One-class classification for cyber intrusion detection in industrial systems

Patric Nader

Thèse
de doctorat
de l'UTT

Patric NADER

One-class Classification for Cyber Intrusion Detection in Industrial Systems

Spécialité :
Optimisation et Sûreté des Systèmes

2015TROY0021 **Année 2015**

utt
université de technologie
Troyes

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Patric NADER

le 24 septembre 2015

One-class Classification for Cyber Intrusion Detection in Industrial Systems

JURY

M. B. RIERA	PROFESSEUR DES UNIVERSITES	Président (Rapporteur)
M. P. BEAUSEROY	PROFESSEUR DES UNIVERSITES	Directeur de thèse
Mme S. CHARBONNIER	MAITRE DE CONFERENCES	Examinateur
M. P. HONEINE	PROFESSEUR DES UNIVERSITES	Directeur de thèse
M. T. MORRIS	ASSOCIATE PROFESSOR	Examinateur
M. A. RAKOTOMAMONJY	PROFESSEUR DES UNIVERSITES	Rapporteur

Personnalités invitées

M. F. CAMPAN	CHEF DE PROJET ONDEO SYSTEMS
M. I. NIKIFOROV	PROFESSEUR DES UNIVERSITES

Acknowledgements

First, I would like to thank my supervisors Paul HONEINE and Pierre BEAUSEROY for providing me with the opportunity to complete my PhD thesis at the University of Technology of Troyes. I especially want to thank Paul HONEINE, whose support and guidance made my thesis work possible. He has been actively interested in my work and has always been available to advise me. I am very grateful for his patience, motivation, enthusiasm, and immense knowledge that make him a great mentor. He has always been guiding me in the right direction, and without his help I could not have finished my thesis successfully. I would like also to thank Pierre BEAUSEROY who managed to be always there for me, despite his busy schedules and enormous responsibilities as the director of the Systems Modeling and Dependability Laboratory (LM2S).

I would like to thank the reviewers, Alain RAKOTOMAMONJY and Bernard RIERA, who have accepted to devote their time to reading and reviewing my work. I would like also to thank the committee members, namely Thomas MORRIS and Sylvie CHABONNIER, for accepting to serve on the committee. A special thanks goes also to Igor NIKIFOROV and Francis CAMPAN.

A special thanks to Thomas Morris and the Mississippi State University SCADA Laboratory for providing the real SCADA datasets, and the French "Agence Nationale de la Recherche"(ANR) for supporting this work under the grant SCALA LM1206E.

My gratitude extends to Regis LENGELLE, the director of the UTT doctoral school. I associate my thanks to Pascale DENIS, Isabelle LECLERCQ and Thérèse KAZARIAN for their availability, understanding, and for all the help they have given me. I would like to thank Bernadette ANDRE and Veronique BANSE for their efforts, support and availability. I would like also to thank all the members of the LM2S Laboratory.

I would also like to take this opportunity to thank all my friends and colleagues who supported me in the past three years. A special thanks goes to Elias KHOURY and Van Long Do, with whom I shared special moments that made this experience unforgettable.

I am very grateful for my parents. Their understanding and their love encouraged me to work hard and to continue pursuing this PhD.

Finally, I am greatly indebted to my devoted wife Sandy MAHFOUZ, the most important person in my life. She has been a constant source of strength and inspiration. Her unconditional love and support without any complaint or regret has enabled me to complete this PhD. Only her determination and constant encouragement made it possible for me to see it through to the end. I owe my every achievement to her.

To my wife and my parents...

Contents

Acknowledgements i

Contents v

List of Figures ix

List of Tables xi

General Introduction 1

1 Industrial Infrastructures and SCADA Systems 7
- 1.1 Supervisory Control and Data Acquisition Systems . . . 8
  - 1.1.1 Background on SCADA systems . . . 8
  - 1.1.2 SCADA architecture . . . 10
  - 1.1.3 SCADA applications . . . 13
- 1.2 Risk Assessment of SCADA Systems . . . 15
  - 1.2.1 Vulnerabilities of SCADA systems . . . 15
  - 1.2.2 Recent attacks against SCADA systems . . . 17
- 1.3 Existing Intrusion Detection Approaches . . . 20
- 1.4 Real SCADA Datasets . . . 22
  - 1.4.1 Gas pipeline testbed . . . 22
  - 1.4.2 Storage tank testbed . . . 25
  - 1.4.3 Water treatment dataset . . . 28
- 1.5 Conclusion . . . 28

2 Kernel Methods in Machine Learning 31
- 2.1 Machine Learning . . . 32
- 2.2 Kernel Methods . . . 34
  - 2.2.1 Kernel functions . . . 34
  - 2.2.2 Reproducing kernel Hilbert space . . . 35
  - 2.2.3 Representer theorem . . . 37
  - 2.2.4 Example of kernel methods: Binary Support Vector Machines . . . 39
- 2.3 RBF Kernels . . . 42
  - 2.3.1 The ℓ_p-norms in kernels . . . 42
  - 2.3.2 Relation between norms . . . 44
  - 2.3.3 Bandwidth parameter . . . 45

2.4 Conclusion 47

3 One-class Classification **49**
3.1 Motivations 50
3.2 Common Approaches 51
3.2.1 Support Vector Data Description 53
3.2.2 One-class Support Vector Machines 55
3.2.3 Slab Support Vector Machines 57
3.2.4 Robust Support Vector Machines 59
3.2.5 Kernel Principal Component Analysis 61
3.2.6 Simple One-class 63
3.3 Truncated Mahalanobis-based One-class 63
3.3.1 Full model approach 64
3.3.2 Advantages of KPCA and kernel whitening 66
3.4 Experimental Results 67
3.4.1 Norm variation 67
3.4.2 Bandwidth parameter 74
3.4.3 One-class approaches 76
3.5 Conclusion 81

4 Sparse One-class Classification **83**
4.1 Introduction 84
4.2 Sparse Truncated Mahalanobis-based One-class 86
4.2.1 The choice of support vectors 86
4.2.2 Sparse formulation 87
4.3 Sparse One-class Framework by Shrinkage 89
4.3.1 Least Angle Regression 90
4.3.2 Least Absolute Shrinkage and Selection Operator 92
4.3.3 Elastic Net 94
4.3.4 Extension to the truncated Mahalanobis distance 95
4.4 Theoretical Results 95
4.4.1 Projection error 95
4.4.2 Error of the first kind 97
4.5 Online One-class Classification 100
4.5.1 The proposed online truncated Mahalanobis-based one-class 101
4.5.1.1 Offline training phase 103
4.5.1.2 Online update phase 104
4.6 Experimental Results 105
4.6.1 Simulated datasets 105
4.6.2 Real datasets 106
4.6.3 Online results 111
4.7 Conclusion 114

General Conclusion and Future Works **117**

A Résumé de la thèse **121**

A.1 Systèmes SCADA . . . 122
A.1.1 Architecture des systèmes SCADA . . . 122
A.1.2 Vulnérabilités des systèmes SCADA . . . 123
A.1.3 Attaques contre SCADA . . . 124
A.1.4 Méthodes de détection existantes . . . 124
A.2 Methodes à noyaux . . . 125
A.2.1 Noyau défini positif et espace de Hilbert associé . . . 126
A.2.2 Variation de la norme des noyaux . . . 128
A.2.3 Le paramètre de largeur de bande . . . 129
A.3 Classification mono-classe . . . 130
A.3.1 Méthodes de classification mono-classe existantes . . . 131
A.3.2 Méthode proposée . . . 131
A.3.3 Résultats expérimentaux . . . 134
A.3.3.1 Variation de la métrique des noyaux . . . 134
A.3.3.2 Heuristique proposée pour choisir σ . . . 135
A.3.3.3 Les approches de classification mono-classe . . . 136
A.4 Representation parcimonieuse pour classification mono-classe . . . 138
A.4.1 Modèle parcimonieux . . . 139
A.4.2 Approche basée sur la distance de Mahalanobis tronquée . . . 139
A.4.2.1 Choix des vecteurs de support . . . 140
A.4.2.2 Formulation parcimonieuse . . . 141
A.4.3 Approche basée sur des méthodes de sélection de variables . . . 142
A.4.3.1 LARS . . . 143
A.4.3.2 LASSO . . . 144
A.4.3.3 Elastic Net . . . 144
A.4.3.4 Extension à la norme de Mahalanobis . . . 145
A.4.4 Classification mono-classe en ligne . . . 146
A.4.4.1 Phase d'apprentissage hors ligne . . . 146
A.4.4.2 Phase de détection/mise à jour en ligne . . . 147
A.4.5 Résultats expérimentaux . . . 148
A.4.5.1 Données simulées . . . 148
A.4.5.2 Données réelles . . . 149
A.4.5.3 Classification mono-classe en ligne . . . 149
A.5 Conclusion et perspectives . . . 150
A.5.1 Travaux futurs . . . 151

Bibliography **153**

List of Figures

1.1 SCADA example 1 . . . 11
1.2 SCADA example 2 . . . 13
1.3 Gas pipeline . . . 23
1.4 Storage tank testbed . . . 27

2.1 The mapping into the RKHS . . . 35
2.2 An illustration of a binary SVM classification . . . 41
2.3 The variation of different norms . . . 44

3.1 An illustration of the hyperplane in SVM . . . 55
3.2 An example of the slab SVM . . . 57
3.3 An example of the Robust SVM . . . 60
3.4 The results on the gas pipeline with the exponential kernel . . . 70
3.5 The results on the gas pipeline with the Gaussian kernel . . . 71
3.6 The behavior of the exponential and the Gaussian kernel with the infinite norm. . . . 72
3.7 Detection of outliers . . . 73
3.8 Error detection and false alarm probabilities as a function of the bandwidth parameter . . . 76
3.9 The decision boundaries on the sinusoidal dataset . . . 78
3.10 The decision boundaries on the square-noise dataset . . . 79

4.1 An illustration of the successive LARS estimates . . . 91
4.2 An example of the hypersphere having one outlier. . . . 98
4.3 A representation of the two concentric hyperspheres. . . . 102
4.4 The results on the sinusoidal dataset . . . 107
4.5 The results on the square-noise dataset . . . 108
4.6 The solution paths of LARS, LASSO and Elastic Net . . . 109

List of Tables

1.1 The attacks against the testbeds, arranged into 7 groups. 26

2.1 The most common positive definite kernel functions. 37
2.2 The expressions of the most common norms. 43

3.1 The confusion matrix of several types of attacks with the SVDD approach using the Gaussian kernel, on the gas pipeline real dataset. 74
3.2 The confusion matrix of several types of attacks with the KPCA approach using the Gaussian kernel, on the gas pipeline real dataset. 74
3.3 Detection rates on the water treatment dataset using the Gaussian kernel with several norms. 74
3.4 Time computational cost of several approaches for computing the bandwidth parameter. 75
3.5 Detection rates for the gas pipeline testbed. 80
3.6 Detection rates for the storage testbed. 80
3.7 Detection rates for the UCI water treatment testbed. 80
3.8 Estimated time (in seconds) for training each approach. 81
3.9 Estimated time (in seconds) to test a new sample for each approach. . . . 81

4.1 Detection rates for the gas pipeline testbed. 111
4.2 Detection rates for the storage testbed. 111
4.3 Detection rates for the UCI water treatment testbed. 111
4.4 Estimated time (in seconds) of each approach. 112
4.5 Detection rates for the gas pipeline testbed. 113
4.6 Detection rates for the storage tank testbed. 113
4.7 Detection rates for the water treatment testbed. 113
4.8 False alarm rates of the online approaches. 113
4.9 Estimated training time (in seconds) of each approach. 114
4.10 Estimated time (in seconds) to test a new sample. 114

General Introduction

The majority of critical infrastructures and industrial systems, such as electrical power grids, oil and natural gas pipelines, chemical processing plants, water distribution systems, wastewater collection systems and nuclear power plants, are controlled nowadays via Supervisory Control And Data Acquisition (SCADA) systems. These systems allow remote monitoring and provide remote access and control to geographically dispersed facilities, which gives the operators working in these facilities the capacity to monitor and control the entire system from a central location in real time.

The principal components of SCADA systems are: a) The *Human Machine Interface (HMI)* allows operators to monitor the state of the process under control and modify its control settings, b) the *Master Terminal Unit (MTU)* stores and processes the information from field devices and transmits control signals, and c) the *Remote Terminal Units (RTU)* receive commands from the MTU to control the local process, acquire data from field devices and transmit it to the MTU. The common protocols (ModBus, Profibus, DNP3) used in the communication between these components present many vulnerabilities regarding the authentication mechanisms between the MTU and the other components, the integrity of the transmitted packets and the anti-replay mechanisms. In addition to the vulnerabilities in the communication between their components, SCADA systems are facing today significant threats of cyberattacks due to the increasing dependence of their communications to public networks, i.e., the internet.

The past decade has witnessed several intentional cyberattacks against critical infrastructures relying on SCADA networks, such as the Maroochy Water Services attack in Australia, disabling the safety monitoring system at Ohio's Davis-Besse nuclear power plant, and the penetration of the U.S. electrical grids. The most complex malware Stuxnet was meant to sabotage the centrifuges used for enriching Uranium in Iran. The diversity of cyberattacks and the complexity of the studied systems make the role of traditional Intrusion Detection Systems (IDS) more difficult. Traditional IDS, such as firewalls, try to match signatures of known cyberattacks with the network traffic, but

they cannot detect new types of cyberattacks not existing in their databases. The cyberattacks threatening critical infrastructures may cause serious economic losses and may impact the health and safety of employees and citizens. The primary objective of this thesis is to detect malicious intrusions once they have already bypassed traditional IDS, by examining the behavior of the studied process. Therefore, the proposed algorithms do not replace the existing detection systems, but they are complementary to their work.

This thesis is a part of the European project SCALA funded by the French "Agence Nationale de la Recherche"(ANR), including academic and non academic partners, namely Ondeo Systems, University of Technology of Troyes and Diateam. The SCALA project focuses on the thematic of the protection of critical infrastructures and networks relying on SCADA systems, and it has three main goals:

- Monitor the network and the system components.
- Detect and localize as fast as possible malicious actions and intrusions.
- Improve the protection of the system against attacks and intrusions.

Two approaches are investigated in the SCALA project:

- The model-based approach is based on the assumption that the analytical redundancy relations of the state variables of the system are known [Tartakovsky et al., 2014]. The statistical models of these variables are integrated in the model-based detection system, and a combined stochastic-dynamic model is used to design the detection algorithm. For more details, see [Do et al., 2015].
- The learning-based approach does not need an exact physical model of the process in order to provide the decision rule [Shawe-Taylor and Cristianini, 2004]. It consists in determining a detection-decision rule by using some previously recorded SCADA observations and commands over a long period of time, and by adapting this decision rule to the evolution of the system.

We investigate in this thesis the second approach with kernel methods in machine learning. In fact, these methods have been widely used in the past few years in the machine learning and data mining fields to discover hidden relations in data, without the need of an exact physical model of the process. To this end, they study the relations within the samples of some training dataset, and elaborate decision rules for detection and classification. In particular, we investigate the use of one-class classification techniques, where the available data refer to a unique class only. This is the case of industrial applications, where the majority of the available data designates the normal functioning modes of the

studied systems, and it is very difficult to acquire data related to malfunctioning modes or critical states. One-class classifiers learn the normal behavior modes of the studied system, and determine decision rules that accept as many normal samples as possible, and detect most of the outliers.

We propose in this thesis several one-class classification approaches for intrusion detection in industrial systems. These approaches aim at helping the traditional IDS in detecting malicious activities and cyberattacks threatening the critical infrastructures. The main objective is to reduce the computational complexity of the existing methods, while maintaining high detection rates. We also propose an online one-class classification approach suitable for real-time and sequential detection applications, where the classifier is updated at each instant. In the following, we give an overview of the organization of this manuscript.

Organization of the Manuscript

This manuscript is composed of four main chapters and a general conclusion. We begin with an introduction on SCADA systems and the security of critical infrastructures in Chapter 1. We give an overview of statistical machine learning and kernel methods in Chapter 2. Chapter 3 focuses on one-class classification for intrusion detection. In Chapter 4, we investigate sparse formulations for one-class classification problems, and we propose an online one-class formulation for sequential detection applications. Next, we outline each chapter of this manuscript.

Chapter 1: Industrial Infrastructures and SCADA Systems

Chapter 1 outlines the increasingly important role of SCADA systems in industrial processes and critical infrastructures. We give an overview of the background of SCADA systems with its different components and applications, and we study the various vulnerabilities that led to several intentional cyberattacks against many critical infrastructures relying on SCADA systems. A review of the common existing techniques for intrusion detection is given, and a description of the real datasets used in this thesis is presented afterwards.

Chapter 2: Kernel Methods in Machine Learning

This Chapter introduces the statistical learning theory based on kernel methods. Machine learning techniques with kernel methods map the training samples into a reproducing kernel Hilbert space in order to detect the hidden relations and patterns within the samples of this training dataset. We outline some of the properties of kernel functions and the main theorems related to kernel methods, namely the Representer theorem. We also investigate the influence of the metric in Radial Basis Function kernels on the decision rule of a one-class classifier, and we propose a simple heuristic for choosing the bandwidth parameter in these kernels.

Chapter 3: One-class Classification

Chapter 3 describes the one-class classification problem, that is used when the available data designate a single class only. In this Chapter, we provide a survey of the most well-known one-class classification methods in the literature. We propose a simple and fast one-class approach for the estimation of the hypersphere enclosing most of the data in the feature space, where we explore the truncated Mahalanobis distance that is used as a novelty measure for detecting the outliers. The proposed truncated Mahalanobis distance allows to mimic the relevant subspace projection of the kernel principal component analysis (KPCA), thus taking advantage of the well-known performance of the KPCA. We apply the corresponding one-class algorithm on simulated datasets as well as on real datasets, and we compare it to well-known one-class classification methods.

Chapter 4: Sparse One-class Classification

Chapter 4 focuses on sparse formulations for one-class classification methods, where this formulation aims at reducing the computational complexity of the corresponding algorithms. We outline the existing sparse methods, and we propose two sparsity-promoting frameworks for one-class classification methods. The first framework is a sparse version of the Mahalanobis-based one-class approach detailed in Chapter 3, while the second one is based on well-known shrinkage methods, namely Least Angle Regression, Least Absolute Shrinkage and Selection Operator, and Elastic Net. The proposed frameworks are backed-up with theoretical results related to the projection error and the error of the first kind. We also propose an online one-class classification approach for real-time and sequential detection applications, as in industrial applications. The relevance of the proposed frameworks is illustrated on simulated and real datasets.

Publications

The research work of this thesis resulted in several international publications. The following outlines these publications:

Journal articles:

- [Nader et al., submitted 2015]: P. Nader, P. Honeine, and P. Beauseroy. One-class classification framework based on shrinkage methods. *Machine Learning*, submitted 2015.
- [Nader et al., 2014c]: P. Nader, P. Honeine, and P. Beauseroy. ℓ_p-norms in one-class classification for intrusion detection in SCADA systems. *Industrial Informatics, IEEE Transactions on*, 10(4):2308-2317, Nov 2014.

Conference articles:

- [Nader et al., 2015b]: P. Nader, P. Honeine, and P. Beauseroy. Shrinkage methods for one-class classification. In *Proc. 23th European Conference on Signal Processing (EUSIPCO)*, Nice, France, 31 August-4 September 2015.
- [Nader et al., 2015a]: P. Nader, P. Honeine, and P. Beauseroy. Online one-class classification for intrusion detection based on the Mahalanobis distance. In *Proc. 23th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, Bruges, Belgium, 22-24 April 2015.
- [Nader et al., 2014a]: P. Nader, P. Honeine, and P. Beauseroy. The role of one-class classification in detecting cyberattacks in critical infrastructures. In *Proc. 9th International Conference on Critical Information Infrastructures Security (CRITIS)*, Limassol, Cyprus, 13-15 October 2014.
- [Nader et al., 2014b]: P. Nader, P. Honeine, and P. Beauseroy. Mahalanobis-based one-class classification. In *Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Reims, France, 21-24 September 2014.
- [Nader et al., 2013]: P. Nader, P. Honeine, and P. Beauseroy. Intrusion detection in SCADA systems using one-class classification. In *Proc. 21th European Conference on Signal Processing (EUSIPCO)*, Marrakech, Morocco, 9-13 September 2013.

Chapter 1

Industrial Infrastructures and Supervisory Control And Data Acquisition Systems

Contents

1.1 Supervisory Control and Data Acquisition Systems 8

1.1.1 Background on SCADA systems 8

1.1.2 SCADA architecture 10

1.1.3 SCADA applications 13

1.2 Risk Assessment of SCADA Systems 15

1.2.1 Vulnerabilities of SCADA systems 15

1.2.2 Recent attacks against SCADA systems 17

1.3 Existing Intrusion Detection Approaches 20

1.4 Real SCADA Datasets 22

1.4.1 Gas pipeline testbed 22

1.4.2 Storage tank testbed 25

1.4.3 Water treatment dataset 28

1.5 Conclusion 28

The security of industrial processes and critical infrastructures has gained a lot of attention in the past few years with the growth of cyberthreats against these systems that are controlled via Supervisory Control and Data Acquisition (SCADA) systems [Boyer, 2009; Kang et al., 2009]. The massive use of Information and Communication Technologies (ICT) in SCADA systems has opened new ways for carrying out cyberattacks against critical infrastructures relying on SCADA networks [Stouffer et al., 2011]. Unauthorized

access to a SCADA system for malicious actions or terrorism purposes could have severe potential consequences on the corresponding physical process. This chapter outlines the important role of SCADA systems in controlling and monitoring critical infrastructures, and the various security vulnerabilities of these system which have increased the risk of cyberthreats against these infrastructures and have led to multiple cyberattacks in the past decades.

The remainder of this chapter is organized as follows. Section 1.1 gives a brief review on SCADA systems and on their architecture. Section 1.2 presents the various vulnerabilities of SCADA networks, and details the recent cyberattacks against these critical infrastructures. Section 1.3 outlines the existing intrusion detection approaches. Section 1.4 details the various real datasets used in the experimental results of this thesis, and Section 1.5 gives a conclusion on the ability of the traditional Intrusion Detection Systems (IDS) to protect the industrial and critical infrastructures.

1.1 Supervisory Control and Data Acquisition Systems

1.1.1 Background on SCADA systems

The role of Supervisory Control and Data Acquisition (SCADA) systems has increased in the past decades in many sectors especially in industrial processes and in critical infrastructure sectors. SCADA systems provide remote access and control to critical infrastructures such as electrical power grids, oil and natural gas pipelines, chemical processing plants, water distribution systems, wastewater collection systems, nuclear power plants, railway transportation systems, pharmaceutical industries, traffic lights, etc. [Stouffer et al., 2006; Bailey and Wright, 2003]. SCADA systems are highly distributed systems used to control geographically dispersed facilities, often scattered over thousands of square kilometers, where centralized data acquisition and control are critical to system operation. They enable the operators to perform centralized monitoring and control for field sites over long-distance communication networks including monitoring alarms and processing status data. Moreover, SCADA systems collect data from one or more distant facilities, display it to the operators graphically or textually, and transfer it to a central computer facility in order to record it and log it in the system databases. This allows the operators to monitor and control the entire system from a central location in real time, and to send control instructions to those facilities. Therefore, SCADA makes it unnecessary for an operator to be assigned to stay at or to frequently visit remote locations when those remote facilities are operating normally. The common types of signals that SCADA systems have to monitor and to control are temperature,

pressure, flow rate, motor speed, level switch, pressure switch, generator status, relays, etc. A real world SCADA system can include hundreds to hundreds of thousands of Input/Output (I/O) devices. As an example, a very simplified water SCADA application would be to monitor water levels at various water sources like reservoirs and tanks; when the water level exceeds some predefined threshold, SCADA activates the pumping system in order to transfer water to secondary lower-level reservoirs.

First generation of SCADA networks operated in isolated environments, with no connectivity to any system outside the network. Nowadays, the extensive use of Information and Communication Technologies (Internet, wireless networks, cell phones) in critical infrastructures has made SCADA networks more and more interconnected with the outside world. Moreover, the majority of the communication protocols have become open solutions and easy to access. Therefore, the vulnerability of these infrastructures to cyberattacks has been increasing excessively. The malicious and terrorist cyberattacks against the critical infrastructures relying on SCADA networks are capable of damaging the entire physical process, cause important economic losses and impact the health and safety of employees and citizens. Moreover, detecting an intrusion and/or a failure in SCADA systems is a difficult task for many reasons, namely the critical infrastructures monitored and controlled via SCADA systems are widely-distributed geographically, the need of a regular feedback from the physical processes, and the real-time performance requirements. These issues are:

- The highly distributed SCADA systems control facilities widely-distributed geographically, often scattered over thousands of square kilometers, with components in locations that often lack physical security. Such geographical dispersion also makes it difficult to physically reset or reload the software on a compromised device. Security solutions in such environments are related to reinforcing the resilience of the application towards such compromises.

- SCADA systems need a regular feedback from the monitored physical processes, and this requires the existence of communication channels that need to be "secured". An attacker does not need to break into the computer to affect such a system, but could penetrate the SCADA secured network in order to inject malicious commands, causing a coordinated streak of physical actions which lead the system to some critical states and respond in an unexpected manner. To protect these infrastructures from this kind of attacks, an understanding of the system and its responses is required.

- Critical infrastructures have real-time requirements. This means that actions must be taken very quickly in the parts of the system that present the anomalies, in order

to compensate for the generation or the transmission of the anomalies elsewhere. Any failure to react in a short time could result in cascading anomalies and possible permanent damage to equipment of the physical process. These interactions have become more and more complex, and require reactions on smaller time-scales. A very quick detection of an intrusion or an anomaly is crucial in order to maintain a high performance of the system.

1.1.2 SCADA architecture

As we showed in the previous section, SCADA systems are used in many business sectors such as transportation, energy, telecommunication and water distribution. Most of these sectors are defined in Europe as critical sectors and, as a consequence, critical installations have to be protected. SCADA systems are cyberphysical systems, consisting of hardware and software components. The hardware allows the transfer of the data and information back and forth between the different components of SCADA systems, and it includes the control center of the studied industrial process, the communication equipments such as radio, telephone line, cable or satellite, and one or more geographically distributed sites which control and monitor the actuators and the sensors. The software is programmed to tell the system what variables to monitor and when to do that, what parameter ranges are acceptable in the normal functioning modes of the studied process, and what response to initiate when the parameters go outside the acceptable values. An example of a SCADA system controlling a nuclear power plant is given in Figure 1.1.

The principal components of SCADA systems are the Human Machine Interface, the Master Terminal Unit, the Remote Terminal Units and the Programmable Logic Controllers [Stouffer et al., 2011]:

- The *Human Machine Interface (HMI)* is both a software and a hardware that allows the human operators and engineers of the industrial system to monitor the state of a process under investigation, modify the control settings to change the objective, and manually override automatic control operations in the event of an emergency. The HMI displays the process status information, historical information, reports, diagnostic data, management information such as scheduled maintenance procedures, logistic information, detailed schematics for particular sensors or machines, expert-system troubleshooting guides and other information to operators, administrators, managers, business partners, and other authorized users. The operator can see a schematic representation of the plant being controlled. For example, a picture of a pump connected to a pipe can show the operator that the pump is running, and how much fluid it is pumping through the pipe at the

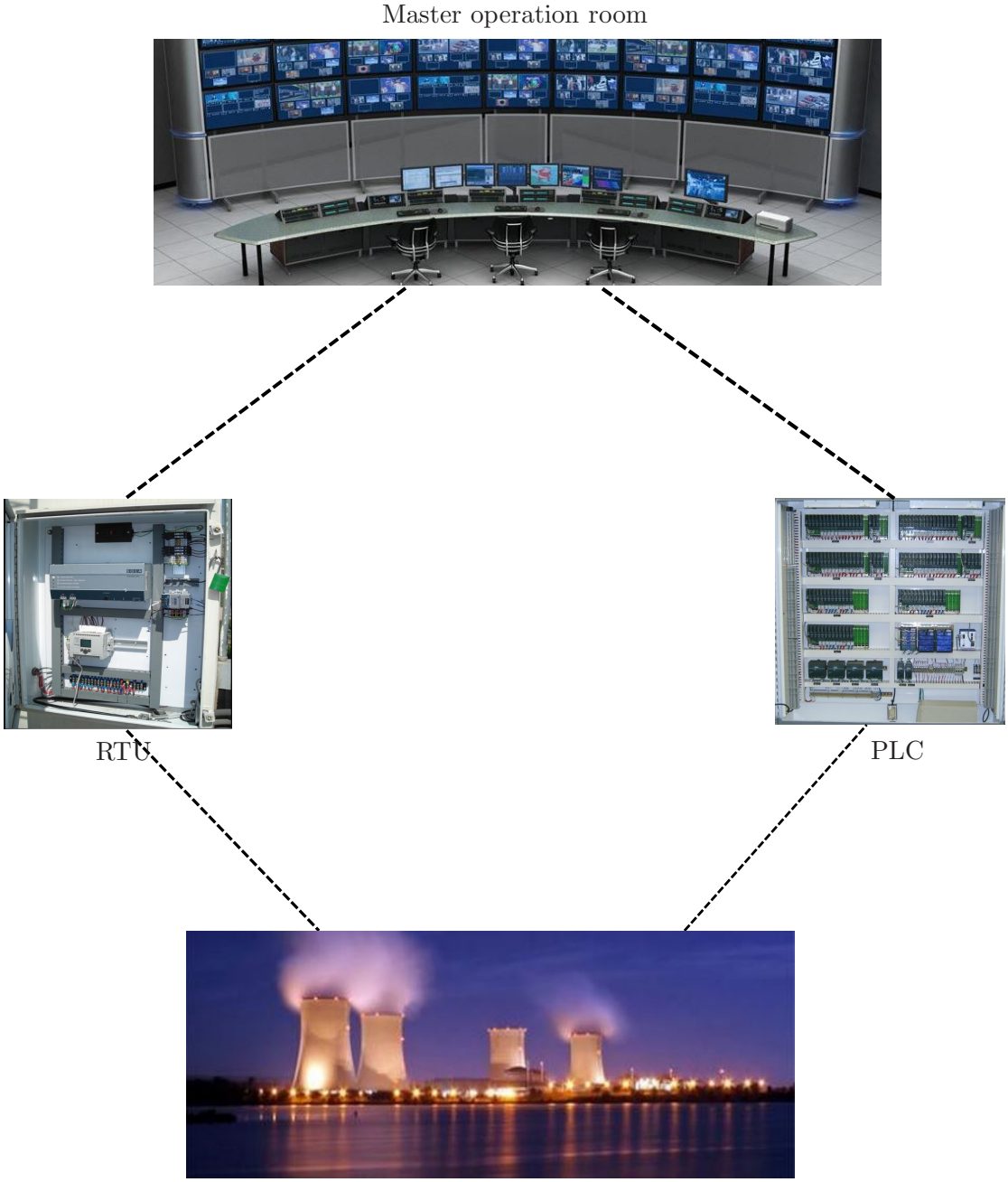

Figure 1.1: An example of a nuclear power plant controlled via a SCADA system with its major components, namely the Human Machine Interface, the Master Terminal Unit [Company, 2015], the Remote Terminal Units and the Programmable Logic Controllers. The PLCs and RTUs, which are connected to the HMI inside the Master operation room, transmit the measurements from the nuclear power plant to the MTU and receive the corresponding control commands.

moment. The HMI also allows an engineer or an operator to configure set points, control algorithms and parameters in the controller. The location, the platform, and the interface may vary a great deal. For example, a HMI could be a dedicated platform in the control center, a laptop on a wireless Local Area Networks (LANs), or a browser on any system connected to the Internet.

- The *Master Terminal Unit (MTU)* or SCADA Server is at the center of each SCADA system, and it is the device that acts as its master component. The MTU issues all the commands, gathers all the data from the different components of SCADA systems, stores some information in its databases, passes other information to associated systems, interfaces with the people who operate the studied process, and transmits control signals to some components based on the processing of the received information. In other words, the MTU is actually in charge of the physical process. To accomplish all these tasks, a very detailed description of all of the sensors and actuators connected to the physical process must be available to the MTU's processor. This description must be arranged in a hierarchical form in order to be handled in the most time-effective manner. In many cases, the MTU is also required to send the data to corporate business computers or computer networks, and all this communication is handled by LANs.

- The *Remote Terminal Units (RTUs)*, also called remote telemetry units, are located at remote sites, and they usually act as Slaves. The RTU is a data acquisition and control unit designed to support SCADA remote stations, it converts sensor signals to digital data, and it is often equipped with wireless radio interfaces to support remote situations where wire-based communications are unavailable. The RTUs first gather information from field devices such as analog values, metered amounts, alarm and status indications, then keep this information available in the memory until the MTU asks for it, and finally code and transmit the information back to the MTU. The RTUs also receive commands from the MTU in order to control the local process, such as to open or close valves, turn switches on and off, turn motors on and off, start and stop pumps, output analog signals that may represent the acceptable pressure range or water levels, etc.

