in analysing a given set of log files is a diligent and dedicated task, which requires constant upgrading, updating, and ability to handle unprecedented samples. The ever changing patterns of anomalies may significantly impair the fluency of a human expert or algorithm. In addition, understanding the semantics of a log file is an ability that can not be easily transferred to other types of log files and to other people – as the structure of different files is not persistent. Therefore, ah-hoc log-based solutions to system maintenance are not strategic, since each log file asks for a specific approach. Ad-hoc methods are computationally very expensive. In other words, ad-hoc methods, as part of a monitoring software, are on the edge of obsolescence since they strongly hamper code reuse for similar solutions.

The present study aims to maintain the QoS of a data center modelled as a non-stationary system.

[1]See Chapter 2 for more details.

The system is modularized in interconnected software components (SwCs)[2]. Models evolve on-the-fly from a data stream extracted from log files to track the non-stationarities of the actual system. Model evolution is based on incremental learning procedures. Parametric and structural model evolution is of utmost importance to accurately track the changing dynamics of the actual system. In particular, structural model evolution is an important emphasis in the present study as it adds not only to offline-trained computational-intelligence models, but also to adaptive models, which very often refer to models supplied with a mechanism to update some parameters only [Astrom book Adaptive Control] [Survey Igor]. Since system maintenance is a practical and non-stationary real-world problem, evolving the structure of granular neural networks [60] and fuzzy rule-based models [59, 61] in a fast one-scan-through-the-data fashion is the hypothesis of this thesis to keep the anomaly detection accuracy and model fluency over time.

Non-stationary systems have an evolving set of possible states $\mathcal{S}$ in a partially-known scenario. In this sense, it is necessary a permanent effort to rectify $\mathcal{S}$ and the probability of each state transition. The intense mutability of the system requires automated approaches to update the solution space without compromising the reliability of the system maintenance.

In this modelling, the system maintenance $\mathcal{M}$ is a stochastic process defined by the extraction of 5 features from a log file f [3] of a software component c

$$\mathcal{M}(s_c, \mathbf{s}, \mathbf{s}_n, f, c) \tag{8.1}$$

in which, s_c is the current state of c, $\mathbf{s}$ is the vector of the last $|\mathbf{s}|$ current states representing the behavior trend, and $\mathbf{s}_n$ is a vector of the most $|\mathbf{s}_n|$ probable next states of c using a forecasting approach, with $s_c, \mathbf{s}, \mathbf{s}_n \in \mathcal{S}$. All the system states are generated by the same method using f of c.

The system maintenance is scalable since a data center can be modularized in SwCs according to convenience as in Fig. 8.1. The log analyses identify s_c of c_i, and its state history can be used to map the inter-dependency among the other monitored modules, using Bayesian networks [121, 122]. The data center is represented as a graph $\mathcal{D} = (\mathcal{C}, \mathcal{R})$ comprising

- $\mathcal{C} = \{c_1, \ldots, c_i, \ldots, c_n\}$ is a set of SwCs, with $|\mathcal{C}| = n$. Each c_i is a SwC with a current state s_c, a state history $\mathbf{s}$, a state prognosis $\mathbf{s}_n$, the type of the used log file f and, the respective system maintenance $\mathcal{M}_i$ (Eq. 8.1).
- $\mathcal{R} \subseteq \{\{c_i, c_j\} \in \mathcal{C} \text{ and } i \neq j\}$ is a set of dependency relations between 2 monitored SwCs, with $|\mathcal{R}| = m$. Each edge has associated a pair of probabilities $(p_{ij},\ p_{ji})$. The p_{ij} represents the probability of a state change in c_i generate a state change in c_j, and a similar reasoning can be done to p_{ji}.

Since c_i can change over time, $\mathcal{D}$ must to be modelled as an evolving system. In nutshell, a naivel solution identifies the current state of $\mathcal{D}$ through a combination of all s_c of $c_i \in \mathcal{C}$, if it makes sense for any particular application. Anyway, this modelling designs a system landscape to the monitoring phase, helping in the inference of the global impact of event occurrences.

[2]Note that SwCs are also non-stationary systems.
[3]See Chapter 4 for more details.

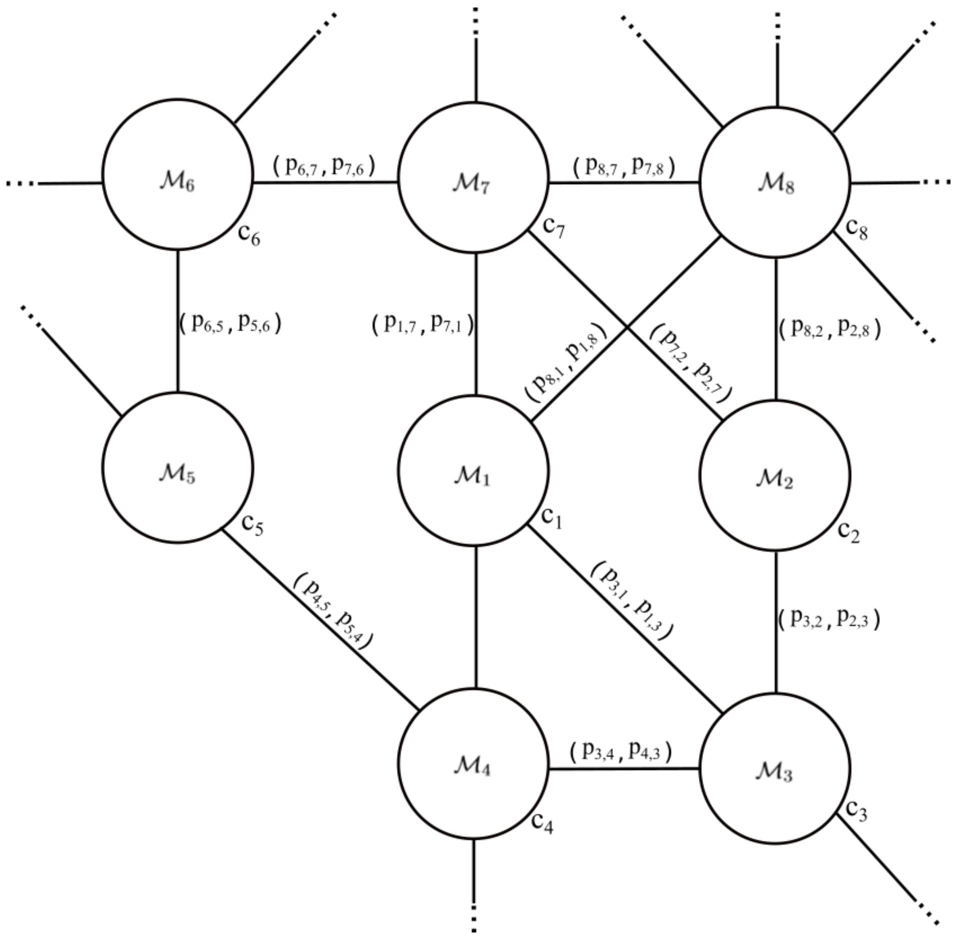


Figure 8.1: Here, it is shown a piece of the graph $\mathcal{D}$, representing the data center maintenance system, in which c_i is a SwC with an associated $\mathcal{M}_i$ system maintenance. The $p_{i,j}$ is the probability of a c_i state change induces a c_j state change, and it is represented as edge weights in the graph.

The functioning dependency of $SwCs$ are associated to sharing limited resources as memory access, data processing, data sharing among different threads, network connection, among others. High level of system coupling can compromise the performance of the whole system. To individualise the order of the task execution, log files strategically register execution points[4] to facilitate the event tracking during the diagnostic.

Considering an execution point as an evidence of the running of functions, procedures, and activities, each log line registers a specific execution point, helping to track back the activities done during the system runtime. The execution flow of a code depends on the value of inner variables, and in turn, they often depend on event occurrences inputted by users or external computing systems. Therefore, to identify which code snippets were executed during a time interval is the first step to understand the occurrence of computing problems recognised by the human perception.

Typically, the system behavior can be modelled through log-based approaches, since log registration is a widespread good practice of software development in industry and scientific communities. Because of the high availability, logs are one of the most important data type to provide a coherent story of the code execution. Normally, a computer system saves an entire narrative in log files, to monitor and debug tasks.

The present solutions aim to identify the current, future and all possible states of a system based exclusively in log mining, in order to monitor the system stability and forecast damages, failures, and anomalies. In addition, maintenance solutions must to be computationally cheap[5], since it is a support activity. Due to the scenario limitation, log-based system maintenance is a hard computing problem, demanding a creative effort to build general-purpose solutions that optimise the analyses reliability.

In this chapter, Health System Models are shown in Sec. 8.1. Section 8.2 describes the naive and granular solutions. The conclusion and next steps are presented in Sec. 8.3.

8.1 Health System Model

To learn s_c, $\mathbf{s_n}$, and $\mathcal{S}$ of SwCs is essential to keep the QoS of a data center in the range stipulated by standards, industry, protocols, or communities. This is the basis to monitor the system health, supporting diagnostic and prognostic activities. Usually, $\mathcal{S}$ are defined by specialised entities as experts, scientific communities, or ad-hoc supervised datasets. In the absence of this information, the knowledge can be generated through case studies, which will be used to model the system behavior. In general, this project phase is very expensive in time and resources, but fundamental to implement an accurate $\mathcal{M}$ design.

Because of the constant demand to update S, identifying S manually is a counterproductive task in non-stationary systems. In order to overcome this technical inconvenience, the proposed methodology focuses on identifying autonomously the state parameters of $\mathcal{M}$ based on a design of system anomalies during Δt in online mode.

[4]The execution points are defined by the programmer through the system code.
[5]Related to memory and CPU time consumption.

In this work, the Health System Model (HSM) monitors the system status through $\mathcal{M}$ (Eq. 8.1), being organized in three levels of abstraction based on information granules [6]:

- anomaly: it is the lowest level of information abstraction. It is based on an anomaly detection classification of pre-processed unsupervised data. The piece of logs are classified in one of 4 classes of anomaly severity.
- state: it is the intermediate level of information abstraction, being formed by a composition of anomalies. The state gives an interpretation of how the anomalies are linked and how they can contributed to the system instability.
- behavior: it is the highest level of information abstraction, composed by a combination of states. The behavior gives a wide landscape in a longer period of time, identifying event occurrences perceived by humans.

The anomaly scheme is a mechanism to translate the lowest-level system speech to the human comprehension. The abstraction level is based on the Granular Computing described in Chapter 3. The information level scheme can used as the basis of system diagnosis and prognosis.

In Chapter 4, the anomaly detection is approached by fuzzy granular classifiers using the $\tilde{\mathbf{u}}_j$ stream [7] to design the 4-class anomaly distribution. Here, it is proposed a methodology in which anomaly compositions are candidates for possible states. The advantage of this proposal is the high applicability, maintainability, and portability to computing systems.

In stationary systems, it is possible to set up $\mathcal{S}$ in offline mode, identifying the probability of state transitions using Bayesian networks [121, 122] to manage the system status. The same approach can be adapted to a non-stationary system considering a finite number of possible states $\mathcal{S}$ during the time interval Δt

$$\mathcal{S}_{\Delta t} = \{s^1, s^2, \ldots, s^k\} \tag{8.2}$$

with $|\mathcal{S}_{\Delta t}| = k$, the number of possible states during Δt. Ad-hoc solutions are the most frequent used approaches to model states since this problem is awfully system-dependent.

Contrary to this trend, the present work addresses this issue through a general-purpose method. Overall, general-purpose approaches can be less precises than ad-hoc versions, since it does not take in advance the system particularities. But even so, the gain of this methodology goes beyond an possible greater efficiency, inasmuch as it makes possible to maintain complex systems.

Further more, the system state is time-dependent as the human mood. Event occurrences can change s_c as life events can change a person mood. According to the person, the pattern changing of the mood can define a particular behavior associated to precise diagnosis as *bipolar disorder*, for example. Similarly, event occurrences can produce a sequence of system states that can characterise a specific system behavior associated to a determined diagnosis as, e.g., *server is down.*

[6] See Chapter 4.

[7] $\tilde{\mathbf{u}}_j$: logging activity metric extracted by log files.

The j-th system behavior, $\mathcal{B}^j$, is modelled as a process defined by a sequence of system states $\mathbf{s}^j \in \mathcal{S}_{\Delta t}$

$$\mathcal{B}^j = \mathbf{s}^j. \tag{8.3}$$

The $\tilde{\mathbf{u}}_j$ stream, taken during Δt to generate $\mathbf{s}^j$, follows the *Normal* distribution $\mathcal{N}(\mu^j, {\sigma^j}^2)$, with $j > 1$, Δt is the time-interval between $j-1$ and j behavior changes, μ^j and σ^j are the mean and the standard variation of $\mathcal{N}(\mu^j, {\sigma^j}^2)$. The technique used to extract $\tilde{\mathbf{u}}_j$ forces its probability distribution $\mathcal{P}(\tilde{\mathbf{u}}_j)$ to $\mathcal{N}(\mu, \sigma^2)$, and it is used to define the behavior pattern during the time interval between two different behavior changes. The changing is identified when the process, based on the $\tilde{\mathbf{u}}_j$ stream, is out-of-control according to the Control Chart (CC) rules [8].

Modelled in this way, the system behavior can be monitored by a classical CC defined in Chapter 4. The CC statistical rules are applied to identify the t_*^j moments in which the process is out-of-control, being necessary to reset the control lines. The value of t_*^j is interpreted as the j-th time instant in which occurs a change of the system behavior, and it is often associated to event occurrences. The nature of the monitored system is described as a sequence of behaviors.

To maintain the process under control, the control lines are recalculated, generating the mean μ^{j+1} and standard deviation σ^{j+1} associated to the $\mathcal{B}^{j+1}$, the $(j+1)$-th system behavior defined by the $\tilde{\mathbf{u}}_j$ stream with $\mathcal{N}(\mu^{j+1}, {\sigma^{j+1}}^2)$. This definition could be used to diagnoses reported problems in $\mathcal{D}$. In the next section, four state models based on the Fuzzy Granular anomaly classification and their respective behavior models are presented.

8.2 Proposed Models

In this section, four solutions based on snapshots of system functioning designed using severity degrees of anomalies are presented. The solutions are associated with the dimensionality of their input. Related to anomaly detection, our best results are achieved using an hour of $\tilde{\mathbf{u}}_j$ stream to produce the anomaly classification, generating a label per minute. Here, it is assumed that the system is under control in the majority of the time.

8.2.1 Naive State Models

The first solution set is based on a mapping of a combination of anomaly labels to a state, and a state set to a behavior. Since each classified period is associated with a distribution of anomaly severity, it can be used to define the system state. In this way, the interpretation of the possible system states $\mathcal{S}$ are

- operative state: it is defined by the occurrence of the anomaly level 0. It is considered as a normal functional status since the related data is the most frequent, representing around 67% of the total.

[8] See Eq.(6.1) Chapter 6 for details.

- semi-operative state: it is defined by the occurrence of the anomaly level 1 and it is also considered as a normal functional status. The system must be monitored more closely than in the previous state, since this anomaly occurrence could be the beginning of an increasing trend of anomalous data. The data classified as level 1 is the second most frequent type of anomaly, representing around 28% of the total.
- failure state: it is defined by the occurrence of the anomaly level 2 and it is considered as an abnormal functional status. The system must be monitored carefully because of the high probability of the data to represent an anomalous trend. The data classified in level 2 is the second most rare type of anomaly, representing around 4.7% of the total. The log analyses have to be done in order to diagnoses the event occurrence and its critical potential.
- fatal state: it is defined by the occurrence of the anomaly level 3 and it is considered as a critical functional status. The anomalous trend is confirmed and there is an ongoing system crisis. The data classified in level 3 is the most rare type of anomaly, representing around 0.3% of the total, and the log analyses are mandatory.

The Naive State Models have a well-defined and immutable set of possible states $\mathcal{S}$, and time-dependent state transitions $\mathcal{S}_t$. In the Unit State Model (USM), $\mathcal{S}$ is a finite crisp set of values $\{0,\ 1,\ 2,\ 3\}$. Otherwise, $\mathcal{S}$ in the Sequence State Model (SSM) is defined in $[0,\ 3]$ with infinite possible states. In the present solution, the infinite states of SSM are summarised in 3 numerical intervals $\{[0,\ 1),\ [1,\ 2),\ [2,\ 3]\}$. Therefore, the $\mathcal{S}$ can be implemented as a $\mathcal{S}_t$ matrix: a 4x4 matrix for USM, and a 3x3 matrix for SSM. Each element $t_{i,j}$ of $\mathcal{S}_t$ represents the current probability of the transition between the states i and j. The state machine of USM and SSM are shown in Fig. 8.2.

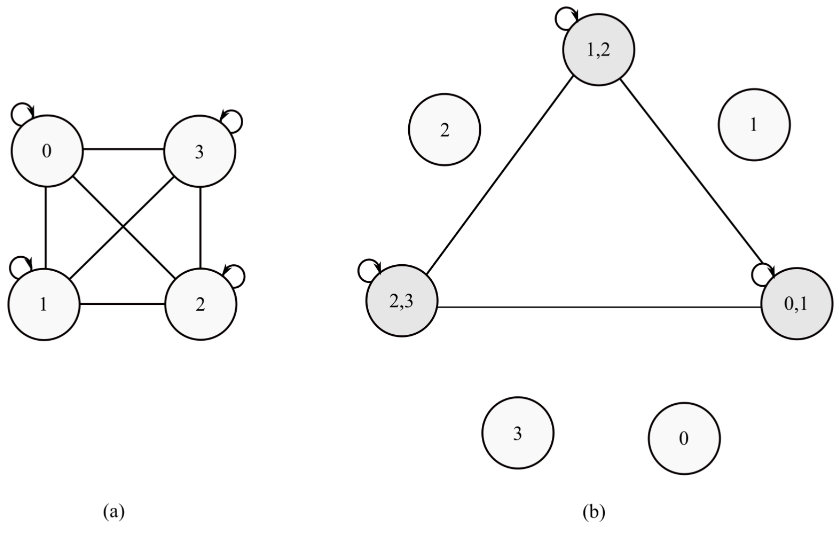


Figure 8.2: The Naive solutions have a well-defined and immutable $\mathcal{S}$, and a time-dependent $\mathcal{S}_t$. The $\mathcal{S}_t$ can be implemented as 4x4 matrix in (a) for USM with 4 possible states $\{0,\ 1,\ 2,\ 3\}$. In (b), the SSM has infinite possible states in $[0,\ 3]$, but they can be interpreted according to the numerical ranges $\{[0,\ 1),\ [1,\ 2),\ [2,\ 3]\}$, following the description of Sub-sec. 8.2.1.