- The *Programmable Logic Controllers (PLCs)* are small industrial computers originally designed to perform the logic functions executed by electrical hardware (relays, drum switches, and mechanical timer/counters). They are connected to the sensors of the physical process, and they convert sensor signals to digital data. The PLCs have more sophisticated embedded control capabilities than RTUs, with a Central Processing Unit (CPU) that allows them to communicate with other devices and to execute the control instructions. PLCs are sometimes implemented as

field devices to serve as RTUs since they are more economical, versatile, flexible, and configurable than special-purpose RTUs; in these cases, the PLCs are often referred to as RTUs.

An illustration of a SCADA system with its major components is given in Figure 1.2 [Blog, 2014].

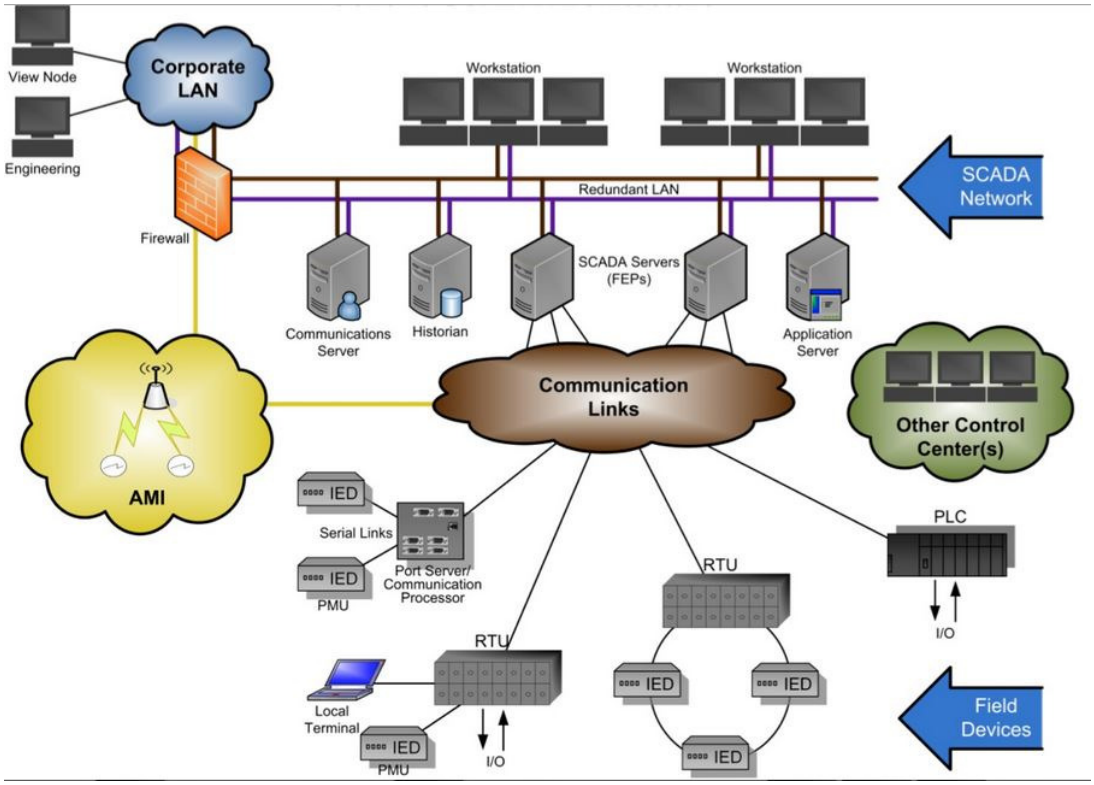

Figure 1.2: An illustration of communications between the different components of a SCADA system, available from [Blog, 2014].

1.1.3 SCADA applications

The SCADA technology has been applied to control and monitor industrial processes and critical infrastructures spread over large areas. The studied systems require frequent, regular, or immediate intervention in some critical cases. It is important to note that over 90% of the critical infrastructures of the United States of America are controlled via SCADA and equivalent industrial systems. Similarly, SCADA systems are used to monitor the majority of the critical installations in Europe. The signals gathered from remote locations include alarms, status indications, measurements related to the studied process, etc. Signals sent from the SCADA's central location to remotes site are usually limited to discrete binary bit changes or to analog values addressed to some devices at

the process. The common analog signals that SCADA systems have to monitor and to control are levels, temperature, pressure, flow rate and motor speed, and an example of an analog signal would be an instruction to change a valve controller set point to 70 percent. On the other hand, the typical digital signals to monitor and to control are level switches, pressure switches, generator status, relays and motors, and an example of a binary bit change would be an instruction ordering a motor to stop. The following examples of such processes illustrate the range of application types that SCADA has been suitable for [UK, 2013]:

- A group of hydroelectric generating stations that include pumps, turbines and motors. These stations are usually located in remote locations, and they are tuned on and off in response to customers demand. These systems can be controlled by opening and closing valves of the turbine. The monitoring of these systems must be done continuously in order to respond relatively quickly to the demands on the electric power grids.

- The oil and gas production facilities that are also spread over large areas. The physical process of these facilities include wells, gathering systems, fluid measurement equipments, and pumps. These systems require relatively simple control such as turning motors on and off, measuring the production quantities regularly, checking the quality of the products, and measuring the temperature in the facilities. It is imperative that similar systems respond quickly to any abnormal condition in the facilities.

- The transportation sectors that include monitoring and control of carriers, traffic lights, transport infrastructure facilities, traffic management operators, port/airport facilities, etc. The transport is one of the most important economic sectors with its infrastructure being essential for the functioning of the entire society. The transport infrastructures are facing multiple challenges to ensure the smooth operation of traffic within their responsibility. These challenges can range from normal traffic to accidents as well as major disruptions due to natural disasters or acts of intentional man-made attacks, such as criminal acts on the premises, robberies of cargo, acts of piracy, hijacking and terrorist attacks.

- The pipelines used to move gas, oil, water or chemical products to the market. The various components of these systems are spread over large areas distant from the central control points. The control system of the pipelines require simple actions such as measuring the pressure of the pipelines regularly, opening and closing valves, and starting or stopping pumps. The pipeline systems must be capable of responding quickly to the market conditions as well as to the leaks of any dangerous or environmentally sensitive materials.

- The electric transmission systems that may cover thousands of square kilometers. The monitoring of these systems can be achieved via measuring the voltage transmitted in the lines, checking the status of the switches and the substations, and monitoring the flow of power into and out of the protected transmission lines. The electric transmission systems can be controlled by opening and closing switches, and they must respond almost immediately to load changes on the lines.

- The water treatment and distribution systems that also cover large territories. These systems require to monitor some parameters such as the tank levels, the system pressure, the chemical treatment substances, and the status of the filters. They can be controlled by opening and closing valves, and by increasing/descreasing the system pressure by turning on or off the pumps. The water treatment and distribution systems must also be capable to respond immediately to the leaks of water or any chemical materials.

Therefore, the field of application of SCADA systems is very wide, and it includes the majority of industrial processes and critical infrastructures. In the past few years, several countries have demonstrated their willingness to develop efficient methods applicable against the increasing cyberthreats that these infrastructures are facing nowadays. Russia appears to want to join the United States of America to create a common front against the cyber-crime and cyber terrorism, and China improves its national capacity in the area of cyber-criminality [Krekel et al., 2009]. Many countries in the European Union have adopted their legislation and organizational response to this real danger, and the European institutions created The National Security Agency Information Systems in 2009 following the reports on cyber terrorism. France has identified the protection of SCADA as a priority for national research and development. Next, we present the risk assessment of SCADA networks, and we outline the recent cyberattacks on critical infrastructures relying on SCADA.

1.2 Risk Assessment of SCADA Systems

1.2.1 Vulnerabilities of SCADA systems

First generations of SCADA systems were independent networks that operated in isolated environments where they rarely shared information with the systems outside their environment, thus with no connectivity to the world outside the network. The SCADA information and command processing were distributed across multiple stations, where each station was responsible for a particular task. The protocols used in the communication between the components of a SCADA system were proprietary, thus very few

people beyond the developers knew enough to determine how secure a SCADA installation was. Nowadays, the industrial systems and the critical infrastructures are more and more interconnected with the outside world via public networks, i.e., the Internet, which has significantly reduced the maintenance costs and increased the capacity of real-time reporting. In addition, the communication protocols have become more standardized and open solutions. However, the interconnection of SCADA systems with the outside world using Internet-based standards, and the integration of the control networks into larger corporate networks in order to share valuable data, have introduced numerous vulnerabilities to these systems and have exposed these critical infrastructures to new sources of potential threats [Ten et al., 2008, 2011].

In addition to the vulnerabilities due to the increasing dependence of their communications to the Internet, SCADA systems are facing today significant threats of cyberattacks due to the vulnerabilities of the communications protocols implemented in these networks [Fovino et al., 2009; Morris and Pavurapu, 2010]. In fact, the common protocols, namely ModBus, Profibus and DNP3, used in the communication between the different components of SCADA systems present many vulnerabilities, regarding the integrity and the authentication mechanisms [Fovino et al., 2010b]. They have been conceived originally for serial communication when security was not crucial for control systems [Fovino et al., 2012]. In particular, these protocols:

- Do not apply any mechanism for checking the integrity of the command packets between Master and Slaves. The received packets could have been intercepted and manipulated by a third party, and their content modified.
- Do not perform any authentication mechanism between Master and Slaves, thus anyone could claim to be the Master, request sensitive information about the physical process, and send malicious control commands to the Slaves.
- Do not apply antirepudiation or antireplay mechanisms.
- Do not provide any mechanism to preserve the network availability of the field devices and SCADA servers.

The threats against critical infrastructures can be originated internally or externally, and can be accidental, natural or intentional ones [Kennedy et al., 2014]. A threat against a SCADA system is considered as an accidental one if it is an external or internal agent that causes harm to system components without any malicious intention. Such harm can be due to negligence or human error, e.g., an employee who uses his infected USB memory stick on the workstations of the facility. When it comes to a natural threat against a SCADA system, it might be any force of nature such as an earthquake, a

storm, a hurricane or a flood, that causes harm by affecting the physical environment or the components of the system. An intentional threat occurs when an external or internal agent causes harm to the system components to willingly affect its functionalities. Intentional security threats can be grouped into three categories: Hackers, insiders and malwares [Urias et al., 2012]:

- The *hackers* are individuals or groups with an advanced understanding of computer langagues and computer networks. Hackers seek and exploit weaknesses in a computer system or computer network, gain access to these networks, collect data flows by intercepting the exchanged packets, and inject false commands with the intention to disrupt the physical system under control. Hackers may be motivated by a multitude of reasons, such as profit, protest, challenge or enjoyment.
- The *insiders* are the personnel of the facility having a legitimate access to the network and may cause damages to the industrial information system or to its infrastructure. Insiders that perform attacks have a distinct advantage over external attackers since they have authorized system access and also may be familiar with the network architecture and system policies/procedures. In addition, there may be less security against insider attacks because most organizations focus on protecting their facilities from external threats.
- The *malwares* are viruses, worms, trojans and spywares that can affect the operating systems and the softwares of the facility. Many of these malwares spread over the Internet using email or malicious webpages to infect unprotected computers, while the others spread through removable media, such as USB memory sticks and external hard drives. These malwares can destroy, damage or infect the information in the computers of the facility, including data on external drivers. They can also take control of the computers and use it to attack other computers.

1.2.2 Recent attacks against SCADA systems

The vulnerabilities in the communication protocols between SCADA components and the intensive use of Internet and communication technologies have increased the cyberthreats and opened new ways for carrying out cyberattacks against critical infrastructures relying on SCADA networks. The past decade has witnessed several intentional cyberattacks against these industrial systems. The people responsible for the majority of these cyberattacks took advantage of vulnerabilities of the critical infrastructures, gained unauthorized access into the SCADA networks monitoring the physical processes, collected data information exchanged between the facilities and the operators, planted malicious malwares disrupting the normal functioning behavior of the system, etc. These

cyberattacks have caused serious damage to the physical process of several critical infrastructures with important economic losses. Next, we detail the most well-known intentional cyberattacks against industrial systems relying on SCADA networks:

- In 2000, an ex-employee of Maroochy Water Services in Australia stole radio equipment attached to a possibly stolen computer, took control of 150 sewage pumping stations, issued radio commands to the sewage equipment he helped install, and released one million liters of untreated sewage into local parks and rivers [Slay and Miller, 2007]. Marine life died, the creek water turned black and the stench was unbearable for the residents of this area.

- In 2003, the Slammer worm penetrated a private computer network at Ohio's Davis-Besse nuclear power plant (USA) and disabled a safety monitoring system for nearly five hours [Christiansson and Luiijf, 2007], despite a belief by the plant personnel that the network was heavily protected by a firewall.

- In 2006, a hacker penetrated a water filtering plant in Pennsylvania (USA) and installed malicious software capable of affecting the normal functioning operations of the plant water treatment [Cárdenas et al., 2011]. The hacker operating on the Internet tapped into an employee's laptop, used his remote access as the point of entry and installed a virus and a spyware in the water plant computer system.

- In 2009, cyberspies penetrated the U.S. electrical grids and left behind software programs that could be used to disrupt the system [Gorman, 2008]. The spies came from China, Russia and other countries, and were believed to be on a mission to navigate and to control the U.S. electrical grids system. The intruders have not sought to damage the power grid or other key infrastructure, but officials warned they could try during a crisis or war.

- In 2010, the complex malware Stuxnet was discovered in Iran. It was targeting the PLCs connected to a nuclear centrifuge used for enriching Uranium. Stuxnet installs a malicious program replacing the PLCs original file in a manner undetectable by the PLC operator [Chen and Abu-Nimeh, 2011]. The ultimate goal of Stuxnet was to sabotage the nuclear plants where the speed fluctuations could have caused the centrifuges to fly apart and to be destroyed [Langner, 2011].

- In 2011, the malware Duqu was discovered with striking similarities to Stuxnet, but apparently with a different objective. Duqu was targeting Microsoft Windows based PCs, and it can be reconfigured remotely in a way to include any kind of malicious functionalities. Duqu generated anomalies in the infected systems that were not easy to spot by any anti-virus product at the time. Indeed, Duqu does

not aim at causing physical damage, but it is an information collecting malware used for cyber espionage, thus cyber/physical attacks based on Duqu might be possible [Bencsáth et al., 2012a].

- In 2012, the malware-based attack Shamoon was discovered. Shamoon has been used for cyber espionage in the energy sector, and it was targeting a petroleum and chemical enterprise in the Middle East [Zhioua, 2013]. The main objective of Shamoon was to wipe data from Microsoft Windows based computers and then to tamper with the Master Boot Record (MBR) of the storage media, making the computer inaccessible. Shamoon resulted in the complete destruction of the content of around 30 000 workstations in this facility.

- In 2012, the most sophisticated malware, namely Flame, was discovered in Hungary. Flame is another information-stealer malware with a modular structure, targeting Microsoft Windows based PCs, and incorporating multiple propagation and attack techniques, as well as special code injection methods. It gathers intelligence in multiple ways, including logging key strokes, saving screenshots, switching on the microphone and the web camera to record audio and video, and browsing through the storage devices connected to the infected computer. It also switches on the Bluetooth radio if available on the infected computer, and saves information about neighboring Bluetooth enabled devices. In addition, it can also use the Bluetooth radio to send information about the victim system to a nearby device possibly controlled by the attackers [Bencsáth et al., 2012b].

- Also in 2012, the malware Gauss that uses a modular structure resembling that of Flame was discovered. This malware has been actively distributed in the Middle East, with the largest number of Gauss infections in Lebanon, in contrast to Flame, which spread primarily in Iran. Similar to Flame and Duqu, Gauss is designed to collect as much information about infected systems as possible. A distinguishing feature of Gauss, however, is that it also steals credentials for various banking systems and social networks, as well as for email and instant messaging accounts, by injecting its own modules into different browsers and intercepting session data, cookies, passwords, and browser history. In particular, the Gauss code includes commands to intercept data required to work with several Lebanese banks, e.g., Bank of Beirut, Byblos Bank, and Fransabank [Expert, 2012].

The cyberattacks threatening the industrial processes and the critical infrastructures relying on SCADA systems have become more and more complex, sophisticated, and hard to detect. These cyberattacks may cause serious economic losses and may impact the health and safety of employees and citizens. In the following, we outline the most known intrusion detection techniques existing in the literature.

1.3 Existing Intrusion Detection Approaches

The security of critical infrastructures has become the ultimate priority of researchers in the past few years with the growth of cyberthreats and the diversity of cyberattacks. Although traditional Intrusion Detection Systems (IDS) update frequently their databases of known attacks, new complex cyberattacks are generated everyday to circumvent security systems and to make their detection nearly impossible [Oman and Phillips, 2007]. For these reasons, researchers have been developing and deploying various IDS to reveal cyberattacks, restrict their impact on the infrastructures, provide more security to the employees and citizens, and limit the economic and human life losses.

A mechanism for collaborative intrusion detection using a centralized server to dispatch activities coming from suspicious IP addresses was proposed in [Gross et al., 2004]. Unfortunately, this approach does not provide any kind of specific technique for identifying high level and complex cyberattacks. Another approach was presented in [Fovino et al., 2010a] that detects the intrusions by monitoring the state evolution of the studied system. The authors analyse the set of commands that are licit when considered in isolation on a single-packet basis, and can disrupt the correct behavior of the system when executed in particular operating states. A prior knowledge on the physical process and on its different critical states is mandatory to build the detection rules. An approach based on the concept of critical state analysis for the detection of a particular type of cyberattacks against a given industrial installation was presented in [Carcano et al., 2011]. The authors used the concept of "critical state proximity" based on the notion of *distance* from critical states to predict whether the system is heading to a dangerous state. This approach focuses on the restrictive assumption that the attacker interferes with the state of the installation forcing a transition from a safe state to a critical one. The cyber security for a wind farm SCADA system, its the vulnerabilities and the impact of cyberattacks on the power system dynamics were investigated in [Yan et al., 2011]. Multiple attack scenarios were developed consequently, and the simulation results show that cyberattacks can cause major problems for a power system, including economy loss, overspeed of wind turbines, and equipment damage. A signature-based approach was proposed in [Yang et al., 2013] that matches signatures of known attacks with the network traffic, but it cannot detect new attacks not existing in their databases. A model-based approach for detecting intrusions in SCADA systems was proposed in [Yang et al., 2014]. This approach needs an exact system model which is not the case for the majority of the critical infrastructures. In [Morris et al., 2011b,a], the authors described a SCADA testbed elaborated in the Mississippi State University Laboratory to investigate cybersecurity vulnerabilities on functional control systems. This testbed includes commercial hardware and software that control physical processes such as gas

pipelines, industrial blowers, smart grid transmission control systems, raised water towers and factory conveyor belts. In order to study cybersecurity vulnerabilities in SCADA systems and to understand their implications and criticality on controlled physical processes, several types of attacks were injected into the network traffic of the system to hide its real functioning state and to disrupt the communication [Gao et al., 2010].

Statistical methods have also been investigated for cyber intrusion detection on SCADA systems. A Bayesian network, implemented in [Bigham et al., 2003], correlates the outputs from several anomaly detectors with other information to reduce the false positive rate. This statistical model relies on the probabilistic relationships and the conditional dependencies between the system's variables. The moving average and the Kalman filter were used for intrusion detection in [Ye et al., 2004; Knorn and Leith, 2008]. These methods operate only on a predefined model-based system having linear relations between its variables. In [Veracini et al., 2011], the probability density estimation was adopted for anomaly detection, which requires prior statistical knowledge on the probabilities of the realizations of each sample in order to estimate the prediction for new samples. Multilayer perceptron and RBF neural networks were also investigated for intrusion detection in [Yu et al., 2006; Golovko et al., 2007; Ghadiri and Ghadiri, 2011]. However, the major drawback of these supervised algorithms is that prior knowledge on different kinds of attacks is required to ensure an accurate detection, while acquiring this prior knowledge on all the existing types of attacks is nearly impossible with the generation of more complex and sophisticated attacks everyday.

The aforementioned intrusion detection techniques have many vulnerabilities towards detecting new and complex cyberattacks. Traditional IDS try to match signatures of known cyberattacks with the network traffic, but they cannot detect new types of cyberattacks not existing in their databases, and the model-based approaches require a prior knowledge on the existing attacks. Therefore, the traditional IDS need help in order to detect new and complicated attacks generated on daily basis, and to provide better protection for industrial systems and critical infrastructures. The diversity of cyberattacks and the complexity of the studied systems make modeling cyberattacks very difficult or even impossible, thus restricting the use of parametric model-based approaches, such as moving average and Kalman filter. This difficulty highlights the potential role of non-parametric methods in detecting intrusions, which could bring to traditional IDS the needed complementary help in order to provide an ultimate protection to the critical infrastructures against cyberattacks. Consequently, the main objective of the methods proposed in this thesis is to provide this important and complementary help to traditional IDS.

1.4 Real SCADA Datasets

As we have seen so far in this chapter, the field of application of SCADA systems covers the majority of industrial processes and critical infrastructures, and the vulnerabilities of these systems have increased the cyberthreats and the potential risks of cyberattacks against the facilities relying on SCADA networks. In order to study the efficiency of the proposed intrusion detection algorithms that are developed throughout this thesis, we apply these algorithms on real SCADA datasets from testbeds that emulate real existing physical processes. The SCADA Laboratory of the Mississippi State University elaborated testbeds to investigate the vulnerabilities of SCADA systems, in which they used commercial hardware and software to control physical processes such as a gas pipeline and a storage tank [Morris et al., 2011b,a]. Different types of attacks were injected in the normal network traffic in order to analyze the behavior of the system in response of these false commands. This section provides in the first place a description on these testbeds, namely the gas pipeline and the storage tank testbeds, and details the attacks against these testbeds. Then, we detail another real dataset used in the experimentation, namely the water treatment plant dataset from the University of California Irvine (UCI) Machine Learning Repository [Bache and Lichman, 2013].

1.4.1 Gas pipeline testbed

The gas pipeline is used to move natural gas or any other petroleum products to the market. The testbed represents a typical SCADA system embracing a MTU gathering data on the studied process and sending control commands to the other components of the SCADA system, RTU and PLCs acquiring data from field devices and sending it to the MTU, and a HMI presenting the different variables of the process to the operators, which allows to monitor and control the real functioning mode of the physical system. The gas pipeline control system contains an air pump that pumps air into the pipeline, a pressure sensor which allows pressure visibility at the pipeline and remotely on the HMI, a release valve and a solenoid release valve to loose air pressure from the pipeline. The control scheme includes an automatic and a manual mode. In the automatic mode, a Proportional Integral Derivative (PID) controller is used to control the pressure in the pipeline, while in the manual mode the operator can supervise the system and take charge over the pump state and the two release valves. Any cyberattack or cyber penetration on the control system of the gas pipeline can cause the loss of visibility and control of the physical process, which leads to financial losses and physical harm to the pipeline and to employees and citizens, as happened in Bellingham-Washington [Abrams and Weiss, 2008]; The HMI became unresponsive and the pipeline leaked 250 000 gallons

of gasoline in nearby streams, creating an explosion that killed three persons and injured eight others. An illustration of the gas pipeline testbed is given in Figure 1.3 [Morris et al., 2011a]. Two datasets were acquired from the gas pipeline testbed.

Figure 1.3: The gas pipeline testbed from the Mississippi State University SCADA Laboratory [Morris et al., 2011a].

Gas pipeline real dataset 1

The first real dataset from the gas pipeline testbed includes samples that measure the pressure at the pipeline in pounds per square inch (PSI). The allowed pressure range in the pipeline is from 0 to 20 PSI, and the current setpoint of the pipeline is fixed at 20 PSI with a margin of 10% which fixes the maximum accepted pressure to 22 PSI. The pipeline operates under three principal modes:

- The first mode is characterized by a very low pressure maintained around 0.1 PSI.
- The second mode keeps the pressure around 10 PSI (the accepted range lays between 9 and 11 PSI).
- The third mode maintains the pressure around 20 PSI (the accepted range is between 18 to 22 PSI).

Several types of false commands and responses were injected into the normal network traffic of the system to make its behavior looks like abnormal and hide its real functioning states, in order to study the vulnerabilities of the system and their implications on the controlled process. These attacks are as follows:

- *The negative response injection attack*: although the pressure cannot be a negative number and it seems trivial to be identified with IDS, the negative pressure is hard to be detected in our case since it has already bypassed the security systems and the firewalls.
- *The single response injection attack*: it injects pressure responses equal to 10.43 PSI, which is the second normal operational mode of the system, and 0.01 PSI which is the first normal operational mode, while the system is running high pressures in the third operational mode.
- *The burst response injection attack*: it injects at a high frequency a single value equals to 20 PSI, while the system is running in several modes.
- *The fast change response injection attack*: it returns measurements that change very fast, opposed to the case of a normal behavior of the pipeline, where the returned values of the pressure are equal to 9.56 PSI and 20 PSI respectively.
- *The slow change response injection attack*: it returns measurements that change in a very slow manner, while the system is running in several modes. The returned values of the pressure remain around 5 PSI and 20 PSI respectively.
- *The wave response injection attack*: it injects pressure responses that vary in a wave form around 9 PSI, and it imitates exactly the second normal functioning mode, while the real system is dealing with high pressures in the third mode.

Therefore, the main difficulty lies in detecting these common and dangerous attacks that imitate the normal behavior of the system, while hiding the real functioning states.

Gas pipeline real dataset 2

The second real dataset from the gas pipeline testbed is more complex and complicated than the first one. In this case, each input sample has 27 attributes representing heterogenous variables which are the command device address, the response device address, the command memory, the response memory, the command packet length, the response packet length, the time interval between packets, the command read/write function codes, the response read/write function codes, the system control mode, the pump state, the manual pump setting, the gas pressure of the pipeline, the set point or target gas pressure, the solenoide state, and the parameters of the PID controller. For this real dataset, 28 types of attacks are injected into the network traffic of the system in order to hide its real functioning state and to disrupt the communication. These attacks are arranged into 7 groups described below:

- *The Naive Malicious Response Injection (NMRI) attacks* inject response packets into the network traffic but lack information about the process being controlled and monitored, thus some of the injected payloads may be invalid.
- *The Complex Malicious Response Injection (CMRI) attacks* are more sophisticated than NMRI, as they require an understanding of the physical process being controlled, try to mask the real state of the studied process and affect the feedback control loop of the system.
- *The Malicious State Command Injection (MSCI) attacks* send malicious commands to remote field devices to change the state of the physical process in such a way to drive the system from a safe state to a critical one.
- *The Malicious Parameter Command Injection (MPCI) attacks* change the controller parameter set points of sensors, actuators, and programmable logic controllers spread over the facility.
- *The Malicious Function Command Injection (MFCI) attacks* are used to disrupt the client server communication link by modifying the command functions of the transmitted packets.
- *The Denial of Service (DOS) attacks* try to stop the proper functioning of the physical system and aim to disable and overwhelm the entire system, either by sending transmissions faster than they can be processed or by sending packets with incorrect payloads. This generates exceptions which crash the running programs or the operating system of the targeted device.
- *The Reconnaissance Attacks (RA)* gather information about the control system network, map the network architecture, in order to identify the devices characteristics such as manufacturer, model number, supported network protocols and system address/memory map. The gathered information may be used for other attacks.

The list of the 28 types of attacks, arranged into 7 groups, is given in Table 1.1.

1.4.2 Storage tank testbed

The storage tank testbed is similar to the oil storage tanks found in the petrochemical industry, which use oil and other products to produce gasoline, kerosene, diesel and other types of plastics. In real configurations, the oil arriving from sea is pumped into the storage tanks to provide a consistent supply of oil to the refinery operation. In this testbed, water was substitued for oil for safety reasons. The control system of

Table 1.1: The attacks against the testbeds, arranged into 7 groups.

Attack index	Attack name	Attack group
1	Address Scan	Reconnaissance
2	Function Code Scan	Reconnaissance
3	Device Identification	Reconnaissance
4	Points Scan	Reconnaissance
5	Memory Dump	Reconnaissance
6	Naive Read Payload Injection	NMRI
7	Invalid Read Payload Injection	NMRI
8	Naive False Error Response	NMRI
9	Negative Sensor Measurement(s)	NMRI
10	Sensor Measurement Grossly Out of Bounds	NMRI
11	Sporadic Sensor Measurement Injection	NMRI
12	Random Sensor Measurement Injection	NMRI
13	Constant Sensor Measurement Injection	CMRI
14	Slope Sensor Measurement Injection	CMRI
15	High Slope Measurement Injection	CMRI
16	Low Slope Measurement Injection	CMRI
17	Replayed Measurement Injection	CMRI
18	Altered System Control Scheme	MSCI
19	Altered Actuator State	MSCI
20	Continually Altered Actuator State	MSCI
21	Altered Proportional Integral Derivative Parameter(s)	MPCI
22	Altered Control Set Point	MPCI
23	Force Listen Mode Only	MFCI
24	Restart Communication	MFCI
25	Clear Data Log	MFCI
26	Change ASCII Input Delimiter	MFCI
27	Invalid Cyclic Redundancy Code (CRC)	DOS
28	Modbus Slave Traffic Jamming	DOS

the storage tank contains primary and secondary storage tanks, a pump to move water from the secondary to the primary tank, a relieve valve that allows water to flow from the primary to the secondary tank, and a sensor that provides the water level in the primary tank as a percentage of its total capacity. Like the gas pipeline testbed, the storage tank testbed also represents a typical SCADA system including a MTU, a HMI, RTUs and PLCs connected and communicating with each other. The operators can place the system in automatic or manual modes, and the pump is turned on and off in a way to keep the water level between the high level and the low level. An alarm is raised if the water level rises beyond the high level or falls within the low level. Cyberattacks on the control systems of a storage tank can cause important financial and human life losses. An illustration of the control system schematic of the storage tank is given in Figure 1.4 [Gao et al., 2010].

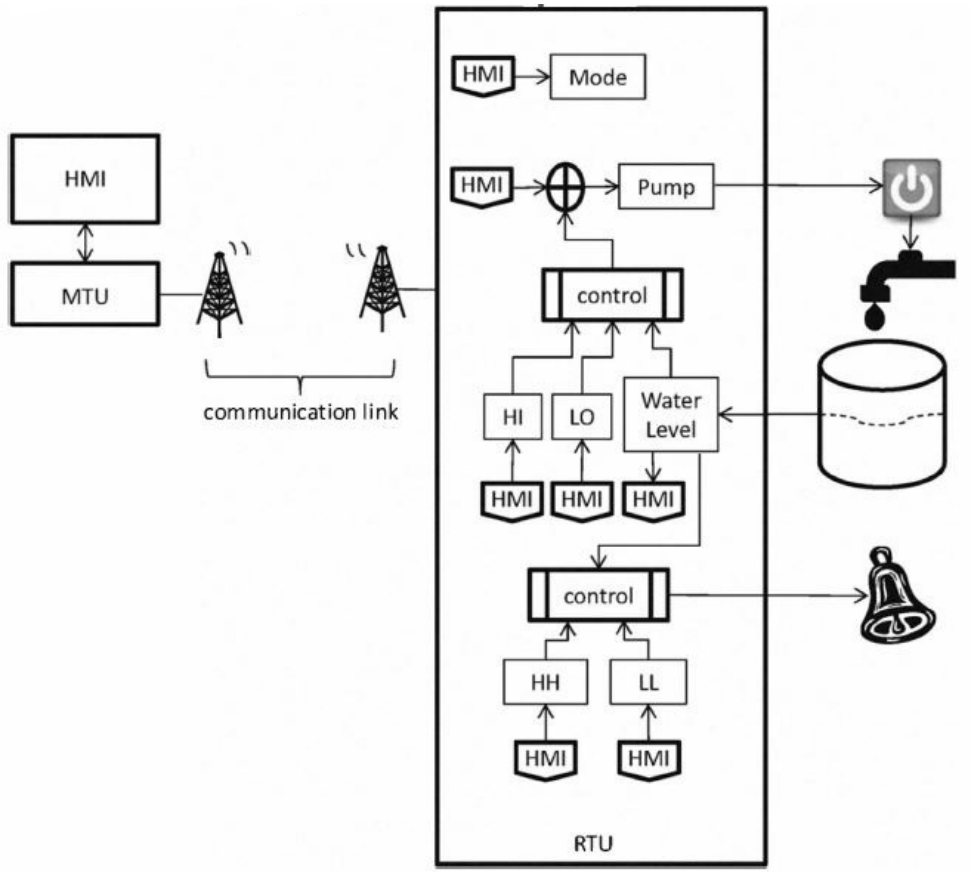

Figure 1.4: The storage tank testbed from the Mississippi State University SCADA Laboratory [Gao et al., 2010].

Each input sample of the storage tank testbed has 24 heterogenous attributes. Some of these variables are common with the gas pipeline testbed, such as the command device address, the response device address, the command memory, the response memory, the command packet length, the response packet length, the time interval between packets, the value of the command read/write function codes, the value of the response read/write function codes, the system control mode, the pump state, the manual pump setting, while other attributes are specific to the storage testbed such as the water level in the tank, the set points related to turning on and off the water pump, and the set points of the high and low water levels related to triggering an alarm due to a system fault. Similar to the second real dataset of the gas pipeline testbed, the same types of attacks were injected into the normal network traffic of the storage tank testbed to hide its real functioning state and to disrupt the communication. See Table 1.1 for more details on the attacks used in this real testbed.

1.4.3 Water treatment dataset

The proposed methods in this thesis will be also applied on another complex dataset, namely the *water treatment plant dataset* from the University of California, Irvine (USA), available from the UCI Machine Learning Repository [Bache and Lichman, 2013]. This dataset comes from the daily measures of sensors in an urban waste water treatment plant. Each sample contains 38 attributes related to the measurements of several important components in the water like input zinc, input PH, input biological demand of oxygen, input suspended solids, input conductivity, input volatile suspended solids, input sediments to secondary settler, output chemical demand of oxygen, output volatile suspended solids, and other attributes. The values of each attribute vary in a different manner, e.g., the range of input PH is between 6.9 and 8.7, input zinc remains between 0.1 and 33.5, input conductivity between 651 and 3230, input suspended solids between 98 and 2008, and input sediments to secondary settler between 0 and 3.5. The available samples represent the normal functioning modes of the water treatment plant, as well as some samples enclosing measurements of abnormal situations like after storms or when solids overload.

The water treatment dataset contains 2.95% of missing attributes. Several approaches existing in the literature solve the missing values problems like event covering [Wong and Chiu, 1987], regularized expectation-maximization [Schneider, 2001], singular value decomposition imputation [Troyanskaya et al., 2001], imputation with K-nearest neighbor [Batista and Monard, 2003], K-means clustering imputation [Li et al., 2004], and the mean imputation [Grzymala-Busse et al., 2005]. We apply in this thesis the mean imputation on the water treatment dataset, for it is a widely used method for dealing with missing data with low computaional complexity [Brown and Kros, 2003]. It turns out that the missing values of this dataset do not correspond to the most relevant attributes, thus replacing them with the mean has negligible impact on the interpretation of the corresponding measurement.

1.5 Conclusion

In this chapter, we reviewed the important role of SCACA systems for remote monitoring and controlling industrial processes and critical infrastructures. We detailed the various applications of SCADA systems, and we showed that the communication protocols used in SCADA networks have many vulnerabilities when it comes to verifying the authentication and the integrity of the transmitted packets. We also showed that the massive use of Information and Communication Technologies has opened new ways for carrying

out cyberattacks against these infrastructures, and the complexity of the attacks has made the task extremely difficult for traditional intrusion detection systems. These systems need help in order to provide an ultimate protection for the critical infrastructures, and this is where comes the role of machine learning with non-parametric approaches in detecting malicious intrusions in industrial processes. These techniques have been very useful for detecting hidden patterns in data, by learning nonlinear systems without the need of an exact physical model.

Chapter 2

Kernel Methods in Machine Learning

Contents

2.1 Machine Learning 32
2.2 Kernel Methods 34
2.2.1 Kernel functions 34
2.2.2 Reproducing kernel Hilbert space 35
2.2.3 Representer theorem 37
2.2.4 Example of kernel methods: Binary Support Vector Machines . 39
2.3 RBF Kernels 42
2.3.1 The ℓ_p-norms in kernels 42
2.3.2 Relation between norms 44
2.3.3 Bandwidth parameter 45
2.4 Conclusion 47

The statistical learning theory provides a framework for studying existing relations within the samples of some training dataset. It also allows to infer decision functions from a given training dataset for classification and regression [Vapnik, 1995; Bousquet et al., 2004]. The resulting optimization problem is an ill-posed one since there is an infinite number of solutions satisfying the learning conditions. Regularization allows to overcome this ill-posedness [Tikhonov and Arsenin, 1977], such as by restricting the space of solutions to regular functions, e.g., with the reproducing kernel Hilbert space (RKHS) [Aronszajn, 1950]. Such space is constructed using a (positive definite) kernel function, which turns out to be a generalization of the inner product, namely the inner

product in a RKHS. Indeed, one can transform conventional linear algorithms into non-linear ones, by expressing these algorithms in terms of inner products and then with a kernel function. This is the well-known "kernel trick" [Aizerman et al., 1964].

In this chapter, we review the kernel functions in machine learning. We give a primer on machine learning in Section 2.1. In Section 2.2, we delineate the kernel methods, the main properties and the advantages of using these methods. We detail in Section 2.3 the most commonly used kernel functions, namely the radial basis functions, we study the influence of the metric used in these kernels, and we propose a simple heuristic for the estimation of the bandwidth parameter. A conclusion is given in Section 2.4.

2.1 Machine Learning

Over the last decades, machine learning techniques have become very popular in the pattern recognition and data mining fields for discovering linear and nonlinear relations with hidden patterns in data [Hofmann et al., 2008; Shawe-Taylor and Cristianini, 2004; Herbrich, 2001]. They provide an elegant way to learn a system without the need of an exact physical model [Greiner et al., 1988; Vert et al., 2004]. They have been applied to solve classification and regression problems in many fields, such as perceptual learning in autonomous robotics applications [Bredeche et al., 2006], credit cards fraud detection [Brause et al., 1999], biomedical signal processing [Strauss et al., 2003], anomaly detection and fault diagnosis in automobile applications [Fujimaki, 2008], and wireless sensor networks [Mahfouz et al., 2014]. The main objective of statistical learning theory is to study the relations within the samples of a training dataset or the relations between the inputs and the outputs of the studied system, and to elaborate decision functions that allow to generalize the performance to new "unseen" samples, i.e., samples not existing in the training dataset. Machine learning techniques can be roughly divided into two main categories: supervised and unsupervised approaches.

The supervised learning approaches study the relation between any sample $\boldsymbol{x}$ of the input space $\mathcal{X}$ and the corresponding label y of the output space $\mathcal{Y}$. They consist in finding a decision function φ that is able to predict the labels of new unseen samples. To this end, the risk functional is minimized as follows:

$$\arg\min_{\varphi} \int_{\mathcal{X}\times\mathcal{Y}} L(\varphi(\boldsymbol{x}), y) P(\boldsymbol{x}, y) d\boldsymbol{x} dy,$$

where L is a loss function that measures the error between the desired output y and the estimated value $\varphi(\boldsymbol{x})$ provided by the learning machine, and $P(\boldsymbol{x}, y)$ is the probability distribution of the data pairs $(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$. Since the distribution $P(\boldsymbol{x}, y)$ is usually

unknown, the decision function shall be inferred from a training dataset of n independent and identically distributed (i.i.d.) observations, namely $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n) \in \mathcal{X} \times \mathcal{Y}$, having $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ the input samples and $y_1, y_2, \ldots, y_n$ the corresponding labels. The minimization of the empirical risk functional is achieved as follows:

$$\varphi^* = \arg\min_{\varphi} \frac{1}{n} \sum_{i=1}^{n} L(\varphi(\boldsymbol{x}_i), y_i), \tag{2.1}$$

where φ^* denotes the optimal function that estimates the output $\varphi^*(\boldsymbol{x})$ for the corresponding input sample $\boldsymbol{x}$. For instance, in binary classification problems, only two classes are involved and the responses belong to the set $\mathcal{Y} = \{-1, +1\}$ for all the training samples. Any sample $\boldsymbol{x}$ is associated to either the first class or the second one by examining the sign of $\varphi^*(\boldsymbol{x})$.

When it comes to unsupervised learning, the only available observations are the samples of the input space $\mathcal{X}$, and these samples are related to a unique class. In contrast to supervised learning, the samples in unsupervised learning do not have a label. The unsupervised approaches seek to find the hidden relations between the input samples without the label information as in the supervised learning. In this case, the estimated function is obtained by solving the following optimization problem:

$$\arg\min_{\varphi} \int_{\mathcal{X}} L(\varphi(\boldsymbol{x}))P(\boldsymbol{x})d\boldsymbol{x},$$

having $P(\boldsymbol{x})$ the probability distribution over $\mathcal{X}$, and L a given loss function. Since the distribution $P(\boldsymbol{x})$ is usually unknown, the decision function shall be inferred from a training dataset of n i.i.d. observations, namely $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathcal{X}$, and the minimization of the empirical risk functional is obtained by:

$$\varphi* = \arg\min_{\varphi} \frac{1}{n} \sum_{i=1}^{n} L(\varphi(\boldsymbol{x}_i)). \tag{2.2}$$

Both supervised and unsupervised learning are ill-posed problems since there exist an infinite number of solutions $\varphi*$ that minimize the empirical risk functional by nullifying (2.1) in supervised learning and (2.2) in the unsupervised case. In order to overcome this problem, one takes advantage of the regularization introduced in [Tikhonov and Arsenin, 1977], which restricts the search of the function φ^* to a space of regular functions. In the following, we review a specific space of regular functions, namely the reproducing kernel Hilbert space [Aronszajn, 1950].

2.2 Kernel Methods

Kernel methods rely on mapping the samples from the input space into a reproducing kernel Hilbert space (RKHS). Linear algorithms are applied on the samples in the RKHS in order to estimate the hidden relations existing within the input samples [Vert et al., 2004]. The algorithms applied on the mapped samples can be expressed as a function of the pairwise inner product between the samples. Kernel methods explore this property by evaluating these inner products with kernel functions, without the need to explicit the nonlinear map. In this section, we give an overview of the kernel functions used in the machine learning and data mining fields, some of their properties and the advantages of using them.

2.2.1 Kernel functions

In order to estimate nonlinear relations and patterns existing among the samples in the input space, kernel methods map the data from this space $\mathcal{X}$ into a higher dimensional feature space $\mathcal{H}$. The mapping is given as follows:

$$\begin{aligned} \phi\colon \mathcal{X} &\longrightarrow \mathcal{H} \\ \boldsymbol{x} &\longmapsto \phi(\boldsymbol{x}), \end{aligned}$$

having ϕ the mapping function. The algorithms are expressed in terms of inner products, thus without any explicit knowledge of the mapping function ϕ. An example of the feature mapping is illustrated in Figure 2.1, where the nonlinear relations in the input space are obtained from linear ones in the feature space.

Consider a training dataset $\{\boldsymbol{x}_i\ , i = 1, \ldots, n\}$. The pairwise inner product between two samples in the feature space $\mathcal{H}$ is denoted:

$$\big\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \big\rangle_{\mathcal{H}},$$

and the squared distance in that space is defined in terms of the inner product as follows:

$$\begin{aligned} \big\| \phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j) \big\|_{\mathcal{H}}^2 &= \big\langle \phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j), \phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j) \big\rangle_{\mathcal{H}} \\ &= \big\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_i) \big\rangle_{\mathcal{H}} - 2\big\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \big\rangle_{\mathcal{H}} + \big\langle \phi(\boldsymbol{x}_j), \phi(\boldsymbol{x}_j) \big\rangle_{\mathcal{H}}. \end{aligned}$$

In machine learning with kernel methods, these expressions are evaluated using a kernel function, namely

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \big\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \big\rangle_{\mathcal{H}}, \quad \text{for any} \quad \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}.$$

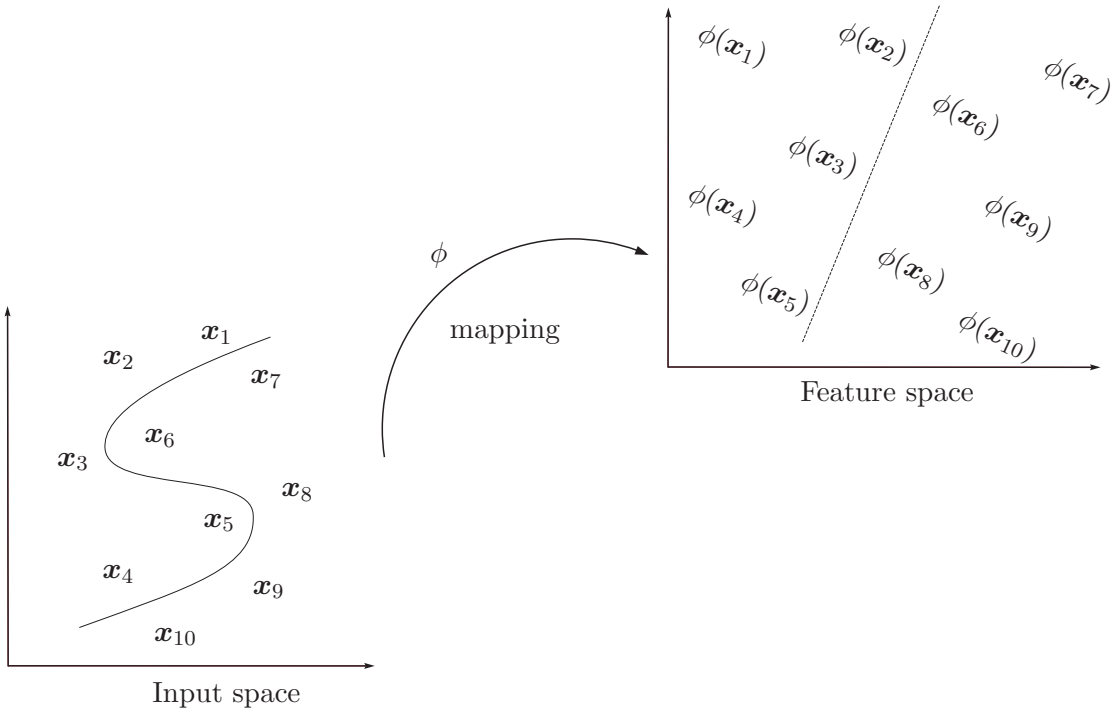

Figure 2.1: An example of the mapping where the existing nonlinear relations in the input space become linear ones in the feature space.

When dealing with a training set $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathcal{X}$, all the corresponding kernel values are gathered in a single kernel matrix $\boldsymbol{K}$ of size $n \times n$, with entries $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The kernel matrix plays an essential role in the learning algorithms. The kernel functions used in kernel methods are positive definite kernel functions. A function k is a positive definite kernel if and only if it is *symmetric*, that is $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_j, \boldsymbol{x}_i)$ for any two samples $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$, and *positive definite*, that is:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0,$$

for any set of samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$, and any set of real numbers $c_1, \ldots, c_n \in \mathcal{R}$.

2.2.2 Reproducing kernel Hilbert space

As we have seen in the previous section, only the inner product between the input samples is needed in kernel methods, without any explicit knowledge of the mapping function ϕ. The input samples are mapped into a higher dimensional feature space, which is an inner product *Hilbert Space* that is *complete* and *separable*. The completeness refers to the property that every Cauchy sequence $\{h_n\}_{n \geq 1}$ of elements of $\mathcal{H}$ converges

to some element $h \in \mathcal{H}$, where a Cauchy sequence satisfies:

$$\sup_{m>n} \|h_n - h_m\| \to 0, \quad \text{as} \quad n \to \infty,$$

and the separability denotes that there exist a countable set of element $h_1, \ldots, h_i$, of $\mathcal{H}$ such that for all $h \in \mathcal{H}$ and $\epsilon > 0$ we have:

$$\|h_i - h\| < \epsilon.$$

The feature space $\mathcal{H}$ is defined by a set of functions, where each one of these functions is a linear combination of the mapped samples. The expression of $\mathcal{H}$ has the following form:

$$\mathcal{H} = \Big\{ \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i), \quad \boldsymbol{x}_i \in \mathcal{X}, \alpha_i \in \mathcal{R}, i = 1, \ldots, n \Big\}. \tag{2.3}$$

Let $f, g \in \mathcal{H}$ be two functions of the feature space given by:

$$f = \sum_{i=1}^{l} \alpha_i \phi(\boldsymbol{x}_i) \tag{2.4}$$

and

$$g = \sum_{j=1}^{n} \beta_j \phi(\boldsymbol{x}_j), \tag{2.5}$$

then the inner product on $\mathcal{H}$ between f and g is constructed as follows:

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{l} \sum_{j=1}^{n} \alpha_i \beta_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i=1}^{l} \alpha_i g(\boldsymbol{x}_i) = \sum_{j=1}^{n} \beta_j f(\boldsymbol{x}_j), \tag{2.6}$$

where $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$, $\alpha_i, \beta_j \in \mathcal{R}$, $l, n \in \mathcal{N}$, and the second and the third equalities come from the definitions of f and g. Taking the special case where $g = \phi(\boldsymbol{x})$ and considering the inner product on $\mathcal{H}$ between f and g in (2.6) gives us the following property:

$$\langle f, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}} = \sum_{i=1}^{l} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) = f(\boldsymbol{x}).$$

This property is known as the *reproducing property* of the kernel. The feature space $\mathcal{H}$ corresponding to the kernel function k satisfying the positive definite property will be referred to as its *reproducing kernel Hilbert space (RKHS)*. Indeed, the kernel k satisfies

the positive definite property since:

$$\begin{aligned}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j) &= \sum_{i,j=1}^{l}\alpha_i\alpha_j\big\langle\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_j)\big\rangle_{\mathcal{H}}\\ &= \Big\langle\sum_{i=1}^{l}\alpha_i\phi(\boldsymbol{x}_i),\sum_{j=1}^{l}\alpha_j\phi(\boldsymbol{x}_j)\Big\rangle_{\mathcal{H}}\\ &= \Big\|\sum_{i=1}^{l}\alpha_i\phi(\boldsymbol{x}_i)\Big\|_{\mathcal{H}}^2\\ &\geq 0.\end{aligned}$$

The advantage of using such a kernel is that it allows to construct learning algorithms in inner product spaces via a kernel function k, without computing the coordinates of the data in that space, and therefore without any explicit knowledge of the mapping function ϕ. This key idea, known as the kernel trick, allows to transform linear algorithms, expressed only in terms of inner products, into nonlinear ones. Table 2.1 shows the most common positive definite kernels in the literature.

Table 2.1: The most common positive definite kernel functions.

Kernel functions	**Expression of $k(\boldsymbol{x}_i,\boldsymbol{x}_j)$**
Linear kernel	$\langle\boldsymbol{x}_i,\boldsymbol{x}_j\rangle$
Polynomial kernel	$\big(\langle\boldsymbol{x}_i,\boldsymbol{x}_j\rangle+d\big)^p$
Gaussian kernel	$\exp\Big(-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2^2}{2\sigma^2}\Big)$
Laplacian kernel	$\exp\Big(-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2}{b}\Big)$
Rational quadratic kernel	$1-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2+d}$
Normalized kernel	$\frac{\langle\boldsymbol{x}_i,\boldsymbol{x}_j\rangle}{\|\boldsymbol{x}_i\|\cdot\|\boldsymbol{x}_j\|}$

Some of the kernels are unit-norm kernels, such as the Gaussian and the Laplacian kernels. A unit-norm kernel satisfies:

$$\|\phi(\boldsymbol{x})\|_{\mathcal{H}}^2 = 1, \quad \text{for all} \quad \boldsymbol{x}\in\mathcal{X}.$$

2.2.3 Representer theorem

The kernel trick plays an important role in transforming linear algorithms into nonlinear ones, as long as these algorithms can be expressed in terms of inner products only. To be operational, the kernel trick is often associated with the representer theorem. The

representer theorem, introduced for spline modeling in [Kimeldorf and Wahba, 1971; Wahba, 1990] and recently generalized to other learning problems in [Schölkopf et al., 2001a; Cucker and Smale, 2002; Kůrková and Sanguineti, 2005], shows that the solutions of a large class of optimization problems can be expressed as a finite linear combination of the training samples in the feature space. Next, we detail this theorem.

Theorem 2.1. *Let $\mathcal{X}$ be an input space, k a kernel function, $\mathcal{H}$ its RKHS and $\phi(\cdot)$ the corresponding mapping function, $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$ a training dataset having $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ the input samples and $y_1, y_2, \ldots, y_n$ the corresponding labels (if available), L a loss function measuring the error between the desired output and the estimated value, and $g(\cdot)$ a strictly increasing function. Then any function $\varphi^* \in \mathcal{H}$ minimizing the following regularized risk functional*

$$\frac{1}{n}\sum_{i=1}^{n} L(\varphi(\boldsymbol{x}_i), y_i) + g(\|\varphi\|_{\mathcal{H}}^2) \tag{2.7}$$

admits a representation of the form

$$\varphi^* = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i). \tag{2.8}$$

Proof. Let $\mathcal{H}_n$ be the subspace of $\mathcal{H}$ spanned by the elements $\{\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \ldots, \phi(\boldsymbol{x}_n)\}$, namely:

$$\mathcal{H}_n = \Big\{\varphi \in \mathcal{H} \colon \varphi = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i), \quad \alpha_i \in \mathcal{R}\Big\}.$$

One can use the orthogonal projection to decompose any element φ of $\mathcal{H}$ into a sum of two functions:

$$\varphi = \varphi^* + \varphi^{\perp},$$

where φ^* lies in the span of $\mathcal{H}_n$, and $\varphi^{\perp}$ lies in the orthogonal complement of $\mathcal{H}_n$. Since each element $\phi(\boldsymbol{x}_i)$ is an element of $\mathcal{H}_n$ and $\varphi^{\perp}$ is orthogonal to each element of this subspace, we have $\langle \varphi^{\perp}, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{H}} = 0$ for $i = 1, 2, \ldots, n$. We make use of the reproducing property to evaluate φ at any $\boldsymbol{x}_i$ as follows:

$$\begin{aligned}
\varphi(\boldsymbol{x}_i) &= \big\langle \varphi, \phi(\boldsymbol{x}_i)\big\rangle_{\mathcal{H}} \\
&= \sum_{j=1}^{n} \alpha_j \big\langle \phi(\boldsymbol{x}_j), \phi(\boldsymbol{x}_i)\big\rangle_{\mathcal{H}} + \big\langle \varphi^{\perp}, \phi(\boldsymbol{x}_i)\big\rangle_{\mathcal{H}} \\
&= \sum_{j=1}^{n} \alpha_j k(\boldsymbol{x}_j, \boldsymbol{x}_i) + \big\langle \varphi^{\perp}, \phi(\boldsymbol{x}_i)\big\rangle_{\mathcal{H}},
\end{aligned}$$

where the right hand side of the last term of this expression is equal to zero by the orthogonality to the elements of $\mathcal{H}_n$. The solution becomes:

$$\varphi(\boldsymbol{x}_i) = \sum_{j=1}^{n} \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

Let $\xi(\varphi)$ denote the risk functional to minimize in (2.7). We apply the Pythagoras theorem in $\mathcal{H}$ on φ:

$$\|\varphi\|_{\mathcal{H}}^2 = \|\varphi^*\|_{\mathcal{H}}^2 + \|\varphi^\perp\|_{\mathcal{H}}^2,$$

which shows that $\xi(\varphi) \geq \xi(\varphi^*)$, with equality if and only if $\|\varphi^\perp\|_{\mathcal{H}} = 0$, since $g(\cdot)$ is a strictly increasing function. As a result, the minimizer of the regularized risk functional satisfies $\|\varphi^\perp\|_{\mathcal{H}} = 0$, thus it belongs to $\mathcal{H}_n$. This concludes the proof. □

2.2.4 Example of kernel methods: Binary Support Vector Machines

In binary classification, the samples are labeled. Each sample belongs to one of the two predefined classes. Consider the training dataset $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$, having the labels belonging to the set $\mathcal{Y} = \{-1, +1\}$ for all the training samples. The binary SVM classifier builds a decision function φ^* in a way to classify new samples not existing in the training dataset into one of the two classes. To this end, the samples are mapped into a high dimensional feature space (RKHS), where the binary SVM classifier finds the optimal hyperplane separating the training samples of these two classes with maximal margin.

Let ρ and $\boldsymbol{w}$ denote the parameters of the hyperplane, defined by any $\phi(\boldsymbol{x})$ verifying the following expression:

$$\big\langle \boldsymbol{w}, \phi(\boldsymbol{x}) \big\rangle_{\mathcal{H}} - \rho = 0.$$

The decision function of the binary SVM classifier is given by:

$$\varphi(\boldsymbol{x}) = \operatorname{sign}\Big(\big\langle \boldsymbol{w}, \phi(\boldsymbol{x}) \big\rangle_{\mathcal{H}} - \rho \Big).$$

To estimate the parameters of the hyperplane, the maximum margin between samples from the two classes yields the following constraints for all $i = 1, 2, \ldots, n$:

$$\begin{cases} \big\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \big\rangle_{\mathcal{H}} - \rho \geq +1 & \text{if} \quad y_i = +1; \\ \big\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \big\rangle_{\mathcal{H}} - \rho \leq -1 & \text{if} \quad y_i = -1. \end{cases}$$

These two cases can be written with a compact notation:

$$y_i \big(\big\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \big\rangle_{\mathcal{H}} - \rho \big) \geq +1, \quad \text{for} \quad i = 1, 2, \ldots, n.$$

In order to separate the samples of the two classes with maximum margin, we solve the following optimisation problem:

$$\min_{\boldsymbol{w},\rho} \frac{1}{2}\|\boldsymbol{w}\|_{\mathcal{H}}^2, \tag{2.9}$$

subject to

$$y_i\big(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{H}} - \rho\big) \geq +1, \quad \text{for} \quad i = 1, 2, \ldots, n.$$

The Lagrangian of this optimization problem is constructed as follows:

$$L(\rho, \boldsymbol{w}, \alpha_i) = \frac{1}{2}\|\boldsymbol{w}\|_{\mathcal{H}}^2 - \sum_{i=1}^{n} \alpha_i \Big(y_i\big(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{H}} - \rho\big) - 1\Big), \tag{2.10}$$

where the α_i's are the Lagrangian multipliers. The Lagrangian has to be minimized with respect to ρ and $\boldsymbol{w}$, and maximized with respect to α_i. The partial derivatives of the Lagrangian with respect to ρ and $\boldsymbol{w}$ give the following relations:

$$\begin{aligned} \frac{\partial L}{\partial \rho} = 0 \qquad &\Longleftrightarrow \qquad \sum_{i=1}^{n} y_i \alpha_i = 0 \\ \frac{\partial L}{\partial \boldsymbol{w}} = 0 \qquad &\Longleftrightarrow \qquad \boldsymbol{w} = \sum_{i=1}^{n} y_i \alpha_i \phi(\boldsymbol{x}_i). \end{aligned}$$

We include these relations in the Lagrangian that becomes:

$$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{2.11}$$

subject to

$$\sum_{i=1}^{n} y_i \alpha_i = 0.$$

This optimization problem is a constrained quadratic programming problem, whose solution is found using off-the-shelf optimization techniques. We note that the optimal hyperplane is given by:

$$\boldsymbol{w} = \sum_{i=1}^{n} y_i \alpha_i \phi(\boldsymbol{x}_i),$$

where we have obtained the expression of the representer theorem of the previous section. The value of ρ is obtained from the training samples lying on the two margins, namely $\phi(\boldsymbol{x}_i) = \pm 1$. These samples are called support vectors since they completely determine the hyperplane. Indeed, it turns out that only this fraction of samples has non-zero α_i's. This property of sparsity made the SVM attractive.

In order to classify any new sample $\boldsymbol{x}$, the decision function of the binary SVM classifier is

$$\varphi(\boldsymbol{x}) = \text{sign}\Big(\sum_{i=1}^{n} y_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) - \rho\Big),$$

where we can encounter one of the following cases:

- $\varphi(\boldsymbol{x}) > +1 \quad \longrightarrow \quad \boldsymbol{x}$ is a sample that belongs to the class $y = +1$.
- $\varphi(\boldsymbol{x}) = +1 \quad \longrightarrow \quad \boldsymbol{x}$ is a sample that belongs to the class $y = +1$ and lies on the margin defined by $\sum_{i=1}^{n} y_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \rho = +1$.
- $\varphi(\boldsymbol{x}) < -1 \quad \longrightarrow \quad \boldsymbol{x}$ is a sample that belongs to the class $y = -1$.
- $\varphi(\boldsymbol{x}) = -1 \quad \longrightarrow \quad \boldsymbol{x}$ is a sample that belongs to the class $y = -1$ and lies on the margin defined by $\sum_{i=1}^{n} y_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \rho = -1$.

As shown in the optimization problem 2.11 and in the above decision function, the mapping does not need to be defined explicitly, since only the kernel function is needed. An illustration in a 2-dimensional space of a binary SVM classification is given in Figure 2.2.

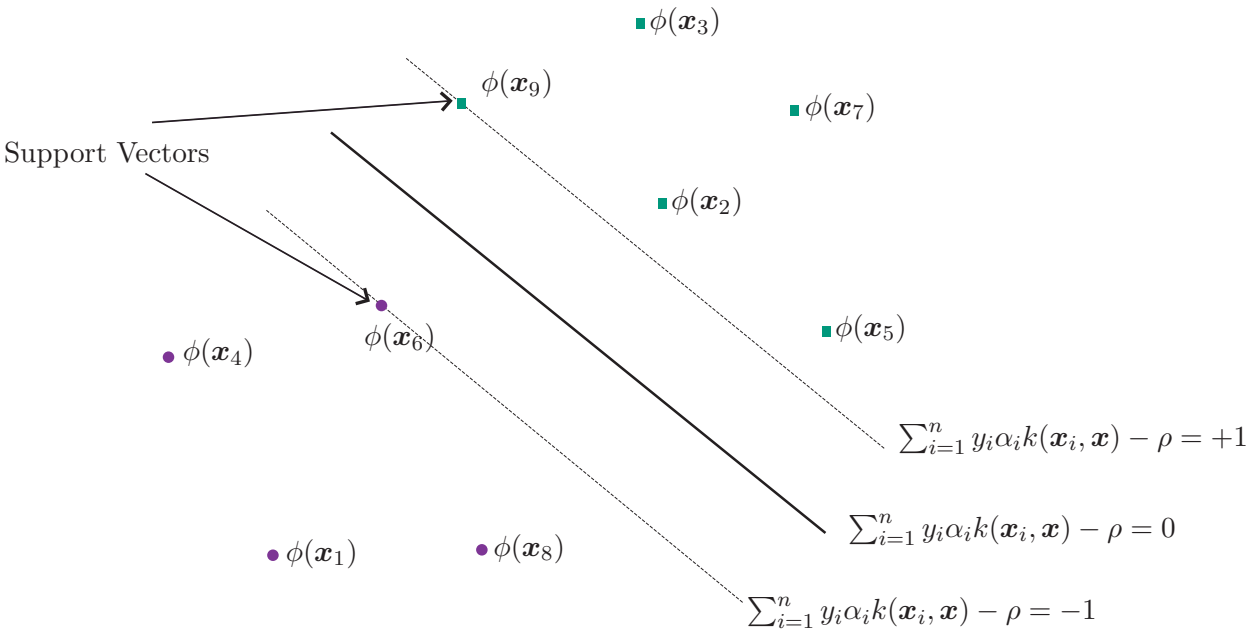

Figure 2.2: An illustration of a binary SVM classification in a 2-dimensional space. The optimal hyperplane defined by $\sum_{i=1}^{n} y_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) - \rho = 0$ separates the two classes with maximal margin.

2.3 RBF Kernels

In order to obtain relevant generalization capacities, the kernel functions that are used must be chosen wisely. We adopt in this thesis the Radial Basis Function (RBF) kernels, since they are the most common and suitable kernels for learning problems [Schölkopf et al., 2001b; Tax and Juszczak, 2002], in particular the Gaussian kernel and the exponential kernel, which follows the form of a Laplace distribution, defined respectively by:

$$\text{Gaussian kernel}: \quad k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma^2}\right)$$
$$\text{Laplacian kernel}: \quad k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2}{b}\right),$$

where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are two input samples, $\|\cdot\|_2$ represents the ℓ_2-norm in the d-dimensional input space, σ denotes the bandwidth parameter of the kernels, and b is a scale parameter that depends on σ. The denominators of both kernels are given in a manner to obtain σ as the standard deviation of the corresponding distribution. In fact, when it comes to the exponential kernel following the Laplacian distribution, the standard deviation is computed as follows:

$$\sigma = \sqrt{2b^2} \quad \Longrightarrow \quad b = \frac{\sigma}{\sqrt{2}}.$$

The bandwidth parameter should be chosen in a way to avoid overfitting and underfitting the data. The performance of kernel methods is highly related to the choice of the metric in the kernel functions, as well as to the choice of the bandwidth parameter. In the following, we detail the impact of varying the norms in the kernel functions and the proposed heuristic for choosing the bandwidth parameter.