Unit State Model

The Unit State Model (USM) is a zero-dimensional model, mapping 1:1 the anomaly label and the current state s_c of c_i. The anomaly is a numerical value, generated by a Granular Anomaly Detection solution (GADs). Remembering, GADs are based on tw_1 [9] and tw_2 [10] sliding windows, providing an online flow of the anomaly classification, that is interpreted by USM as a s_c stream. In this scenario, s_c is updated each tw_2 minutes. USM is the function

$$USM(j) = s_c^j. \tag{8.4}$$

in which j marks the j-th generated s_c, with $j > 1$. The set of possible states $\mathcal{S}_i$ of c_i is

$$\mathcal{S}_i \in \{0,\ 1,\ 2,\ 3\}. \tag{8.5}$$

The behavior study defines a functioning pattern, being necessary in the diagnosis phase. The j-th system behavior of c_i, $\mathcal{B}_i^j$, is defined as a sequence of the last k s_c generated by GADs during Δt

$$\mathcal{B}_i^j = \{s_c^1,\ s_c^2,\ \ldots,\ s_c^l,\ \ldots,\ s_c^k\}, \tag{8.6}$$

in which $s_c^l \in \mathcal{S}_i$. A graphical representation of USM is given by Fig. 8.3-a. The s_c^l interpretation is directly given by the state descriptions in Sec. 8.2.1.

Sequence State Model

The Sequence State Model (SSM) is an one-dimensional model based on the h:1 mapping, in which a sequence of the last h anomaly labels given by GADs is mapped in the current state s_c^{*j}. SSM is the function

$$SSM(j) = s_c^{*j} \tag{8.7}$$

In this work, the s_c^{*j} is summarized using the following weighted average aggregation function

$$s_c^* = \frac{\sum\limits_{i=1}^{h} s_c^i * p^i}{\sum\limits_{i=1}^{h} p^i} \tag{8.8}$$

[9] tw_1: the aggregation time interval.

[10] tw_2: the frequency, in which the anomaly classification is generated, is based on tw_1. The solution uses $tw_1 = 60$ minutes and $tw_2 = 1$ minute, i.e., it's generated an anomaly classification based on the log processing of 60 minutes each minute.

with

$$p^i = \begin{cases} p_0 & \text{if } s_c^i = 0; \\ p_1 & \text{if } s_c^i = 1; \\ p_2 & \text{if } s_c^i = 2; \\ p_3 & \text{if } s_c^i = 3; \end{cases} \tag{8.9}$$

in which $\sum_{\beta=0}^{3} p_\beta = 1$. The p_β are weights given by the normalised inverse of the expected frequency of the anomaly labels[11]

$$[p_0,\ p_1,\ p_2,\ p_3] = \frac{1}{T}\left[\frac{1}{0.67},\ \frac{1}{0.28},\ \frac{1}{0.047}\ ,\frac{1}{0.003}\right] \tag{8.10}$$

in which T is the normalisation factor. So

$$[p_0,\ p_1,\ p_2,\ p_3] = [0,\ 0.01,\ 0.06,\ 0.93] \tag{8.11}$$

This approach permits infinite possible system states S^* in

$$\mathcal{S}^* \in [0, 3], \tag{8.12}$$

in which $\mathcal{S}^*$ is in a numerical interval. As in the previous modelling, the system behavior $\mathcal{B}_i^*$ is based on a sequence of s_c^*, in which $s_c^* \in \mathcal{S}^*$ (Eq. 8.6). Figure 8.3-b summarises graphically the SSM details. The state interpretation is lightly different of USM. Instead of a precise identification of the system state, it is provided a fuzzy value between two described states in Sec. 8.2.1.

8.2.2 Granular State Models

The Granular State Models (GSM) are a set of solutions based on information granules (IGs) learned by GADs. Differently of the Naive solutions (Sub-sec. 8.2.1) in which the states are a combination of numerical values, here, the states are a combination of IG as defined in Chapter 3.

Besides the usual anomaly label, these solutions have s_c associated to IG ellipses, which gives a higher dimensionality of its classification[12]. The composition of the ellipse and the anomaly label determine IG, and consequently, the state. Hence, the ellipse characteristics[13] increases the dimensionality of the solution space, aggregating information about the system health. Because the infinite possibilities related to the IG attributes, the $\mathcal{S}$ in GSM must to be updated considering the IG fuzzy nature.

Figure 8.4 shows the state machine that represents both, the Information Granule State Model (IGSM) and the Granular Set State Model (GSSM). In IGSM, each node is a IG of the set of the

[11]See Chapter 4 for more details.
[12]The IG is given by eGFC.
[13]As its format and its location in 2D space.

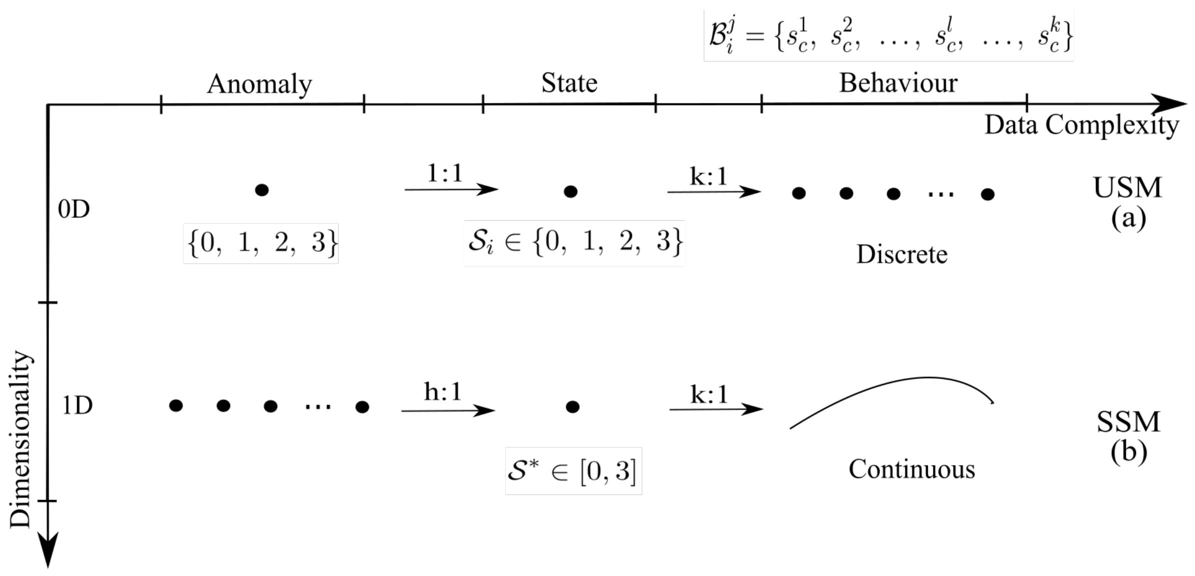

$\mathcal{B}_i^j = \{s_c^1,\ s_c^2,\ \dots,\ s_c^l,\ \dots,\ s_c^k\}$
Anomaly
State
Behaviour
Data Complexity
Dimensionality
0D
1D
1:1
k:1
h:1
$\{0,\ 1,\ 2,\ 3\}$
$\mathcal{S}_i \in \{0,\ 1,\ 2,\ 3\}$
Discrete
USM (a)
$\mathcal{S}^* \in [0,3]$
Continuous
SSM (b)

Figure 8.3: This figure summarises the naive models. The principal model characteristics are shown, i.e., how the anomaly, the state, and the behavior are graphically represented. The dimensionality of the anomaly inputted in the models is highlighted as well as the mappings between anomaly and state; and state and behavior. The complete descriptions of (a) and (b) are in Sub-sec. 8.2.1.

IGs that are still remembered by the system at the moment t_γ. In GSSM, each node is a set of IGs remembered by the system at the moment t_γ. The t_γ moment is defined in Δt. The edges represent the possible state transition, in which the dotted lines and the solid lines were, respectively, not verified yet and observed before the moment t_γ.

Information Granule State Model

The Information Granule State Model (IGSM) is a two-dimensional model based on a 1:1 mapping between the IG generated by GADs and the current state s_c°. IGSM is the function

$$IGSM(j) = s_c^{\circ j}. \tag{8.13}$$

The $s_c^{\circ j}$ is modelled as the last activated IG. The IG is activated when it absorbs a new data. The last activated granule is a IG of the set of remembering IGs that absorbs the last data inputted in a GADs. The j-th $s_c^\circ \in \mathcal{S}^\circ$ is defined by[14]

$$s_c^{\circ j} = IG(j) \tag{8.14}$$

[14]In which s_c° and $\mathcal{S}^\circ$ are associated to c_i.

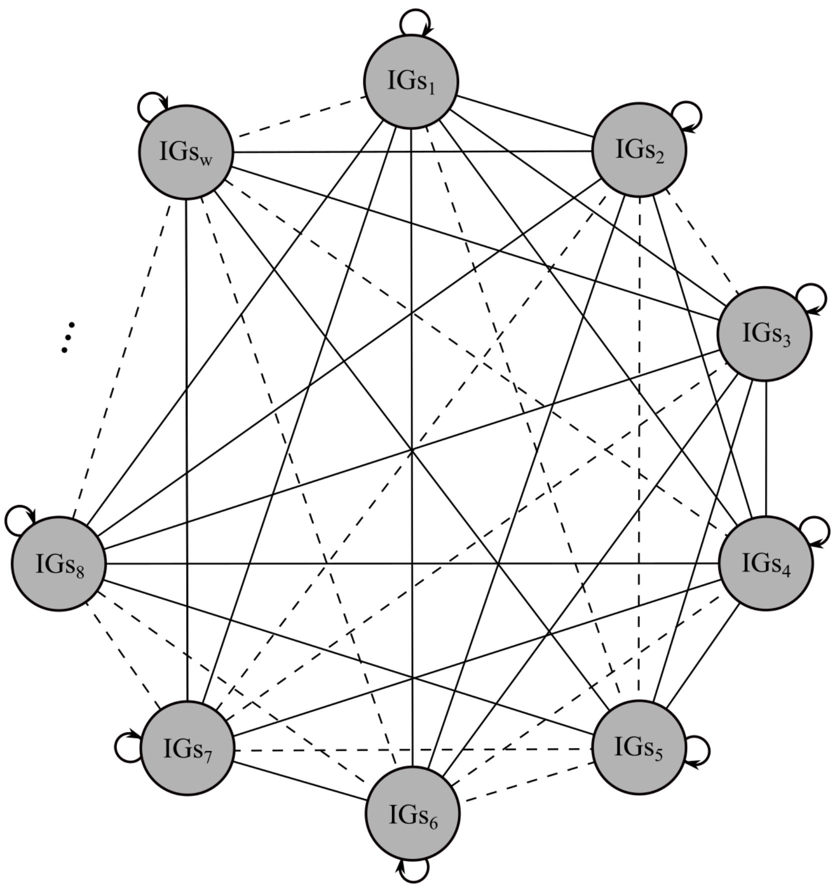


Figure 8.4: The Granular solutions have a well-defined and mutable $\mathcal{S}$ and $\mathcal{S}_t$. Here, the nodes are the representation of the possible states during Δt, in which the IGs_γ means a single IG in IGSM, and a set of IGs in GSSM. As in Sub-sec. 8.2.1, $\mathcal{S}_t$ can be implemented as wxw matrix in both, in which w is quantity of activated elements. The dotted line represents a possible state transition that was not verified until the moment t. In the other hand, the solid line represents the observed state transitions until the moment t.

with

$$IG(j) = (\mathbf{c_p^j}, \mathbf{r^j}, a_l^j). \tag{8.15}$$

The $IG(j)$ is generated by a GADs using f of c_i, activated at j-th snapshot. IG is associated with an anomaly label and the ellipse centered at the point $\mathbf{c_p} = (c_x, c_y)$, with the points $\mathbf{r} = (r_b, r_s)$ used to generate radius vectors in the 2D space, and a_l^j is the IG anomaly label. In this way, IG is given by a tuple defined in $(\mathbb{R}^2, (\mathbb{R}^2, \mathbb{R}^2), \mathcal{S}_i)$. Hence, the set of possible states $\mathcal{S}^\circ$ is the function

$$\mathcal{S}^\circ : j \mapsto (\mathbb{R}^2, (\mathbb{R}^2, \mathbb{R}^2), \mathcal{S}_i), \tag{8.16}$$

in which $\mathcal{S}^\circ$ maps the time instant j to a IG, being $\mathcal{S}_i$ defined in Eq. 8.5.

As the previous models, the IGSM behavior model, $\mathcal{B}_i^\circ$, is a s_c° sequence as in Eq. 8.6 using the data generated during Δt. In this particular case, it is a sequence of labelled ellipses representing the last $|\mathcal{B}_i^\circ|$ activated IGs. Figure 8.5-a shows graphically the IGSM details. In this model, the system state is an ellipse representing the intensity of the associated anomaly label, and can be compared to the other granules that compose the behavior trend, giving a more complete design of the evolution of the system status.

Granule Set State Model

The Granule Set State Model (GSSM) is a three-dimensional model based on the I^j:1 mapping between the j-th set of $|I^j|$ activated IGs generated by GADs and the current state $s_c^\dagger$. GSSM is the function

$$GSSM(j) = s_c^{\dagger j}, \tag{8.17}$$

in which $s_c^{\dagger j}$ is the j-th $s_c^\dagger \in \mathcal{S}^\dagger$ defined as

$$s_c^{\dagger j} = I^j. \tag{8.18}$$

The j-th set of IGs, I^j, is formed by

$$I^j = \{IG_1^j,\ IG_2^j,\ \dots,\ IG_\alpha^j,\ \dots,\ IG_{|I^j|}^j\}. \tag{8.19}$$

The IG_α^j is defined as $IG(j)$ in Eq. 8.15, and $\mathcal{S}^\dagger$ is the function that maps j in the last $|I^j|$ activated IGs

$$\mathcal{S}^\dagger : j \mapsto (\mathbb{R}^2, (\mathbb{R}^2, \mathbb{R}^2), \mathcal{S}_i)^+. \tag{8.20}$$

$\mathcal{S}^\dagger$ maps j to at least a IG [15]. The GADs implementation have the h_r variable to control the speed in which a IG is forgotten, given in rounds. Consequently, h_r also controls the quantity of activated IGs

[15]The notation $+$ is used to indicate that $\mathcal{S}^\dagger$ associates one or more IGs with j.

at j.

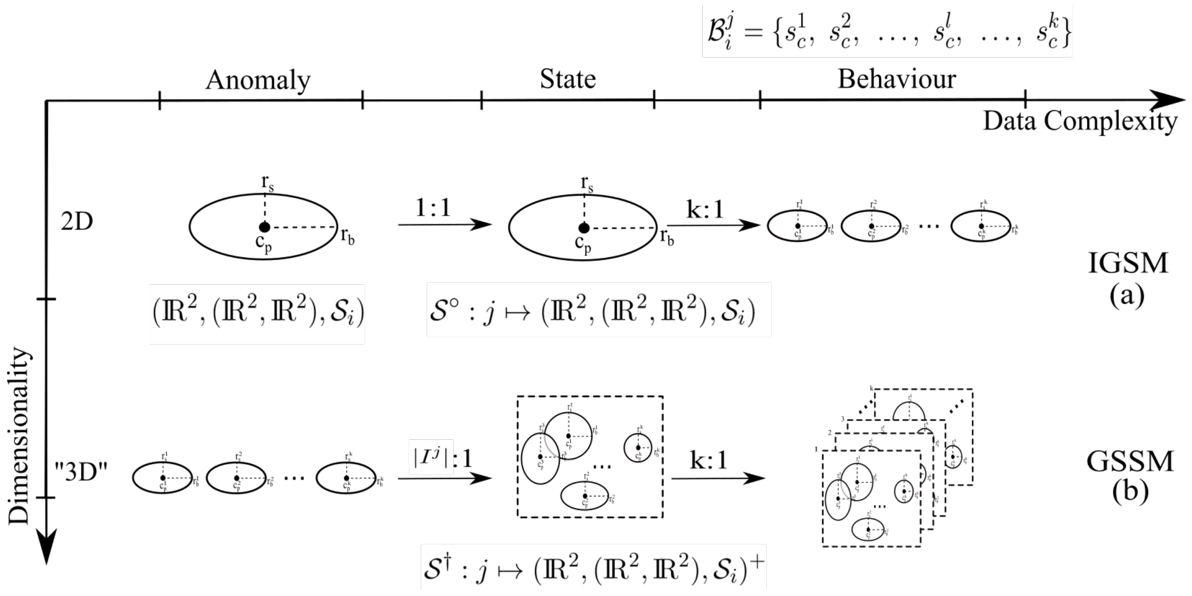

$\mathcal{B}_i^j = \{s_c^1,\ s_c^2,\ \ldots,\ s_c^l,\ \ldots,\ s_c^k\}$
Anomaly
State
Behaviour
Data Complexity
Dimensionality
2D
"3D"
r_s
c_p
r_b
1:1
k:1
$|I^j|$:1
$(\mathbb{R}^2, (\mathbb{R}^2, \mathbb{R}^2), \mathcal{S}_i)$
$\mathcal{S}^\circ : j \mapsto (\mathbb{R}^2, (\mathbb{R}^2, \mathbb{R}^2), \mathcal{S}_i)$
$\mathcal{S}^\dagger : j \mapsto (\mathbb{R}^2, (\mathbb{R}^2, \mathbb{R}^2), \mathcal{S}_i)^+$
IGSM
(a)
GSSM
(b)

Figure 8.5: This figure summarises the granular models. The principal model characteristics are shown, i.e., how the anomaly, the state, and the behavior are graphically represented. The dimensionality of the anomaly inputted in the models is highlighted as well as the mappings between anomaly and state; and state and behavior. The complete descriptions of (a) and (b) are in Sub-sec. 8.2.2.