2.3.1 The ℓ_p-norms in kernels

Since we are working on industrial processes, (e.g., measuring the pressure inside a gas pipeline, the temperature of a boiling water reactor, the water levels in the storage tanks, monitoring the flow of power in transmission lines, checking the states of the valves and the pumps of the system, etc.), the value of each variable is very important to evaluate the state of the studied physical process, and it is essential to predict if the process is leading to a critical state. Therefore, we need kernels that emphasize simultaneous small changes in several features as well as large variations in a single one. We propose to replace the ℓ_2-norm in the RBF kernels with other norms in order to study the influence of these metrics on the decision function of the classifiers.

Consider a training dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ in a d-dimensional space $\mathcal{X}$. For various values of p, the expression of the ℓ_p-norm between two vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ from $\mathcal{X}$ is given as follows:

$$\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_p = \left(\left|x_{i_1} - x_{j_1}\right|^p + \left|x_{i_2} - x_{j_2}\right|^p + \cdots + \left|x_{i_d} - x_{j_d}\right|^p\right)^{\frac{1}{p}}.$$

For instance, the *Manhattan* distance, also known as the *city-block* norm or ℓ_1-norm, which measures how close are the samples in each direction, has this expression:

$$\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_1 = \sum_{m=1}^{d} \left|x_{i_m} - x_{j_m}\right|, \quad \text{for } i, j = 1, \ldots, n.$$

The expressions of the most common norms that are used in the literature are given in Table 2.2.

Table 2.2: The expressions of the most common norms.

Norm p	$\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_p$
1	$\sum_{m=1}^{d} \left\|x_{i_m} - x_{j_m}\right\|$
2	$\sqrt{\sum_{m=1}^{d} \left(x_{i_m} - x_{j_m}\right)^2}$
∞	$\max \left\|x_{i_m} - x_{j_m}\right\|$
p	$\left(\sum_{m=1}^{d} \left\|x_{i_m} - x_{j_m}\right\|^p\right)^{\frac{1}{p}}$

In order to understand the impact of the metric ℓ_p in kernel functions, Figure 2.3 illustrates the variation in the behavior of different norms in a 2-dimensional space. Each sample has two characteristics, namely feature 1 and feature 2, and p takes one of the values $\frac{3}{4}, 1, \frac{3}{2}, 2, 3, 4, 7$ and ∞. Each color represents equidistant contours with reference to the origin O. It is obvious from this figure that each norm operates differently on simultaneous variation of multiple feature values. For instance, when we consider the ℓ_2-norm, also called the Euclidean distance, a large variation in the value of any feature has a much greater effect than simultaneous variations of both features. Indeed, for this metric, the samples B and C are equidistant from the origin O, the samples A and E are equidistant from O, whereas A and E are much further than B and C. However, for the ℓ_1-norm, C and D are equidistant from the origin O and much closer to O than B, while this same sample B is as far from the origin as C with the ℓ_2-norm. Thus in the ℓ_1-norm case, a simultaneous small change in several features is as important as a large variation in a single one. Therefore, the norms with a small value of p are particularly

sensitive on simultaneous variations of multiple features, while the ones with higher p are more sensitive to large variations in any single feature.

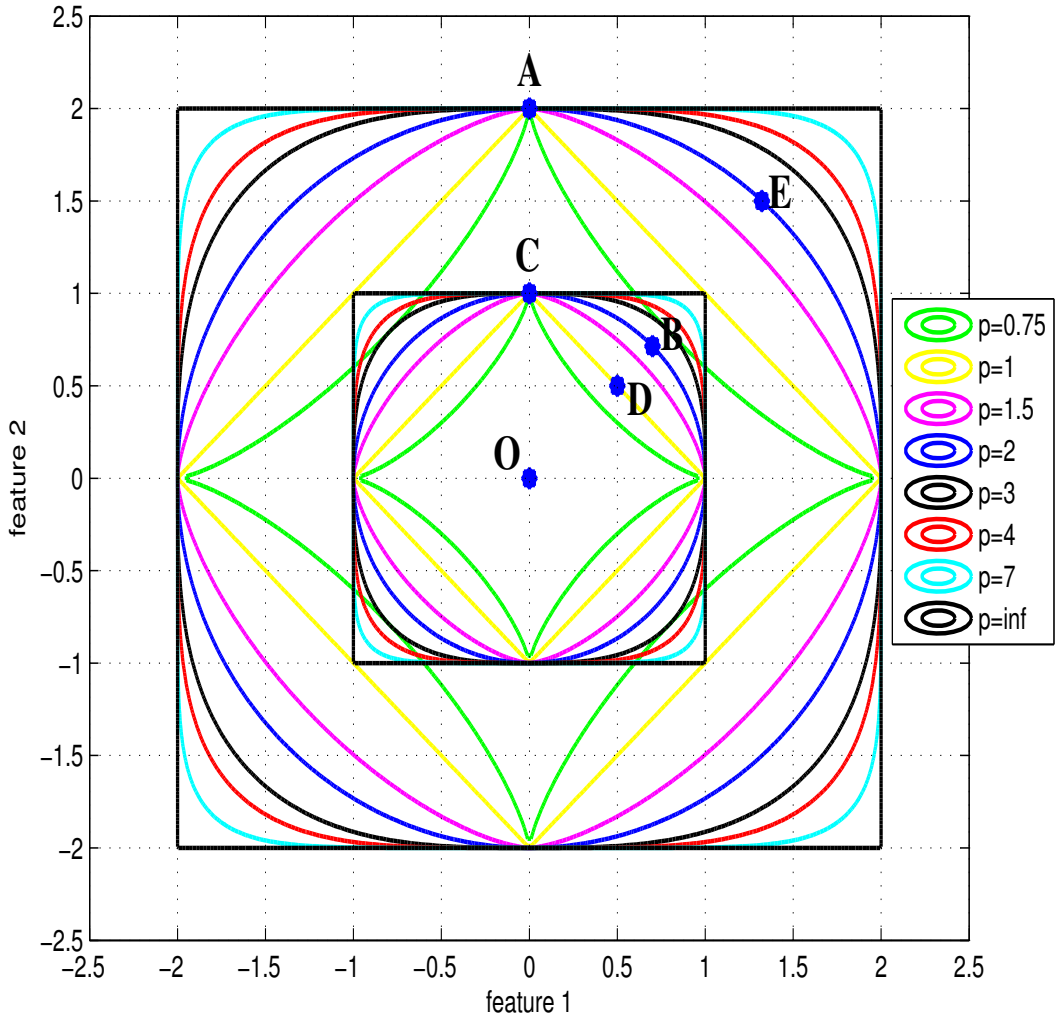

Figure 2.3: The variation of different ℓ_p-norms for p ranging from $p = 0.75$ to $p = \infty$, where each color represents equidistant contours with reference to the origin O. The norms become more sensitive on simultaneous variations of multiple features as p decreases, while the ones with higher p are more sensitive to large variations in single feature directions.

2.3.2 Relation between norms

Consider the Euclidean distance, namely $p = 2$, and the Manhattan distance, namely $p = 1$. One can easily show that the ℓ_2-norm of any vector is bounded by its ℓ_1-norm, namely

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \leq \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_1.$$

In addition, the succeeding relation follows directly from the Cauchy-Schwarz inequality and depends on the dimension d of the space:

$$\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_1 \leq \sqrt{d}\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2.$$

Figure 2.3 illustrates the relations between the ℓ_1-norm and the ℓ_2-norm, where the distances in the case of the Manhattan distance are shorter than the ones in the Euclidean distance case. The mapped samples when using the ℓ_2-norm are more spread out in the feature space than with the ℓ_1-norm, and the relations between their corresponding Laplacian-based exponential and Gaussian kernels are given respectively by:

$$\exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2}{\frac{\sigma}{\sqrt{2}}}\right) \geq \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_1}{\frac{\sigma}{\sqrt{2}}}\right).$$

$$\exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma^2}\right) \geq \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_1^2}{2\sigma^2}\right).$$

The relations between the ℓ_1-norm and the ℓ_2-norm can be generalized to other norms, since we have:

$$\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_{p+a} \leq \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_p$$

for any $p > 0$ and $a \geq 0$, and the generalized Cauchy-Schwarz inequality becomes:

$$\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_p \leq \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_r \leq d^{\left(\frac{1}{r}-\frac{1}{p}\right)}\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_p$$

for any $0 < r < p$. Indeed, the variation of the norm in RBF kernels affects the distribution of the data in the feature space. Moreover, this distribution, depending on the value of p, has an important impact on the resulting decision function of the classifier. Finally, the generalized relations between different norms for the Laplacian-based exponential and Gaussian kernels respectively for any $0 < r < p$ are given by:

$$\exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_p}{\frac{\sigma}{\sqrt{2}}}\right) \geq \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_r}{\frac{\sigma}{\sqrt{2}}}\right).$$

$$\exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_p^2}{2\sigma^2}\right) \geq \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_r^2}{2\sigma^2}\right).$$

2.3.3 Bandwidth parameter

One of the problems when using radial basis functions in machine learning is the choice of the bandwidth parameter of the kernel. The value of this parameter must be chosen wisely since it plays a crucial role in defining the decision function of the classifier. On

one hand, a small value of σ causes the kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to be almost equal to zero for any pair of distinct input samples, which leads to a very tight decision function that overfits the training samples. On the other hand, a large value of σ makes $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to be almost equal to 1, the classifier underfits the data and we obtain a loose decision function.

Several approaches were proposed in the literature in order to choose the bandwidth parameter of RBF kernels. In [Shi and Malik, 2000], the value of the bandwidth parameter was set between 10 to 20 percent of the maximum distance between the training samples; this range works on some cases in image segmentation only, and cannot be generalized to other types of applications. A grid-search of 11 values was proposed for the bandwidth parameter in [Soares et al., 2004] following a geometric series of factor 4. However, this cross-validation technique remains the most expensive in terms of time consumption and does not always lead to the optimal choice in classification problems. A more restricted range was suggested in [Cherkassky and Ma, 2004] depending on the input data and faster than the grid-search, but with poorer results. In [Evangelista et al., 2007], the notion of the coefficient of variance was introduced in order to find the optimal bandwidth parameter, but it did not give the optimal performance on several simulated data. Recently in [Gurram and Kwon, 2011], a two-step iterative heuristic was proposed using a gradient descent algorithm to optimize the bandwidth parameter, but the convergence of the proposed algorithm is not guaranteed and the computational requirements are still important.

The majority of the research work in the literature has been concentrated on supervised learning problems, since it is easier to construct theories and algorithms on labeled data than on unlabeled data. This thesis faces many challenges since it focuses on unsupervised learning. The heuristic proposed in this thesis is inspired by the work in [Haykin, 1998]. Since σ depends on the distribution of the training dataset and on the number of input samples as well as on the fraction of samples considered as support vectors, the estimation of σ should take into consideration all these factors. In addition, we can estimate from the training dataset the fraction of support vectors. Therefore, we propose to use in the classification algorithms of this thesis the following expression for computing the bandwidth parameter σ:

$$\sigma = \frac{d_{\max}}{\sqrt{2M}},$$

where $d_{\max}$ refers to the maximal distance between the samples of the training dataset, and M represents the estimated number of support vectors among this dataset, which is equivalent to the fraction of support vectors multiplied by the total number of the training samples. The relation between the spread of the training dataset and the

number of support vectors defines the bandwidth parameter σ. We bear in mind that the metric of the distance in $d_{\max}$ is the same as the one in the expressions of the kernel functions, e.g., the same ℓ_p-norm appears in $d_{\max}$ and in the kernels for any value of p. The experimental results in the next chapter show that this expression ensures that the extreme cases (overfitting and underfitting the data) are avoided, and the optimization of this parameter is obtained with minimal computational cost, without the need for the time consuming cross-validation step. This heuristic avoids all the time consuming of the other methods existing in the literature. Moreover, the empirical results show that it leads to a very good choice of the bandwidth parameter σ, when it comes to the error detection rates of the classifier.

2.4 Conclusion

In this chapter, we reviewed the role of kernel functions in machine learning. We showed that kernel methods rely on mapping the samples from the input space into a higher dimensional feature space (RKHS). The algorithms applied on the samples in the RKHS were expressed as a function of the pairwise inner product between the samples. The inner products were evaluated by kernel functions, without the need to explicit the nonlinear map. We outlined the main properties of kernel functions, and we detailed the binary SVM for classification problems. We studied the influence of the metric in radial basis function kernels, and we proposed a simple heuristic for choosing the bandwidth parameter with minimum computational cost. In this thesis, we focus on one-class classification techniques, which are useful when the only available samples refer to a single class. The next chapter is devoted to one-class classification.

Chapter 3

One-class Classification

Contents

3.1 Motivations 50

3.2 Common Approaches 51

3.2.1 Support Vector Data Description 53

3.2.2 One-class Support Vector Machines 55

3.2.3 Slab Support Vector Machines 57

3.2.4 Robust Support Vector Machines 59

3.2.5 Kernel Principal Component Analysis 61

3.2.6 Simple One-class 63

3.3 Truncated Mahalanobis-based One-class 63

3.3.1 Full model approach 64

3.3.2 Advantages of KPCA and kernel whitening 66

3.4 Experimental Results 67

3.4.1 Norm variation 67

3.4.2 Bandwidth parameter 74

3.4.3 One-class approaches 76

3.5 Conclusion 81

Over the last decades, machine learning techniques with kernel methods have been widely used since they provide a powerful tool for discovering linear and nonlinear relations in data. In Chapter 2, we reviewed the kernel methods used in machine learning and data mining, by detailing the binary SVM for classification problems. In this Chapter, we outline the importance of one-class classification in machine learning, where the available data refer to a unique class only. This problem appears naturally in many domains, such as in industrial applications where only the normal behavior of the studied system is available.

The remainder of this chapter is organized as follows. We begin with the motivations behind the increasing popularity of one-class classification techniques in Section 3.1. In Section 3.2, we give a review of the existing one-class classification approaches, and we outline the most common one-class algorithms used in the literature. We propose in Section 3.3 the truncated Mahalanobis-based on-class, which is a simple and fast approach for estimating the center of the one-class classifer. We delineate the experimental results in Section 3.4, and we give a conclusion in Section 3.5.

3.1 Motivations

In binary and multi-class classification problems, the decision function of the classifier is supported by the presence of samples from each class, and the associated algorithms are designed to classify any new sample into one of the several pre-defined classes [Mathur and Foody, 2008; Rocha and Klein Goldenstein, 2014]. In several applications as in industrial systems, the only available data designate the normal functioning modes of the studied physical process, while the data related to the malfunctioning modes and to critical states are difficult to obtain. When it comes to industrial processes and detecting machine faults and intrusions, the number of the failure modes and the increasing number of new generated attacks may not be bounded in general [Ten et al., 2008]. This is the reason why researchers have been developing in the last few years algorithms to solve one-class problems, where the available dataset refers only to a single class [Hoffmann, 2007; Chandola et al., 2009].

The one-class classifiers learn the normal behavior modes of the studied system. They develop decision functions in order to test new samples not existing in the training dataset, in a way to accept as many normal samples as possible and to detect the outliers, namely any sample that does not belong to the same distribution of the training dataset [Khan and Madden, 2010]. One-class classification algorithms have been applied in many fields, namely for recognizing emotional and non-emotional facial expressions applications [Zeng et al., 2006], detecting masqueraders from legitimate users in mobile applications [Mazhelis, 2006], Windows registry anomaly detection [Stolfo et al., 2005], visual object recognition in the context of Human-Robot Interaction (HRI) [Wang et al., 2004], time-series novelty detection [Ma and Perkins, 2003], network anomaly detection applications [Zhang et al., 2008], seizure analysis from intracranial Electroencephalography (EEG) signals [Gardner et al., 2006], and recently for intrusion detection in industrial systems and critical infrastructures [Nader et al., 2013, 2014b,c].

In order to describe the application of these algorithms on the physical processes, let $\boldsymbol{x}_i$ be an input sample designating the states related to the normal functioning modes of

the studied system. Two cases can be considered for the learning training phase. On one hand, the $\boldsymbol{x}_i$ can represent several characteristics at a specific time, such as the gas pressure inside the pipeline, the temperature of the industrial blower, the status of the switches and the substations, the water level of the raised water tower, the time interval between two successive measurements, the states of the pumps and valves, and other measurements. On the other hand, $\boldsymbol{x}_i$ can include characteristics from the same component which vary as a function of time, such as the pressure inside the pipeline, or the concentration of some sensitive substances in the chemical processing plant, or the load voltage in the transmission lines, or the water level in storage tanks, or the temperature of the system for three or more consecutive instants. In both cases, the learning phase exploits the hidden relations between the training samples, and the novelty detection occurs when any sample corresponds to a suspicious behavior of the system. In the following, we outline the different one-class classification approaches existing in the literature.

3.2 Common Approaches

Several formulations were proposed in the literature for one-class classification problems. Researchers have been facing many challenges to elaborate relevant one-class algorithms, namely in reducing the computational cost of these algorithms, in improving the detection accuracy, and in avoiding both overfitting and underfitting the data. The one-class Support Vector Machines (one-class SVM), proposed in [Schölkopf et al., 1998a, 2001b], uses a hyperplane in order to separate the mapped data from the origin with maximum margin. This approach requires to solve a constrained quadratic programming problem, thus it is greedy in terms of computational cost. The Support Vector Data Description (SVDD) was introduced in [Tax and Duin, 1999, 2004], and it estimates the hypersphere with minimum radius enclosing most of the training data, and allowing to a fraction of the training samples to remain outside this hypersphere. The resulting optimization problem is essentially similar to the one-class SVM, while they are equivalent when unit-norm kernels are used, such as the Gaussian kernel. Neither one-class SVM nor SVDD take into consideration the heterogenous nature of the mapped data, namely the scale variation in each direction. An attempt to overcome the scale variation drawback is proposed in [Tax and Juszczak, 2002] with a kernel whitening normalization by rescaling the data to have equal variance in each direction of the feature space. The resulting optimization problem incorporated an eigen decomposition problem as well as the conventional constrained quadratic programming problem. In [Azami et al., 2014], the use of the ℓ_0 pseudo-norm in a SVDD formulation was proposed, and this approach provided an iterative procedure by solving a constrained quadratic programming problem at each

iteration, which is very expensive in terms of computational cost. A first attempt for a fast and a simple one-class approach is introduced in [Noumir et al., 2012b] to overcome the drawbacks of existing algorithms, and it computes the Euclidean distance in the feature space between the mapped samples and the center of the data. This approach is faster than the aforementioned methods, but the use of the Euclidean distance in the decision function of the classifier leads to a high sensitivity towards the presence of outliers in the training dataset. The One-Class Neighbor Machine (OCNM), introduced in [Munoz and Moguerza, 2006; Ahmed et al., 2007], is another Euclidean-based one-class classification approach, which inherits the same sensitivity towards outliers than the one in [Noumir et al., 2012b]. The slab Support Vector Machine (slab SVM), described in [Schölkopf et al., 2005; Tao et al., 2005; Eigensatz et al., 2008], aims at finding a slab (two parallel hyperplanes) that encloses the samples that are maximally separated from the origin. Similarly to the one-class SVM, this approach requires to solve a constrained quadratic programming problem and it is expensive in terms of computational cost. The "Robust SVM" algorithm, introduced in [Song et al., 2002] for binary and multi-class classification problems, was modified in [Amer et al., 2013] for anomaly detection in one-class classification problems. This algorithm aims at reducing the influence of the existing outliers on the decision boundary of the standard one-class SVM classifier, by introducing a new slack variable related to the distance between each sample and the center of the data in the feature space. This approach is less sensitive than the standard SVM towards outliers, yet it still requires to solve a constrained quadratic programming problem. Other approaches that use the covariance information to learn the kernel in one-class SVM were proposed in [Tsang et al., 2006; Wang et al., 2006]. These approaches require to solve a second order cone programming problem, and their complexity is cubic with the size of the training dataset. The one-class kernel Fisher discriminant classifier, introduced in [Roth, 2006] for outlier detection, measures the deviation of the samples from a Gaussian distribution. The Gaussian assumption in this approach is violated in the majority of the cases. Kernel Principal Component Analysis (KPCA), introduced in [Schölkopf et al., 1998b] for several applications, was adapted in [Hoffmann, 2007] for one-class classification problems, by projecting the data into the subspace spanned by the most relevant eigenvectors of the covariance matrix. The reconstruction error used as a novelty measure has a relatively low computational cost, yet this approach loses the sparsity of SVM and SVDD.

In the following, we detail the most common one-class classification algorithms in the literature, namely the ones to which we compare our algorithms. Consider a training dataset $\boldsymbol{x}_i$, $i = 1, \ldots, n$, in a d-dimensional input space $\mathcal{X} \subset \mathcal{R}^d$. The training samples are mapped into a reproducing kernel Hilbert space $\mathcal{H}$ via a mapping function ϕ. The RKHS is associated to a kernel function k.

3.2.1 Support Vector Data Description

Support Vector Data Description (SVDD) was introduced in [Tax and Duin, 1999, 2004] in order to get a good description around a training dataset. SVDD computes a spherically shaped decision boundary with minimum radius enclosing most of the training samples in the feature space. Samples that lay outside this hypersphere are considered as outliers, and they should be detected by the SVDD classifier.

The hypersphere that encompasses with minimum radius most of the data $\phi(\boldsymbol{x}_i)$ in the feature space $\mathcal{H}$ is characterized by its center $\boldsymbol{a}$ and its radius $R > 0$. The SVDD algorithm minimizes its volume by minimizing R^2. To avoid a large description that does not represent the data very well, the presence of outliers in the training set is allowed by introducing a slack variable $\xi_i \geq 0$ for each training sample $\boldsymbol{x}_i$, which allows to penalize the samples lying outside the hypersphere. This boils down to the following constrained optimization problem:

$$\min_{\boldsymbol{a},R,\xi_i} R^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i, \tag{3.1}$$

subject to

$$\|\phi(\boldsymbol{x}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2 \leq R^2 + \xi_i \quad \text{and} \quad \xi_i \geq 0 \quad \forall i = 1, ..., n.$$

The predefined parameter $\nu \in (0, 1)$ regulates the trade-off between the volume of the hypersphere and the number of outliers. Its value represents an upper bound on the fraction of outliers and a lower bound on the fraction of support vectors (the support vectors refer to the data *on* and *outside* the boundary). The Lagrangian of this optimization problem is constructed as follows:

$$L(R, \boldsymbol{a}, \alpha_i, \gamma_i, \xi_i) = R^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\Big(R^2 + \xi_i - \|\phi(\boldsymbol{x}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2\Big) - \sum_{i=1}^{n}\gamma_i\xi_i, \tag{3.2}$$

where the α_i's and the γ_i's are the Lagrangian multipliers. The partial derivatives of the Lagrangian with respect to R, ξ_i and $\boldsymbol{a}$ are nullified, which gives the following relations:

$$\begin{aligned}
\frac{\partial L}{\partial R} = 0 \quad &\Longleftrightarrow \quad \sum_{i}^{n}\alpha_i = 1, \\
\frac{\partial L}{\partial \xi_i} = 0 \quad &\Longleftrightarrow \quad 0 \leq \alpha_i \leq \frac{1}{\nu n}, \\
\frac{\partial L}{\partial \boldsymbol{a}} = 0 \quad &\Longleftrightarrow \quad \boldsymbol{a} = \frac{\sum_i^n \alpha_i\phi(\boldsymbol{x}_i)}{\sum_{i=1}^n \alpha_i} \\
&\qquad\qquad = \sum_{i}^{n}\alpha_i\phi(\boldsymbol{x}_i),
\end{aligned}$$

where the expression of the center of the hypersphere $\boldsymbol{a}$ serves as the representer theorem. Incorporating these relations into the Lagrangian gives us the following objective functional to be maximized with respect to α_i:

$$L = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_i) - \sum_{i,j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{3.3}$$

subject to

$$\sum_{i=1}^{n} \alpha_i = 1 \quad \text{and} \quad 0 \;\leq\; \alpha_i \;\leq\; \frac{1}{\nu n}.$$

We now have a convex constrained quadratic programming problem, whose solution is found using any off-the-shelf optimization technique. For instance, one can use the MATLAB function quadprog. The values of the Lagrangian multipliers depend on whether the constraint $\|\phi(\boldsymbol{x}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2 \leq R^2 + \xi_i$ is satisfied by the corresponding sample $\boldsymbol{x}_i$. We encounter one of these three cases:

$$\begin{aligned}
\|\phi(\boldsymbol{x}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2 < R^2 \quad &\Longleftrightarrow \quad \alpha_i = 0, \\
\|\phi(\boldsymbol{x}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2 = R^2 \quad &\Longleftrightarrow \quad 0 < \alpha_i < \frac{1}{\nu n}, \\
\|\phi(\boldsymbol{x}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2 > R^2 \quad &\Longleftrightarrow \quad \alpha_i = \frac{1}{\nu n}.
\end{aligned}$$

The radius of the optimal hypersphere is obtained with the distance in the feature space $\mathcal{H}$ from the center $\boldsymbol{a}$ to any sample $\phi(\boldsymbol{x}_k)$ on the boundary, where these samples satisfy the following constraint $0 < \alpha_k < \frac{1}{\nu n}$:

$$\begin{aligned}
R^2 &= \|\phi(\boldsymbol{x}_k) - \boldsymbol{a}\|_{\mathcal{H}}^2 \\
&= \Big\langle \phi(\boldsymbol{x}_k) - \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_k) - \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i) \Big\rangle_{\mathcal{H}} \\
&= k(\boldsymbol{x}_k, \boldsymbol{x}_k) - 2 \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_k, \boldsymbol{x}_i) + \sum_{i,j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j).
\end{aligned}$$

In order to evaluate a new sample $\boldsymbol{x}$, the decision rule is obtained by evaluating the distance between the center $\boldsymbol{a}$ and the sample $\phi(\boldsymbol{x})$ in the feature space. If this distance is smaller than the radius, the new sample $\boldsymbol{x}$ is considered as a normal sample:

$$\|\phi(\boldsymbol{x}) - \boldsymbol{a}\|_{\mathcal{H}}^2 \leq R^2;$$

Otherwise, $\boldsymbol{x}$ is considered as an outlier.

3.2.2 One-class Support Vector Machines

Support Vector Machines (SVM), initially studied in [Vapnik, 1995] for binary and multiclass classification as well as regression problems, were adapted to solve one-class classification problems in [Schölkopf et al., 1998a, 2001b; Decoste and Schölkopf, 2002]. In contrast to SVDD which defines the hypersphere enclosing most of the samples in the feature space, the one-class SVM finds the hyperplane that separates the samples from the origin with maximum margin. The one-class SVM algorithm develops a decision function that evaluates which side of the hyperplane each sample falls on. An example of the hyperplane in a 2-dimensional feature space is given in Figure 3.1.

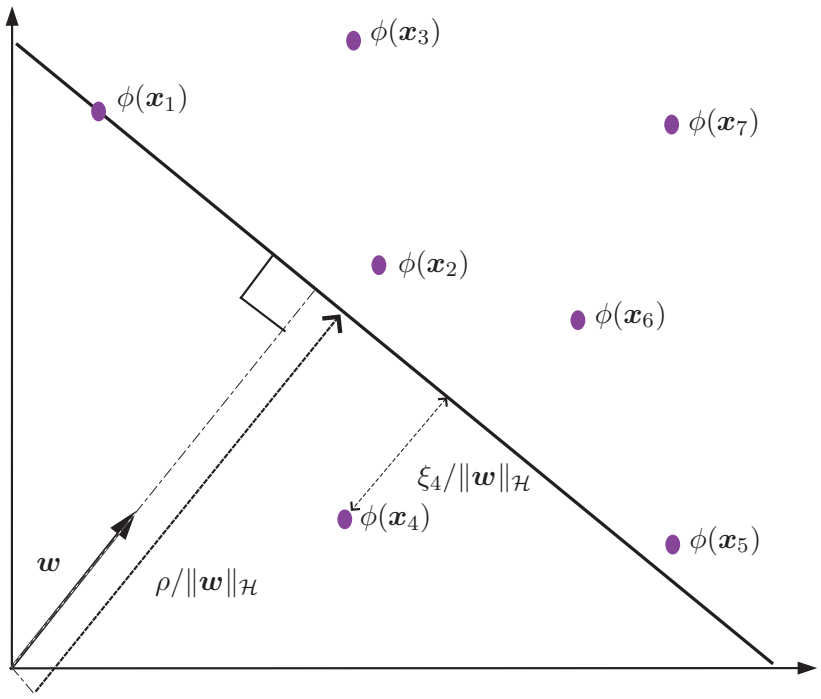

Figure 3.1: An illustration of the hyperplane separating the mapped samples from the origin with maximum margin in a 2-dimensional feature space, where ρ and $\|\boldsymbol{w}\|$ are the parameters of this hyperplane. The sample $\phi(\boldsymbol{x}_4)$ is the only outlier in this example, and it is penalized by the associated slack variable ξ_4. The distance between the hyperplane and the outlier is $\xi_4/\|\boldsymbol{w}\|_{\mathcal{H}}$, and the one between the hyperplane and the origin is $\rho/\|\boldsymbol{w}\|_{\mathcal{H}}$.

In order to separate the samples from the origin with maximum margin, we solve the following constrained quadratic problem:

$$\min_{\boldsymbol{w},\rho,\xi_i} \frac{1}{2}\|\boldsymbol{w}\|_{\mathcal{H}}^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho, \tag{3.4}$$

subject to

$$\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{H}} \geq \rho - \xi_i \quad \text{and} \quad \xi_i \geq 0 \quad \forall i = 1, ..., n,$$

where the slack variables $\xi_i \geq 0$ penalize the excluded samples, and the tunable parameter ν represents an upper bound on the fraction of outliers. The Lagrangian of this optimization problem is constructed as follows:

$$L(\rho, \boldsymbol{w}, \alpha_i, \gamma_i, \xi_i) = \frac{1}{2}\|\boldsymbol{w}\|^2_{\mathcal{H}} + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho - \sum_{i=1}^{n}\alpha_i\Big(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle - \rho + \xi_i\Big) - \sum_{i=1}^{n}\gamma_i\xi_i, \quad (3.5)$$

where the α_i's and the γ_i's are the Lagrangian multipliers. The partial derivatives of the Lagrangian with respect to ρ, ξ_i and $\boldsymbol{w}$ are nullified, which leads to the following relations:

$$\begin{aligned}
\frac{\partial L}{\partial \rho} = 0 \quad &\Longleftrightarrow \quad \sum_{i=1}^{n}\alpha_i = 1, \\
\frac{\partial L}{\partial \xi_i} = 0 \quad &\Longleftrightarrow \quad 0 \leq \alpha_i \leq \frac{1}{\nu n}, \\
\frac{\partial L}{\partial \boldsymbol{w}} = 0 \quad &\Longleftrightarrow \quad \boldsymbol{w} = \sum_{i=1}^{n}\alpha_i\phi(\boldsymbol{x}_i).
\end{aligned}$$

We include these relations into the Lagrangian to obtain the following objective functional to be minimized with respect to α_i:

$$L = \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad (3.6)$$

subject to

$$\sum_{i=1}^{n}\alpha_i = 1 \quad \text{and} \quad 0 \;\leq\; \alpha_i \;\leq\; \frac{1}{\nu n}.$$

Once again, we have a convex constrained quadratic programming problem.

The decision function for evaluating any new sample $\boldsymbol{x}$ has the following form:

$$\begin{aligned}
f(\boldsymbol{x}) &= \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{H}} - \rho \\
&= \sum_{i=1}^{n}\alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) - \rho,
\end{aligned}$$

where we can encounter one of the following three cases:

- $f(\boldsymbol{x}) > 0 \longrightarrow$ $\boldsymbol{x}$ is a normal sample, i.e., lies among the training dataset.
- $f(\boldsymbol{x}) = 0 \longrightarrow$ $\boldsymbol{x}$ is a normal sample that lies on the hyperplane.
- $f(\boldsymbol{x}) < 0 \longrightarrow$ $\boldsymbol{x}$ is considered an outlier as it lies between the hyperplane and the origin.

The one-class SVM algorithm becomes equivalent to the SVDD when the kernel function used in the learning process satisfies $k(\boldsymbol{x}, \boldsymbol{x}) = \text{constant}$, which is the case with the Gaussian kernel having $k(\boldsymbol{x}, \boldsymbol{x}) = 1$.

3.2.3 Slab Support Vector Machines

The slab Support Vector Machines (slab SVM) described in [Schölkopf et al., 2005; Tao et al., 2005; Eigensatz et al., 2008] is a modified version of the standard one-class SVM algorithm, in which a change in the geometric setup leads to a reformulation of the standard optimization problem of the SVM. In contrast to the case of the standard one-class SVM, the slab SVM aims at finding a region bounded by two parallel hyperplanes, called a slab, that encloses the samples in the feature space, in a way to maximally separate this slab from the origin. An example of the area enclosed between two hyperplanes is given in Figure 3.2.