The $s_c^{\dagger j}$ variable is a set of IGs, i.e., a vector of labelled ellipses generated by GADs using f of c_i, considering the j-th snapshot. As the previous models, the GSSM behavior model, $\mathcal{B}_i^\dagger$, is a $s_c^\dagger$ sequence given by Eq. 8.6. Figure 8.5-b represents graphically the GSSM details. In this particular case, it is a sequence of set of labelled ellipses representing the last $|\mathcal{B}_i^\dagger|$ activated set of IGs of c_i during Δt.

8.3 Summary

In this chapter, four state models based on GADs have been presented. Increasing the model complexity, the graphical representation of the models are the point, the vector, the 2D ellipse, and the sequence of 2D ellipses to USM, SSM, IGSM, and GSSM, respectively, as shown in Fig. 8.6. The anomaly input is used to classify the model dimensionality, and consequently its implementation complexity. It is expected the increasing of accuracy of the prediction with the increasing of the model complexity. On the other hand, the complexity increasing makes state interpretation more difficult.

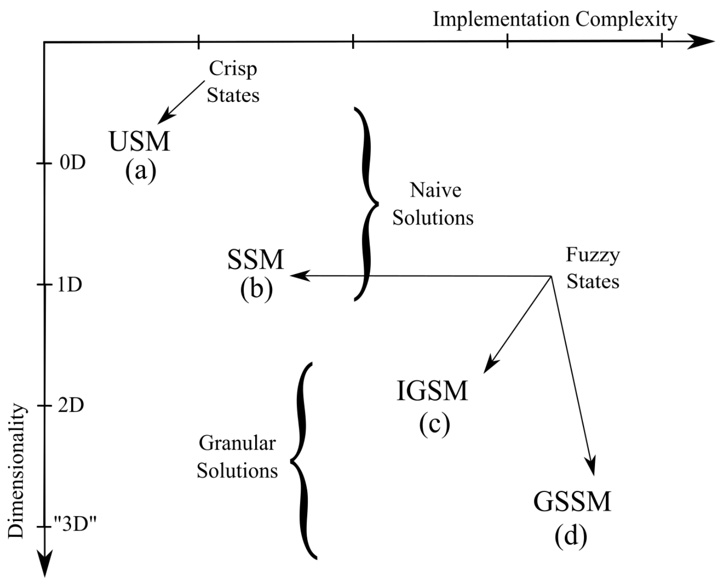


Figure 8.6: This figure compares the proposed models, classifying them in Naive or Granular solutions, organising them according to the dimensionality of the design of the inputted anomaly and the implementation complexity. This summarising shows the fuzzy or crisp nature of the state in all models. The complete descriptions of (a) and (b), (c) and (d) are in Sub-sec. 8.2.1, and 8.2.2 respectively.

Chapter 9

Conclusion

This chapter summarises the thesis, highlights its contributions, and best results achieved in Sec. 9.1, and provides directions for future investigation in Sec. 9.2.

9.1 Summary

Computing systems are embedded in the society to support the synergistic collaboration between humans and machines. Beyond the economic factor, the energy consumption and the heat generation are strongly affected by the performance of computing systems. The quality of the software functioning may invalidate any optimization attempt. Data-driven machine learning algorithms are the basis of the human-centered applications in the ages of information, in which their interpretability become one of the most important hot topics.

In particular, software development and maintenance are critical disciplines in the computer science, being both the basis of the automatising growth, supporting the evolution of the computing systems. Data center maintenance systems are a software to manage a set of software. Multi-users, cross-platform, and multi-property software cohabit in the same ecosystem, complicating and squishing even more the system stabilising, collapsing until its inoperability.

Data centers are evolving scenarios that tend naturally to crash. In general, online machine learning approaches are used as auxiliary tools to uphold the sustenance of the log-based computing systems. Using logs to promote the system preservation is an inexpensive and smart choice as most software registers the inner events in logs. The ease of obtaining logs is counterbalanced by the difficulty of handling them, since they are Big Data assembled in large-flow streams of messy information.

To help in this task, the Granular Computing paradigm controls the abstraction level of a problem through the granularization degree of the system information. This mechanism permits that the observer changes convenently the problem perspective. Data-stream driven modelling and the adaptation mechanisms address properly the uncertainty of the functioning behaviour of computing systems through learning algorithms. Granular paradigm provides the control mechanism to adjust the abstraction level of the information, and Fuzzy set, the mathematical foundation to approach the data uncertainty. Furthermore, system maintenance is an ultimately decision-making problem, as such it is needed to explain the taken decisions, mainly because of the appeal of the Explainable Artificial Intelligence to get transparent and intelligible solutions in human-centered applications.

In particular, the background scenario is the Worldwide LHC Computing Grid (WLCG), a world-wide grid infrastructure involving more than 170 computing centers over 42 countries supporting the Large Hadron Collider (LHC) experiments. The main WLCG computing center is at CERN, the research center that holds the biggest particle accelerator of the world and is responsible for the main-stream research in Nuclear and Particle Physics. In total, the LHC experiments rely on more than 400 PB of disk storage, 700 PB of tape storage, almost 1 million CPU cores, and both, dedicated and shared network infrastructures. In this complex infrastructure, the system maintenance problem was designed.

To address the problem, a framework is proposed, being inspired in a heath care metaphor. The main areas of the human health care were mapped in the respective system maintenance area, coupling them in a solution pipeline. Each partial solution is a complete tool by itself, being its output explainable. The puzzle of smart tools provides an useful composition, in which modules can collaborate with each others.

Anomaly detection (AD) solutions are the core of this thesis. The used methodology is capable to format logs, initially a messy of verbose data, in a stream of numerical and well-behaved vectors, preserving the original system characteristics. This step assures and facilitates the data handling, supervising them using a clever self-learning method based on control charts. The modified data is finally classified according with its severity in a four-level scheme through three different fuzzy-granular algorithms, i.e., (i) Fuzzy set-Based evolving Model (FBeM), (ii) evolving Granular Neural Network (eGNN), and (iii) evolving Gaussian Fuzzy Classifier (eGFC). All methods are online classifiers based on fuzzy rules set extracted even by intense data streams. In practice, the data stream is clustered in information granules according with the fuzzy rules defined by each method. In (i), FBeM is a fuzzy-granular approach in which the fuzzy rules are trapezoidal membership functions. In (ii), eGNN is a neuro-fuzzy granular network constructed incrementally from an online data stream. Its processing units are fuzzy neurons and granules, and the network architecture is the result of a gradual construction. In (iii), eGFC is a semi-supervised evolving classifier derived from an online granular-computing framework, employing Gaussian membership functions to cover the data space with fuzzy granules. Essentially, regarding anomaly detection accuracy, FBeM achieved $(85.64 \pm 3.69)\%$; eGNN reached $(96.17 \pm 0.78)\%$; and eGFC obtained $(92.48 \pm 1.21)\%$. These methods can individuate anomaly occurrences with a high probability in non-stationary scenarios.

AD outputs can be analysed in three ways. At first, it is possible to identify if an anomaly is occurring at the moment, as a system monitoring. Second, the set of granules can provide a screenshot state of the system as a pictorial design of the anomaly distribution, giving the functioning current scenario. Based on this idea, it is proposed four different state models. And third, AD outputs provide the anomaly history of a system, and can be used as input to a prediction method. In addiction, AD implementations work as filters, selecting the pieces of logs that have the higher probability to have useful information to be extracted by log parsing, minimising the content processing.

Log Parsing (LP) is used to extract the formal grammar of logs through an evolving fuzzy-granular classifier, the evolving Log Parsing (eLP). The method is based on an evolving syntactic analysis considering each template, extracted by log processing, as a formal grammar. eLP is inspired in Interval-Based Evolving Modeling (IBeM) and uses interval arithmetic to compose the fuzzy rules set. Each rule is used to recognise the syntax of a message type. The rule receives a numerical vector that represents the indexes of the template words found in the new sample. This vector is used to verify how much a sample belongs to a rule, and consequently to the correspondent granule. The most active granule is adapted using the new sample, if it is activated enough. At each round, the rules set is updated, merging similar granules, and forgetting the inactive one when it is necessary. This method has all the basic adaptation operations of the fuzzy granular learning algorithms. LP outputs are a numerical data stream of log message identifications[1], in which each one is associated with a textual template. Since the method identifies the message type through the template recognising, its parameters extraction is an obvious step. Both, message type and features, can be used as diagnosis information. For the first time in the evolving intelligent systems literature, the proposed method, eLP, is able to process streams of words and sentences. eLP reached $(94 \pm 2)\%$ of template effectiveness. Besides being competitive, eLP particularly generates a log grammar, and presents a higher level of model interpretability.

9.2 Future Research

Engineering applications were the pioneers in the systems maintenance area, considering initially simple implementations with a small number of possible known states, and increasing progressively the complexity level of the approached problems. However, computing systems are a much harder problem conceptually. They permit an infinite set of possible, and partially unknown, states, even in simple cases. The origin of this problem is the intrinsic difficult to proceed the formal verification of a programming code - the classical attempt of Software Engineering to create programs as reliable as possible, avoiding bugs, errors, failures, misprogramming, etc.

Obviously, it is impossible to ensure the correctness of complex implementations. Programming is a sophisticate ability that permits countless solutions to the same problem through manifold paradigms. Because of that, the system maintenance will be always a topic in interest to Computer Science, in which the tendency is to increase its intricacy with the increasing of computational complexity of applications. There is a long way until the ultimate system maintenance goal be achieved - the complete autonomy of the maintenance process.

In Chapter 2, the proposed framework can provide numerous insights about future works, specially in the Identification Part. The implementation of a fuzzy-granular predictive maintenance using the supervised dataset[2] could be addressed directly, predicting anomaly trends through FBeM, IBeM, eGNN, or eFGC, since all of them are both classifiers and predictors. Another prompt possibility is a

[1]It is a tuple of numbers (t_{ID}, f_D), the template id and its effectiveness, by for simplicity, we are considering just t_{ID} in this moment.

[2]The dataset is generated by any anomaly detection solution addressed in Chapter 4.

prediction approach using the output of the log parsing, in which the results could be combined with the previous anomaly forecasting, since it is interesting minimising the consumption of computing resource. The parsing output is a sequence of message types, represented by numbers. This numerical sequence would be the input for a forecasting system based on content[3], which would be triggered when an anomalous trend has been identified by the previous predictor. Using the same strategy applied in system monitoring, the anomaly detection[4] coupled with message type identification could be used to diagnosis an anomalous occurrences, helping in the recognition of the anomaly origin using logs content.

Changing the perspective of the approach from health care to speech translation, it is possible to envision new application scenarios. To help to describe system events, codes generate logs that are used to track them. Each software generates a different set of logs because logs are strongly dependent on code functionalities. A software has to tell event stories through log generation, creating its own language, unintelligible to other software and human beings, originally. In this way, logs, seen as a primitive language, can be designated through a formal grammar (log parsing approach) and a dictionary of event meanings via ontology. The couple grammar-dictionary provides the basis for the semantic analysis, defining a proper communication standard, even if it is unidirectional initially. The semantic analysers are the basis for a prescriptive maintenance in the Intervention Part of the framework in Chapter 2.

Imagining even further into the future, since the linguistic expression of a system is comprehensible, it would be possible to teach it to other software using an intermediary translation machine. A similar strategy has been used in compilers of programming languages for a long time, when a language is translated to an assembly version as an intermediary step to generate the binary code. In this way, system-to-system protocols could be used to communicate inner events in the sphere of the software operations, could be used to avoid conflicts or inconsistencies through decision taken autonomously using distributed sentinels or a centralised maintenance watcher.

Considering the future of fuzzy-granular algorithms, eLP is the first step to consider fuzzy-granular approaches as candidate to Natural Language Processing strategy, since it is used to process textual data to generate formal grammars. Even if it is applied in a simple and limited language as logs, to take advantage of the conceptual similarity between fuzzy sets and formal grammars could be a promising approach. Considering also the characteristics of natural language, as ambiguity and need for contextualisation, it is an extra incentive to bet on fuzzy-granular approaches.

[3]Using FBeM, IBeM, eGNN, or eGFC, as before.

[4]Differently of the prediction approach, this solution uses an anomaly detection classifier coupled with the log parsing (classifier).

List of Tables

7.1 eGNN Performance in Multi-class Classification of System Anomalies with 99% of confidence . . . 90
7.2 Performance of the Algorithms in Multi-class Classification of System Anomalies with 99% of confidence, ranked in ascending order. . . . 91
7.3 eGNN Performance in Multi-class Classification of System Anomalies with 99% of confidence . . . 96
7.4 The eLP Experimental Meta-parameters' range . . . 97
7.5 The eLP Experimental Results . . . 98

C.1 Parameters used by the unsupervised learning methods . . . 145
C.2 Computational resource consumption during I and II stages of the proposed framework 146

List of Figures

2.1 Maintenance System (MS) classification according with the MS intervention strategy related to the moment of the failure occurrence. 22
2.2 The flowchart represents the medical attendance with the main steps of the process. . . 28
2.3 The mapping between the occurrence and language spaces is the interpretation of happened events, increasing the abstraction level of the perceived information. 29
2.4 Maintenance framework based on an approach of Health Maintenance. 30

4.1 eGNN: Evolving neuro-fuzzy network architecture for classification 57
4.2 Fuzzy neuron model 58

5.1 Example of interval $I_k^i = [l_k^i, L_k^i]$, and expansion region E_k^i 75

6.1 SRM architecture and relations among its main modules: Front-end, Back-end, and databases 80
6.2 Control chart procedure to tag log data 82

7.1 FBeM Classification accuracy versus amount of rules for an array of meta-parameters . 92
7.2 eGNN classification accuracy versus amount of rules for a variety of meta-parameters . . 92
7.3 Example of FBeM confusion matrix in anomaly classification. 93
7.4 Example of eGNN confusion matrix in anomaly classification. 93
7.5 Example of eFGC confusion matrix in anomaly classification 94
7.6 The time evolution of the evolving factors: granulation ρ and number of rules, and the model accuracy until the convergence at the 3 first graphics. At the last 2 graphics the eGFC Gaussian classes. 95
7.7 The multi-dimensional ellipsoidal geometry of eGFC granules using the first four attributes of the log stream. The colours of the centers refer to the control chat of Fig. 6.2, i.e., green: normal system condition; yellow, orange and red: low, medium and high anomaly severity 96
7.8 Example of logs in which the template is extracted easily. 98
7.9 Example of logs in which extracting the template can be tricky. 99
7.10 Log message that could harm the template extraction in Fig. 7.9. 99

7.11 The template effectiveness (a-b-c) and the model interpretability (d-e-f) are deeply compare in regards to the number of rules (a-d), h_r (b-e), and ρ (c-f). 101

7.12 The template's size (a-b-c) and the number of rules (d-e) are deeply compare in regards to the number of ρ (a-d), h_r (b-e), and number of rules (c). In (f), it is highlighted the relationship between the model interpretability-template effectiveness. 102

8.1 Here, it is shown a piece of the graph $\mathcal{D}$, representing the data center maintenance system, in which c_i is a SwC with an associated $\mathcal{M}_i$ system maintenance. The $p_{i,j}$ is the probability of a c_i state change induces a c_j state change, and it is represented as edge weights in the graph. 105

8.2 The Naive solutions have a well-defined and immutable $\mathcal{S}$, and a time-dependent $\mathcal{S}_t$. The $\mathcal{S}_t$ can be implemented as 4x4 matrix in (a) for USM with 4 possible states $\{0,\ 1,\ 2,\ 3\}$. In (b), the SSM has infinite possible states in $[0,\ 3]$, but they can be interpreted according to the numerical ranges $\{[0,\ 1),\ [1,\ 2),\ [2,\ 3]\}$, following the description of Sub-sec. 8.2.1. 109

8.3 This figure summarises the naive models. The principal model characteristics are shown, i.e., how the anomaly, the state, and the behavior are graphically represented. The dimensionality of the anomaly inputted in the models is highlighted as well as the mappings between anomaly and state; and state and behavior. The complete descriptions of (a) and (b) are in Sub-sec. 8.2.1. 112

8.4 The Granular solutions have a well-defined and mutable $\mathcal{S}$ and $\mathcal{S}_t$. Here, the nodes are the representation of the possible states during Δt, in which the IGs_γ means a single IG in IGSM, and a set of IGs in GSSM. As in Sub-sec. 8.2.1, $\mathcal{S}_t$ can be implemented as wxw matrix in both, in which w is quantity of activated elements. The dotted line represents a possible state transition that was not verified until the moment t. In the other hand, the solid line represents the observed state transitions until the moment t. 113

8.5 This figure summarises the granular models. The principal model characteristics are shown, i.e., how the anomaly, the state, and the behavior are graphically represented. The dimensionality of the anomaly inputted in the models is highlighted as well as the mappings between anomaly and state; and state and behavior. The complete descriptions of (a) and (b) are in Sub-sec. 8.2.2. 115

8.6 This figure compares the proposed models, classifying them in Naive or Granular solutions, organising them according to the dimensionality of the design of the inputted anomaly and the implementation complexity. This summarising shows the fuzzy or crisp nature of the state in all models. The complete descriptions of (a) and (b), (c) and (d) are in Sub-sec. 8.2.1, and 8.2.2 respectively. 116

A.1 Template of the multi-class confusion Matrix 127

B.1 The evolving granular hyperparameters are used in all granular approaches. 130

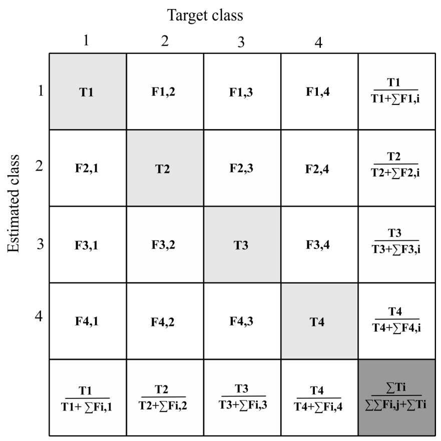
B.2 Methodology summarization in 3 modules: (1) pre-processing, (2) weak tagging, and (3) fuzzy-granular classification. 133
B.3 Pre-processing phase is composed by data extracting, transforming data distribution and data structuring. 134
B.4 Weak tagging is based on a Control Chart classification. 134
B.5 Anomaly Detection fuzzy-granular approaches receive as input the data pair, and generate a more specific classification output, detailing each possible class in information granules. 136
B.6 Example of Anomaly detection output given by FBeM classification, generating a non-linear and time-varying decision boundary of the classes. Granule border is fuzzy defined by a smooth frontier through a trapezoidal membership function. 136
B.7 evolving granular Log Parser receives as input the log message, and identify its template. 138

C.1 Flowchart of the proposed DMF framework 139
C.2 Confusion matrix as classification performance measure 145
C.3 Example of a peak of log production (day 23) as a sign of an anomalous occurrence . . . 146
C.4 Evolving behavior of the StoRM log data during the observed anomalous period: (a) percentage of anomalies detected; (b) amount of generated anomalous periods using a 4-day time window 147
C.5 Confusion matrices of the classifiers: (a) One-class SVM, (b) IF, and (c) LOF. The IF method has shown the best result for accuracy (69.5%). To understand the third line and column, see Figure C.2 147

Appendix A

How to read the Multi-Class Confusion Matrix

Consider confusion matrices to multi-class classification problem, as shown in Figure A.1. The y-axis indicates the estimated classes, and the x-axis, the target classes. The 2x2 matrix aligned on the top-left side provides the classification results related to the possible classes (normal and anomaly), i.e., True Positive (TP), False Positive (FP) or Type I Error, False Negative (FN) or Type II Error, and True Negative (TN). These values are expressed as a percentage mean and a standard deviation. The square at the right-down corner provides the global accuracy of a method. The other squares of the last column give the sensitivity (TPR) and specificity (TNR), from up to down, and the other squares of the last line give the precision (PPV) and negative precision value (NPV), from left to right.



	Target class 1	2	3	4	
Estimated class 1	T1	F1,2	F1,3	F1,4	$\frac{T1}{T1+\sum F1,i}$
2	F2,1	T2	F2,3	F2,4	$\frac{T2}{T2+\sum F2,i}$
3	F3,1	F3,2	T3	F3,4	$\frac{T3}{T3+\sum F3,i}$
4	F4,1	F4,2	F4,3	T4	$\frac{T4}{T4+\sum F4,i}$
	$\frac{T1}{T1+\sum Fi,1}$	$\frac{T2}{T2+\sum Fi,2}$	$\frac{T3}{T3+\sum Fi,3}$	$\frac{T4}{T4+\sum Fi,4}$	$\frac{\sum Ti}{\sum\sum Fi,j+\sum Ti}$

Figure A.1: Template of the multi-class confusion Matrix

Appendix B

Tutorial of the Log-based evolving Maintenance Service

This tutorial gives an overview of the methodology and explains how to use the first version of the service-driven eMF to detect and interpret anomalous behavior of log-based systems. It is addressed to attend needs of the INFN technical staff to apply the solution operatively. Even this work is done for INFN computing center, the tutorial is general-purpose and can be applied in any case, from an service to a whole computing infrastructure. For more details about the fuzzy-granular methods and a complete description on updating, removing, conflict resolution, and merging procedures, see Chapters 4 and 5. The theoretical basis to a deep methodology understanding is presented in Chapters 2 and 3.

Log files are unstructured big data, and as such have to be processed in a massive way to provide useful information about the system behavior. System maintenance is a complex data mining that evolves complementary problems to build the final service, involving monitoring, detection, diagnosis, and prediction of events, usually failures.

The present methodology is a general-purpose service-driven software. The evolving Maintenance Framework (eMF) is a composition of fuzzy-granular components, easily adaptable to different log-based systems. The data granularity of the output can be set appropriately to a practical scenario. All components are set to produce a human-intelligible solution through the hyperparameters settings: (1) h_r, that control the model memory, and (2) the granularity ρ, that define the size of the information granules. In this way, logs are classified in levels of anomaly, in which each possible class is clustered in information granules. They identify sub-classes of anomalies inside of the class, providing more accurate and localised data analysis.

Figure B.1 shows the hyperparameters used in all granular approaches of this work: (i) the granularity ρ, and (ii) the memory controller, h_r. In (i), ρ defines the size of the information granule as a mechanism of abstraction control. Expanding ρ, the abstraction level is raised, having a more generic model, facilitating its structural stability since a small granules set is able to absorb more samples. On

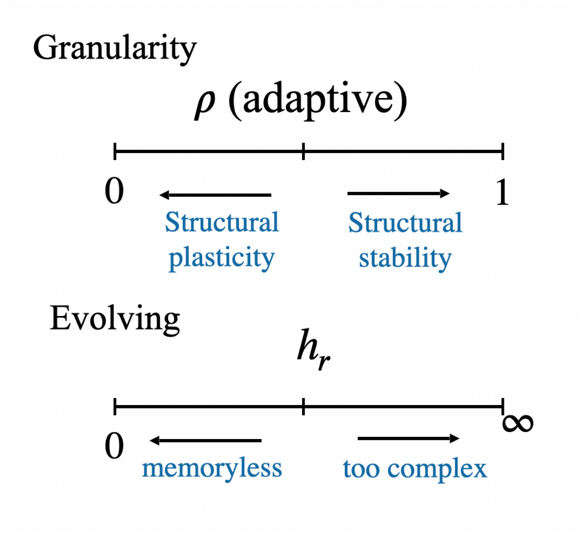


Figure B.1: The evolving granular hyperparameters are used in all granular approaches.

the other hand, if ρ contracts, a more detailed model of the data is generated with a bigger structural plasticity. In (ii), h_r defines how much the model remembers about the inactive granules, in which the inactivity is defined by the absence of data absorption.

B.1 How to choose the log file

A good programming practice guidelines include the use of life cycles of software development, modular programming, and code reuse. The logging activity of a computing system is extracted from a previously chosen log file used as an indicator of the system behavior, $indicator_{file}$. The strategy to choose $indicator_{file}$ is out of the scope of this tutorial.

In anyway, the used data is a critical point of the framework and it must be sensitive enough to anomaly occurrences, otherwise occasionally the methodology can be invalidated and the analysis quality, impaired. It is essential to choose event-based data, i.e., each line written in the log file has to be related to one or more event occurrences during the system runtime. A software can produce one or more candidates to be $indicator_{file}$ with different sensitivity to event occurrences. In these cases, the selection of the best candidate can be made by an expert, if it is possible, or by brute-force experiments otherwise. Brute-force experiments are designed to monitor the file candidates and identify each one has the data analysis that better represent the observed events, bringing better information for the failures identification.

The present software is based on log file, log record, and logging activity. Usually, a log record is made by a timestamp, i.e. the moment of the log writing, followed by the message itself describing the execution point of the software. Log file is a file in which the lines are log records. Log data is a set of log files generated by a software component. In general, it is an unstructured, redundant, and ad-hoc data generated with a high rate, having to be processed in order to have a real-time useful information about the system behavior. Logging activity (log_{act}) is a metric that measures the intensity of system activity, being used as indicator of the system behavior. Logging activity is the rate of log records written in a log file, measured by number of lines per time unit.

Log data are used to support technical staff to trace events occurring during system runtime. Each log record is linked to an execution point of the software. The sequence of execution points provides the event definition. Different system events can have sub-sequences in common, and impact differently on the decision made by the maintenance system.

System maintenance is a challenging topic specially in complex systems as computing centers. Since, this problem can be applied both a service of a computer and in a whole computing center, it is appropriate to modulate the solution to decrease the code complexity and person-power cost, providing a good code reuse and a modular software product.

System maintenance problem goes beyond anomaly monitoring and its detection, extending for diagnosis and prediction of new failures. It is possible to identify and to relate many interesting maintenance sub-problems, providing a complete solution [6]. The rest of this tutorial is structured as follows. Section B.2 presents related questions and implementation decisions. Section B.3 describes the three software components: (1) pre-processing, (2) weak tagging, and (3) the evolving fuzzy-rule-based classification method to identify the anomalous occurrences, and the diagnosis module based on log parsing.

B.2 About Implementation Decisions

Here, some related questions and the implementation decisions are approached to clarify crucial points of project design.

B.2.1 How can we approach big data problems?

One of the main concern related to big data problems is to deal with a high rate production of unstructured data. The data features create the necessity to produce general-purpose modular solutions to a problem class in order to provide a real-time response to the application. The main strategy is to transform the messy big data in a meaningful small data, considering human-centered problems in which the solution interpretability is essential. In the system maintenance, the process implies the transformation of big data into a comprehensive and viewable data that summarises the system behavior, maintaining the behavior representability.

B.2.2 Why are fuzzy-set solutions a good idea to log-based predictive maintenance systems?

A context-sensitive event-based log file is a good $indicator_{file}$. In general, event occurrences are expressed in the log file through log records. There is no rules related with how many log records are written during the event occurrence and how to define an event occurrence. Log records are hand-made, depending deeply of the programmer style. Because of the imprecise nature of data, different events can be defined by the same indicative log records, being a challenge to filter their occurrences. To identify a log record trace, one or more different event occurrences can have a similar or even the same log record trace, making clear the fuzzy-nature of the log data.

B.2.3 Why is it a good idea to use granular computing in a data-driven problem?

Granular computing provides the abstract level of the solution output, can be adjusted conveniently. The control chart self-learning strategy gives a weak classification, improved by the clustering of the classified data into information granules. Each class has at least a granule. A granule clusters similar data inside a class, and the information similarity is related to the granularity settings, and consequently, to the information abstraction. In such manner, information granules represent inner patterns of a class.

B.2.4 Why is it important to identify intervals of anomaly behavior of a service?

The system behavior can be monitored by log_{act}. Anomalous patterns indicate the occurrence of an unusual event in the system. To identify pattern occurrences is important to diagnose what it is happening during the service execution. Anomalous intervals have more probability to have traces of unusual events, such as cyberattacks. The frequency of anomalous event occurrence give a health metric of a system.

B.2.5 Why is it a good idea to do service-oriented maintenance system?

To identify the data processing unit of an algorithm is a good programming practice, permitting the code reuse, saving implementation time and decreasing the code coupling and complexity.

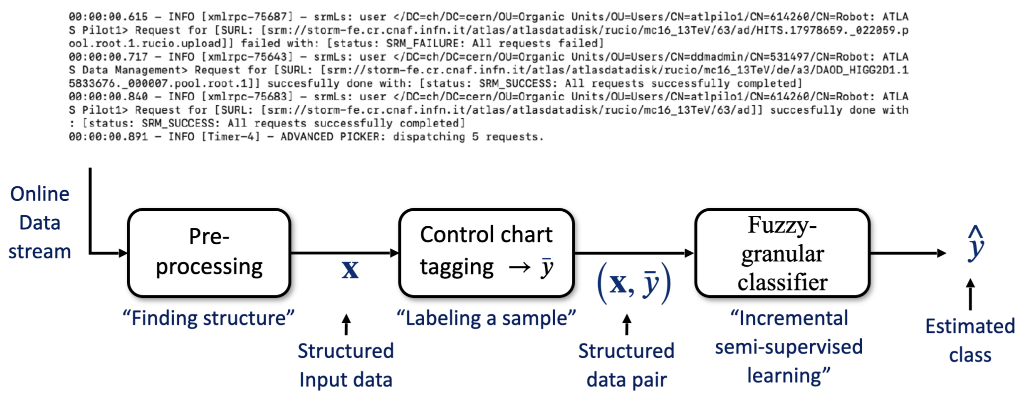
B.3 Software Description

The eMF software is compound by (1) anomaly detection, (2) log parsing, and (3) proposed extensions. Each module is summarised in this section. In (3), even though they have not yet been tested for this purpose, fuzzy-granular methods are used to recognise patterns through data and can be used to identify log patterns of anomalies and/or log messages.

B.3.1 Anomaly detection

The first part of this software, defined by the anomaly detection module, are composed by three components: (1) log pre-processing, (2) weak tagging, and (3) Fuzzy-granular method, i.e., FBeM, IBeM, eGNN, and eGFC.

Figure B.2 shows the modules that compose the proposed methodology. In (1), the pre-processing phase structures the log data stream summarising, in time-windows, the behavior of a service operation. In (2), the structured data is labelling with a weak tagging by a Control Chart self-learning approach. In (3), the supervised data pass through an incremental supervised learning to improve the estimation of the anomaly class of the original log data in one of the four possibilities classes, from normal to high anomaly level.



```
00:00:00.615 - INFO [xmlrpc-75687] - srmLs: user </DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=atlpilo1/CN=614260/CN=Robot: ATLA
S Pilot1> Request for [SURL: [srm://storm-fe.cr.cnaf.infn.it/atlas/atlasdatadisk/rucio/mc16_13TeV/63/ad/HITS.17978659._022059.p
ool.root.1.rucio.upload]] failed with: [status: SRM_FAILURE: All requests failed]
00:00:00.717 - INFO [xmlrpc-75643] - srmLs: user </DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=ddmadmin/CN=531497/CN=Robot: ATLA
S Data Management> Request for [SURL: [srm://storm-fe.cr.cnaf.infn.it/atlas/atlasdatadisk/rucio/mc16_13TeV/de/a3/DAOD_HIGG2D1.1
5833676._000007.pool.root.1]] succesfully done with: [status: SRM_SUCCESS: All requests successfully completed]
00:00:00.840 - INFO [xmlrpc-75683] - srmLs: user </DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=atlpilo1/CN=614260/CN=Robot: ATLA
S Pilot1> Request for [SURL: [srm://storm-fe.cr.cnaf.infn.it/atlas/atlasdatadisk/rucio/mc16_13TeV/63/ad]] succesfully done with
: [status: SRM_SUCCESS: All requests successfully completed]
00:00:00.891 - INFO [Timer-4] - ADVANCED PICKER: dispatching 5 requests.
```

Figure B.2: Methodology summarization in 3 modules: (1) pre-processing, (2) weak tagging, and (3) fuzzy-granular classification.

Log Pre-processing

At the first component, $indicator_{file}$ is used to generate log_{act}. The metric is defined as the number of lines generated during the time-window w_1:

$$log_{act} = \sum_{w_1}^{w_1} 1 \tag{B.1}$$

the log_{act} stream is $s_1 = \{log_{act}^1, log_{act}^2, \ldots\}$. In this version, w_1 is set as 10 seconds, the minimum possible value to the *Storm* Service $indicator_{file}$ example (*storm-backend.log*). A second time-series s_2 is the mean of m samples of s_1:

$$mean_{l_a}^{m_{start}} = \frac{\sum\limits_{i=m_{start}}^{m_{end}} log_{act}^i}{m} \tag{B.2}$$

in which $m = m_{end} - m_{start}$ with $m = 30$, $m_{start} = \{0, 1, 2, 3, \ldots\}$, generating $s_2 = \{mean_{a}^0, mean_{l_a}^1, mean_{l_a}^2, \ldots\}$.

Input Data

Based on the previously defined s_2 time-series, we generate the attributes $\mathbf{x} = \{x_1, \ldots, x_j, \ldots, x_5\}$. Attributes x_j mean $\overline{\mu}$, $\sigma(\mu_j \forall j)$, $min(\mu_j \forall j)$, $max(\mu_j \forall j)$, and $max(\Delta\mu_j)$, respectively. The latter means the maximum difference of amplitude of two consecutive μ_j, in which $\mu_j \subset w_j$, i.e., μ_j is contained in the j-th window, in which each window is composed by m consecutive s_2 samples.

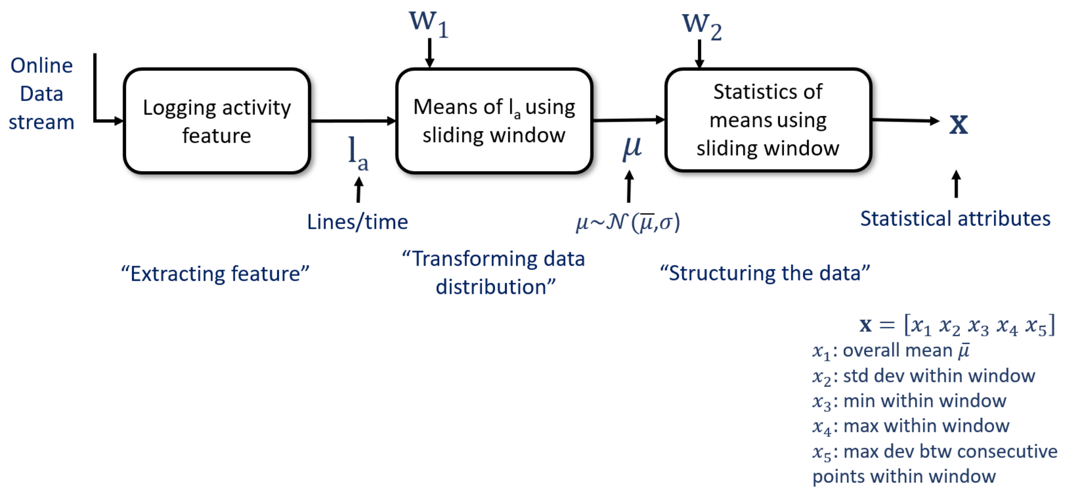
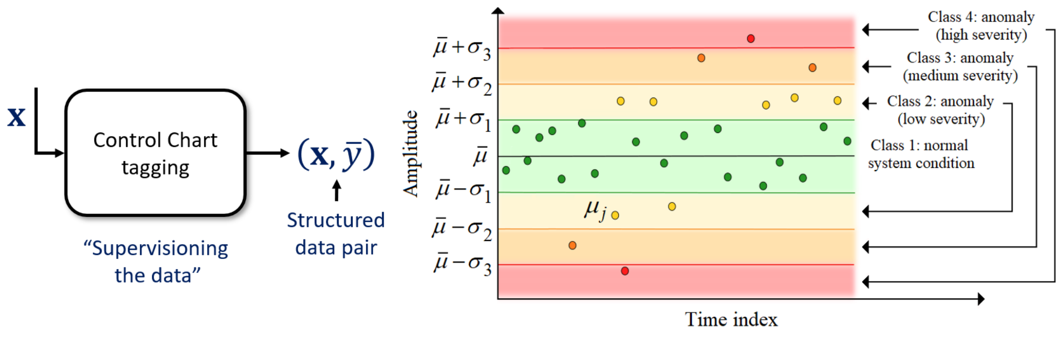


Figure B.3: Pre-processing phase is composed by data extracting, transforming data distribution and data structuring.

Figure B.3 indicates the three steps of the pre-processing phase: (1) extracting the feature log_{act} of the log data stream, (2) calculating the mean of m samples of s_1, normalising the probability distribution of the input data, and (3) generating a structured data stream s_3 with five columns.