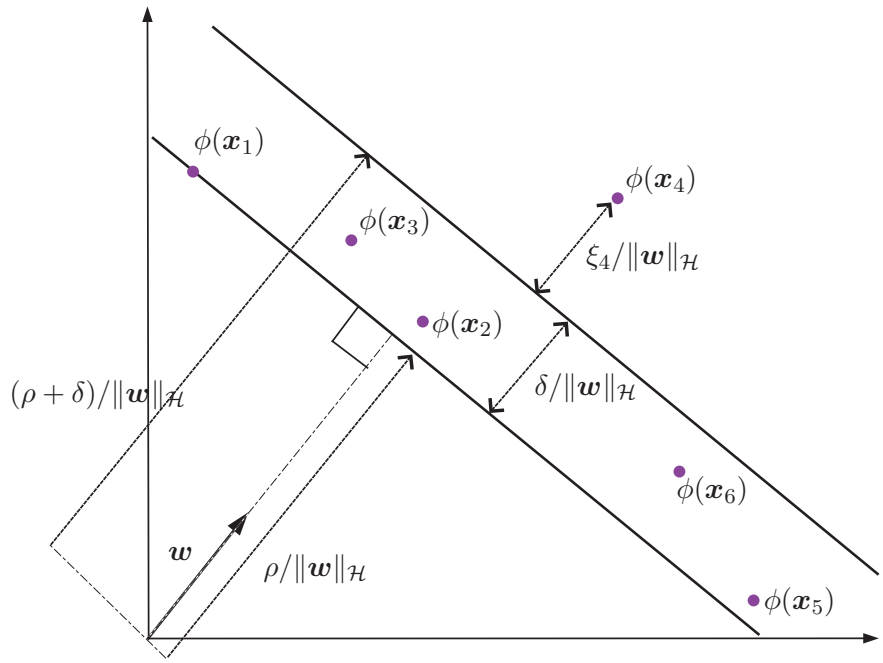

Figure 3.2: An example of the slab bounded by two hyperplanes separating the mapped samples from the origin with maximum margin in a 2-dimensional feature space, where ρ, $\|\boldsymbol{w}\|$ and δ are the parameters of this slab. The distance between the two hyperplanes is given by $\delta/\|\boldsymbol{w}\|_{\mathcal{H}}$. The outlier $\phi(\boldsymbol{x}_4)$ is penalized by the associated slack variable ξ_4.

The constrained optimization problem that needs to be solved in order to find the two hyperplanes of the slab SVM is given as follows:

$$\min_{\boldsymbol{w},\rho,\xi_i} \frac{1}{2}\|\boldsymbol{w}\|_{\mathcal{H}}^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho, \tag{3.7}$$

subject to

$$0 \leq \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle_{\mathcal{H}} - \rho + \xi_i \leq \delta,$$

where ξ_i denote the slack variables penalizing the excluded samples, and ν represents an upper bound on the fraction of outliers. Considering the Lagrangian of the above constrained optimization problem, and incorporating the relations from its partial derivatives with respect to $\boldsymbol{w}$, ρ and ξ_i gives us the following objective functional to be minimized with respect to the Lagrangian multipliers α_i and β_i:

$$L = \frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i - \beta_i)(\alpha_j - \beta_j) k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \delta \sum_{i=1}^{n} \beta_i, \tag{3.8}$$

subject to

$$0 \;\leq\; \alpha_i, \beta_i \;\leq\; \frac{1}{\nu n}, \quad \text{for} \quad \forall i = 1, ..., n,$$
$$\text{and} \quad \sum_{i=1}^{n} (\alpha_i - \beta_i) = 1.$$

This dual problem is a constrained quadratic programming problem, which can be solved using standard quadratic programming techniques.

A new sample $\boldsymbol{x}$ is considered as a normal one if it lies between the "lower" hyperplane, defined by

$$\langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}} - \rho = 0,$$

and the "upper" hyperplane, given by

$$\langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}} - \rho - \delta = 0,$$

where $\boldsymbol{w}$ has this expression:

$$\boldsymbol{w} = \sum_{i=1}^{n} (\alpha_i - \beta_i) \phi(\boldsymbol{x}_i).$$

For each new sample $\boldsymbol{x}$, we can encounter one of the following cases:

- $0 < \sum_{i=1}^{n} (\alpha_i - \beta_i) k(\boldsymbol{x}_i, \boldsymbol{x}) - \rho < \delta \quad \longrightarrow \quad \boldsymbol{x}$ is a normal sample, i.e., lies between the lower and the upper hyperplanes.
- $\sum_{i=1}^{n} (\alpha_i - \beta_i) k(\boldsymbol{x}_i, \boldsymbol{x}) - \rho = 0 \quad \longrightarrow \quad \boldsymbol{x}$ is a normal sample that lies on the lower hyperplane.
- $\sum_{i=1}^{n} (\alpha_i - \beta_i) k(\boldsymbol{x}_i, \boldsymbol{x}) - \rho - \delta = 0 \quad \longrightarrow \quad \boldsymbol{x}$ is a normal sample that lies on the upper hyperplane.

- $\sum_{i=1}^{n}(\alpha_i - \beta_i)k(\boldsymbol{x}_i, \boldsymbol{x}) - \rho < 0$ or $\sum_{i=1}^{n}(\alpha_i - \beta_i)k(\boldsymbol{x}_i, \boldsymbol{x}) - \rho - \delta > 0 \longrightarrow \boldsymbol{x}$ is an outlier since it lies outside the slab defined by the two hyperplanes.

3.2.4 Robust Support Vector Machines

The Robust SVM algorithm is another modified version of the standard SVM, and it was adapted for one-class classification problems in [Amer et al., 2013]. The main objective of this algorithm is to make the SVM algorithm more robust towards outliers, by reducing the influence of the existing outliers on the decision boundary of the classifier. The non-zero slack variables used in standard SVM to allow the samples to lie on the other side of the decision boundary are dropped from the optimization objective, and they are replaced with another slack variable related to the distance between the samples and the center of the data in the feature space. This will cause the decision boundary to be shifted towards the normal samples.

Let $\boldsymbol{c}_n$ denote the empirical center of the data in the feature space. The expression of $\boldsymbol{c}_n$ is given as follows:

$$\boldsymbol{c}_n = \frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}_i).$$

The new slack variable introduced in the optimization problem represents the ratio between the squared distance of the samples to the center $\boldsymbol{c}_n$ and the maximal value of this distance $\text{dist}_{\max}$, and it is computed as follows:

$$\begin{aligned}
\xi_i &= \left(\left\|\phi(\boldsymbol{x}_i) - \frac{1}{n}\sum_{k=1}^{n}\phi(\boldsymbol{x}_k)\right\|_{\mathcal{H}}^2\right)/\text{dist}_{\max} \\
&= \left(k(\boldsymbol{x}_i, \boldsymbol{x}_i) - \frac{2}{n}\sum_{k=1}^{n}k(\boldsymbol{x}_i, \boldsymbol{x}_k) + \frac{1}{n^2}\sum_{i,k=1}^{n}k(\boldsymbol{x}_i, \boldsymbol{x}_k)\right)/\text{dist}_{\max} \\
&\approx \left(k(\boldsymbol{x}_i, \boldsymbol{x}_i) - \frac{2}{n}\sum_{k=1}^{n}k(\boldsymbol{x}_i, \boldsymbol{x}_k)\right)/\text{dist}_{\max},
\end{aligned}$$

where the approximation comes from dropping the last term in the expression of the distance which is constant. The Robust SVM solves the following constrained optimization problem:

$$\min_{\boldsymbol{w},\rho}\frac{1}{2}\|\boldsymbol{w}\|_{\mathcal{H}}^2 - \rho, \tag{3.9}$$

subject to

$$\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{H}} \geq \rho - \lambda\xi_i,$$

where ξ_i are the slack variables and λ a regularization parameter. Incorporating the relations from the partial derivatives of the Lagrangian of this optimization problem with respect to $\boldsymbol{w}$ and ρ gives the following objective functional to be minimized with

respect to the Lagrangian multipliers α_i:

$$L = \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \lambda \sum_{i=1}^{n} \xi_i \alpha_i, \tag{3.10}$$

subject to

$$0 \leq \alpha_i \leq 1, \quad \text{for} \quad \forall i = 1, ..., n,$$
$$\text{and} \quad \sum_{i=1}^{n} \alpha_i = 1.$$

This is a constrained quadratic programming problem. The use of this type of slack variables can be justified as follows. The separation margin for each mapped sample can be thought as $\rho - \lambda\xi_i$. For any outlier, its distance to the center $\boldsymbol{c}_n$ is greater than the one of normal samples, so the augmented term $\lambda\xi_i$ is relatively large, and the coefficient α_i associated with this sample should be close to zero. Therefore, the decision function of the classifier will be less affected by outliers. An example that illustrates the different decision hyperplanes of a standard SVM and the Robust SVM is given in Figure 3.3.

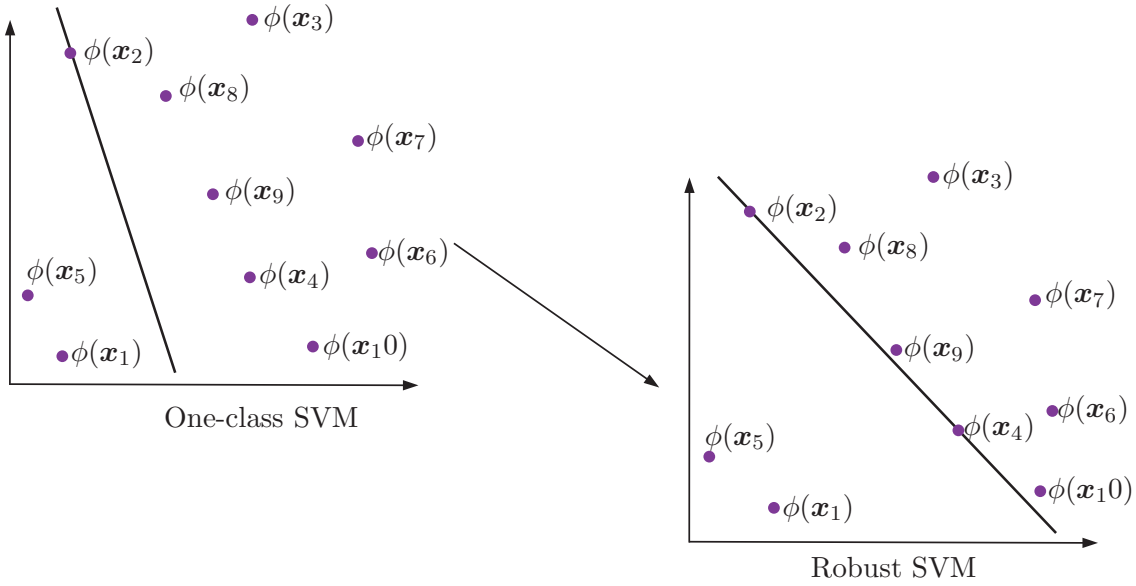

Figure 3.3: A comparison between the decision hyperplanes of the one-class SVM approach and the Robust SVM. The modified slack variables in the Robust SVM reduce the influence of the outliers on the decision hyperplane of the classifier, which will be shifted towards the samples that represent the normal class.

3.2.5 Kernel Principal Component Analysis

Kernel Principal Component Analysis (KPCA) is a nonlinear extension of PCA in a kernel-defined feature space, where using the linear kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle$ is equivalent to performing the original PCA [Schölkopf et al., 1998b]. KPCA extracts the subspace with maximum variance of the data, and allows dimensionality reduction and denoising by projecting the samples into that subspace. Thanks to the intrinsic properties of the KPCA, it takes into account the heterogeneous variance of the distribution of the data in the feature space. KPCA was investigated in [Hoffmann, 2007] for one-class classification by introducing the *reconstruction error* as a measure of novelty.

In order to extract the subspace with maximum variance of the data, the KPCA algorithm seeks the eigenvectors $\boldsymbol{v}^k$ associated to the largest eigenvalues λ^k of the covariance matrix $\boldsymbol{\Sigma}$, defined in the feature space by

$$\boldsymbol{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} \big(\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n\big)\big(\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n\big)^T,$$

where $\boldsymbol{c}_n$ is the empirical center of the data in the feature space:

$$\boldsymbol{c}_n = \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i).$$

The eigenvalues and eigenvectors of the covariance matrix satisfy

$$\lambda^k \boldsymbol{v}^k = \boldsymbol{\Sigma} \boldsymbol{v}^k.$$

It is easy to see that each each eigenvector $\boldsymbol{v}^k$ is a linear combination of the samples and takes the following form:

$$\boldsymbol{v}^k = \sum_{i=1}^{n} \alpha_i^k \widetilde{\phi}(\boldsymbol{x}_i),$$

where $\widetilde{\phi}(\boldsymbol{x}_i)$ is the centered version of $\phi(\boldsymbol{x}_i)$ in the feature space, namely

$$\widetilde{\phi}(\boldsymbol{x}_i) = \phi(\boldsymbol{x}_i) - \frac{1}{n} \sum_{j=1}^{n} \phi(\boldsymbol{x}_j).$$

The coefficients α_i are given by solving the following eigen decomposition problem

$$n\lambda^k \boldsymbol{\alpha}^k = \widetilde{\boldsymbol{K}} \boldsymbol{\alpha}^k,$$

where the kernel function $\widetilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ corresponding to the centered version $\widetilde{\phi}(\boldsymbol{x}_i)$ is computed as follows:

$$\begin{aligned}\widetilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \langle \widetilde{\phi}(\boldsymbol{x}_i), \widetilde{\phi}(\boldsymbol{x}_j) \rangle_{\mathcal{H}} \\ &= k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \frac{1}{n}\sum_{r=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{x}_r) - \frac{1}{n}\sum_{r=1}^{n} k(\boldsymbol{x}_r, \boldsymbol{x}_j) + \frac{1}{n^2}\sum_{r,s=1}^{n} k(\boldsymbol{x}_r, \boldsymbol{x}_s).\end{aligned}$$

In fact, this centered kernel matrix is used in the optimization problem without the need to compute directly the covariance matrix $\boldsymbol{\Sigma}$.

The reconstruction error of any sample $\boldsymbol{x}$ measures the squared distance in the feature space between the centered sample $\widetilde{\phi}(\boldsymbol{x})$ and its projection onto the subspace spanned by the most relevant eigenvectors of the covariance matrix. Let $\mathcal{P}$ be the projection operator onto the subspace spanned by the q eigenvectors $\boldsymbol{v}^1, \boldsymbol{v}^2, \cdots, \boldsymbol{v}^q$. The reconstruction error is computed as follows:

$$\left\|\widetilde{\phi}(\boldsymbol{x}) - \mathcal{P}\widetilde{\phi}(\boldsymbol{x})\right\|_{\mathcal{H}}^2 = \left\langle \widetilde{\phi}(\boldsymbol{x}), \widetilde{\phi}(\boldsymbol{x}) \right\rangle_{\mathcal{H}} - 2\left\langle \widetilde{\phi}(\boldsymbol{x}), \mathcal{P}\widetilde{\phi}(\boldsymbol{x}) \right\rangle_{\mathcal{H}} + \left\langle \mathcal{P}\widetilde{\phi}(\boldsymbol{x}), \mathcal{P}\widetilde{\phi}(\boldsymbol{x}) \right\rangle_{\mathcal{H}}. \quad (3.11)$$

Since the projection operator $\mathcal{P}$ satisfies the two main properties:

- $\mathcal{P}$ is idempotent $\longrightarrow$ $\mathcal{P}^2 = \mathcal{P}$,
- $\mathcal{P}$ is self-adjoint $\longrightarrow$ $\left\langle \mathcal{P}\widetilde{\phi}(\boldsymbol{x}), \widetilde{\phi}(\boldsymbol{x}') \right\rangle_{\mathcal{H}} = \left\langle \widetilde{\phi}(\boldsymbol{x}), \mathcal{P}\widetilde{\phi}(\boldsymbol{x}') \right\rangle_{\mathcal{H}}$,

then the reconstruction error's expression (3.11) is simplified as follows:

$$\left\|\widetilde{\phi}(\boldsymbol{x}) - \mathcal{P}\widetilde{\phi}(\boldsymbol{x})\right\|_{\mathcal{H}}^2 = \widetilde{k}(\boldsymbol{x}, \boldsymbol{x}) - \left\langle \mathcal{P}\widetilde{\phi}(\boldsymbol{x}), \mathcal{P}\widetilde{\phi}(\boldsymbol{x}) \right\rangle_{\mathcal{H}},$$

where

$$\mathcal{P}\widetilde{\phi}(\boldsymbol{x}) = \sum_{l=1}^{q} \left\langle \widetilde{\phi}(\boldsymbol{x}), \boldsymbol{v}^l \right\rangle_{\mathcal{H}} \frac{\boldsymbol{v}^l}{\|\boldsymbol{v}^l\|_{\mathcal{H}}}.$$

Since the eigenvectors are orthonormal, we obtain:

$$\left\langle \mathcal{P}\widetilde{\phi}(\boldsymbol{x}), \mathcal{P}\widetilde{\phi}(\boldsymbol{x}) \right\rangle_{\mathcal{H}} = \sum_{l=1}^{q} \left\langle \widetilde{\phi}(\boldsymbol{x}), \boldsymbol{v}^l \right\rangle_{\mathcal{H}}^2,$$

and the expression of the reconstruction error in the feature space becomes:

$$\left\|\widetilde{\phi}(\boldsymbol{x}) - \mathcal{P}\widetilde{\phi}(\boldsymbol{x})\right\|_{\mathcal{H}}^2 = \widetilde{k}(\boldsymbol{x}, \boldsymbol{x}) - \sum_{l=1}^{q} \left\langle \widetilde{\phi}(\boldsymbol{x}), \boldsymbol{v}^l \right\rangle_{\mathcal{H}}^2. \quad (3.12)$$

After evaluating the reconstruction error for all the samples of the training dataset, an error threshold is fixed based on the predefined number of outliers. In order to

decide whether a new sample belongs to the same distribution as the training dataset, its reconstruction error is evaluated. If this error is smaller than the threshold, the corresponding sample is treated as a normal one; Otherwise, it is considered as an outlier.

3.2.6 Simple One-class

The main drawback of the existing one-class classification algorithms, specifically the ones derived from the SVM and SVDD, is the high computational complexity since a constrained quadratic programming problem has to be solved. In order to overcome the time consumption drawback of the existing algorithms, a fast and simple one-class approach was introduced in [Noumir et al., 2012b]. This approach finds a hypersphere that encloses most of the samples, by estimating its center without solving any quadratic programming problem. The decision function in this approach is based on the Euclidean distance in the feature space between the samples and the center of the hypersphere.

Let $\boldsymbol{c}_n$ be the center of the data in the feature space estimated using all the available training samples, namely $\boldsymbol{c}_n = \frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}_i)$. The expression of the squared distance between any sample $\phi(\boldsymbol{x})$ and the center $\boldsymbol{c}_n$ is given as follows:

$$\left\|\phi(\boldsymbol{x}) - \frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}_i)\right\|_{\mathcal{H}}^2 = k(\boldsymbol{x},\boldsymbol{x}) - \frac{2}{n}\sum_{i=1}^{n}k(\boldsymbol{x},\boldsymbol{x}_i) + \frac{1}{n^2}\sum_{i,j=1}^{n}k(\boldsymbol{x}_i,\boldsymbol{x}_j).$$

After evaluating this distance between all the training samples and the estimated center, the simple one-class algorithm defines a threshold based on the estimated fraction of outliers among the training dataset. The decision function for a new sample $\boldsymbol{x}$ is defined by its Euclidean distance to the center. If this distance is greater than the predefined threshold, the sample is considered as an outlier.

3.3 Truncated Mahalanobis-based One-class

In the previous Section, we outlined the one-class classification approaches existing in the literature, and we detailed the most common algorithms for one-class problems. The main drawback of most of these approaches is the high computational cost of the quadratic programming problems that need to be solved. Another drawback of some of these approaches is the sensitivity of the algorithms towards the presence of outliers among the training dataset. In order to overcome the drawbacks of the aforementioned algorithms, we propose a simple and fast one-class classification approach. The one-class classifier of the proposed approach is defined by the hypersphere enclosing the training

samples in the feature space, and we estimate the center of this hypersphere without solving any quadratic programming problem, by following the work of [Noumir et al., 2012b]. In opposition to their work where the Euclidean distance was used, we propose a new novelty measure based on the truncated Mahalanobis distance in the feature space. In fact, the Mahalanobis distance is a multivariate dissimilarity that takes into account the scatter of the data in that space [Mahalanobis, 1936]. To take advantage of the properties of KPCA [Hoffmann, 2007] and kernel whitening [Tax and Juszczak, 2002], we propose the truncated Mahalanobis distance, which uses the most relevant axes in the feature space.

3.3.1 Full model approach

The truncated Mahalanobis-based one-class approach finds the hypersphere enclosing the training samples in the feature space. This approach uses the truncated Mahalanobis distance between the studied sample and the center of the training data in that space as a novelty measure that classifies it as a normal one or an outlier.

The expectation of the samples in the feature space has the following form:

$$\mathrm{E}[\phi(\boldsymbol{x})] = \int_{\mathcal{X}} \phi(\boldsymbol{x}) P(\boldsymbol{x}) d\boldsymbol{x},$$

having $P(\boldsymbol{x})$ the probability distribution of the training samples over $\mathcal{X}$. Since the distribution $P(\boldsymbol{x})$ is usually unknown, one can estimate this expectation by the empirical center of the training dataset, with

$$\boldsymbol{c}_n = \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i).$$

The empirical center of the training dataset respresents the center of the hypersphere. The Mahalanobis distance between a sample $\phi(\boldsymbol{x})$ and $\boldsymbol{c}_n$ is defined as follows:

$$\big\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\big\|_{\boldsymbol{\Sigma}}^2 = \big(\phi(\boldsymbol{x}) - \boldsymbol{c}_n\big)\boldsymbol{\Sigma}^{-1}\big(\phi(\boldsymbol{x}) - \boldsymbol{c}_n\big), \tag{3.13}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the data in the feature space, namely

$$\boldsymbol{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} \big(\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n\big)\big(\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n\big)^T. \tag{3.14}$$

Without any explicit knowledge of the mapping function $\phi(\cdot)$, the covariance matrix cannot be expressed in terms of the data $\phi(\boldsymbol{x})$ in the feature space. To overcome this

problem, we use the singular value decomposition of the covariance matrix $\boldsymbol{\Sigma}$ as follows:

$$\boldsymbol{\Sigma} = \boldsymbol{V}^T \boldsymbol{D} \boldsymbol{V},$$

having $\boldsymbol{V}$ the matrix of eigenvectors $\boldsymbol{v}^k$ of $\boldsymbol{\Sigma}$, $\boldsymbol{D}$ the diagonal matrix with the corresponding eigenvalues λ^k, for $k = 1, 2, \cdots, n$, where each pair $(\boldsymbol{v}^k, \lambda^k)$ satisfies

$$\lambda^k \boldsymbol{v}^k = \boldsymbol{\Sigma} \boldsymbol{v}^k.$$

From the definition of the matrix $\boldsymbol{\Sigma}$, it is easy to see that each eigenvector is a linear combination of the training samples $\phi(\boldsymbol{x}_i)$ in the feature space, namely:

$$\boldsymbol{v}^k = \sum_{i=1}^{n} \alpha_i^k \big(\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n\big) = \sum_{i=1}^{n} \alpha_i^k \Big(\phi(\boldsymbol{x}_i) - \frac{1}{n} \sum_{j=1}^{n} \phi(\boldsymbol{x}_j)\Big).$$

By incorporating the expression of $\boldsymbol{v}^k$ in the eigen decomposition of $\boldsymbol{\Sigma}$, namely $\lambda^k \boldsymbol{v}^k = \boldsymbol{\Sigma} \boldsymbol{v}^k$, the coefficients α_i^k are given by solving the eigen decomposition problem

$$n \lambda^k \boldsymbol{\alpha}^k = \widetilde{\boldsymbol{K}} \boldsymbol{\alpha}^k, \tag{3.15}$$

where the matrix $\widetilde{\boldsymbol{K}}$ of entries $\widetilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the centered version of $\boldsymbol{K}$, and the corresponding mapping function is given by: $\widetilde{\phi}(\boldsymbol{x}_i) = \phi(\boldsymbol{x}_i) - \boldsymbol{c}_n$.

Since $\boldsymbol{V}$ is an orthogonal matrix, the inverse of the covariance matrix $\boldsymbol{\Sigma}^{-1}$ can be expressed as follows:

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{V}^T \boldsymbol{D}^{-1} \boldsymbol{V} = \boldsymbol{V}^T \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{V}. \tag{3.16}$$

Next, equation (3.13) takes this form $\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\|_{\boldsymbol{\Sigma}}^2 = \boldsymbol{a}^T \boldsymbol{a}$ having:

$$\boldsymbol{a} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{V} \Big(\phi(\boldsymbol{x}) - \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i)\Big),$$

where each entry a^k of $\boldsymbol{a}$ is associated to an eigenvector $\boldsymbol{v}^k$, with:

$$\begin{aligned} a^k &= (\lambda^k)^{-\frac{1}{2}} \Big(\sum_{i=1}^{n} \alpha_i^k k(\boldsymbol{x}_i, \boldsymbol{x}) - \frac{1}{n} \sum_{i,j=1}^{n} \alpha_i^k k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i^k \frac{1}{n} \sum_{j=1}^{n} k(\boldsymbol{x}_j, \boldsymbol{x}) + \sum_{i=1}^{n} \alpha_i^k \frac{1}{n^2} \sum_{j,j'=1}^{n} k(\boldsymbol{x}_j, \boldsymbol{x}_{j'}) \Big) \\ &= (\lambda^k)^{-\frac{1}{2}} \sum_{i=1}^{n} \alpha_i^k \widetilde{k}(\boldsymbol{x}_i, \boldsymbol{x}). \end{aligned}$$

Finally, the Mahalanobis distance in equation (3.13) is computed in the feature space as follows:

$$\big\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\big\|_{\boldsymbol{\Sigma}}^2 = \sum_{k=1}^{n} (\lambda^k)^{-1} \Big(\sum_{i=1}^{n} \alpha_i^k \widetilde{k}(\boldsymbol{x}_i, \boldsymbol{x}) \Big)^2. \tag{3.17}$$

After computing the Mahalanobis distance between each training sample $\phi(\boldsymbol{x}_i)$ and the center $\boldsymbol{c}_n$, and fixing in advance the number of outliers n_{out} in the training dataset, we set a threshold R which represents the radius of the one-class hypersphere. The decision function of our classifier considers a new sample $\boldsymbol{x}$ as an outlier if its Mahalanobis distance to the center in the feature space is greater than this threshold, namely

$$\left\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\right\|_{\boldsymbol{\Sigma}} > R;$$

Otherwise, the sample is considered as a normal one. We propose in the following a one-class approach that takes the advantages of the KPCA by projecting the samples onto some relevant subspace. To this end, we explore a "truncated" Mahalanobis distance.

3.3.2 Advantages of KPCA and kernel whitening

As detailed in the previous section, the Mahalanobis distance investigates all the eigenvectors of the covariance matrix. We propose to "truncate" it, by selecting a set of eigenvectors. This corresponds to projecting the samples onto the subspace spanned by these eigenvectors of the covariance matrix $\boldsymbol{\Sigma}$. We accentuate the fact that the choice of the number of eigenvectors has an impact on the decision function of the classifier. On one hand, a small number of eigenvectors cannot give sufficient information on the data, and this leads to a loose description boundary that underfits the data. On the other hand, a large number of eigenvectors leads to inaccurate results and to a description boundary that overfits the data. Instead of using all the eigenvectors $\boldsymbol{v}^k$ for the projection operation, we make use of the advantages in the KPCA approach, where only the eigenvectors associated to the largest eigenvalues are taken into consideration. The remaining ones are considered to be associated to noise. Therefore, the Mahalanobis distance is approximated by the truncated Mahalanobis distance in the feature space.

The performance of the one-class classification algorithms depends on the heterogenous scaling of the data in the feature space. Therefore, we also adopt the kernel whitening normalization of the eigenvectors as proposed in [Tax and Juszczak, 2002], where the variance of the mapped data is constant in all directions. This normalization rescales the training data to have unit variance in each feature direction. In fact, the variance of the samples in the feature space along an eigenvector $\boldsymbol{v}^k$ is given by

$$\frac{1}{n}(\boldsymbol{\alpha}^k)^T \widetilde{\boldsymbol{K}}\widetilde{\boldsymbol{K}}\boldsymbol{\alpha}^k.$$

In order to obtain a constant variance for all feature directions, and using equation (3.15), we obtain:

$$(n\lambda^k)^2\|\boldsymbol{\alpha}^k\|^2 = 1 \quad \Longrightarrow \quad \|\boldsymbol{\alpha}^k\| = \frac{1}{n\lambda^k} \quad \text{for all} \quad k = 1, 2, \dots n.$$

3.4 Experimental Results

In this Section, we detail the experimental results of one-class classification algorithms on simulated datasets as well as on real datasets. We investigate two RBF kernels, namely the Gaussian kernel and the exponential kernel that follows the form of a Laplace distribution, as follows:

$$\text{Gaussian kernel}: \quad k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_p^2}{2\sigma^2}\right),$$

$$\text{Laplacian kernel}: \quad k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_p}{\frac{\sigma}{\sqrt{2}}}\right),$$

where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are two input samples, $\|\cdot\|_p$ represents the ℓ_p-norm in the d-dimensional input space, and σ denotes the bandwidth of the kernel. In the first place, we study the influence of the metric p in kernels on the decision functions of the one-class classifiers. We detail afterwards the results of the simple heuristic for choosing the bandwidth parameters σ. Finally, we present a comparison between the proposed truncated Mahalanobis-based one-class approach and other well-known algorithms, on simulated and on real datasets.

3.4.1 Norm variation

In order to study the influence of the kernel metric on the decision functions of the classifiers, we compare two well-known one-class classification methods, namely SVDD and KPCA. The kernel functions used in these methods are the Gaussian and the exponential kernels. We vary the value of the kernels metric p between $\frac{3}{4}$ and ∞. We apply these one-class algorithms on the gas pipeline real dataset from the Mississippi State University SCADA Laboratory and on the water treatment plant dataset from the UCI Machine Learning Repository.

Let $x(t)$ be the pressure in the pipeline at instant t. The definition of the input vectors should be made to draw attention to the fact that the pressure measurements of two consecutive instants in the normal functioning modes of the system must be close to each other. Furthermore, a significant difference in the pressure between two consecutive

instants may be a strong sign of a cyberattack. For these reasons, the time series is folded into 2-dimensional input vectors composed of the pressure at instant t and the difference in the pressure between instants t and $t-1$, namely

$$\boldsymbol{x}_t = [x(t) \quad x(t) - x(t-1)].$$

The data are divided into a training set and a test set. The training phase of each classification algorithm is made on a training set of 2000 samples, and includes only 131 samples related to the transitional states that should be considered as outliers. The test phase is conducted on five different test sets containing the cyberattacks detailed in Chapter 1 as follows:

- Slow response injection attack: 335 normal samples and 209 outliers.
- Fast response injection attack: 153 normal samples and 276 outliers.
- Burst response injection attack: 143 normal samples and 233 outliers.
- Single response injection attack: 125 normal samples and 128 outliers.
- Wave response injection attack: 418 normal samples and 114 outliers.

The outliers in the test sets represent the attacks that have to be detected by the one-class classification algorithms.

The results on the real gas pipeline dataset for the exponential and the Gaussian kernels are shown in Figures 3.4 and 3.5 respectively. The bandwidth parameter for each norm is computed as detailed in the previous chapter, namely

$$\sigma = \frac{d_{max}}{\sqrt{2M}},$$

where d_{max} refers to the maximal distance between any two samples in the input space, and M represents the estimated number of support vectors among the training dataset.

As illustrated in the figures, the decision boundary in each case encloses the samples accepted as normal ones, while the samples considered as outliers are outside the boundary. The use of the ℓ_2-norm in the exponential kernel with the SVDD approach (Figure 3.4) gives a good description of the training dataset, while the ℓ_∞-norm overfits the data. The ℓ_1-norm and the $\ell_{\frac{3}{4}}$-norm lead to a tighter boundary than with the ℓ_2-norm. We have similar results when using the exponential kernel with the KPCA approach. When the Gaussian kernel is used in KPCA (Figure 3.5), the ℓ_2-norm and the ℓ_1-norm lead to almost the same good result, the $\ell_{\frac{3}{4}}$-norm overfits the data, and the decision boundaries

for the remaining norms underfit the training dataset. We also have the same results when using the Gaussian kernel with the SVDD approach. We note that for the values of p greater than $p = 2$, e.g., $p = 3, 4, 7, ...$, the results of the SVDD and the KPCA approaches are worse than with the ℓ_1-norm and with the ℓ_2-norm, and the decision boundaries of the classifiers lead to inaccurate results. The different behavior of the two kernels for the infinite norm is illustrated in Figure 3.6. The first contour level of the Gaussian kernel corresponds to the first four contour levels of the exponential kernel. Therefore, as the value of p increases, the same contour levels become more "expanded" with the Gaussian kernel and tighter with the exponential kernel. This is the reason why the infinite norm overfits with the exponential kernel and underfits with the Gaussian kernel.