Weak Tagging: Control Chart Approach

The time-series $\mathbf{x}$ is a series of statistical features. As one of them is the mean, we can use a control chart to make a weak tagging. A control chart is a time-series graphic used to monitor a process behavior, phenomenon or variable using the Central Limit Theorem [118]. It is based on the fact that the mean $\mu(x)$ of a random variable x follows a normal distribution.



x
Control Chart tagging
(x, ȳ)
Structured data pair
"Supervisioning the data"
Amplitude
μ̄+σ3
μ̄+σ2
μ̄+σ1
μ̄
μ̄−σ1
μ̄−σ2
μ̄−σ3
μj
Time index
Class 4: anomaly (high severity)
Class 3: anomaly (medium severity)
Class 2: anomaly (low severity)
Class 1: normal system condition

Figure B.4: Weak tagging is based on a Control Chart classification.

Let $\mathrm{x}_j = [x_1 \ \ldots \ x_j \ \ldots \ x_5]$, $j \geqslant 1$, be a sequence of statistical values from the means of m s_2. Figure B.4 shows how an instance s_3 is labelled using a control chart. To simplify the notation, s_3^j is named here as μ_j. The mean of $\bar{\mu}$ is

$$\bar{\mu} = \frac{1}{m}\sum_{j=1}^{m}\mu_j. \tag{B.3}$$

The k-th upper and lower horizontal lines in relation to $\bar{\mu}$ refer to the k-th standard deviation,

$$\sigma_k(s_3) = k * \sqrt{\frac{1}{m}\sum_{j=1}^{m}(\bar{\mu} - \mu_j)^2}. \tag{B.4}$$

If $\mu_j \subset [\bar{\mu} - k\ \sigma(\mu_j \forall j),\ \bar{\mu} + k\ \sigma(\mu_j \forall j)]$ and $\mu_j \not\subset [\bar{\mu} - (k-1)\ \sigma(\mu_j \forall j),\ \bar{\mu} + (k-1)\ \sigma(\mu_j \forall j)]$, then, for $k = 1$, the instance is tagged as 'Class 1', which means normal system operation. If $k = 2, 3, 4$, then μ_j is tagged as 'Class 2', 'Class 3' and 'Class 4', respectively, see Fig. B.4. These mean low, medium, and high-severity anomaly. The probability that μ_j is within each class is 67%, 28%, 4.7%, and 0.3%, respectively. The online anomaly detection problem is unbalanced, and the labels provided by the control chart approach are weak.

The self-learning control chart approach alleviates the burden of obtaining manually-labeled data, which can be impractical. Instead, weak labels are obtained with the understanding that they are imperfect because of the assumption of a unique, global, and time-invariant normal distribution. However, labels can nonetheless be used to support supervised learning and create a strong classification model using evolving fuzzy-granular methods. In order to illustrate the fuzzy-granular improvement, it is presented an example of fuzzy-granular method. For more details about it or to be aware about other available similar methods, see the Chapter 4.

Fuzzy-Granular Classifiers

This thesis approaches three evolving fuzzy-granular methods to anomaly detection problem: FBeM, eGNN, and eFGC. The methods are online and adaptive, generating real-time applications. All of them uses a evolving granular mechanism to classify data in four classes according with the system behavior previously modulate by the pre-processing step.

Figure B.5 abstracts the data flow of a fuzzy-granular classifier. The classifier receives the hyperparameter values and a data pair $(\mathbf{x}, \bar{y})$, in which $\mathbf{x}$ is the statistics summarization of system logs generated during a time-window, and $\bar{y}$ is the control chart weak tagging. Its output is the association of $(\mathbf{x}, \bar{y})$ to an information granule. The $(\mathbf{x}, \bar{y})$ class can not change in the fuzzy-granular classification, but it is refined by a data clustering inside its class (the granule). In this way, the data is classified by criticality by the self-learning control chart method, and by data similarity by the fuzzy-granular methods.

Figure B.6 shows an example of the methodology output. Each point represents a time interval of the system execution. In this case, it is shown just two classes, in which the 'class 1' is represented by green points and the 'class 2' by red points. The information granules appear as rectangles in which

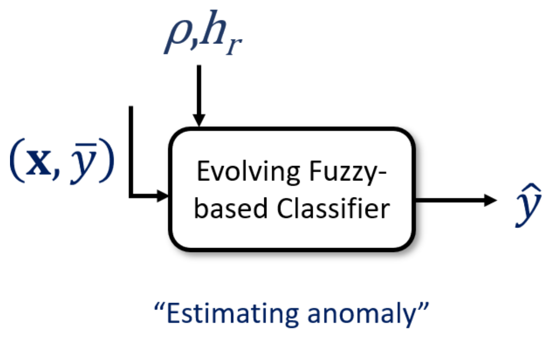


Figure B.5: Anomaly Detection fuzzy-granular approaches receive as input the data pair, and generate a more specific classification output, detailing each possible class in information granules.

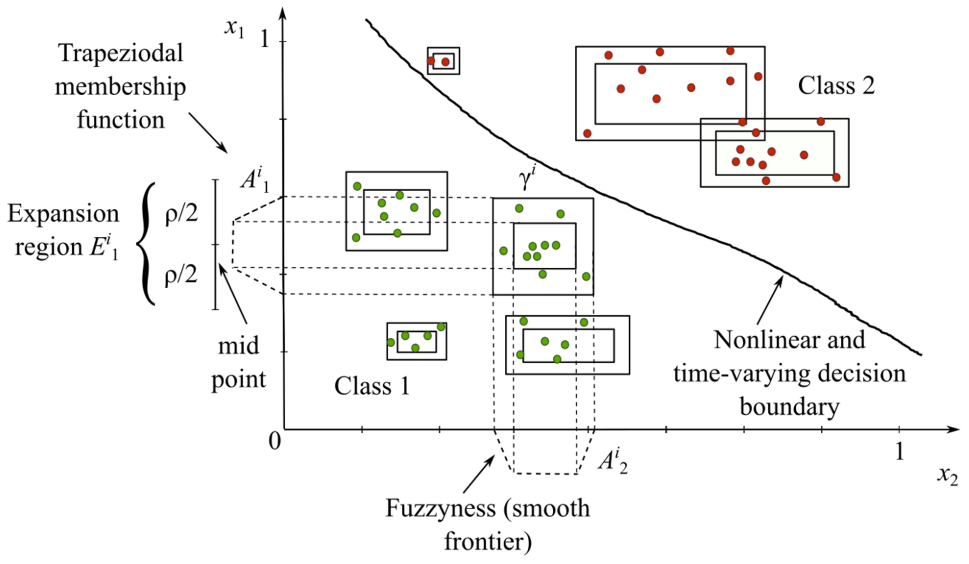

x_1
1
Trapeziodal membership function
A^i_1
Expansion region E^i_1
ρ/2
ρ/2
mid point
γ^i
Class 1
Class 2
Nonlinear and time-varying decision boundary
0
A^i_2
1
x_2
Fuzzyness (smooth frontier)

Figure B.6: Example of Anomaly detection output given by FBeM classification, generating a nonlinear and time-varying decision boundary of the classes. Granule border is fuzzy defined by a smooth frontier through a trapezoidal membership function.

the inner one is the granule boundary that can be expanded until the limits of the external rectangle. In this way, more similar data are clustered inside the classes in information granules format, being used as input data to the evolving log parser.

B.3.2 Log Parsing

Logs are one of the most valued sources of information for large-scale service maintenance [102]. They can be seen as written expressions, with poor syntax. In fact, log files are created by using an austere and restricted idiom. Most learning and monitoring approaches handles logs as strings. Many studies link the importance of addressing anomaly detection together with log parsing. The data-parsing task tends to increase the ability of a model to recognise anomalous behaviors through data structuring. This work considers log parsing as an abstraction function that generates information in a structured way, thus summarising multiple log data.

Here it is presented the evolving granular log parser called eLP to extract a formal grammar from textual data streams, and stores local grammars into information granules. Granules are described by word vectors and an interval rule base, which is evolved on the fly in a textual scenario. In particular, while the vast majority of evolving intelligent methods are focused on numerical point-wise data processing, interval data streams are considered by the granular methods in [112] [113], and fuzzy data streams are evaluated in [60] [114] [62]. Nevertheless, for the first time in the evolving intelligence literature, the eLP method is able to process streams of words and sentences.

We assume that the highly anomaly data previously classified (Sec. B.3.1) have more probability to be linked to the anomalous event, being primarily processed by log parser in order to minimise costs. Since decision-making systems can be impacted by both excess of irrelevant details or lack of important information, the analysed data must to be previously selected, avoiding contamination of the used data.

In this step, the log stream, $\tilde{L}$ is observed by instances $\mathbf{x}^{[h]}$, $h = 1, 2, \ldots$, in which $\mathbf{x}^{[h]}$ is a log message composed by a timestamp followed by the message itself, is written in a file continuously. $\mathbf{x}^{[h]}$ is associated with $(\mathbf{m}^i, \mathbf{a'}^{[h]})$, i.e., a template $\mathbf{m}^i \in M = \{\mathbf{m}^1, \ldots, \mathbf{m}^c\}$, and an attribute vector $\mathbf{a'}^{[h]}$, in which both of them have the word-vector format and a word w is formed by one or more characters. $\mathbf{a'}^{[h]}$ is the expression of the particularities of the occurred event.

The eLP method generates a set of granules in which each granule is defined by a formal grammar $\mathcal{G}$ [115]. The classification method aims to associate logs to a structured template format, identifying in this way its message type that is unique and indicates the execution of a specific piece of software.

Figure B.7 summarises the Log Parser functioning. eLP receives a log stream and generates a stream of message types, identified by numerical identifiers,

B.3.3 Event Tracking

To couple the two previous approaches, the highest anomalous data is parsed and analysed according with the information granule in which it is associated. Since each log message has a timestamp, the granule data can be organized temporally in order to identify the occurrence of an anomalous

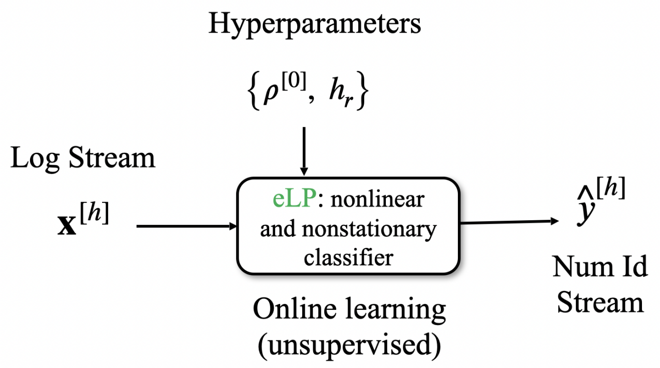


Figure B.7: evolving granular Log Parser receives as input the log message, and identify its template.

event, extracting its details from the message attributes. Numerical patters can be identified by the fuzzy-granular methods already mentioned to correlate numerical sequences to failures, threats or bugs.

B.3.4 Code Availability and Setting up

The software implementation is available at the GitHub link [1]. In case of doubt, contact Leticia Decker by email leticia.deckerde@unibo.it or letsdecker@gmail.com with the subject 'Support – Fuzzy-Granular System Maintenance'.

[1]Available from December 2022 on `https://github.com/letsdecker`.

Appendix C

Non-granular Log-based approaches

The present work approaches the log-based system maintenance problem from the traditional perspective. The aim is to detect software faults, bugs, threats, and infrastructural problems. It is proposed a general-purpose methodology to anomaly detection in computer grids using unstructured, textual, and unsupervised data. The solution consists in recognising periods of anomalous activity based on content and information extracted from user log events. This study has particularly compared One-class SVM, Isolation Forest (IF), and Local Outlier Factor (LOF) algorithms. IF provides the best fault detection accuracy, 69.5%.

C.1 Intelligent Diagnostic Maintenance Framework

In this section, an intelligent Diagnostic Maintenance Framework (DMF) is addressed, aiming to identify anomalous time intervals via log mining. The framework is formed by 3 component blocks: (I) log preprocessing (LP); (II) anomaly detection clustering (ADC); and (III) anomaly selection (AS). The block diagram of the proposed solution is shown in Figure C.1.

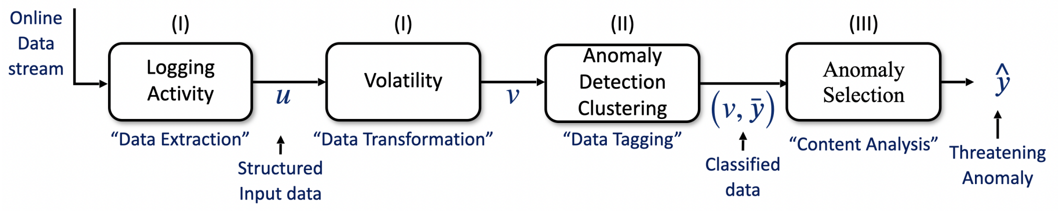


Figure C.1: Flowchart of the proposed DMF framework

In (I), the logging activity, **u**, is extracted from the online log stream (Data Extraction). A new stream is generated, which is later transformed into a volatility **v** stream (Data Transformation) of the running service. In this step, the textual and unstructured data set is formatted as a numerical data. Thereafter, the **v** stream, whose elements summarise the log data during a time window, is classified as normal or anomalous, as illustrated in the Tagging Data block (II). In this step, a sample can be classified by a ML solution. We consider: (i) One-class SVM; (ii) Isolation Forest (IF); and (iii) the Local Outlier Factor (LOF) methods. We aim at separating anomalous and usual time periods. Next, in (III), classification results are evaluated using Term Frequency-Inverse Document Frequency ($TFIDF$) sentiment analysis (Content Analysis). $TFIDF$ identifies which anomaly threaten the system stability, since not every anomaly is a menace. The methodology is a highly interpretable and a general-purpose solution used to identify the percentage of anomalous data are the best sources to a feature extraction maintaining a polynomial complexity of time execution. The computational complexity of the framework depends on the chosen clustering method in (II), varying from $\mathcal{O}(n)$ to $\mathcal{O}(n^3)$. The process is detailed below.

C.1.1 Log Preprocessing

Logs are time series of textual sentences. They register event occurrences in the source code executed during the system run-time. Log Preprocessing (LP) involves three streams of data. From the first, raw log stream, a second, log activity stream $\mathbf{u}$, is extracted. Subsequently, the latter is converted in a volatility stream $\mathbf{v}$. The LP implementation has complexity linear in the number of lines of the log data, $\mathcal{O}(n)$.

C.1.2 Volatility

Originally used to detect oscillations in stock markets, the volatility can be applied to $\mathbf{u}$ to measure trend changes. An increase of $\mathbf{u}$ indicates a decrease of the system stability. The volatility $\mathbf{v}$ at the instant k is

$$v^k = \frac{1}{m}\sqrt{\sum_{j=1}^{m}(\mu(\mathbf{u}_{X^k}) - u^j_{X^k})} \tag{C.1}$$

in which

$$\mathbf{u}_{X^k} = [u^1_{X^k}, u^2_{X^k}, \dots, u^m_{X^k}] \tag{C.2}$$

$\mathbf{u}_{X^k}$ is a set, with $|\mathbf{u}_{X^k}| = w_{vol} = m$. $\mathbf{u}_{X^k}$ with $k = 1, 2, \dots$ is a time series of $\mathbf{u}_{X^k}$ associated to the landmark time window w_{vol}. $\mathbf{v}$ is a time series of the standard deviation of $\mathbf{u}_{X^k}$, in which each element is bounded by $[0, \infty]$; $u^j_{X^k}$ is the j-th log activity in the k-th $\mathbf{u}_X$. The metric is used as input data of classifiers. Since an online solution is proposed, the computational complexity to generate $\mathbf{v}$ is $\mathcal{O}(\frac{n}{w_{vol}})$, which reduces to $\mathcal{O}(n)$ since w_{vol} is constant; the memory complexity is $\mathcal{O}(1)$, since each element of the $\mathbf{v}$ stream is scanned only once. Each 'anomalous $\mathbf{v}$' provides the time interval of an anomaly occurrence, which is sufficient to identify the exact piece of log data that should be highlighted in further analysis.

C.1.3 Anomaly Detection Clustering

Unsupervised methods do not require a training dataset, being the most widely applicable techniques. These methods generally assume that usual instances are much more frequent than anomalies, and for this reason, such techniques can suffer from high false alarm rate. Based on that, the clustering step aims to identify anomalies in the log production, and potentially correlates them with failure/error/threat occurrences.

Outlier detection, or AD, is a hot issue in Data Mining. Classifiers differ in the strategies used to generate class boundaries. In this subsection, a brief overview of the methods we used to cluster log data, i.e., One-class SVM, IF, and LOF, is given. The Scikit-learn library to Python 3 is used to implement the methods.

One-Class SVM is an unsupervised method to classify new data as similar or different related to the known fraction of the dataset. It generates a hyper-plane to optimally separate anomalous and usual instances based on support vectors. Its computational complexity varies from $\mathcal{O}(n^2)$ to $\mathcal{O}(n^3)$, depending on the dataset.

IF is a well-known branch of outlier detection methods, being sensitive to global outliers only, and having a weak detection performance for local outliers. The IF implementation returns an anomaly score for each sample. The method isolates the anomalous samples by randomly choosing a feature, and then setting a random split threshold between the possible range values of this feature. The number of splitting steps required to isolate an input data is equal to the path length from the root to the leaf node in the tree generated by the recursive algorithm. The path length is a measure of the sample normality averaged over a forest of such random trees. Random partitioning generates considerably smaller paths for anomalous samples compared to usual samples. IF computational complexity is $\mathcal{O}(n)$.

In general, LOF implementations are better for local outlier detection, and are known for having a higher time complexity. Its complexity depends on the size of the dataset, i.e., $\mathcal{O}(n)$ for smaller datasets, tending to $\mathcal{O}(n^2)$ for larger dataset. This unsupervised AD method computes the local density deviation of a given input related to its neighbors. Anomaly identification is based on feature density as outlier samples show a considerably lower density than its nearest neighbors. Typically, the number of neighbors must be greater than the minimum expected number of cluster samples – thus allowing local outliers in the cluster – and smaller than the maximum number of local outliers. Since this information generally is not available, taking 20 neighbors is usually the employed strategy.

C.1.4 Anomaly Selection

In this phase, we proceed information retrieval, i.e., a content-mining processing that searches for specific information over the classified log data. For simplicity, the naive version of $TFIDF$ method is chosen. The approach is based on the calculus of TF and IDF in a collection of documents. The TF ranks the item i based on its importance $TF_{i,j}$ on the document j,

$$TF_{i,j} = \frac{n_{i,j}}{N_j} \tag{C.3}$$

being $n_{i,j}$ the number of occurrences of the term i in the document j; and N_j the total number of items in j. The IDF counts the number of relevant documents in the collection related to the term i,

$$IDF_{i,D} = log\left[\frac{N}{1 + |j \in D : i \in j|},\right] \tag{C.4}$$

in which D is the collection of documents with $|D| = N$, and $1 + |j \in D : i \in j|$ indicates the number of documents in which the term i appears at least once. $TFIDF$ is the result of the multiplication of the terms. The higher the $TF_{i,j}$ and the $IDF_{i,D}$, the higher the importance of the term i in D, i.e., the term i must be frequent in a small number of documents j in D to be ranked as important.

As unsupervised data is used, this step validates the quality of the data classification. The $TFIDF$ score is generated to each word, giving a ranking of words. According to the $TFIDF$ rank, a time period is tagged to confirm or refute the previous unsupervised classification, using sentiment analysis as the basis. This analysis is detailed in the Methodology Section.

The $TFIDF$ sentiment analysis has computational complexity $\mathcal{O}(knm)$, in which m is the quantity of words in each line in n, and k is the size of the reference list of words. Since $m \ll n$, m can be considered constant, as well as k, and, consequently, the computational complexity converges to $\mathcal{O}(n)$

C.1.5 Algorithm

The algorithm below summarises the DMF framework discussed in the previous subsections and shown in Figure C.1. It is implemented in Python 3, using Scikit-learn libraries. The LP stage is implemented in lines 6-17, in which in lines 8-10, it is calculated $\mathbf{u}_{X^k}$ that is used to generate $\mathbf{v}$ in lines 11-16. As an online method, the LP stage is done continuously, and the data is classified as soon it is available. In lines 8 and 13, i and j are used in w_{log} and w_{vol}, respectively. In lines 17-22, the ADC module considers one of the 3 described ML methods. The anomalous $\mathbf{v}$ can be saved to further maintenance analysis. In lines 23-27, the AS module validates the classification using $TFIDF$ as Sentiment Analysis method.

The DMF Framework

```
1:  //Indices initialization:
2:  $t = k = 1$
3:  //Landmark time-window length:
4:  Read $w_{log}$//to generate the $\mathbf{u}$ stream
5:  Read $w_{vol}$//to generate the $\mathbf{v}$ stream
6:  //Log pre-processing
7:  while not $endOfLogFile$ do
8:      Read sample set $X^t = \{x^i | i \in [\overline{w_{log}}, \underline{w_{log}}]\}$
9:      Extract logging activity $\mathbf{u}^j_{X^k}$
10:     Increment $t$
11:     If $t == w_{vol}$ then
12:         While $k < t$ do
```

```
13:         Calculate $v^k(\{u^j_{X^t} | j \in [\overline{w_{vol}}, \underline{w_{vol}}]\})$
14:         Increment $k$
15:       **endWhile**
16:    **endIf**
17:    //Clustering for Anomaly Detection
18:    $t = 1$, Use an unsupervised method to tag $v^k$ as normal or anomaly
19:    **If** $v^k$ is anomalous **then**
20:      Insert in $\mathbf{v}^k$ in $\mathbf{Y}$
21:    **endIf**
22: **enfWhile**
23: Return $\mathbf{Y}$, the set of anomalous periods
24: //Anomaly Selection
25: **For** $i \in \mathbf{Y}$ **do**
26:    Apply on $i$ the $TFIDF$ Sentiment Analysis
27: **endFor**
```

C.2 Methodology

The described framework provides a general-purpose methodology to approach the online AD problem considering typical log data available in computing centers. As usual, this data must be event-oriented, i.e., the textual lines written in the log file must be indexed to real-time system occurrences. Therefore, logs can be directly converted to a stream, changing the data structure from sentence-based to well-behaved numerical vectors. However, the stream cannot distinguish between up and down variations, highlighting trend changing only in the service behavior. It is used to identify event precursors. The present paper uses real-world logs from the INFN-CNAF computing center. No privileged information is extracted to improve classification quality. The classification results are based exclusively in the metric.

C.2.1 About the Dataset

Logs consist of a sequence of lines formed by the timestamp and the log message content. Timestamps record the instant in which the log message was written. Each log file has a set of possible messages. The log production depends on events occurred during a time window, reaching up to millions of lines per day for a single log file from a single service. As a result, logs are preprocessed using data re-sampling and scaling.

Usually, computing centers have a large amount of logs available from hundreds of running subsystems on their infrastructure. Since logs are often heterogeneous, unstructured, and textual, they significantly vary in format and semantics from system to system. For this reason, developing a general-purpose solution using logs is an extremely challenging and important task.

The proposed solution is designed to support any kind of logs. Content analysis provided by the AS module is an additional step used to identify the percentage of threatening anomalies within the anomalous data. The log analysis is focused on an 8-day dataset of the StoRM service, from May 21 to May 28, 2019 – a time period known as being potentially anomalous from previous analyses. The INFN-CNAF IT staff could not provide insights on what happened during this period. Figure C.3 shows a peak of log production on May 23. Log production was around 3 times greater than the usual production, evidencing an anomaly.

C.3 Experimental Design

Consider a collection of documents, D. Each document j has a list of terms sorted in descending order in relation to the $TFIDF$ score. To validate the document class label, lists of words labeled as positive and negative are selected from the analysed documents. It is assumed that negative words are more frequent in anomalous context, and positive words are more frequent in usual context. This simplification is not always true since the anomalous contexts can be due to the increase of network activity or other occurrences not necessarily associated with threats.

In the end, the scored list of words of the document of is used in sentiment analysis. The lists are built manually using the set of documents and non-specialised linguistic knowledge. For example, words such as “error”, “failure”, “abort”, “broken”, and “warn” are negative; whereas “available”, “successfully”, “completed”, and “updated” are positive. The sums of all positive and negative scores are used to validate the class label associated with the j-th document, i.e., if the sum of the positive scores is greater than the sum of negatives, then the document is classified as normal; otherwise, it is anomalous. Better reference lists improve the reliability of the validation process. Since, e.g., a 1-day log file contains a sequence of normal and anomalous time periods, we separated them in 2 groups according to their classes. For the $TFIDF$ analysis, we built two sets of documents: (i) a single anomalous document, and a set of normal documents (D_1), or (ii) a single normal document, and a set of anomalous documents (D_2). This strategy is used to get higher the scores of anomalous terms in D_1, and to get higher the scores of the normal terms in D_2, since $TFIDF$ analysis takes into account the term relevance related to the set of documents. As an example, consider the existence of 20 anomalous and 21 normal time intervals during a day. The set D_1 uses 1 anomalous and 21 normal documents; D_2 uses 1 normal and 20 anomalous documents.

C.4 Data Analysis

Content-based classification is used to generate confusion matrices for the unsupervised ML methods. The aim is to evaluate the methodology accuracy through the AS strategy. Consider confusion matrices to binary classification, as shown in Figure C.2. The y-axis indicates the estimated classes, and the x-axis, the target classes. The 2x2 matrix aligned on the top-left side provides the classification results related to the possible classes (normal and anomaly), i.e., True Positive (TP), False Positive (FP) or Type I Error, False Negative (FN) or Type II Error, and True Negative (TN). These values are expressed as a percentage mean and a standard deviation in Results Section. The square at the right-down corner provides the global accuracy of a method. The other squares of the last column give the sensitivity (TPR) and specificity (TNR), from up to down, and the other squares of the last line give the precision (PPV) and negative precision value (NPV), from left to right.

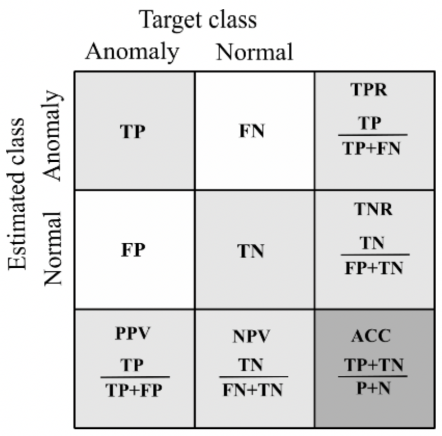


Figure C.2: Confusion matrix as classification performance measure

C.5 Results

This section describes the experimental results of the DMF experiments, as described in the Methodology Section. The LP step is the same for all experiments, as previously described in the Log Preprocessing sub-section. The AS procedure uses the $TFIDF$ algorithm (see the Anomaly Selection sub-section) to validate a classifier of the ADC module and identify which anomalies represent threats.

The w_{log} and w_{vol} time-windows are set to 5 and 30 minutes, respectively. The stream is previously normalised using the highest daily value, avoiding a scale-factor bias towards higher values. The clustering algorithms are One-Class SVM, IF, and LOF (addressed in the Intelligent DMF Section). The initial hyper-parameters of the methods are summarized in Table C.1.

Table C.1: Parameters used by the unsupervised learning methods

Algorithms	Initial Hyper-parameters
One-class SVM	$kernel = rbf$, $v = 0.2$
IF	$random_{state} = 1$, $n_{estimators} = 100$, $grid^{*}_{search}$
LOF	$n_{neighbors} = 20$, $grid_{search}{}^{1}$

The experiments are done using a stressful 8-day StoRM log data scenario, from May 21 to May 28 2019, with an anomaly epicentre on May 23, which produced a 3.87GB log file (around 3 times greater than a usual log file, see Figure C.3). The underlying scenario is the worst case of the AD framework since the methodology considers the anomaly as a rare occurrence, and at least a clear anomaly occurred in the period.

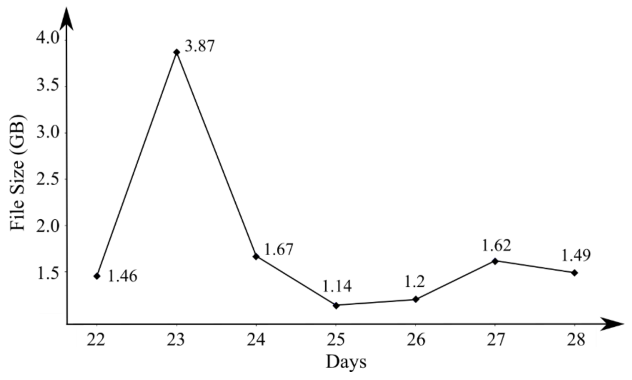


Figure C.3: Example of a peak of log production (day 23) as a sign of an anomalous occurrence

Table C.2 shows the DMF performance in terms of execution time, and percentage of memory and CPU usage in an Intel i5 quad-core computer with 16GB of RAM considering the epicentre log file. The percentage of CPU usage is related to a single core. According with the results, the IF (Isolation Forest) implementation is the most efficient method in execution time and CPU usage, being around 4% better than the other methods. IF and LOF are the best methods considering memory usage – although this result is not statistically different. All experiments were run 10 times considering the LP and ADC stages of the proposed framework to generate the Table C.2. The Table C.2 results express the workability of the framework in a low-cost solution in terms of time consumption of analysis execution, keeping the complexity of the code implementation in a feasible level. Since it is used just standard python libraries, the code maintenance is also facilitated.

Table C.2: Computational resource consumption during I and II stages of the proposed framework

Algorithms	Execution Time(s)	CPU Usage(%)	Memory Usage(%)
One-class SVM	124.4 ± 3.2	98.8 ± 0.5	25.1 ± 8.7
IF	106.6 ± 1.4	80.1 ± 14.2	18.2 ± 8.5
LOF	120.4 ± 2.8	97.9 ± 0.8	15.7 ± 0.7

Figures C.4 shows the statistics related with: the percentage of time associated to anomaly periods (Figure C.4.a); and amount of anomalous periods (Figure C.4.b). A sliding windows of 4 days is used to generate means and standard deviations showed in the figures. Since clustering is based on outlier detection, the time related to anomalous periods is much smaller than the time associated to normal periods as a premise. This premise can be observed especially for the IF implementation (Figure C.4.a). Concurrently, IF splits a window in more time periods than the other methods, even though it classifies a smaller amount of time periods as anomalous. The double skill of the IF behavior promote the best experimental classification result, detailed later in Figure 5.b. Both (LOF and IF) algorithms are slightly affected by the anomalous peak – which is expressed by plateaus and narrowed deviations

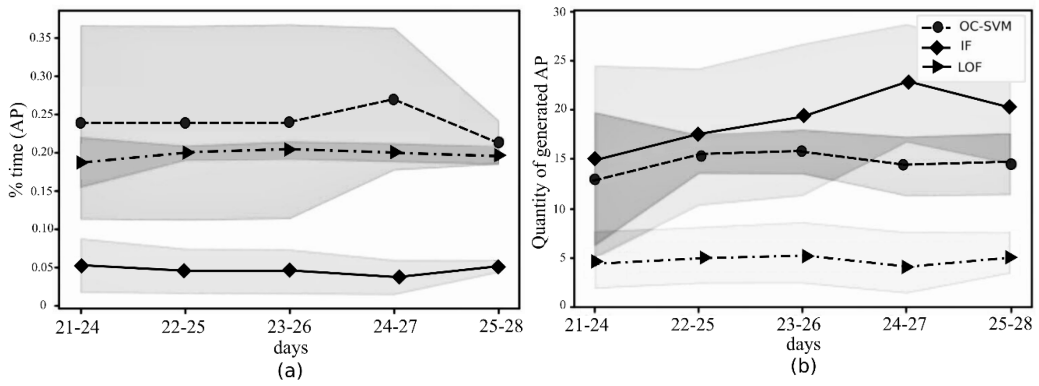
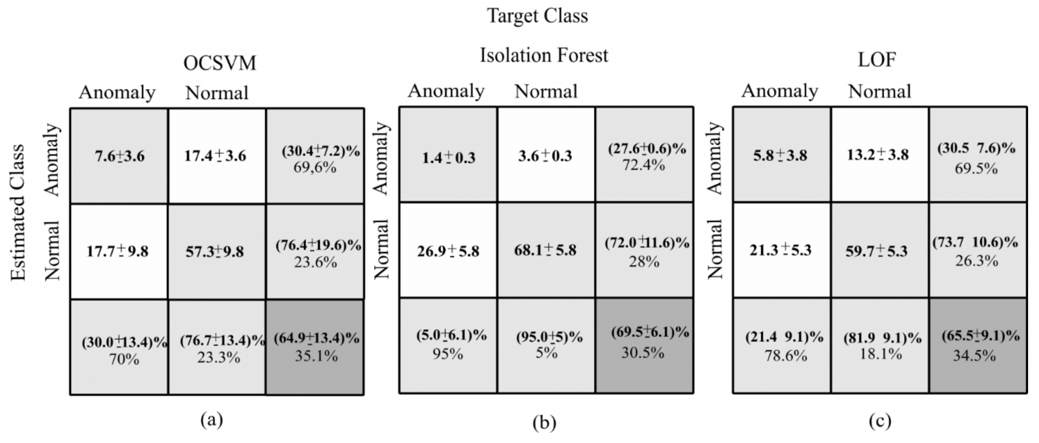


Figure C.4: Evolving behavior of the StoRM log data during the observed anomalous period: (a) percentage of anomalies detected; (b) amount of generated anomalous periods using a 4-day time window

in Figures C.4.a-b. One-class SVM shows a higher number of time periods classified as anomalous, with larger standard deviation throughout the whole time period, which contrasts with the well-behaved standard deviations for LOF and IF. The quality of classification of One-class SVM is more affected during the anomalous occurrences compared to the quality of the other two classification approaches. The standard deviation in Figure C.4.a can be considered a measurement of stability of the log-based clustering method in a stressful scenario.

Figures C.5 shows the results of the classifiers through the AS method. IF (Figure C.5.b) provides the best accuracy (69.5%) and best negative precision (95.0%), but the worst result for precision (5.0%). All methods have comparable results related to sensitivity (around 30%), and specificity (around 72%). As expected for outlier classifiers, all methods have shown a higher rate for Type I error.



Target Class

Estimated Class

(a) OCSVM

	Anomaly	Normal	
Anomaly	7.6±3.6	17.4±3.6	(30.4±7.2)% 69,6%
Normal	17.7±9.8	57.3±9.8	(76.4±19.6)% 23.6%
	(30.0±13.4)% 70%	(76.7±13.4)% 23.3%	(64.9±13.4)% 35.1%

(b) Isolation Forest

	Anomaly	Normal	
Anomaly	1.4±0.3	3.6±0.3	(27.6±0.6)% 72.4%
Normal	26.9±5.8	68.1±5.8	(72.0±11.6)% 28%
	(5.0±6.1)% 95%	(95.0±5)% 5%	(69.5±6.1)% 30.5%

(c) LOF

	Anomaly	Normal	
Anomaly	5.8±3.8	13.2±3.8	(30.5 7.6)% 69.5%
Normal	21.3±5.3	59.7±5.3	(73.7 10.6)% 26.3%
	(21.4 9.1)% 78.6%	(81.9 9.1)% 18.1%	(65.5±9.1)% 34.5%

Figure C.5: Confusion matrices of the classifiers: (a) One-class SVM, (b) IF, and (c) LOF. The IF method has shown the best result for accuracy (69.5%). To understand the third line and column, see Figure C.2

The precision and the sensitivity are strongly impacted by the assumption that all anomalies are closely associated to a threat, failure, or a system infrastructural problem. In fact, there are scenarios that are not threatening, but impact the log production and, as a result, are also considered as anomalies time intervals, e.g., an increasing of network traffic or resource consumption in the cloud system.

To improve the methodology classification, it is needed to evolve the dictionary used by the $TFIDF$ sentiment analysis through an online approach. Presently, the valuation dictionary is static, being the bigger framework limitation. An online extraction of terms can promote a scouting of the dictionary terms, helping to individuate the thread type. Other possibility is to split the anomalous occurrences in more classes based on the cause of the anomalous trend. This problem-solve approach would improve the classification results in return for reducing the framework applicability, since the strategy migrates from a general-purpose to an ad-hoc solution, decreasing its interpretability. In addiction, the $TFIDF$ sentiment analysis can be used as self-learning method to classify anomalous periods, and generate supervised datasets. The best methodology advantage is the high level of the framework workability, keeping the a low-cost of time consumption related with analysis execution with best results around 70% of accuracy. As a support activity, system maintenance must be computationally cheap besides being desirable to minimise the code complexity.