The error probabilities of the different types of cyberattacks for the Gaussian kernel using the ℓ_2-norm and the ℓ_1-norm are detailed in Tables 3.1 and 3.2. The ℓ_1-norm outperforms the ℓ_2-norm in several cases especially when it comes to decreasing the error of the second type (the outliers and the transitional states accepted as normal data). In particular, the wave response injection and the slow response injection data contain small simultaneous variation of the features, and this explains why the ℓ_1-norm outperforms the other norms. On the other hand, the burst response injection data contains sudden variation of both features, which explains the better results of the ℓ_2-norm. The best results are achieved with the slow and the single attacks having error detection probabilities around 99.25%, then the burst attack with 88%. The results for the wave attack (error detection probability around 65%) and burst attack (error detection probability around 70%) are not acceptable when dealing with security applications. The corresponding responses were injected in the normal operational modes of the studied system in order to imitate its behavior, which makes the detection of these attacks very difficult. Moreover, since these injections have already bypassed the traditional security systems (IDS and firewalls), the detection of the malicious attacks by operators comes mostly far too late after some severe consequences on the industry. For instance, the gas pipeline is dealing with high pressure while it appears to the deceived operator that it is working on low values; if the operator acts on the false information and increases the input gas pressure, he/she risks in putting the system in a highly dangerous state. This is where machine learning techniques, specifically one-class classification algorithms, play a crucial role to learn the industrial systems in order to detect all kinds of intrusions and avoid physical, financial and human lives losses. An example of the detection of outliers for the different types of attacks with the SVDD approach is illustrated in Figure 3.7.

After being successfully tested on the 2-dimensional gas pipeline dataset, the one-class classification algorithms are now tested on a very complex dataset, namely the water treatment plant dataset from the UCI Machine Learning Repository. This dataset

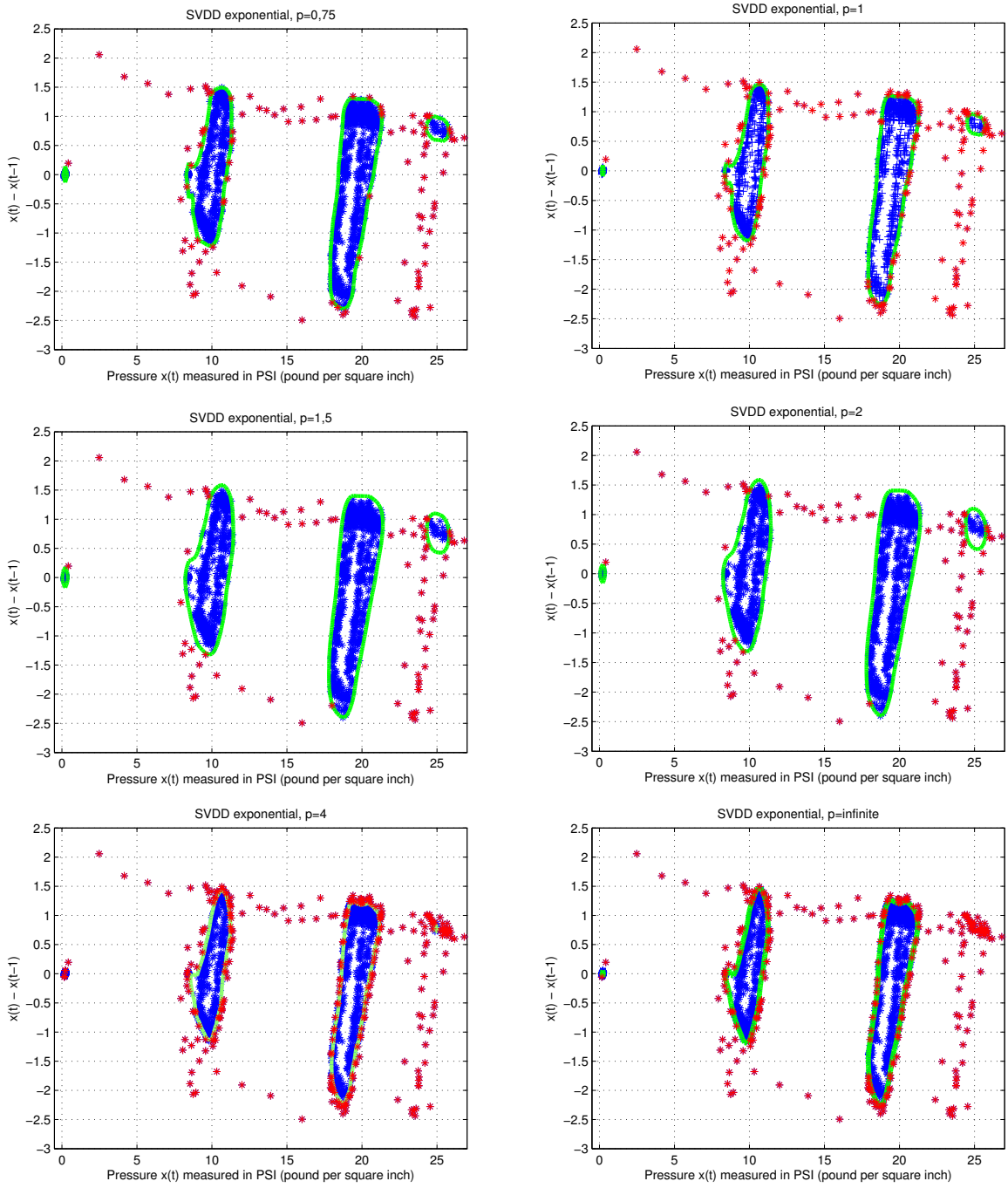

Figure 3.4: The exponential kernel is applied on the gas pipeline real dataset with the SVDD approach using several norms. The description boundaries are given by the green lines, the outliers correspond to the red samples while the normal samples are in blue. The best description boundaries are obtained with the ℓ_2-norm (middle right) and the ℓ_1-norm (top right). The ℓ_1-norm leads to a description boundary that is tighter than the one obtained with the ℓ_2-norm.

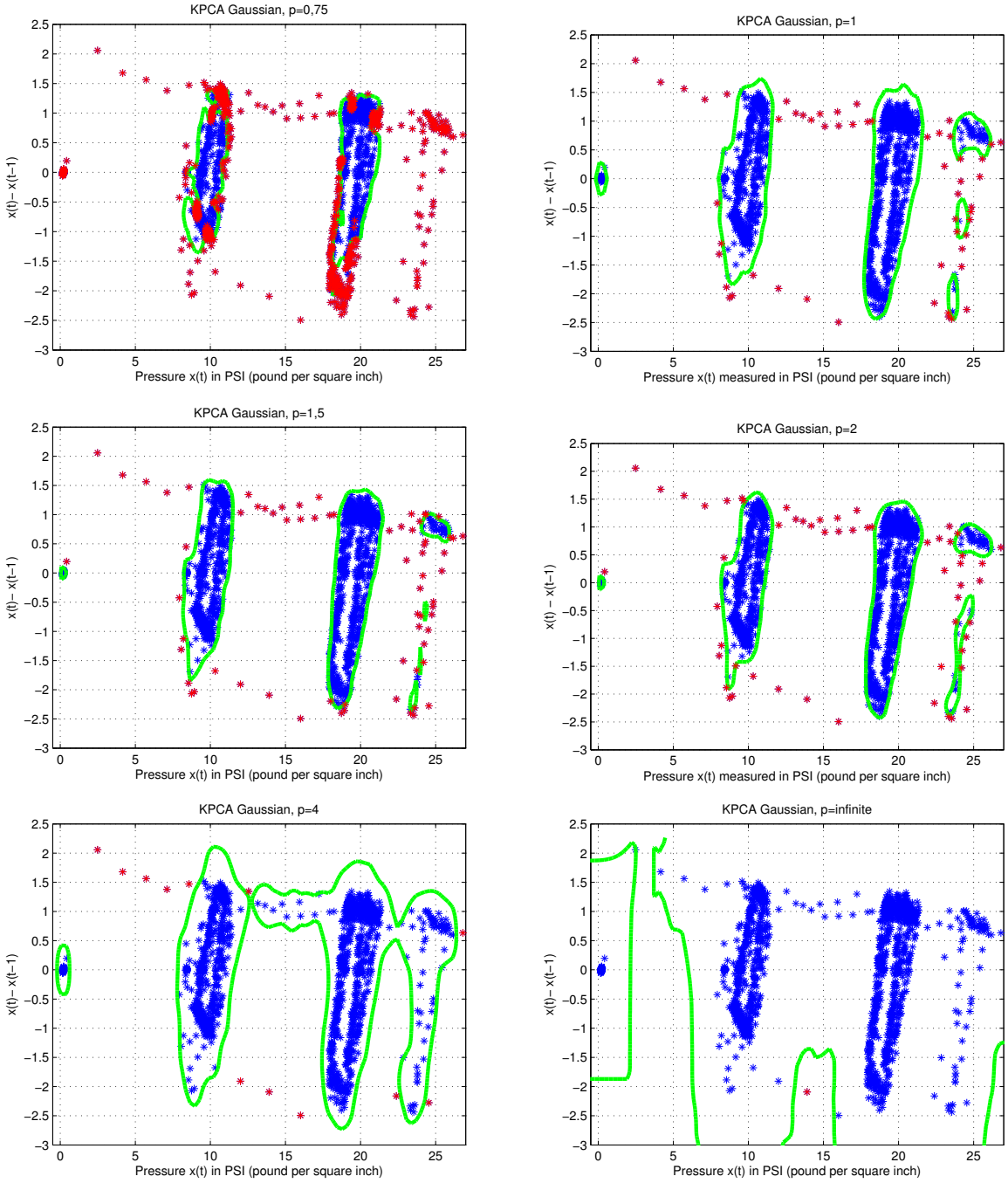

Figure 3.5: The Gaussian kernel is applied on the gas pipeline real dataset with the KPCA approach using several norms. The description boundaries are given by the green lines, the outliers correspond to the red samples while the normal samples are in blue. The ℓ_2-norm (middle right) and the ℓ_1-norm (top right) lead to almost identical results, while the other norms have loose descriptions or overfit the data.

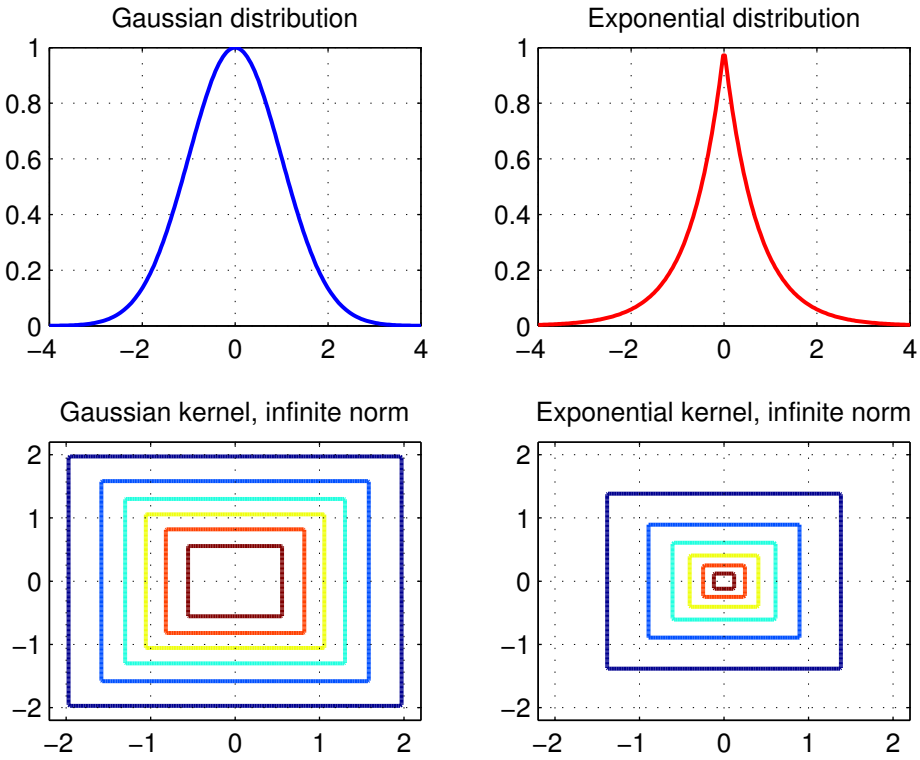

Figure 3.6: The behavior of the exponential and the Gaussian kernel with the infinite norm. As the value of p increases, the same contour levels (having the same contour colors) become more "expanded" with the Gaussian kernel (bottom left) and "tighter" with the exponential kernel (bottom right), which explains the overfitting of the infinite norm with the exponential kernel and the underfitting with the Gaussian kernel.

comes from the daily measures of sensors in a urban waste water treatment plant, where each sample contains 38 attributes related to the measurements of several important components in the water like input zinc, input PH, etc. The training dataset contains 513 samples related to four different normal situations while the test set encloses measurements of abnormal situations like after storms or when solids overload. The regularization parameter in the SVDD approach is fixed at 0.1, and the number of the most relevant eigenvectors in the KPCA approach is equal to 40. The results on the water treatment dataset are shown in Table 3.3. Two main observations can be drawn from these results: The KPCA approach outperforms the SVDD for all the studied cases and the ℓ_2-norm gives better results than the other norms. In fact, the attributes of this dataset contains important variation of multiple features in consecutive samples, which explains the better result of the ℓ_2-norm. Furthermore, since each sample is a vector of 38 dimensions, the projection of the data onto the subspace with maximum variance has allowed to the KPCA to get a better description of the training dataset than SVDD. Therefore, the best result is achieved when combining the ℓ_2-norm with the KPCA approach, with an error detection rate equals to 92.1%.

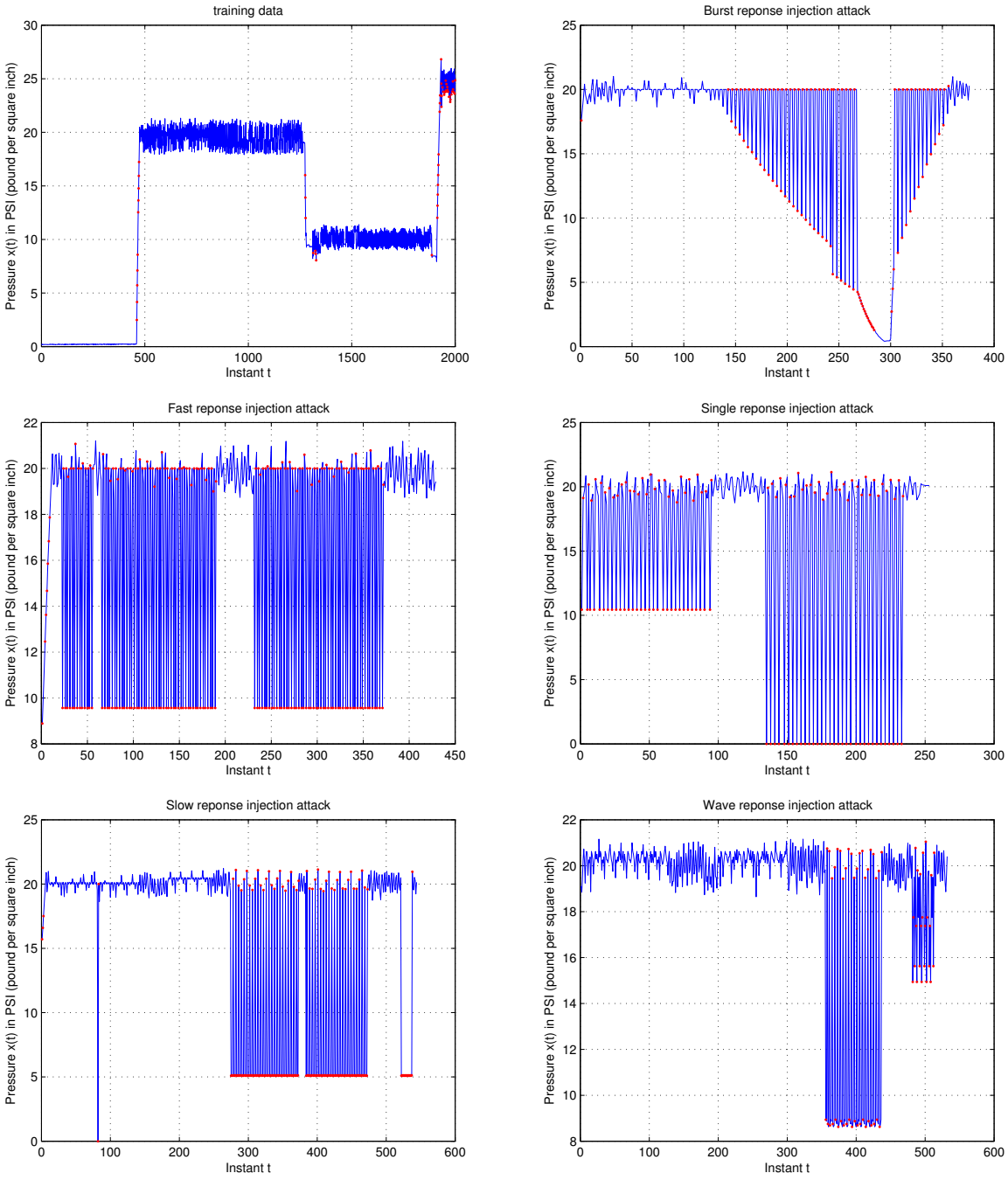

Figure 3.7: Detection of outliers for several types of attacks with the SVDD approach using the Gaussian kernel with ℓ_1-norm. The blue samples refer to the samples accepted as normal ones while the red samples are considered as outliers.

Table 3.1: The confusion matrix of several types of attacks with the SVDD approach using the Gaussian kernel, on the gas pipeline real dataset.

		Gaussian l_2-norm		Gaussian l_1-norm	
		Normal	Outlier	Normal	Outlier
Slow injection	Normal	99.7	0.3	99.7	0.3
	Outlier	0.9	99.1	0.5	99.5
Fast injection	Normal	99.4	0.6	99.4	0.6
	Outlier	11.6	88.4	11.6	88.4
Burst injection	Normal	99.3	0.7	99.3	0.7
	Outlier	33.9	66.1	34.3	65.7
Single injection	Normal	99.2	0.8	99.2	0.8
	Outlier	0.8	99.2	0.8	99.2
Wave injection	Normal	99.8	0.2	99.3	0.7
	Outlier	35.1	64.9	34.2	65.8

Table 3.2: The confusion matrix of several types of attacks with the KPCA approach using the Gaussian kernel, on the gas pipeline real dataset.

		Gaussian l_2-norm		Gaussian l_1-norm	
		Normal	Outlier	Normal	Outlier
Slow injection	Normal	99.4	0.6	99.7	0.3
	Outlier	0.9	99.1	0.5	99.5
Fast injection	Normal	98.3	1.7	99.4	0.6
	Outlier	11.6	88.4	11.6	88.4
Burst injection	Normal	99.3	0.7	99.3	0.7
	Outlier	27.9	72.1	31.3	68.7
Single injection	Normal	98.4	1.6	99.2	0.8
	Outlier	0.8	99.2	0.8	99.2
Wave injection	Normal	98.8	1.2	98.1	1.9
	Outlier	35.1	64.9	34.2	65.8

Table 3.3: Detection rates on the water treatment dataset using the Gaussian kernel with several norms.

approach	$p = 1$	$p = 1.5$	$p = 2$	$p = 4$	$p = \infty$
SVDD	50.3	71.4	78.6	59.1	44.6
KPCA	64.3	78.6	92.1	64.7	49.7

3.4.2 Bandwidth parameter

In order to demonstrate that the proposed heuristic for computing the bandwidth parameter σ leads to a very good result without any time computational cost, we compare it to three common methods existing in the literature:

- The standard 5-fold cross-validation divides the training dataset into 5 equally sized folds. Subsequently, k iterations of training and validation are performed, such that within each iteration a different fold is held-out for validation and the remaining $k-1$ are used for learning. The candidate values of σ follow a geometric progression with factor 2, namely $[0.5, 1, 2, \ldots, 1024]$.
- The large range grid proposed in [Soares et al., 2004] selects 11 values of σ following a geometric progression with factor 4, namely $[0.25, 1, 4, 16, 64, 256, 1000, 4000, 16000, 64000, 256000]$. The 7th element was set by the authors to 1000 rather than to 1024.
- The restricted range grid proposed in [Cherkassky and Ma, 2004] selects a 5-range grid that depends on the input range of the training dataset. The input samples are prescaled to [0 1] range. The bandwidth parameter takes one of the following values: $[0.1, 0.2, 0.3, 0.4, 0.5]$.

We investigate the error detection probabilities and the false alarm probabilities of the cyberattacks detailed previously in terms of the bandwidth parameter σ, using the Gaussian kernel with the ℓ_2-norm. The results are illustrated in Figure 3.8. The interval of good values for the bandwidth parameter is between 0.9 and 1.5, which represents the best compromise between a high detection rate and a low false alarm rate. The proposed heuristic leads to $\sigma = 0.94$ that lays in this interval. These good results achieved with the proposed heuristic confirm its relevance.

Furthermore, the estimated time of each approach for computing the bandwidth parameter is given in Table 3.4. These results are obtained on the gas pipeline dataset using the Gaussian kernel with the ℓ_2-norm. The proposed heuristic is clearly hundreds of times faster than the other methods. In fact, the computation of σ is related to the number of input samples of the training dataset, the distribution of this training dataset and to the fraction of support vectors, but it is independent of the one-class algorithm used in the experiments. For the other approaches, SVDD requires more time than KPCA to compute σ since a constrained quadratic problem has to be solved.

Table 3.4: Time computational cost of several approaches for computing the bandwidth parameter.

approach	5-fold CV	large range	limited range	proposed heuristic
SVDD	8h 5min	2h 58min	1h 26min	14.7 sec
KPCA	3h 47min	1h 32min	34.6 min	14.7 sec

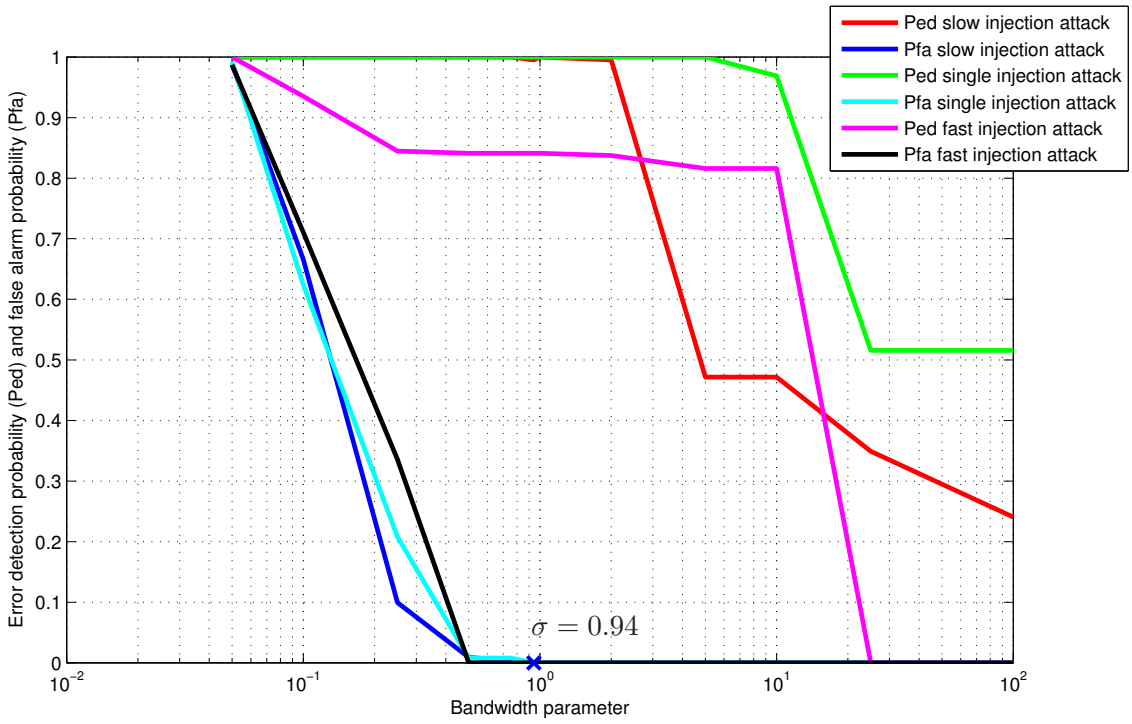

Figure 3.8: The error detection (P_{ed}) and the false alarm (P_{fa}) probabilities for three types of cyberattacks as a function of the bandwidth parameter σ. The proposed heuristic leads to $\sigma = 0.94$, which represents the best compromise between a high detection rate and a low false alarm rate.

3.4.3 One-class approaches

In this section, we provide a comparison in the performance between the proposed truncated Mahalanobis-based one-class classification algorithm and other well-known one-class approaches, namely SVDD, KPCA, simple one-class, slab SVM and robust SVM. The Gaussian kernel is used in the simulations, with its standard expression given by $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma^2})$. Since the behavior of the one-class SVM becomes identical to the SVDD with the unit-norm Gaussian kernel, we do not provide the results for the one-class SVM.

The one-class classification algorithms are applied in the first place on two simulated datasets, namely the sinusoidal and the square noise datasets [Hoffmann, 2007]. The main objective is to define a good description boundary that follows the distribution of the training dataset, in a way to enclose the normal samples while avoiding the extremes cases, namely overfitting and underfitting the data. We note that the sinusoidal dataset contains 95 samples, and the square noise has 450 samples including the noisy samples that have to be detected as outliers. We compared the results of the proposed algorithm with the aforementioned one-class classification approaches. The regularization parameters of all the one-class approaches, which represent an upper bound on the fraction

of outliers among the training samples, are set to 10% of the training dataset. In addition, the number of the most relevant eigenvectors taken into account for the projection operation in the KPCA and the proposed truncated Mahalanobis-based one-class is set to 40, which is the same value taken in [Hoffmann, 2007]. The decision boundaries are shown in Figures 3.9 and 3.10. The best results on the sinusoidal and the square noise datasets are achieved with the proposed truncated Mahalanobis-based one-class algorithm, which leads to the tightest decision boundaries that follows the distribution of the training samples in the best possible way. The KPCA approach leads to a good result, while the SVDD, simple one-class, slab SVM and robust SVM lead to loose decision boundaries that do not describe the exact distribution of the training samples. The good result of the proposed algorithm is due to the strong properties of the truncated Mahalanobis distance estimated via the projection onto the subspace spanned by the eigenvectors associated to the largest eigenvalues of the covariance matrix. This projection makes the proposed algorithm less sensitive to the choice of σ.

The one-class classification algorithms are now applied on two real datasets from the Mississipi State University SCADA Laboratory, namely the gas pipeline and the storage tank [Morris et al., 2011a], and on a third real dataset from the University of California Irvine (UCI) Machine Learning Repository [Bache and Lichman, 2013], namely the water treatment plant dataset. As detailed in Chapter 1, the gas pipeline is used to move natural gas or other petroleum products to the market, and the storage tank testbed is similar to the oil storage tanks found in the petrochemical industry. Each input sample has 27 attributes for the gas pipeline and 24 attributes for the storage tank, and these attributes represent heterogenous variables, such as gas pressure, water level, pump state, etc. In addition, 28 types of attacks are injected into the network traffic of the system in order to hide its real functioning state and to disrupt the communication. These attacks are arranged into 7 groups, as given in Table 1.1. When it comes to the water treatment plant dataset, it comes from the daily measures of sensors in an urban waste water treatment plant, where each input sample contains 38 attributes related to the measurements of several important components in the water like input zinc, input PH, input suspended solids, etc. The training set contains samples related to four different normal situations while the test set encloses measurements of abnormal situations like after storms or when solids overload.

The first criterion for one-class classification algorithms is the capacity to detect the outliers when testing samples not existing in the training dataset. These one-class algorithms are tested on nearly 100 000 samples related to the aforementioned attacks, and the detection rates on the gas pipeline, the storage tank and the water treatment plant testbeds are given in Tables 3.5, 3.6 and 3.7. The worst detection rates are achieved with the simple one-class approach, which can be explained by its high sensitivity to the

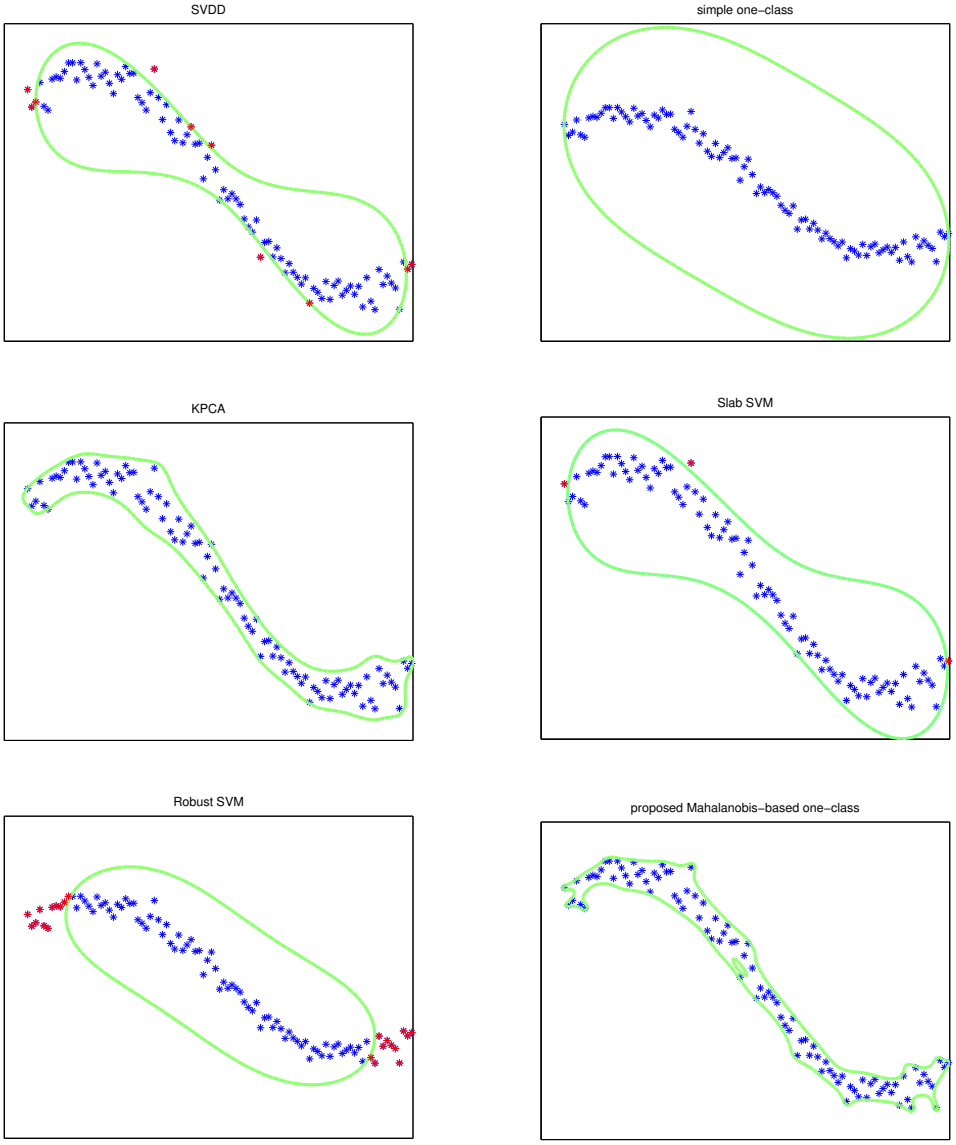

Figure 3.9: The decision boundaries on the sinusoidal dataset for the studied one-class classification algorithms. The description boundaries are given by the green lines, the red samples are the ones considered as outliers while the normal samples are in blue. The best decision boundary is obtained with the proposed truncated Mahalanobis-based one-class algorithm, and it is tighter than the one with KPCA, while the other approaches lead to loose decision boundaries.

presence of outliers among the training dataset. The SVDD, KPCA, simple one-class, slab SVM and robust SVM have good detection rates for some types of cyberattacks. The proposed truncated Mahalanobis-based approach outperforms all the other one-class approaches, and gives the best detection rates for all the types of attacks on the real datasets, expect for the MSCI and MFCI attacks on the gas pipeline testbed. When it comes to the false alarm rates, all the one-class approaches have a false alarm rate around 1%.