C.6 Conclusion

A general-purpose Diagnostic Maintenance Framework to the log-based anomaly detection problem is described using standard implementation methods. Event-based time-varying log data, as an input data stream, is considered. The framework provides a methodology to classify unsupervised, textual, and unstructured data; and to detect anomalous occurrences through data clustering. Classifier validation is given by a $TFIDF$ sentiment analysis. Three machine learning methods, namely, One-class SVM, Isolation Forest, and Local Outlier Factor, are compared. Isolation Forest has shown a slightly better result than the remaining methods, with an accuracy around 70%. The classifiers precision and sensitivity, as reported in Confusion Matrices, are low for all methods – since the $TFIDF$ validation considered only anomalies that cause threats, failures, or infrastructural damages. Detected anomalies can also represent an uncommon use of the system services (highlighted by the increase of the data throughput, task executions, latency, packet loss rate), but not necessarily be associated to system damage.

An alternative to improve sensitivity is to reduce the FN rate (anomalous data classified as normal) through changing ML methods and hyper-parameters; and emphasise data processing with focus on the data classified as anomalous. This strategy is based on the fact that the amount of anomalous instances is much smaller than that of normal instances, varying from 5% to 35% of the total amount of log data. Another alternative is to develop methods that aim at improving the quality of lists of sentiment words used to validate the clustering result.

Adaptive classification methods can be better guided by means of the accuracy feedback given by the $TFIDF$ sentiment analysis. In the future, we will identify the type of message associated to anomalous time windows through image recognition based on $TFIDF$ graphics generated from classified logs. In any case, the general-purpose methodology and framework to the log-based AD problem described in the present work can be applied in any of these real-world scenarios. Another important point to highlight is the expressive superiority of the experimental results of the fuzzy-granular approaches related to the traditional ML methods to system maintenance problems, since fuzzy-granular strategies take advantage of the uncertain and imprecise nature of logs.

Bibliography

[1] M.M. Lehman. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9):1060–1076, 1980. (cit. on pp. 13, 14)

[2] Israel Herraiz, Daniel Rodriguez, Gregorio Robles, and Jesus M. Gonzalez-Barahona. The evolution of the laws of software evolution: A discussion based on a systematic literature review. *ACM Comput. Surv.*, 46(2), December 2013. (cit. on pp. 13, 14)

[3] Alessandro Di Girolamo et al. Operational intelligence for distributed computing systems for exascale science. In *24th Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP), AU*, pages 1–8, 2020. (cit. on pp. 19)

[4] W. Herr and B. Muratori. Concept of luminosity. In *CAS - CERN Accelerator School: Intermediate Course on Accelerator Physics*, 2006. (cit. on pp. 19)

[5] Johannes Albrecht et al. A Roadmap for HEP Software and Computing R&D for the 2020s. *Comput Softw Big Sci*, 3, 2019. (cit. on pp. 19)

[6] Leticia Decker de Sousa et al. Big data analysis for predictive maintenance at the INFN-CNAF data center using machine learning approaches. In *Proc. of the 25th Conf. of Open Innovations Association (FRUCT). Helsinki, Finland*, pages 448–451, 2019. (cit. on pp. 20, 22, 131)

[7] INFN. Infn-cnaf. url = https://www.cnaf.infn.it/wlcg-tier-1-data-center/. (cit. on pp. 20)

[8] CERN. High-luminosity lhc. url = https://home.cern/science/accelerators/high-luminosity-lhc. (cit. on pp. 21)

[9] T Diotalevi, A Falabella, B Martelli, D Michelotto, L Morganti, D Bonacorsi, L Giommi, and S Rossi Tisbeni. Collection and harmonization of system logs and prototypal analytics services with the elastic (elk) suite at the infn-cnaf computing centre. In *International Symposium on Grids & Clouds 2019 (ISGC 2019)*, Taipei, Taiwan, 2019. Proceedings of Science. (cit. on pp. 21)

[10] L Giommi, T Diotalevi, A Falabella, B Martelli, L Morganti, L Rinaldi, D Bonacorsi, S Rossi Tisbeni, E Ronchieri, and A Ceccanti. Towards predictive maintenance with machine learning at the infn-cnaf computing centre. In *International Symposium on Grids & Clouds 2019 (ISGC 2019)*, Taipei, Taiwan, 2019. Proceedings of Science. (cit. on pp. 21, 79)

[11] Simone Rossi Tisbeni. Big data analytics towards predictive maintenance at the INFN-CNAF computing centre. Master's thesis, U. of Bologna, 2019. (cit. on pp. 21)

[12] Francesco Minarini. Anomaly detection prototype for log-based predictive maintenance at INFN-CNAF. Master's thesis, University of Bologna, 2019. (cit. on pp. 21, 47)

[13] Francesco Minarini and Leticia Decker. Time-series anomaly detection applied to log-based diagnostic system using unsupervised machine learning approach. In *Proc. of the 27th Conf. of Open Innovations Association (FRUCT). Trento , Italy*, pages 343–348, 2020. (cit. on pp. 21, 47)

[14] Riyaz Ahamed Ariyaluran Habeeb et al. Real-time big data processing for anomaly detection: A survey. *Int. Journal of Information Management*, 45:289–307, 2019. (cit. on pp. 21)

[15] Bernard Schmidt et al. Predictive maintenance of machine tool linear axes: A case from manufacturing industry. *Procedia Manufacturing*, 17:118–125, 2018. (cit. on pp. 21)

[16] Chuan-Jun Su et al. Real-time big data analytics for hard disk drive predictive maintenance. *Computers & Electrical Engineering*, 71:93–101, 2018. (cit. on pp. 21, 46)

[17] Amruta Ambre and Narendra Shekokar. Insider threat detection using log analysis and event correlation. In *Procedia Computer Science*, volume 45, pages 436–445. Elsevier B.V., 2015. (cit. on pp. 21)

[18] Claudio Ciccotelli. *Practical Fault Detection and Diagnosis in Data Centers.* PhD thesis, Sapienza Università di Roma, Roma, Italy, dec 2016. (cit. on pp. 21)

[19] Shikha Agrawal and Jitendra Agrawal. Survey on anomaly detection using data mining techniques. In *Procedia Computer Science*, volume 60-1, pages 708–713. Elsevier B.V., 2015. (cit. on pp. 21, 23)

[20] Riyaz Ahamed Ariyaluran Habeeb, Fariza Nasaruddin, Abdullah Gani, Ibrahim Abaker Targio Hashem, Ejaz Ahmed, and Muhammad Imran. Real-time big data processing for anomaly detection: A survey. In *International Journal of Information Management*, volume 45, pages 289–307. Elsevier Ltd, apr 2019. (cit. on pp. 21, 23)

[21] Nuno Pereira et al. Building a microscope for the data center. In *Int. Conf. on Wireless Algorithms, Systems, and Applications*, pages 619–630. Springer, 2012. (cit. on pp. 21)

[22] Chuan Jun Su and Shi Feng Huang. Real-time big data analytics for hard disk drive predictive maintenance. *Computers and Electrical Engineering*, 71:93–101, oct 2018. (cit. on pp. 21, 25)

[23] Jeanderson Candido, Mauricio Aniche, and Arie van Deursen. Log-based software monitoring: a systematic mapping study. *PEERJ COMPUTER SCIENCE*, MAY 6 2021. (cit. on pp. 21, 23)

[24] Flavio Trojan and Rui Marçal. Proposal of maintenance-types classification to clarify maintenance concepts in production and operations management. *Journal of Business Economics*, 8:562–574, 2017. (cit. on pp. 22)

[25] Arthur Vervaet. MoniLog: An Automated Log-Based Anomaly Detection System for Cloud Computing Infrastructures. In *2021 IEEE 37TH INTERNATIONAL CONFERENCE ON DATA ENGINEERING (ICDE 2021)*, IEEE International Conference on Data Engineering, pages 2739–2743. IEEE, IEEE COMPUTER SOC, 2021. 37th IEEE International Conference on Data Engineering (IEEE ICDE), ELECTR NETWORK, APR 19-22, 2021. (cit. on pp. 23)

[26] Sarah M. Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, oct 2016. (cit. on pp. 23)

[27] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, nov 2017. (cit. on pp. 23)

[28] Li Zhang. The Research of Log-Based Network Monitoring System. In Lee, G, editor, *ADVANCES IN INTELLIGENT SYSTEMS*, volume 138 of *Advances in Intelligent and Soft Computing*, pages 315–320, HEIDELBERGER PLATZ 3, D-14197 BERLIN, GERMANY, 2012. SPRINGER-VERLAG BERLIN. International Conference on Control Systems (ICCS 2012), Hong Kong, PEOPLES R CHINA, MAR 01-02, 2012. (cit. on pp. 23)

[29] Omair Shafiq, Reda Alhajj, and Jon G. Rokne. Handling incomplete data using Semantic Logging based Social Network Analysis Hexagon for Effective Application Monitoring and Management. In Wu, X and Ester, M and Xu, G, editor, *2014 PROCEEDINGS OF THE IEEE/ACM INTERNATIONAL CONFERENCE ON ADVANCES IN SOCIAL NETWORKS ANALYSIS AND MINING (ASONAM 2014)*, pages 634–641, 345 E 47TH ST, NEW YORK, NY 10017 USA, 2014. IEEE; Assoc Comp Machinery; Assoc Comp Machinery SIGKDD; IEEE Comp Soc; IEEE TCDE; Springer; Tencent Internet; Soc Inst; Venustech Informat Technol Co Ltd, IEEE. IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Beijing, PEOPLES R CHINA, AUG 17-20, 2014. (cit. on pp. 23)

[30] Diana Martinez-Mosquera, Sergio Lujan-Mora, Gabriel Lopez, and Lauro Santos. Data Cleaning Technique for Security Logs Based on Fellegi-Sunter Theory. In Wrycza, S and Maslankowski, J, editor, *INFORMATION SYSTEMS: RESEARCH, DEVELOPMENT, APPLICATIONS, EDUCATION*, volume 300 of *Lecture Notes in Business Information Processing*, pages 3–12, HEIDELBERGER PLATZ 3, D-14197 BERLIN, GERMANY, 2017. SPRINGER-VERLAG BERLIN. (cit. on pp. 23)

[31] Okoye Kingsley, Abdel-Rahman H. Tawil, Usman Naeem, Syed Islam, and Elyes Lamine. Using Semantic-based Approach to Manage Perspectives of Process Mining: Application on Improving Learning Process Domain Data. In Joshi, J and Karypis, G and Liu, L and Hu, X and Ak, R and Xia, Y and Xu, W and Sato, AH and Rachuri, S and Ungar, L and Yu, PS and Govindaraju, R and Suzumura, T, editor, *2016 IEEE INTERNATIONAL CONFERENCE ON BIG DATA (BIG DATA)*, pages 3529–3538, 345 E 47TH ST, NEW YORK, NY 10017 USA, 2016. IEEE; IEEE Comp Soc; Natl Sci Fdn; Cisco; Huawei; Elsevier; Navigant; Johns Hopkins Whiting Sch

Engn, IEEE. 4th IEEE International Conference on Big Data (Big Data), Washington, DC, DEC 05-08, 2016. (cit. on pp. 24)

[32] ZQ Liu and F Yan. Fuzzy neural network in case-based diagnostic system. *IEEE TRANSACTIONS ON FUZZY SYSTEMS*, 5(2):209–222, MAY 1997. (cit. on pp. 24)

[33] XZ Wang. Knowledge discovery through mining process operational data. In Mujtaba, IM and Hussain, MA, editor, *APPLICATION OF NEURAL NETWORKS AND OTHER LEARNING TECHNOLOGIES IN PROCESS ENGINEERING*, pages 287–328, 2001. Workshop on Application of Neural Networks and Other Learning Technologies in Process Engineering, IMPERIAL COLL, LONDON, ENGLAND, MAY 12, 1999. (cit. on pp. 24)

[34] Soya Park, Ketan Talaulikar, and Chris Metz. RCV: Network Monitoring and Diagnostic System with Interactive User Interface. In Smari, WW, editor, *2016 INTERNATIONAL CONFERENCE ON COLLABORATION TECHNOLOGIES AND SYSTEMS (CTS)*, pages 578–583. Honeywell Int Inc; Knowledge Based Syst Inc; Ball Aerosp & Technologies Corp; Intel Corp; Microsoft Res; Springer Verlag, 2016. 17th International Conference on Collaboration Technologies and Systems (CTS), Orlando, FL, OCT 31-NOV 04, 2016. (cit. on pp. 24)

[35] Daniel L. C. Mack, Gautam Biswas, Xenofon D. Koutsoukos, and Dinkar Mylaraswamy. Learning Bayesian Network Structures to Augment Aircraft Diagnostic Reference Models. *IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING*, 14(1):358–369, JAN 2017. (cit. on pp. 24)

[36] Bernard Schmidt and Lihui Wang. Predictive maintenance of machine tool linear axes: A case from manufacturing industry. In *Procedia Manufacturing*, volume 17, pages 118–125. Elsevier B.V., 2018. (cit. on pp. 25)

[37] A.R. Mashhadi, W. Cabe, and S Behdad. Moving towards real-time data-driven quality monitoring: A case study of hard disk drives. In *Procedia Manufacturing*, volume 26, pages 1107–1115. Elsevier B.V., 2018. (cit. on pp. 25)

[38] Ruben Sipos, Dmitriy Fradkin, Fabian Moerchen, and Zhuang Wang. Log-based predictive maintenance. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 1867–1876, New York, NY, USA, 2014. Association for Computing Machinery. (cit. on pp. 25)

[39] Maren David Dangut, Zakwan Skaf, and Ian K. Jennions. An integrated machine learning model for aircraft components rare failure prognostics with log-based dataset. *ISA TRANSACTIONS*, 113:127–139, JUL 2021. (cit. on pp. 25)

[40] Maren David Dangut, Zakwan Skaf, and Ian K. Jennions. Rare Failure Prediction Using an Integrated Auto-encoder and Bidirectional Gated Recurrent Unit Network. *IFAC PAPERSONLINE*, 53(3):276–282, 2020. 4th International-Federation-of-Automatic-Control (IFAC) Workshop on Advanced Maintenance Engineering, Services and Technologies (AMEST), Cambridge, ENGLAND, SEP 10-11, 2020. (cit. on pp. 25)

[41] Clemens Gutschi, Nikolaus Furian, Josef Suschnigg, Dietmar Neubacher, and Siegfried Voessner. Log-based predictive maintenance in discrete parts manufacturing. In Teti, R and DAddona, DM, editor, *12TH CIRP CONFERENCE ON INTELLIGENT COMPUTATION IN MANUFACTURING ENGINEERING*, volume 79 of *Procedia CIRP*, pages 528–533. Int Acad Prod Engn; Fraunhofer Joint Lab Excellence Adv Prod Technol; CIRP, 2019. 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering (CIRP ICME), Naples, ITALY, JUL 18-20, 2018. (cit. on pp. 25)

[42] Ruben Sipos, Dmitriy Fradkin, Fabian Moerchen, and Zhuang Wang. Log-based Predictive Maintenance. In *PROCEEDINGS OF THE 20TH ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING (KDD'14)*, pages 1867–1876. Assoc Comp Machinery; ACM SIGKDD; ACM SIGMOD, 2014. 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), New York, NY, AUG 24-27, 2014. (cit. on pp. 26)

[43] Paul Abbott. *Mosby's Dictionary of Medicine, Nursing and Health Professions*, pages 1–2134. Mosby International, United States, 7th edition, 2005. (cit. on pp. 26)

[44] Daniel Leite. *Evolving Granular Systems*. PhD thesis, University of Campinas, Campinas, Brazil, july 2012. (cit. on pp. 33, 43, 44)

[45] Daniel Leite, Pyramo Costa, and Fernando Gomide. Granular approach for evolving system modeling. In Eyke Hüllermeier, Rudolf Kruse, and Frank Hoffmann, editors, *Computational Intelligence for Knowledge-Based Systems Design*, pages 340–349, Berlin, Heidelberg, 2010. Springer. (cit. on pp. 33, 48)

[46] Witold Pedrycz, Andrzej Skowron, and Vladik Kreinovich. *Handbook of Granular Computing*. Wiley-Interscience, USA, 2008. (cit. on pp. 33)

[47] Lotfi A. Zadeh. Generalized theory of uncertainty (gtu)—principal concepts and ideas. *Computational Statistics & Data Analysis*, 51(1):15–46, 2006. The Fuzzy Approach to Statistical Analysis. (cit. on pp. 38)

[48] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965. (cit. on pp. 38)

[49] Geoffrey Webb, Loong Lee, Bart Goethals, and François Petitjean. Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery*, 32, 09 2018. (cit. on pp. 43, 44)

[50] Igor Škrjanc, José Iglesias, Araceli Sanchis, Daniel Leite, Edwin Lughofer, and Fernando Gomide. Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey. *Inf. Sci.*, 490:344–368, 2019. (cit. on pp. 45, 51, 52, 72)

[51] L. A. Cordovil, P. H. Coutinho, I. Bessa, M. F. D'Angelo, and R. Palhares. Uncertain data modeling based on evolving ellipsoidal fuzzy information granules. *IEEE Transactions on Fuzzy Systems*, page 11p. DOI: doi.org/10.1109/TFUZZ.2019.2937052, 2019. (cit. on pp. 45, 48)

[52] Cristiano Garcia, Daniel Leite, and Igor Škrjanc. Incremental missing-data imputation for evolving fuzzy granular prediction. *IEEE T Fuzzy Syst.*, pages 1–15, 2019. DOI: 10.1109/T-FUZZ.2019.2935688. (cit. on pp. 45, 54)

[53] Richard Hyde, Plamen Angelov, and A. Mackenzie. Fully online clustering of evolving data streams into arbitrarily shaped clusters. *Inf. Sci.*, 382:1–41, 2016. (cit. on pp. 45)

[54] P. Silva, H. Sadaei, R. Ballini, and F. Guimaraes. Probabilistic forecasting with fuzzy time series. *IEEE Transactions on Fuzzy Systems*, page 14p. DOI: doi.org/10.1109/TFUZZ.2019.2922152, 2019. (cit. on pp. 45)