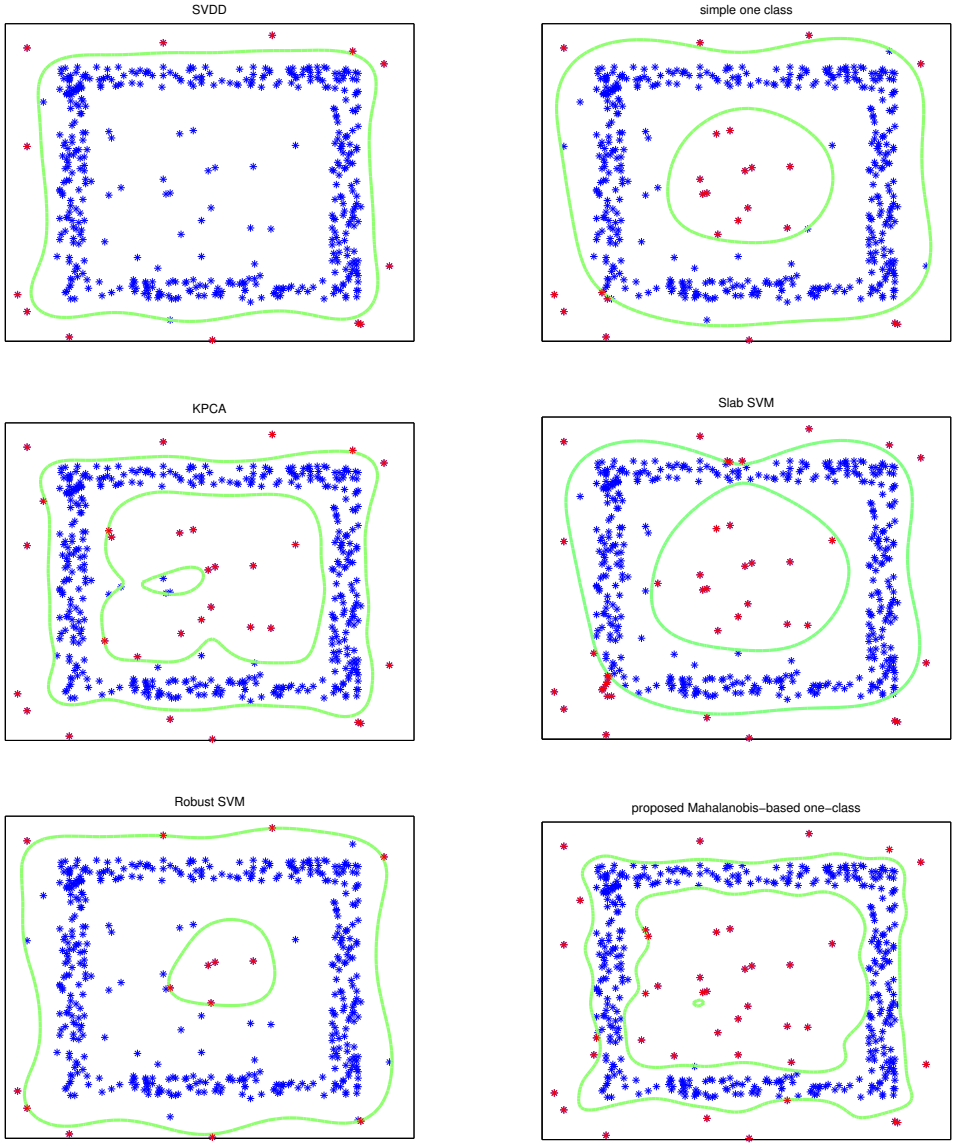

Figure 3.10: The decision boundaries on the square-noise dataset for the studied one-class classification algorithms. The description boundaries are given by the green lines, the outliers correspond to the red samples while the normal samples are in blue. The proposed truncated Mahalanobis-based one-class algorithm leads to the best description boundary that follows the distribution of the training samples.

The second criterion for one-class approaches is the time consumption of the algorithms. Table 3.8 shows the estimated time for training, and Table 3.9 outlines the estimated time to test each new sample of the real datasets. These results show that the simple one-class approach is the fastest algorithm on the gas and storage testbeds, but with the poorest results. The proposed truncated Mahalanobis-based approach is slightly slower than the simple one-class, but it is twice faster than the KPCA, up to 10 times faster than the SVDD and 50 times faster than the slab SVM. Furthermore, our approach is the fastest regarding the time needed to test a new sample except for the storage

testbed, and it is up to 4.5 times faster than SVDD. These results are very important since the proposed truncated Mahalanobis-based approach leads to the best detection rates, and it is up to 50 times faster than the other one-class algorithms.

Table 3.5: Detection rates for the gas pipeline testbed.

						Proposed approach
	SVDD	KPCA	Slab SVM	Robust SVM	simple one-class	Mahalanobis-based one-class
NMRI	98.1	98.7	98.4	92.9	91.7	**99.6**
CMRI	99.5	**99.8**	99.5	98.5	95.4	**99.8**
MSCI	**89.1**	86.2	86.2	54.9	22.6	83.1
MPCI	98.2	98.6	96.9	98.1	94.1	**99.1**
MFCI	**89.9**	89.3	89.4	64.7	31.6	85.1
DOS	96.1	96.8	96.3	95.5	68.5	**97.7**
RA	**99.8**	**99.8**	**99.8**	99.7	98.1	**99.8**

Table 3.6: Detection rates for the storage testbed.

						Proposed approach
	SVDD	KPCA	Slab SVM	Robust SVM	simple one-class	Mahalanobis-based one-class
NMRI	95.1	97.1	92.2	94.1	88.2	**98.8**
CMRI	61.2	75.3	63.5	59.7	46.2	**82.4**
MSCI	97.3	98.1	96.9	98.1	96.3	**98.7**
MPCI	98.6	99.5	99.1	99.2	97.6	**99.7**
MFCI	97.9	**99.9**	98.7	98.1	40.6	**99.9**
DOS	71.7	79.9	73.4	59.8	55.3	**83.3**
RA	97.8	99.5	98.1	98.7	95.9	**99.7**

Table 3.7: Detection rates for the UCI water treatment testbed.

						Proposed approach
Approach	SVDD	KPCA	Slab SVM	Robust SVM	simple one-class	Mahalanobis-based one-class
P_{ed}	78.6	92.1	81.6	74.7	64.8	**95.2**

Table 3.8: Estimated time (in seconds) for training each approach.

						Proposed approach
	SVDD	KPCA	Slab SVM	Robust SVM	simple one-class	Mahalanobis-based one-class
gas	70.23	18.31	302.74	61.32	**9.23**	10.08
storage	123.72	20.14	557.23	102.28	**10.41**	11.89
UCI	12.91	4.58	78.95	10.78	1.79	**1.57**

Table 3.9: Estimated time (in seconds) to test a new sample for each approach.

						Proposed approach
	SVDD	KPCA	Slab SVM	Robust SVM	simple one-class	Mahalanobis-based one-class
gas	0.039	0.019	0.035	0.037	0.011	**0.010**
storage	0.043	0.032	0.038	0.041	**0.015**	0.019
UCI	0.031	0.017	0.028	0.029	0.073	**0.071**

3.5 Conclusion

In this chapter, we investigated the increasing role of one-class classification techniques in several applications, specifically when the only available data designate the normal functioning modes of the studied physical process, and we showed the importance of these techniques in detecting the intrusions and the malicious cyberattacks in critical infrastructures. We proposed a simple and fast one-class classification approach in which the one-class classifier is defined by the hypersphere enclosing the training samples in the feature space. The center of this hypersphere is estimated without solving any quadratic optimization problem. We proposed to use the truncated Mahalanobis distance in the feature space as a novelty measure, by projecting the samples onto the subspace spanned by the eigenvectors associated to the largest eigenvalues of the covariance matrix. We applied the proposed method on simulated datasets as well as on real datasets, and we compared this approach to well-known one-class approaches, namely SVDD, KPCA, simple one-class, slab SVM and robust SVM. The results showed that the proposed approach had the best description boundaries, the best detection rates and the fastest algorithm on simulated and real datasets. We also studied the influence of the metric in RBF kernels, and we tested the proposed fast heuristic for choosing the bandwidth parameter of these kernels, which led to a relevant value of this parameter having the best detection rates and the lowest false alarm rates. In the next Chapter, we investigate sparse and online approaches for one-class classification.

Chapter 4

Sparse One-class Classification

Contents

4.1 Introduction 84

4.2 Sparse Truncated Mahalanobis-based One-class 86

4.2.1 The choice of support vectors 86

4.2.2 Sparse formulation 87

4.3 Sparse One-class Framework by Shrinkage 89

4.3.1 Least Angle Regression 90

4.3.2 Least Absolute Shrinkage and Selection Operator 92

4.3.3 Elastic Net 94

4.3.4 Extension to the truncated Mahalanobis distance 95

4.4 Theoretical Results 95

4.4.1 Projection error 95

4.4.2 Error of the first kind 97

4.5 Online One-class Classification 100

4.5.1 The proposed online truncated Mahalanobis-based one-class . . 101

4.6 Experimental Results 105

4.6.1 Simulated datasets 105

4.6.2 Real datasets 106

4.6.3 Online results 111

4.7 Conclusion 114

As detailed in the previous Chapter, one-class classification methods have gained a lot of interest in a large number of applications where the only available data designate a unique class, as in industrial applications. The one-class algorithms learn the normal behavior of the systems, in a way to detect any suspicious sample that does not belong

to the same distribution of the training dataset. This chapter focuses on sparse one-class formulations, in which only a small fraction of the training samples contributes effectively to the decision function of the classifier. We propose two frameworks for sparse one-class classification. The first one is a sparse formulation of the truncated Mahalanobis-based one-class approach detailed in Chapter 3, while the second framework is based on well-known shrinkage methods [Hastie et al., 2001], namely Least Angle Regression [Efron et al., 2004], Least Absolute Shrinkage and Selection Operator [Tibshirani, 1996; Osborne et al., 1999], and Elastic Net [Zou and Hastie, 2005; Zhou, 2013]. We also propose an online sparse one-class classification approach in which the classifier is improved sequentially at each instant, where each new sample is taken into account in order to update the decision rule of the classifier.

The remainder of this chapter is organized as follows. In Section 4.1, we outline the motivations behind the increasing need of low computational approaches, and we give a brief introduction to sparse approximation. We describe the first proposed sparse one-class framework in Section 4.2. The second framework based on shrinkage methods is detailed in Section 4.3. In Section 4.4, we provide theoretical results related to the projection error and the error of the first kind. The proposed online one-class classification framework is detailed in Section 4.5. The results on simulated and real datasets are given in Section 4.6, and the conclusion in Section 4.7.

4.1 Introduction

The existing one-class classification methods are, most of the time, very expensive in terms of computational cost in order to infer the decision rules of the classifiers, as we have detailed in Chapter 3. Some of these one-class approaches need to solve a constrained quadratic programming problem, others require the optimization of a second order cone programming problem, while others incorporate an eigen decomposition problem. The time computational cost of these algorithms grows with the number of samples in the training datasets. The need for reducing the computational complexity has increased in the past years, with the growing interest in the study of sparse representations [Tropp and Wright, 2010]. The main objective of the sparse approximation theory or sparse representation modeling of the data is to describe and approximate a target signal using linear combinations of few other elementary signals drawn from a fixed collection, known as the dictionary [Elad, 2010]. In general, the choice of a relevant dictionary is done either by building a sparsifying dictionary based on a mathematical model of the data, or by learning a dictionary on a training dataset [Rubinstein et al., 2010].

Some of the one-class classification methods detailed in the previous chapter result in a sparse solution. In the SVDD approach, the center of the hypersphere enclosing the samples is a linear combination of the so-called support vectors of the description. These support vectors, representing a small fraction of the training dataset, are collected in the so-called dictionary. In the one-class SVM, the hyperplane separating the samples from the origin with maximum margin depends also on support vectors. Moreover, the hyperplanes in the Slab SVM and Robust SVM are defined by linear combinations of support vectors. Although these approaches provide sparse solutions, they are expensive in terms of computational cost since they require to solve constrained quadratic programming problems.

Since the existing one-class classification methods are, either sparse but expensive in terms of computational complexity, or lack the sparsity property, we propose in this chapter two frameworks for sparse one-class classification. The first framework is a sparse formulation of the proposed truncated Mahalanobis-based one-class approach detailed in Chapter 3. When it comes to the second framework, we revisit well-known shrinkage methods, namely Least Angle Regression [Efron et al., 2004], Least Absolute Shrinkage and Selection Operator [Tibshirani, 1996; Osborne et al., 1999], and Elastic Net [Zou and Hastie, 2005; Zhou, 2013], by adapting their algorithms to become suitable for one-class classification problems. The main objective of the proposed frameworks is to derive a relevant sparse model using only a small fraction of the training dataset, while maintaining low-computation algorithms for large training datasets.

The expectation of the samples in the feature space has the following form:

$$\mathrm{E}[\phi(\boldsymbol{x})] = \int_{\mathcal{X}} \phi(\boldsymbol{x}) P(\boldsymbol{x}) d\boldsymbol{x},$$

having $P(\boldsymbol{x})$ the probability distribution of the training samples over $\mathcal{X}$. Since the distribution $P(\boldsymbol{x})$ is usually unknown, one can estimate this expectation by the empirical center of the training dataset in this space, with

$$\boldsymbol{c}_n = \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i).$$

We propose to approximate the empirical center using a small fraction of the training samples, where only these samples are taken into account in the decision functions of the classifiers. This sparse center is a linear combination of a set of samples in the feature space, namely:

$$\boldsymbol{c}_{\mathcal{A}} = \sum_{j=1}^{n} \beta_j \phi(\boldsymbol{x}_j),$$

where only a small fraction of the coefficients β_j in the center's expression are nonzero, namely:

$$\boldsymbol{c}_{\mathcal{A}} = \sum_{j \in \mathcal{A}} \beta_j \phi(\boldsymbol{x}_j),$$

where $\mathcal{A} \subset \{1, 2, \ldots, n\}$. The center $\boldsymbol{c}_{\mathcal{A}}$ has to be determined efficiently to represent the first order moment of the distribution of the training dataset. Therefore, we define $\boldsymbol{c}_{\mathcal{A}}$ by the approximation of the empirical center $\boldsymbol{c}_n$. In order to estimate the sparse center, we propose to minimize the error of approximating the empirical center $\boldsymbol{c}_n$ with $\boldsymbol{c}_{\mathcal{A}}$. Next, we detail the sparse one-class classification frameworks.

4.2 Sparse Truncated Mahalanobis-based One-class

In order to provide a sparse approach that reduces the computational complexity of the algorithm while maintaining a good description boundary around the data, we consider a sparse formulation of the truncated Mahalanobis-based one-class of Chapter 3. The one-class classifier is defined by the hypersphere enclosing the training samples in the feature space. In order to estimate a sparse center, we propose to approximate the estimated center $\boldsymbol{c}_n$ of the training samples using some support vectors, by analogy to the SVDD approach where the support vectors are the training samples that lie on and outside the hypersphere. Since the sparse center is a linear combination of these support vectors, only these samples are taken into account in the estimation of the truncated Mahalanobis distance in the RKHS. The selection of the support vectors is based on ad-hoc sparsification criteria, such as the distance criterion or the coherence criterion. We briefly present these two criteria before describing in detail the sparse formulation.

4.2.1 The choice of support vectors

Many sparsification criteria have been proposed in the literature for the selection of the support vectors, in order to define relevant data-driven dictionaries. Of particular interest are approaches with low computational complexity, often for online learning [Honeine, 2015]. In the following, we restrict the presentation to two well-known sparsification criteria, the coherence criterion initially studied in [Honeine et al., 2007; Richard et al., 2009] for nonlinear adaptive filtering and the SVDD-inspired distance criterion [Noumir et al., 2012b; Noumir, 2012].

The coherence describes the behavior of dictionaries in sparse approximation problems. The coherence is defined by the largest (in absolute value) cross-correlation in a set of

samples, hence its expression is given as follows:

$$\mu = \max_{i \neq j} |\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}|.$$

The coherence expression can be written in terms of a kernel function, which corresponds to the largest absolute value of the off-diagonal entries of the kernel matrix, namely[1]:

$$\mu = \max_{i \neq j} |k(\boldsymbol{x}_i, \boldsymbol{x}_j)|.$$

A set is considered incoherent when μ is small, and this parameter is equal to zero for orthonormal bases. It is therefore natural to consider the least coherent set as the relevant set of support vectors.

The distance criterion relies on the Euclidean distance in the feature space between the samples and the estimated center of the data, namely:

$$\left\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\right\|_{\mathcal{H}}^2 = k(\boldsymbol{x}, \boldsymbol{x}) - \frac{2}{n} \sum_{j=1}^{n} k(\boldsymbol{x}, \boldsymbol{x}_j) + \frac{1}{n^2} \sum_{i,j=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

The set $\mathcal{A}$ containing the indices of the furthest samples to the center is given as follows:

$$\mathcal{A} = \left\{ i, \quad \left\|\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n\right\|_{\mathcal{H}}^2 > R^2 \right\},$$

having R the radius/threshold, often obtained from a predefined number of outliers. This set defines the support vectors. We note that regardless of the sparsification criterion used for selecting the support vectors, the following algorithm remains unchanged.

4.2.2 Sparse formulation

In order to provide a sparse formulation for the truncated Mahalanobis-based one-class approach, approximating the center $\boldsymbol{c}_n$ with the sparse center $\boldsymbol{c}_{\mathcal{A}}$ is achieved by minimizing the Mahalanobis distance between $\boldsymbol{c}_n$ and $\boldsymbol{c}_{\mathcal{A}}$ as follows:

$$\arg\min_{\beta_i} \left\| \frac{1}{n} \sum_{l=1}^{n} \phi(\boldsymbol{x}_l) - \sum_{i \in \mathcal{A}} \beta_i \phi(\boldsymbol{x}_i) \right\|_{\boldsymbol{\Sigma}}^2.$$

[1] This definition of the coherence corresponds to unit-norm kernels, i.e., $k(\boldsymbol{x}, \boldsymbol{x}) = 1$ for every sample $\boldsymbol{x} \in \mathcal{X}$; Otherwise, we have to substitute $\frac{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{k(\boldsymbol{x}_i, \boldsymbol{x}_j) k(\boldsymbol{x}_i, \boldsymbol{x}_j)}}$ for $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in this definition.

The partial derivative of this cost function with respect to each β_i is computed and set to zero, and for each feature direction the following expression is nullified:

$$(\lambda^k)^{-\frac{1}{2}}\Big(\frac{1}{n}\sum_{l=1}^{n}\sum_{k\in\mathcal{A}}k(\boldsymbol{x}_l,\boldsymbol{x}_k)-\sum_{j,k\in\mathcal{A}}\beta_j k(\boldsymbol{x}_j,\boldsymbol{x}_k)\Big)\sum_{r=1}^{n}\alpha_r^k(\phi(\boldsymbol{x}_r)-\boldsymbol{c}_n).$$

This boils down to the following:

$$\frac{1}{n}\sum_{l=1}^{n}\sum_{k\in\mathcal{A}}k(\boldsymbol{x}_l,\boldsymbol{x}_k)=\sum_{j,k\in\mathcal{A}}\beta_j k(\boldsymbol{x}_j,\boldsymbol{x}_k).$$

The coefficients β_i are computed through the matrix notation:

$$\boldsymbol{\beta}=\boldsymbol{K}_{\mathcal{A}}^{-1}\boldsymbol{k}, \tag{4.1}$$

where the entries of the kernel matrix $\boldsymbol{K}_{\mathcal{A}}$ are $k(\boldsymbol{x}_j,\boldsymbol{x}_k)$ for $j,k\in\mathcal{A}$, and $\boldsymbol{k}$ is the column vector with entries

$$\frac{1}{n}\sum_{k\in\mathcal{A}}k(\boldsymbol{x}_l,\boldsymbol{x}_k),\quad \text{for}\quad l=1,\ldots,n.$$

In order to avoid non-invertible singular matrix $\boldsymbol{K}_{\mathcal{A}}$, one can include a regularization parameter ν, namely

$$\boldsymbol{\beta}=(\boldsymbol{K}_{\mathcal{A}}+\nu\boldsymbol{I})^{-1}\boldsymbol{k}.$$

The classifier fixes a threshold R based on a predefined number of outliers n_{out}. The decision function for any new sample $\boldsymbol{x}$ is to evaluate the squared Mahalanobis distance between $\phi(\boldsymbol{x})$ and the sparse center $\boldsymbol{c}_{\mathcal{A}}$ as follows:

$$\begin{aligned}\big\|\phi(\boldsymbol{x})-\boldsymbol{c}_{\mathcal{A}}\big\|_{\boldsymbol{\Sigma}}^2=&\sum_{k=1}^{m}\frac{1}{\lambda^k}\Big(\sum_{i=1}^{n}\alpha_i^k k(\boldsymbol{x}_i,\boldsymbol{x})-\sum_{i=1}^{n}\sum_{j\in\mathcal{A}}\alpha_i^k\beta_j k(\boldsymbol{x}_i,\boldsymbol{x}_j)\\&-\sum_{i=1}^{n}\alpha_i^k\frac{1}{n}\sum_{j=1}^{n}k(\boldsymbol{x}_j,\boldsymbol{x})+\sum_{i=1}^{n}\alpha_i^k\frac{1}{n}\sum_{j=1}^{n}\sum_{l\in\mathcal{A}}\beta_l k(\boldsymbol{x}_j,\boldsymbol{x}_l)\Big)^2.\end{aligned}$$

If this squared distance is greater than the radius R^2, the sample is considered as an outlier; Otherwise, it is considered as a normal sample. Similarly to the full-model truncated Mahalanobis-based approach, the above Mahalanobis distance can be truncated by projecting the training samples onto the subspace spanned by the eigenvectors associated to the largest eigenvalues of the covariance matrix.

4.3 Sparse One-class Framework by Shrinkage

The previous approach focused on separating the sparsification criterion from the approximation of the hypersphere centre. The framework proposed in this section considers solving jointly the two problems, namely the selection of the support vectors and the approximation of the resulting center. The optimization problem within this framework takes the following form:

$$\arg\min_{\boldsymbol{\beta}} \Big\| \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) - \sum_{j=1}^{n} \beta_j \phi(\boldsymbol{x}_j) \Big\|_{\mathcal{H}}^2, \tag{4.2}$$

subject to some sparsity-inducing constraints. Such constraints include that the ℓ_0-norm of $\boldsymbol{\beta}$ shall not exceed some predefined threshold. For computational reasons, the ℓ_0-norm is often replaced by the ℓ_1-norm, i.e., $\sum |\beta|$, which is the closest convex norm to the ℓ_0-norm.

It is important to draw attention to the fact that the optimization problem (4.2) has a form similar to the one in shrinkage methods used for regression problems, which takes the form of

$$\arg\min_{\beta} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2,$$

subject to some sparsity-inducing constraints, such as $\sum |\beta|$ is upper bounded. These shrinkage methods have been usually used in regression problems for the selection of the most relevant features, where only the corresponding coefficients remain nonzero. We propose to revisit three well-known shrinkage methods, namely Least Angle Regression, Least Absolute Shrinkage and Selection Operator, and Elastic Net. We modify their algorithms in order to estimate the sparse center $\boldsymbol{c}_{\mathcal{A}}$, by adapting them to solve the optimization problem (4.2), which selects the most relevant samples among the training dataset. Once the most relevant samples are selected and the corresponding coefficients estimated, the decision function of the one-class classifier is the (squared) Euclidean distance in the feature space between any new sample $\boldsymbol{x}$ and the sparse center $\boldsymbol{c}_{\mathcal{A}}$, which is given by the following expression:

$$\|\phi(\boldsymbol{x}) - \boldsymbol{c}_{\mathcal{A}}\|_2^2 = k(\boldsymbol{x}, \boldsymbol{x}) - 2 \sum_{i=1}^{n} \beta_i k(\boldsymbol{x}_i, \boldsymbol{x}) + \sum_{i,j=1}^{n} \beta_i \beta_j k(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

Next, we detail the modified shrinkage methods, by revisiting the corresponding optimization problems and the resulting solutions.

4.3.1 Least Angle Regression

The first shrinkage method studied for this framework is the Least Angle Regression (LARS), which builds a model sequentially by augmenting the set of the most relevant samples, one sample at a time. The modification of the LARS algorithm for one-class classification allows to solve the following optimization problem:

$$\arg\min_{\boldsymbol{\beta}} \Big\| \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) - \sum_{j=1}^{n} \beta_j \phi(\boldsymbol{x}_j) \Big\|_{\mathcal{H}}^2, \tag{4.3}$$

subject to $\sum |\beta| < t$, for some parameter t. Let $\widehat{\boldsymbol{c}}_{\mathcal{A}_k}$ be the estimated sparse center at step k, $\mathcal{A}_k$ the subset of indices of the most relevant samples, and $(\boldsymbol{c}_n - \widehat{\boldsymbol{c}}_{\mathcal{A}_k})$ the current residual. LARS considers the sample having the largest absolute correlation with the current residual $(\boldsymbol{c}_n - \widehat{\boldsymbol{c}}_{\mathcal{A}_k})$, and projects the other samples on this first one. LARS repeats the selection process until a new sample has the same correlation level with the current residual, and continues in a direction that preserves equiangularity between the set of the most relevant samples, until a third one enters this set. LARS continues equiangularly between these three samples until a forth one enters this set, and so on. An example of the successive LARS estimates is illustrated in Figure 4.1, where the algorithm starts at $\widehat{\boldsymbol{c}}_{\mathcal{A}_0}$, and the equiangular vectors are updated in a way to preserve equal angles with the original axes.

The LARS algorithm begins at $\widehat{\boldsymbol{c}}_{\mathcal{A}} = \boldsymbol{0}$, and updates $\widehat{\boldsymbol{c}}_{\mathcal{A}}$ at each step. Let $\boldsymbol{X}$ be the matrix of the samples in the feature space, namely:

$$\boldsymbol{X} = \big(\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \ldots, \phi(\boldsymbol{x}_n)\big).$$

Let $\boldsymbol{X}_{\mathcal{A}}$ denote the matrix containing the retained most relevant samples based on the greatest absolute correlation criterion, and $\boldsymbol{K}_{\mathcal{A}}$ the $|\mathcal{A}| \times |\mathcal{A}|$ corresponding kernel matrix, where $|\mathcal{A}|$ denotes the cardinality of $\mathcal{A}$. The expression of the current estimate of the sparse center takes the form:

$$\widehat{\boldsymbol{c}}_{\mathcal{A}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}.$$

LARS considers the sample having the largest absolute correlation with the current residual, where the vector of current correlations is defined as follows:

$$\begin{aligned} \widehat{\boldsymbol{corr}} &= \boldsymbol{X}^T(\boldsymbol{c}_n - \widehat{\boldsymbol{c}}_{\mathcal{A}}) \\ &= \frac{1}{n} \sum_{i,j=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i,j=1}^{n} \widehat{\beta}_j k(\boldsymbol{x}_i, \boldsymbol{x}_j), \end{aligned}$$

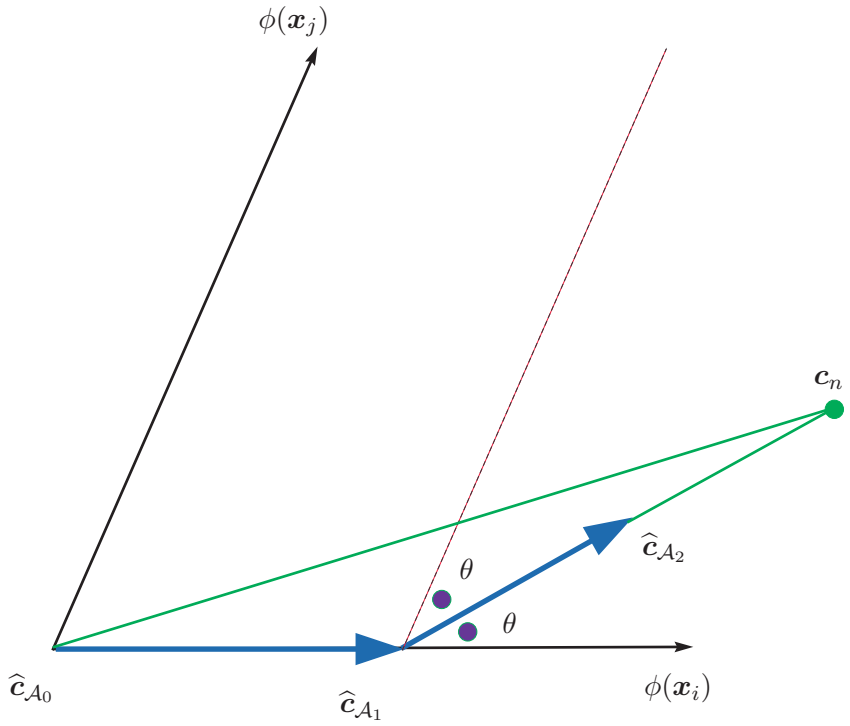

Figure 4.1: An illustration of the successive LARS estimates in a simple 2-dimensional space, where the algorithm starts at $\widehat{\boldsymbol{c}}_{\mathcal{A}_0} = \mathbf{0}$. In this example, the first residual $(\boldsymbol{c}_n - \widehat{\boldsymbol{c}}_{\mathcal{A}_0})$ makes a smaller angle with $\phi(\boldsymbol{x}_i)$ than with $\phi(\boldsymbol{x}_j)$, so we start moving in the direction of $\phi(\boldsymbol{x}_i)$ and $\widehat{\boldsymbol{c}}_{\mathcal{A}_1} = \beta_1 \phi(\boldsymbol{x}_i)$. At the next step, the current residual $(\boldsymbol{c}_n - \widehat{\boldsymbol{c}}_{\mathcal{A}_1})$ makes equal angles θ with $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$, so we have to move in a direction that preserves this equiangularity, as given with $\widehat{\boldsymbol{c}}_{\mathcal{A}_2}$.

having $\widehat{\beta}_j$ the current estimates of the center's coefficients. The next step is to project all the samples onto the subspace spanned by the samples of $\mathcal{A}$, in a way to preserve equal angles between these samples. The equiangular vector needed for the projection operation has the following form:

$$\boldsymbol{u}_{\mathcal{A}} = \boldsymbol{X}_{\mathcal{A}} \boldsymbol{w}_{\mathcal{A}},$$

where the weight vector $\boldsymbol{w}_{\mathcal{A}}$ making equal angles with the columns of $\boldsymbol{X}_{\mathcal{A}}$, the matrix $\boldsymbol{G}_{\mathcal{A}}$ related to the set $\mathcal{A}$, and the scalar $A_{\mathcal{A}}$ are given by:

$$\begin{aligned} \boldsymbol{w}_{\mathcal{A}} &= A_{\mathcal{A}} \boldsymbol{G}_{\mathcal{A}}^{-1} \mathbf{1}_{\mathcal{A}}, \\ \boldsymbol{G}_{\mathcal{A}} &= \boldsymbol{s}^T \boldsymbol{K}_{\mathcal{A}} \boldsymbol{s}, \\ A_{\mathcal{A}} &= (\mathbf{1}_{\mathcal{A}}^T \boldsymbol{G}_{\mathcal{A}}^{-1} \mathbf{1}_{\mathcal{A}})^{-\frac{1}{2}}, \end{aligned}$$

and $\boldsymbol{s}$ denotes the vector of the signs of the current correlations with entries:

$$s_j = \text{sign}\{\widehat{corr}_j\}, \quad \text{for} \quad j = 1, 2, \ldots, |\mathcal{A}|.$$

After computing $\boldsymbol{X}_{\mathcal{A}}$, $A_{\mathcal{A}}$, and $\boldsymbol{u}_{\mathcal{A}}$, the previous estimate $\widehat{\boldsymbol{c}}_{\mathcal{A}}$ is updated to:

$$\widehat{\boldsymbol{c}}_{\mathcal{A}+} = \widehat{\boldsymbol{c}}_{\mathcal{A}} + \widehat{\gamma}\boldsymbol{u}_{\mathcal{A}}$$

using the equiangular vector, where

$$\widehat{\gamma} = \min_{j=1,\ldots,|\mathcal{A}^c|} \left\{ \frac{\widehat{C} - \widehat{corr}_j}{A_{\mathcal{A}} - a_j}, \frac{\widehat{C} + \widehat{corr}_j}{A_{\mathcal{A}} + a_j} \right\},$$

having min the minimum over the positive components, $\mathcal{A}^{\mathcal{C}}$ the complementary set of $\mathcal{A}$, a_j an element of the inner product vector defined by

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{X}^T \boldsymbol{u}_{\mathcal{A}} \\ &= \boldsymbol{X}^T \boldsymbol{X}_{\mathcal{A}} \boldsymbol{w}_{\mathcal{A}} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{|\mathcal{A}|} k(\boldsymbol{x}_i, \boldsymbol{x}_j) \boldsymbol{w}_{\mathcal{A}}, \end{aligned}$$

and $\widehat{C} = \max_j\{|\widehat{corr}_j|\}$. Finally, the coefficients $\boldsymbol{\beta}$ are updated as follows:

$$\boldsymbol{\beta}_{new} = \widehat{\boldsymbol{\beta}} + \widehat{\gamma}\boldsymbol{s}^T \boldsymbol{w}_{\mathcal{A}}. \tag{4.4}$$

This algorithm inherits the drawbacks of the conventional LARS algorithm. The main drawback is with highly correlated samples, which may limit its application to high dimensional data. Another drawback is its sensitivity to noise. The following algorithm aims at overcoming these drawbacks.