[55] Rajasekar Venkatesan, Meng Er, Mihika Dave, Mahardhika Pratama, and Shiqian Wu. A novel online multi-label classifier for high-speed streaming data applications. *Evolving Systems*, pages 303–315, 2016. (cit. on pp. 45)

[56] P. V. Souza, T. Rezende, A. Guimaraes, V. Araujo, L. Batista, G. Silva, and V. Silva. Evolving fuzzy neural networks to aid in the construction of systems specialists in cyber attacks. *Journal of Intelligent & Fuzzy Systems*, 36(6):6773–6763, 2019. (cit. on pp. 45)

[57] C. Bezerra, B. Costa, L. A. Guedes, and P. Angelov. An evolving approach to data streams clustering based on typicality and eccentricity data analytics. *Information Sciences*, 518:13–28, 2020. (cit. on pp. 45)

[58] M. Pratama, W. Pedrycz, and E. Lughofer. Online tool condition monitoring based on parsimonious ensemble+. *IEEE T Cybernetics*, 50(2):664–677, 2020. (cit. on pp. 45)

[59] Daniel Leite, Rosangela Ballini, Pyramo Costa Jr, and Fernando Gomide. Evolving fuzzy granular modeling from nonstationary fuzzy data streams. *Evolving Systems*, 3:65–79, 2012. (cit. on pp. 46, 50, 51, 54, 55, 104)

[60] Daniel Leite, Pyramo Costa Jr, and Fernando Gomide. Evolving granular neural networks from fuzzy data streams. *Neural Networks*, 38:1–16, 2013. (cit. on pp. 46, 48, 49, 56, 57, 58, 68, 104, 137)

[61] Leticia Decker, Daniel Leite, Luca Giommi, and Daniele Bonacorsi. Real-time anomaly detection in data centers for log-based predictive maintenance using an evolving fuzzy-rule-based approach. In *IEEE World Congress on Computational Intelligence (WCCI, FUZZ-IEEE), Glasgow*, page 8p, 2020. (cit. on pp. 46, 104)

[62] Daniel Leite, Leticia Decker, Marcio Santana, and Paulo Souza. EGFC: Evolving Gaussian fuzzy classifier from never-ending semi-supervised data streams - with application to power quality disturbance detection and classification. In *IEEE World Congress on Computational Intelligence (WCCI – FUZZ-IEEE), Glasgow*, page 8p, 2020. (cit. on pp. 46, 51, 55, 68, 72, 137)

[63] Shikha Agrawal et al. Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, 60:708–713, 2015. (cit. on pp. 46)

[64] S. He, J. Zhu, P. He, and M. R. Lyu. Experience report: System log analysis for anomaly detection. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pages 207–218, 2016. (cit. on pp. 46)

[65] M. Chen, A. Zheng, J. Lloyd, M. Jordan, and E. Brewer. Failure diagnosis using decision trees. In *Autonomic Computing, International Conference on*, pages 36–43, Los Alamitos, CA, USA, may 2004. IEEE Computer Society. (cit. on pp. 46)

[66] Mostafa Farshchi, Jean-Guy Schneider, Ingo Weber, and John Grundy. Metric selection and anomaly detection for cloud operations using log and metric correlation analysis. *Journal of Systems and Software*, 137:531 – 549, 2018. (cit. on pp. 46)

[67] R. Vinayakumar, K. P. Soman, and P. Poornachandran. Long short-term memory based operation log anomaly detection. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 236–242, 2017. (cit. on pp. 46)

[68] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. DeepLog. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, October 2017. (cit. on pp. 47)

[69] Lin Zhang, Xueshuo Xie, Kunpeng Xie, Zhi Wang, Ye Lu, and Yujun Zhang. An Efficient Log Parsing Algorithm Based on Heuristic Rules. In Yew, PC and Stenstrom, P and Wu, J and Gong, X and Li, T, editor, *ADVANCED PARALLEL PROCESSING TECHNOLOGIES (APPT 2019)*, volume 11719 of *Lecture Notes in Computer Science*, pages 123–134. China Comp Federat, 2019. 13th International Symposium on Advanced Parallel Processing Technologies (APPT), Tianjin, PEOPLES R CHINA, AUG 15-16, 2019. (cit. on pp. 47, 68)

[70] Bin Xia, Yuxuan Bai, Junjie Yin, Yun Li, and Jian Xu. LogGAN: a log-level generative adversarial network for anomaly detection using permutation event modeling. *Information Systems Frontiers*, June 2020. (cit. on pp. 47)

[71] Liang Bao, Qian Li, Peiyao Lu, Jie Lu, Tongxiao Ruan, and Ke Zhang. Execution anomaly detection in large-scale systems through console log analysis. *Journal of Systems and Software*, 143:172 – 186, 2018. (cit. on pp. 47)

[72] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo. Failure prediction in ibm bluegene/l event logs. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 583–588, 2007. (cit. on pp. 47)

[73] Bingming Wang, Shi Ying, Guoli Cheng, Rui Wang, Zhe Yang, and Bo Dong. Log-based anomaly detection with the improved k-nearest neighbor. *International Journal of Software Engineering and Knowledge Engineering*, 30(02):239–262, 2020. (cit. on pp. 47)

[74] Sarah M Erfani et al. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition*, 58:121–134, 2016. (cit. on pp. 47)

[75] Max Landauer, Florian Skopik, Markus Wurzenberger, and Andreas Rauber. System log clustering approaches for cyber security applications: A survey. *Computers & Security*, 92:101739, 2020. (cit. on pp. 47)

[76] Bingming Wang, Shi Ying, and Zhe Yang. A log-based anomaly detection method with efficient neighbor searching and automatic k neighbor selection. *Scientific Programming*, 2020:1–17, June 2020. (cit. on pp. 47)

[77] João Henriques, Filipe Caldeira, Tiago Cruz, and Paulo Simões. Combining k-means and XGBoost models for anomaly detection using log datasets. *Electronics*, 9(7):1164, July 2020. (cit. on pp. 47)

[78] F. T. Liu, K. M. Ting, and Z. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008. (cit. on pp. 47)

[79] Amir Farzad and T. Aaron Gulliver. Unsupervised log message anomaly detection. *ICT Express*, 6(3):229 – 237, 2020. (cit. on pp. 47)

[80] Witold Pedrycz. *Granular Computing : An Introduction*, volume 45, pages 309–328. Springer, 2000. (cit. on pp. 47, 53)

[81] Leticia Decker, Daniel Leite, Fabio Viola, and Daniele Bonacorsi. Comparison of evolving granular classifiers applied to anomaly detection for predictive maintenance in computing centers. In *IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), Bari*, page 8p, 2020. (cit. on pp. 47)

[82] Leticia Decker, Daniel Leite, Luca Giommi, and Daniele Bonacorsi. Real-time anomaly detection in data centers for log-based predictive maintenance using an evolving fuzzy-rule-based approach. In *IEEE World Congress on Comput. Intell. (WCCI, FUZZ-IEEE), Glasgow*, page 8p, 2020. (cit. on pp. 47)

[83] Daniel Leite, Rosangela Ballini, Pyramo Costa Jr, and Fernando Gomide. Evolving fuzzy granular modeling from nonstationary fuzzy data streams. *Evolving Systems*, 3:65–79, 2012. (cit. on pp. 47)

[84] Daniel Leite, Pyramo Costa Jr, and Fernando Gomide. Evolving granular neural networks from fuzzy data streams. *Neural Networks*, 38:1–16, 2013. (cit. on pp. 47)

[85] Daniel Leite, Leticia Decker, Marcio Santana, and Paulo Souza. EGFC: Evolving Gaussian fuzzy classifier from never-ending semi-supervised data streams - with application to power quality disturbance detection and classification. In *IEEE World Congress on Computational Intelligence (WCCI – FUZZ-IEEE), Glasgow*, page 8p, 2020. (cit. on pp. 47)

[86] E. Soares, P. Costa, B. Costa, and D. Leite. Ensemble of evolving data clouds and fuzzy models for weather time series prediction. *Appl. Soft Comput.*, 64:445–453, 2018. (cit. on pp. 48)

[87] Ammar Shaker and Edwin Lughofer. Self-adaptive and local strategies for a smooth treatment of drifts in data streams. *Evolving Systems*, 5(4):239–257, 2014. (cit. on pp. 48)

[88] Xiaowei Gu and Plamen P. Angelov. Self-organising fuzzy logic classifier. *Inf. Sci.*, 447:36 – 51, 2018. (cit. on pp. 48)

[89] Bilal Mirza, Zhiping Lin, and Nan Liu. Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. *Neurocomputing*, 149:316 – 329, 2015. Advances in neural networks Advances in Extreme Learning Machines. (cit. on pp. 48)

[90] M. Pratama, Jie Lu, Edwin Lughofer, Guangquan Zhang, and Sreenatha Anavatti. Scaffolding type-2 classifier for incremental learning under concept drifts. *Neurocomputing*, 191:304 – 329, 2016. (cit. on pp. 48)

[91] Charles Aguiar and Daniel Leite. Unsupervised Fuzzy eIX: Evolving internal-external fuzzy clustering. In *IEEE Evolving and Adaptive Intelligent Systems (EAIS 2020), Bari - IT*, pages 1–8, 2020. (cit. on pp. 48)

[92] Youngin Kim and Cheong Park. An efficient concept drift detection method for streaming data under limited labeling. *IEICE Transactions on Information and Systems*, E100.D:2537–2546, 2017. (cit. on pp. 48)

[93] D. Leite, G. Andonovski, I. Skrjanc, and F. Gomide. Optimal rule-based granular systems from data streams. *IEEE Transactions on Fuzzy Systems*, 28(3):583–596, 2020. (cit. on pp. 48)

[94] D. Leite, G. Andonovski, I. Skrjanc, and F. Gomide. Optimal rule-based granular systems from data streams. *IEEE T Fuzzy Syst.*, 2019. doi: 10.1109/TFUZZ.2019.2911493. (cit. on pp. 51, 52, 53)

[95] Witold Pedrycz and Fernando Gomide. *An Introduction to Fuzzy Sets: Analysis and Design.* MIT Press, 2000. (cit. on pp. 52, 58)

[96] Daniel Leite and Igor Škrjanc. Ensemble of evolving optimal granular experts, owa aggregation, and time series prediction. *Inf. Sci.*, 504:95–112, 2019. (cit. on pp. 52)

[97] Ronald R. Yager. Measures of specificity. In Okyay Kaynak, Lotfi A. Zadeh, Burhan Türkşen, and Imre J. Rudas, editors, *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications*, pages 94–113, Berlin, Heidelberg, 1998. Springer. (cit. on pp. 53)

[98] Eduardo A. Soares, Heloisa A. Camargo, Suzana J. Camargo, and Daniel F. Leite. Incremental gaussian granular fuzzy modeling applied to hurricane track forecasting. *2018 IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8, 2018. (cit. on pp. 55)

[99] Gleb Beliakov, Ana Pradera, and Tomasa Calvo. Aggregation functions: A guide for practitioners. In *Studies in Fuzziness and Soft Computing*, 2007. (cit. on pp. 58)

[100] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):183–190, 1988. (cit. on pp. 63)

[101] A. Kishor, A. K. Singh, and N. R. Pal. Orness measure of owa operators: A new approach. *IEEE Transactions on Fuzzy Systems*, 22(4):1039–1045, 2014. (cit. on pp. 63)

[102] Weibin Meng, Ying Liu, Federico Zaiter, Shenglin Zhang, Yihao Chen, Yuzhe Zhang, Yichen Zhu, En Wang, Ruizhi Zhang, Shimin Tao, Dian Yang, Rong Zhou, and Dan Pei. LogParse: Making Log Parsing Adaptive through Word Classification. In *2020 29TH INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS AND NETWORKS (ICCCN 2020)*, IEEE International Conference on Computer Communications and Networks. IEEE; IEEE Commun Soc, 2020. 29th International Conference on Computer Communications and Networks (ICCCN), ELECTR NETWORK, AUG 03-06, 2020. (cit. on pp. 67, 137)

[103] C. Bertero, M. Roy, C. Sauvanaud, and G. Tredan. Experience report: Log mining using natural language processing and application to anomaly detection. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pages 351–360, 2017. (cit. on pp. 67)

[104] Tong Xiao, Zhe Quan, Zhi-Jie Wang, Kaiqi Zhao, and Xiangke Liao. LPV: A Log Parser Based on Vectorization for Offline and Online Log Parsing. In Plant, C and Wang, H and Cuzzocrea, A and Zaniolo, C and Wu, X, editor, *20TH IEEE INTERNATIONAL CONFERENCE ON DATA MINING (ICDM 2020)*, IEEE International Conference on Data Mining, pages 1346–1351. IEEE; IEEE Comp Soc; Univ Calabria; Mininglamp Technol, 2020. 20th IEEE International Conference on Data Mining (ICDM), ELECTR NETWORK, NOV 17-20, 2020. (cit. on pp. 67)

[105] Donghwan Shin, Zanis Ali Khan, Domenico Bianculli, and Lionel Briand. A Theoretical Framework for Understanding the Relationship Between Log Parsing and Anomaly Detection. In Feng, L and Fisman, D, editor, *RUNTIME VERIFICATION (RV 2021)*, volume 12974 of *Lecture Notes in Computer Science*, pages 277–287, 2021. 21st International Conference on Runtime Verification (RV), ELECTR NETWORK, OCT 11-14, 2021. (cit. on pp. 67)

[106] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R. Lyu. Tools and Benchmarks for Automated Log Parsing. In *2019 IEEE/ACM 41ST INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: SOFTWARE ENGINEERING IN PRACTICE (ICSE-SEIP 2019)*, pages 121–130. IEEE; Assoc Comp Machinery; IEEE Comp Soc; Special Interest Grp Software Engn; Tech Council Software Engn, 2019. IEEE/ACM 41st International Conference on Software Engineering - Software Engineering in Practice (ICSE-SEIP), Montreal, CANADA, MAY 25-31, 2019. (cit. on pp. 67)

[107] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 33–40, 2017. (cit. on pp. 67)

[108] Jin Wang, Yangning Tang, Shiming He, Changqing Zhao, Pradip Kumar Sharma, Osama Alfarraj, and Amr Tolba. Logevent2vec: Logevent-to-vector based anomaly detection for large-scale logs in internet of things. *Sensors*, 20(9), May 2020. (cit. on pp. 67)

[109] Ya Zhong, Yuanbo Guo, and Chunhui Liu. FLP: a feature-based method for log parsing. *ELECTRONICS LETTERS*, 54(23):1334–1335, NOV 15 2018. (cit. on pp. 67)

[110] Shaohan Huang, Yi Liu, Carol Fung, Rong He, Yining Zhao, Hailong Yang, and Zhongzhi Luan. Paddy: An Event Log Parsing Approach using Dynamic Dictionary. In *NOMS 2020 - PROCEEDINGS OF THE 2020 IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM 2020: MANAGEMENT IN THE AGE OF SOFTWARIZATION AND ARTIFICIAL INTELLIGENCE*, IEEE IFIP Network Operations and Management Symposium. IEEE; IFIP; IEEE Commun Soc, 2020. IEEE/IFIP Network Operations and Management Symposium (NOMS), ELECTR NETWORK, APR 20-24, 2020. (cit. on pp. 67)

[111] Guojun Chu, Jingyu Wang, Qi Qi, Haifeng Sun, Shimin Tao, and Jianxin Liao. Prefix-Graph: A Versatile Log Parsing Approach Merging Prefix Tree with Probabilistic Graph. In *2021 IEEE 37TH INTERNATIONAL CONFERENCE ON DATA ENGINEERING (ICDE 2021)*, IEEE International Conference on Data Engineering, pages 2411–2422. IEEE, 2021. 37th IEEE International Conference on Data Engineering (IEEE ICDE), ELECTR NETWORK, APR 19-22, 2021. (cit. on pp. 68)

[112] Daniel F. Leite, Pyramo Costa, and Fernando Gomide. Interval-based evolving modeling. In *2009 IEEE Workshop on Evolving and Self-Developing Intelligent Systems*, pages 1–8, 2009. (cit. on pp. 68, 72, 137)

[113] Daniel Leite, Pyramo Costa, and Fernando Gomide. Granular approach for evolving system modeling. In *Proceedings of the Computational Intelligence for Knowledge-Based Systems Design, and 13th International Conference on Information Processing and Management of Uncertainty*, IPMU'10, page 340–349, Berlin, Heidelberg, 2010. Springer-Verlag. (cit. on pp. 68, 137)

[114] Daniel Leite and Fernando Gomide. *Evolving Linguistic Fuzzy Models from Data Streams*, pages 209–223. Springer, Berlin, Heidelberg, 2012. (cit. on pp. 68, 137)

[115] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools (2nd Edition)*. Addison Wesley, August 2006. (cit. on pp. 68, 69, 137)

[116] Gabriella Airenti. Is a naturalistic theory of communication possible? *Cognitive Systems Research*, 11(2):165–180, 2010. (cit. on pp. 68)

[117] Leticia D. de Sousa et al. Event detection framework for wireless sensor networks considering data anomaly. In *2012 IEEE Symposium on Computers and Communications (ISCC)*, pages 000500–000507, July 2012. (cit. on pp. 81)

[118] Peihua Qiu. *Introduction to Statistical Process Control*. Wiley: India, 2014. (cit. on pp. 81, 134)

[119] J. M. Alonso, L. Magdalena, and Gil González-Rodríguez. Looking for a good fuzzy system interpretability index: An experimental approach. *Int. J. Approx. Reason.*, 51:115–134, 2009. (cit. on pp. 83)

[120] D.D. Nauck. Measuring interpretability in rule-based classification systems. In *The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ '03.*, volume 1, pages 196–201 vol.1, 2003. (cit. on pp. 83)

[121] Noureddine Zerhouni Rafael Gouriveau, Kamal Medjaher. *Health Assessment, Prognostics, and Remaining Useful Life – Part B*, chapter 5, pages 109–136. John Wiley & Sons, Ltd, 2016. (cit. on pp. 104, 107)

[122] Noureddine Zerhouni Rafael Gouriveau, Kamal Medjaher. *Health Assessment, Prognostics and Remaining Useful Life – Part A*, chapter 4, pages 67–107. John Wiley & Sons, Ltd, 2016. (cit. on pp. 104, 107)