4.3.2 Least Absolute Shrinkage and Selection Operator

The second shrinkage algorithm modified for the estimation of the sparse center is the Least Absolute Shrinkage and Selection Operator (LASSO). The objective in the LASSO involves minimizing the residual sum of squares, the same entity as in ordinary least squares (OLS) regression and LARS, subject to a bound on the sum of the absolute value of the coefficients. In other words, LASSO minimizes the residual sum of squares under a constraint on the ℓ_1-norm of the coefficient vector. It is easy to see that the ℓ_1-norm constraint induces sparsity in the solution. The LASSO shrinks the estimated coefficients towards the origin and sets some of them to zero, in a way to retain the most relevant samples and to discard the other ones. The main advantage of LASSO is with large volume datasets, where the coefficients of irrelevant samples are shrunk to zero.

The LASSO solves the following optimization problem:

$$\arg\min_{\boldsymbol{\beta}} \Big\| \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) - \sum_{j=1}^{n} \beta_j \phi(\boldsymbol{x}_j) \Big\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \tag{4.5}$$

for a given tuning parameter $\lambda > 0$. This parameter controls the sparsity level of the solution. The solution path of the LASSO, namely the solutions for all the values of λ, can be generated by some modifications of the LARS algorithm detailed previously. Indeed, the sign of any nonzero coefficient β_j must agree with the sign s_j of the corresponding current correlation $\widehat{corr}_j$, namely

$$\operatorname{sign}(\beta_j) = \operatorname{sign}(\widehat{corr}_j) = s_j,$$

for any $j \in \mathcal{A}$ [Efron et al., 2004]. Unlike in LARS, the coefficients in LASSO do not change signs during the update step since they are piecewise linear. Let $\widehat{\boldsymbol{d}}$ be the vector defined as follows:

$$\widehat{\boldsymbol{d}} = \begin{cases} s_j w_{\mathcal{A}j} & \text{for any} \quad j \in \mathcal{A}; \\ 0 & \text{otherwise.} \end{cases}$$

To update the coefficients as in equation (4.4), we have:

$$\beta_j(\gamma) = \widehat{\beta}_j + \gamma d_j \quad \text{for} \quad j \in \mathcal{A}.$$

Therefore, $\beta_j(\gamma)$ changes sign at

$$\gamma_j = -\frac{\beta_j}{d_j},$$

having the first such change occurring at

$$\widetilde{\gamma} = \min_{\gamma_j > 0} \{\gamma_j\}.$$

The sign restriction is violated when $\widetilde{\gamma} < \widehat{\gamma}$, and $\beta_j(\gamma)$ cannot be a LASSO solution; $\beta_j(\gamma)$ has changed sign while $c_j(\gamma)$ has not. The index j of the corresponding sample $\boldsymbol{x}_j$ is removed from the set of the most relevant samples, namely

$$\mathcal{A} = \mathcal{A} \setminus \{j\},$$

and the algorithm moves to the next equiangular direction. Therefore, this modification allows the set of the most relevant samples to increase or decrease one at a time until the LARS algorithm leads to all LASSO solutions.

This modified version of the LASSO allows to outperform the LARS algorithm by adding/removing one sample at a time. On the other hand, this algorithm inherits

the drawbacks of the conventional LASSO. The main drawback remains with high correlated variables, where LASSO tends to arbitrarily select only one variable from the group and ignores the others, thus it cannot do group selection.

4.3.3 Elastic Net

The Elastic Net is a LARS-derived regularization and variable selection method that overcomes the limitations of LARS and LASSO methods, specifically when it comes to high correlated variables. The Elastic Net optimization problem combines ℓ_1 and ℓ_2 penalties of the LASSO and ridge regression methods, thus Elastic Net produces a sparse model with both continuous shrinkage and variable selection. In addition, unlike LARS and LASSO, Elastic Net has a grouping effect where strongly correlated samples are in or out of the model together. The Elastic Net has the advantage of including automatically all the highly correlated variables in the group, and it was compared to a stretchable fishing net that retains "all the big fish" [Zou and Hastie, 2005]. In addition, the entire Elastic Net solution paths can be directly computed from the LARS algorithm.

The so-called naïve Elastic Net optimization problem is defined as follows:

$$\arg\min_{\boldsymbol{\beta}} \Big\| \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) - \sum_{j=1}^{n} \beta_j \phi(\boldsymbol{x}_j) \Big\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2,$$

for some given tuning parameters $\lambda_1, \lambda_2 > 0$, and it becomes a pure LASSO optimization when $\lambda_2 = 0$, and a ridge regression when $\lambda_1 = 0$. This optimization problem incurs a double amount of shrinkage from ridge and LASSO procedures, which introduces unnecessary extra bias compared with pure LASSO or ridge regression. In order to improve the prediction performance, the coefficients of the naïve version of Elastic Net are rescaled to obtain the Elastic Net coefficients as follows:

$$\boldsymbol{\beta}_{(\text{Elastic Net})} = (1 + \lambda_2)\boldsymbol{\beta}_{(\text{naïve Elastic Net})}.$$

This rescaling of the coefficients will undo the double amount of shrinkage.

The naïve Elastic Net problem can be transformed into an equivalent LASSO problem as in the optimisation problem (4.5), by replacing the parameter λ with [Zou and Hastie, 2005]:

$$\frac{\lambda_1}{\sqrt{1 + \lambda_2}}.$$

As detailed in the previous section, a simple modification in the LARS algorithm leads to all the LASSO solution paths. Therefore, the proposed LARS algorithm leads to all the Elastic Net solution paths.

4.3.4 Extension to the truncated Mahalanobis distance

The main drawback of using the Euclidian distance is its sensitivity to the scale variation in each direction. To overcome this drawback, we propose to extend this current sparse framework in a way to replace the Euclidian distance in the decision function of the classifier with the truncated Mahalanobis distance. In fact, the Mahalanobis distance takes into account the distribution of the training samples. As detailed in the previous Section, the (squared) Mahalanobis distance between any sample $\phi(\boldsymbol{x})$ and the center $\boldsymbol{c}_{\mathcal{A}}$ in the feature space is given as follows:

$$\left\|\phi(\boldsymbol{x}) - \boldsymbol{c}_{\mathcal{A}}\right\|_{\boldsymbol{\Sigma}}^2 = \sum_{k=1}^{m} \frac{1}{\lambda^k} \Big(\sum_{i=1}^{n} \alpha_i^k k(\boldsymbol{x}_i, \boldsymbol{x}) - \sum_{i=1}^{n} \sum_{j \in \mathcal{A}} \alpha_i^k \beta_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i^k \frac{1}{n} \sum_{j=1}^{n} k(\boldsymbol{x}_j, \boldsymbol{x}) + \sum_{i=1}^{n} \alpha_i^k \frac{1}{n} \sum_{j=1}^{n} \sum_{l \in \mathcal{A}} \beta_l k(\boldsymbol{x}_j, \boldsymbol{x}_l) \Big)^2.$$

The Mahalanobis distance in the feature space is estimated via the projection of the training samples onto the subspace spanned by the eigenvectors associated to the largest eigenvalues of the covariance matrix. Therefore, the Mahalanobis distance is approximated by the truncated Mahalanobis distance.

4.4 Theoretical Results

In this Section, we provide theoretical results on the error of projecting the center of the data $\boldsymbol{c}_n$ and on the first kind error. Let $\mathcal{P}$ be the projection operator onto the subspace spanned by the eigenvectors associated to the largest eigenvalues of the covariance matrix $\boldsymbol{\Sigma}$. The Mahalanobis distance is approximated by the truncated Mahalanobis distance, by replacing $\boldsymbol{\Sigma}$ by the corresponding approximation $\widehat{\boldsymbol{\Sigma}}$ as follows:

$$\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\|_{\boldsymbol{\Sigma}} \approx \|\mathcal{P}\phi(\boldsymbol{x}) - \mathcal{P}\boldsymbol{c}_n\|_{\widehat{\boldsymbol{\Sigma}}}.$$

Next, we study the relevance of this approximation.

4.4.1 Projection error

Theorem 4.1. *Given a training dataset $\boldsymbol{x}_i$, $i = 1, \ldots, n$, with its covariance matrix $\boldsymbol{\Sigma}$ defined in the feature space as given in 3.14. The error of projecting the center of the data $\boldsymbol{c}_n$ onto the subspace spanned by the k eigenvectors associated to the largest*

eigenvalues of $\mathbf{\Sigma}$ *can be upper bounded by*

$$\frac{1}{n^2}\sum_{i=k+1}^{n}\lambda_i,$$

where $\lambda_{k+1},\ldots,\lambda_n$ *are the smallest eigenvalues of* $\mathbf{\Sigma}$*, i.e., the eigenvalues associated to the eigenvectors unused in the projection operation.*

Proof. Let $\mathbf{I}$ denote the identity operator. The quadratic error of approximating $\boldsymbol{c}_n$ by its projection onto the subspace defined by the projection operator $\mathcal{P}$ is upper-bounded as follows:

$$\begin{aligned}
\big\|(\mathbf{I}-\mathcal{P})\boldsymbol{c}_n\big\|_{\widehat{\mathbf{\Sigma}}}^2 &= \big\|\frac{1}{n}\sum_{i=1}^{n}(\mathbf{I}-\mathcal{P})\phi(\boldsymbol{x}_i)\big\|_{\widehat{\mathbf{\Sigma}}}^2 \\
&\leq \frac{1}{n^2}\sum_{i=1}^{n}\big\|(\mathbf{I}-\mathcal{P})\phi(\boldsymbol{x}_i)\big\|_{\widehat{\mathbf{\Sigma}}}^2 \\
&\leq \frac{1}{n^2}\sum_{i=1}^{n}\big\|\phi(\boldsymbol{x}_i)-\mathcal{P}\phi(\boldsymbol{x}_i)\big\|_{\widehat{\mathbf{\Sigma}}}^2 \\
&\leq \frac{1}{n^2}\Big(\sum_{i=1}^{n}\big\|\phi(\boldsymbol{x}_i)\big\|_{\widehat{\mathbf{\Sigma}}}^2-\sum_{i=1}^{n}\big\|\mathcal{P}\phi(\boldsymbol{x}_i)\big\|_{\widehat{\mathbf{\Sigma}}}^2\Big),
\end{aligned}$$

where the first inequality follows from the triangular inequality, and the third one comes from Pythagoras theorem given an orthogonal projection. According to [Shawe-Taylor and Cristianini, 2004, Chapter 6] we have:

$$\sum_{i=1}^{n}\big\|\phi(\boldsymbol{x}_i)\big\|_{\widehat{\mathbf{\Sigma}}}^2=\sum_{i=1}^{n}\lambda_i,$$

and

$$\begin{aligned}
\sum_{i=1}^{n}\big\|\mathcal{P}\phi(\boldsymbol{x}_i)\big\|_{\widehat{\mathbf{\Sigma}}}^2 &= \sum_{i=1}^{n}\sum_{j=1}^{k}\mathcal{P}_{\boldsymbol{v}^j}\big(\phi(\boldsymbol{x}_i)\big)^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{k}(v_i^j)^2\lambda_i \\
&= \sum_{i=1}^{n}\lambda_i\sum_{j=1}^{k}(v_i^j)^2,
\end{aligned}$$

where v_i^j is the i-th element of the eigenvector $\boldsymbol{v}^j$. Since the eigenvectors $\boldsymbol{v}^j$ are orthogonal, we must have the following relation:

$$\sum_{j=1}^{k}(v_i^j)^2 \leq 1,$$

for all i. Therefore, we obtain the following:

$$\sum_{i=1}^{n}\left\|\mathcal{P}\phi(\boldsymbol{x}_i)\right\|_{\widehat{\boldsymbol{\Sigma}}}^2 \leq \sum_{i=1}^{k}\lambda_i.$$

Consequently, the error of projecting the center $\boldsymbol{c}_n$ onto the subspace spanned by the k eigenvectors associated to the largest eigenvalues of the covariance matrix can be upper bounded by:

$$\begin{aligned}\left\|(\mathbf{I}-\mathcal{P})\boldsymbol{c}_n\right\|_{\widehat{\boldsymbol{\Sigma}}}^2 &\leq \frac{1}{n^2}\left(\sum_{i=1}^{n}\lambda_i - \sum_{i=1}^{k}\lambda_i\right)\\ &\leq \frac{1}{n^2}\sum_{i=k+1}^{n}\lambda_i.\end{aligned}$$

□

4.4.2 Error of the first kind

Let $\boldsymbol{c}_\infty$ denote the expectation of the data in the feature space, namely

$$\mathrm{E}[\phi(\boldsymbol{x})] = \int_{\mathcal{X}}\phi(\boldsymbol{x})P(\boldsymbol{x})d\boldsymbol{x}.$$

In the following, we consider that the samples of the training dataset are generated from the same distribution. The following theorems gives an upper bound on the probability of a new sample lying outside a hypersphere centered on the empirical mean.

Theorem 4.2. *Consider the hypersphere in the feature space centered on $\boldsymbol{c}_n$ whith the radius R_1 given by*

$$R_1 = \max_{i=1,\cdots,n}\|\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n\|.$$

By the symmetry of the i.i.d. assumption, the probability that a new sample $\boldsymbol{x}$ drawn from the same distribution as the training dataset, lies outside this hypersphere is upper-bounded by

$$\mathrm{P}\big(\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\| > R_1 + 2\epsilon_1\big) \leq \frac{1}{n+1},$$

having ϵ_1 the error of approximating $\boldsymbol{c}_\infty$ with $\boldsymbol{c}_n$.

The proof of this Theorem is given in [Shawe-Taylor and Cristianini, 2004]. Figure 4.2 illustrates an example of the hypersphere centered on $\boldsymbol{c}_n$ and enclosing all the training samples. The following theorem generalizes this result by considering outliers in the training dataset and a sparse representation of the center.

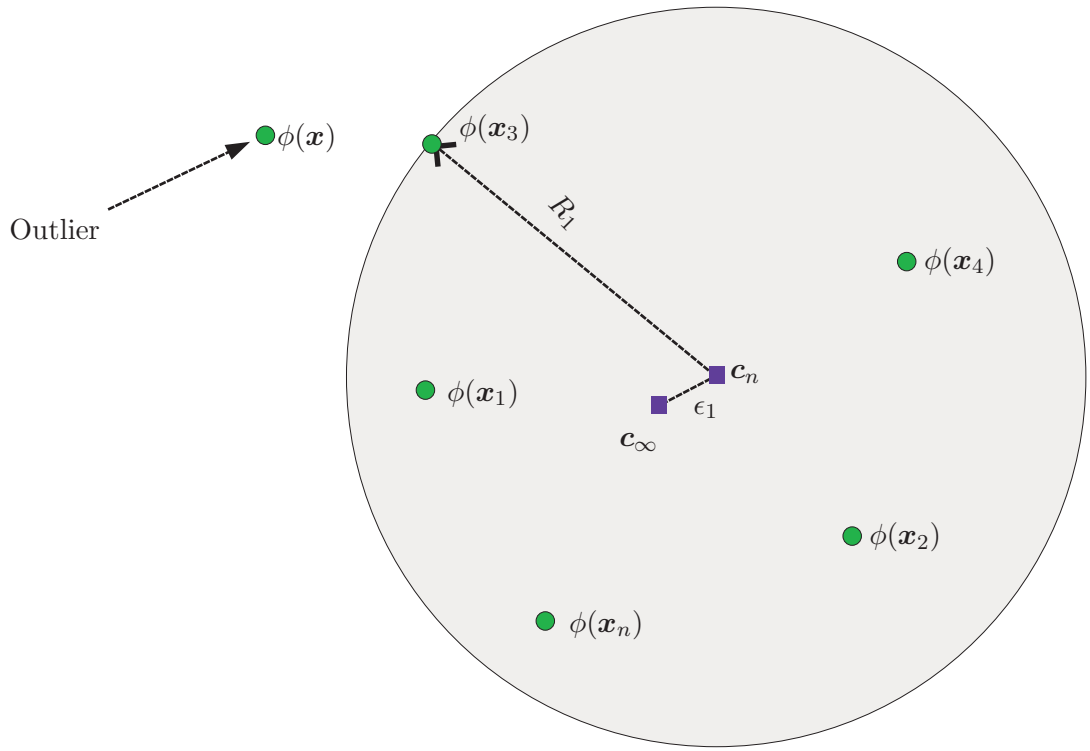

Figure 4.2: An example of the hypersphere centered on $\boldsymbol{c}_n$ and enclosing all the training samples, while the new sample $\boldsymbol{x}$ is an outlier since it lies outside the hypershpere.

Theorem 4.3. *Consider the hypersphere in the subspace spanned by the eigenvectors associated to the largest eigenvalues of the covariance matrix and centered on $\mathcal{P}\boldsymbol{c}_\mathcal{A}$ with radius $R_{n_{out}}$ excluding n_{out} outliers. The probability that a new sample $\boldsymbol{x}$ drawn from the same distribution as the training dataset, lies outside this hypersphere is upper-bounded as follows:*

$$\mathrm{P}\left(\|\mathcal{P}\phi(\boldsymbol{x}) - \mathcal{P}\boldsymbol{c}_\mathcal{A}\|_{\widehat{\boldsymbol{\Sigma}}} > R + 2(\epsilon_3 + \epsilon_4)\right) \leq \frac{n_{out}+1}{n+1},$$

having ϵ_3 the error between $\mathcal{P}\boldsymbol{c}_\infty$ and $\mathcal{P}\boldsymbol{c}_n$, and ϵ_4 the error between $\mathcal{P}\boldsymbol{c}_n$ and $\mathcal{P}\boldsymbol{c}_\mathcal{A}$, namely

$$\epsilon_3 = \|\mathcal{P}\boldsymbol{c}_n - \mathcal{P}\boldsymbol{c}_\infty\|_{\widehat{\boldsymbol{\Sigma}}},$$
$$\epsilon_4 = \|\mathcal{P}\boldsymbol{c}_n - \mathcal{P}\boldsymbol{c}_\mathcal{A}\|_{\widehat{\boldsymbol{\Sigma}}}.$$

Proof. Consider the hypersphere centered on the sparse center $\boldsymbol{c}_\mathcal{A}$, with its radius defined by considering n_{out} outliers from the training dataset. We apply the triangular inequality

twice on the distance between $\phi(\boldsymbol{x})$ and $\boldsymbol{c}_{\mathcal{A}}$, and we get the following relations:

$$\begin{aligned}\|\phi(\boldsymbol{x}) - \boldsymbol{c}_{\mathcal{A}}\|_{\mathcal{H}} &\leq \|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\|_{\mathcal{H}} + \|\boldsymbol{c}_n - \boldsymbol{c}_{\mathcal{A}}\|_{\mathcal{H}} \\ &\leq \|\phi(\boldsymbol{x}) - \boldsymbol{c}_\infty\|_{\mathcal{H}} + \|\boldsymbol{c}_\infty - \boldsymbol{c}_n\|_{\mathcal{H}} + \|\boldsymbol{c}_n - \boldsymbol{c}_{\mathcal{A}}\|_{\mathcal{H}} \\ &\leq \|\phi(\boldsymbol{x}) - \boldsymbol{c}_\infty\|_{\mathcal{H}} + \epsilon_1 + \epsilon_2,\end{aligned}$$

having ϵ_2 the error of approximating $\boldsymbol{c}_n$ with $\boldsymbol{c}_{\mathcal{A}}$. On the other hand, we have for any training sample $\boldsymbol{x}_i$:

$$\begin{aligned}\|\phi(\boldsymbol{x}_i) - \boldsymbol{c}_{\mathcal{A}}\|_{\mathcal{H}} &\geq \|\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n\|_{\mathcal{H}} - \|\boldsymbol{c}_n - \boldsymbol{c}_{\mathcal{A}}\|_{\mathcal{H}} \\ &\geq \|\phi(\boldsymbol{x}_i) - \boldsymbol{c}_\infty\|_{\mathcal{H}} - \|\boldsymbol{c}_\infty - \boldsymbol{c}_n\|_{\mathcal{H}} - \|\boldsymbol{c}_n - \boldsymbol{c}_{\mathcal{A}}\|_{\mathcal{H}} \\ &\geq \|\phi(\boldsymbol{x}_i) - \boldsymbol{c}_\infty\|_{\mathcal{H}} - \epsilon_1 - \epsilon_2.\end{aligned}$$

From these two inequalities, and by the symmetry of the i.i.d. assumption, the probability of a new sample $\boldsymbol{x}$ drawn from the same distribution as the training dataset lying outside this hypersphere is bounded by

$$\begin{aligned}\mathrm{P}\big(\|\phi(\boldsymbol{x}) - \boldsymbol{c}_{\mathcal{A}}\|_{\mathcal{H}} > R_{n_{out}} + 2(\epsilon_1 + \epsilon_2)\big) &\leq \mathrm{P}\big(\|\phi(\boldsymbol{x}) - \boldsymbol{c}_\infty\|_{\mathcal{H}} > R_{n_{out}}\big) \\ &\leq \frac{n_{out}+1}{n+1}.\end{aligned}$$

Now consider the hypersphere centered on the projected sparse center $\mathcal{P}\boldsymbol{c}_{\mathcal{A}}$ with the same radius. We apply the triangular inequality twice on the truncated Mahalanobis distance between the projected sample $\mathcal{P}\phi(\boldsymbol{x})$ and the projected sparse center $\mathcal{P}\boldsymbol{c}_{\mathcal{A}}$, and we get the following relations:

$$\begin{aligned}\|\mathcal{P}\phi(\boldsymbol{x}) - \mathcal{P}\boldsymbol{c}_{\mathcal{A}}\|_{\widehat{\boldsymbol{\Sigma}}} &\leq \|\mathcal{P}\phi(\boldsymbol{x}) - \mathcal{P}\boldsymbol{c}_n\|_{\widehat{\boldsymbol{\Sigma}}} + \|\mathcal{P}\boldsymbol{c}_n - \mathcal{P}\boldsymbol{c}_{\mathcal{A}}\|_{\widehat{\boldsymbol{\Sigma}}} \\ &\leq \|\mathcal{P}\phi(\boldsymbol{x}) - \mathcal{P}\boldsymbol{c}_\infty\|_{\widehat{\boldsymbol{\Sigma}}} + \|\mathcal{P}\boldsymbol{c}_\infty - \mathcal{P}\boldsymbol{c}_n\|_{\widehat{\boldsymbol{\Sigma}}} + \|\mathcal{P}\boldsymbol{c}_n - \mathcal{P}\boldsymbol{c}_{\mathcal{A}}\|_{\widehat{\boldsymbol{\Sigma}}} \\ &\leq \|\mathcal{P}\phi(\boldsymbol{x}) - \mathcal{P}\boldsymbol{c}_\infty\|_{\widehat{\boldsymbol{\Sigma}}} + \epsilon_3 + \epsilon_4.\end{aligned}$$

For any training sample $\boldsymbol{x}_i$ we have:

$$\begin{aligned}\|\mathcal{P}\phi(\boldsymbol{x}_i) - \mathcal{P}\boldsymbol{c}_{\mathcal{A}}\|_{\widehat{\boldsymbol{\Sigma}}} &\geq \|\mathcal{P}\phi(\boldsymbol{x}_i) - \mathcal{P}\boldsymbol{c}_n\|_{\widehat{\boldsymbol{\Sigma}}} - \|\mathcal{P}\boldsymbol{c}_n - \mathcal{P}\boldsymbol{c}_{\mathcal{A}}\|_{\widehat{\boldsymbol{\Sigma}}} \\ &\geq \|\mathcal{P}\phi(\boldsymbol{x}_i) - \mathcal{P}\boldsymbol{c}_\infty\|_{\widehat{\boldsymbol{\Sigma}}} - \|\mathcal{P}\boldsymbol{c}_\infty - \mathcal{P}\boldsymbol{c}_n\|_{\widehat{\boldsymbol{\Sigma}}} - \|\mathcal{P}\boldsymbol{c}_n - \mathcal{P}\boldsymbol{c}_{\mathcal{A}}\|_{\widehat{\boldsymbol{\Sigma}}} \\ &\geq \|\mathcal{P}\phi(\boldsymbol{x}_i) - \mathcal{P}\boldsymbol{c}_\infty\|_{\widehat{\boldsymbol{\Sigma}}} - \epsilon_3 - \epsilon_4.\end{aligned}$$

From these two inequalities, and by the symmetry of the i.i.d. assumption, the probability of a new sample $\boldsymbol{x}$ drawn from the same distribution as the training dataset lying

outside this hypersphere is bounded by

$$\begin{aligned} \mathrm{P}\big(\|\mathcal{P}\phi(\boldsymbol{x}) - \mathcal{P}\boldsymbol{c}_{\mathcal{A}}\|_{\widehat{\boldsymbol{\Sigma}}} > R_{n_{out}} + 2(\epsilon_3 + \epsilon_4)\big) &\leq \mathrm{P}\big(\|\mathcal{P}\phi(\boldsymbol{x}) - \mathcal{P}\boldsymbol{c}_{\infty}\|_{\widehat{\boldsymbol{\Sigma}}} > R_{n_{out}}\big) \\ &\leq \frac{n_{out}+1}{n+1}. \end{aligned}$$

□

The theoretical results on the error of the first kind give an upper bound on the probability of false alarm.

4.5 Online One-class Classification

The one-class classification frameworks proposed so far in this thesis are considered as offline learning or batch learning techniques, since all the training samples are available at once. The estimation of the decision rule of the classifier does not change after the initial training phase. Many real-life applications can be more naturally viewed as online rather than batch learning problems. Indeed, the data are often collected continuously in time, in a sequential fashion, and more importantly, the relations to be learned between the data may also evolve in time [Blum, 1998]. The key difference between online learning and batch learning techniques is that, in online learning, the decision rule of the classifier is updated after the arrival of every new sample, and we get more and more samples as time goes on. On the other hand, batch techniques are used when one has access to the entire training dataset at once [Kivinen et al., 2004]. Researchers have been facing many challenges to elaborate relevant online one-class classification algorithms, by minimizing the time consumption of the algorithm with the reduction of the complexity of the classifier, while improving the detection accuracy by minimizing the false alarm rates.

Several incremental and decremental SVM algorithms were proposed for online learning, where the classifier is updated by adding/removing samples based on the new incoming observations [Cauwenberghs and Poggio, 2001; Karasuyama and Takeuchi, 2010]. These approaches cannot be extended for one-class classification problems. Li and Long [2002] proposed an online maximum margin algorithm that updates the classifier by solving a quadratic programming problem, which increases the model complexity and the time computational costs. Gentile [2002] proposed a modified approach to reduce the computational costs, but often with poorer results. Desobry et al. [2005] proposed to train the classifier twice at each iteration, and the detection is performed by comparing the present sample set with the immediate past set. The repeated batch training leads to

high computational costs. Gomez-Verdejo et al. [2011] introduced an adaptive online one-class SVM that stores the new samples for many iterations before incorporating them into the training set, which is time consuming. Tax and Laskov [2003] proposed an online version of the standard SVDD, in which the classifier is constrained to have a limited number of samples in memory in order to be applicable to very large datasets. A new sample is added to the model at each instant, while an old sample is removed to maintain a fixed window size. Although this approach maintains a relatively constant estimated training time, it needs to solve a constrained quadratic programming problem at each iteration, thus it requires high computational cost. In order to overcome the quadratic programming problem and to reduce the computational cost of existing approaches, Zhang et al. [2009] used the linear optimization of the quarter-sphere SVM as proposed in [Laskov et al., 2006]. Similarly to the online SVDD, the new sample is added and the old one is removed to maintain a fixed window size, while the constrained quadratic programming problem is approximated and replaced by a linear optimization problem. This approach is faster than the ones in [Desobry et al., 2005; Tax and Laskov, 2003], but the repeated training results in a delay in the processing of new samples. Another attempt to overcome the quadratic programming problem is detailed in [Davy et al., 2006], where an iterative update of the coefficients is used at each time step. This online approach remains greedy in terms of computational cost. A fast online one-class approach was proposed in [Noumir et al., 2012a], where the coherence criterion is used to select the support vectors among the training set. This approach considers a least-squares optimization problem instead of the common constrained quadratic programming problem, where the model complexity is controlled by the coherence criterion as a sparsification rule. This criterion is coupled with a simple updating rule for online learning, which yields a low computational demanding algorithm.

4.5.1 The proposed online truncated Mahalanobis-based one-class

In order to overcome the drawbacks of the existing methods, we propose an online approach for one-class classification. This method is an extended version of the sparse one-class truncated Mahalanobis-based approach proposed in Section 4.2. For this purpose, we modify the decision function of the classifier to become suitable for online applications, by defining two concentric hyperspheres enclosing the support vectors of the description. The first main advantage of using two hyperspheres instead of a single one is the isolation of the outliers outside the decision boundary, without including these samples in the online update step of the classifier, which makes the proposed algorithm more robust to outliers. Another advantage of this approach is the small number of support vectors which reduces the computational costs of the algorithm. A representation of

the two concentric hyperspheres that define the proposed online one-class classification method is illustrated in Figure 4.3.

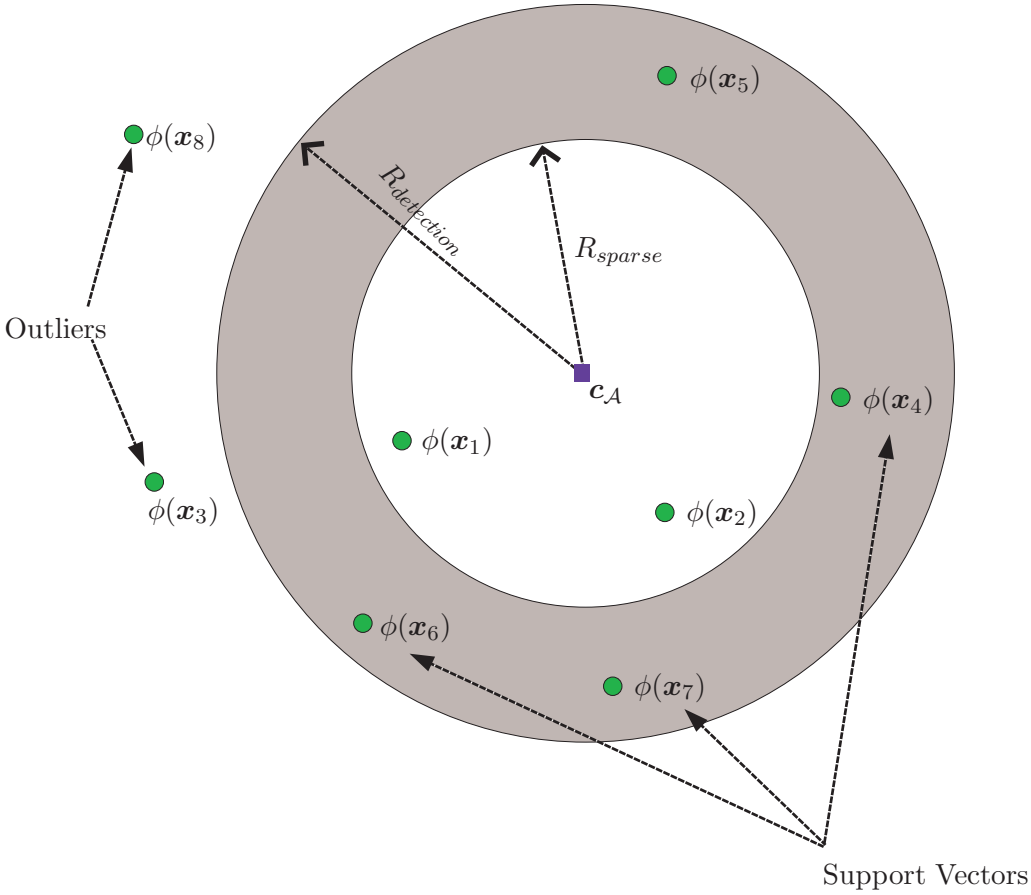

Figure 4.3: The proposed online one-class classification defined by two concentric hyperspheres in the feature space. The isolation of the outliers outside the decision boundary makes the proposed algorithm more robust to these samples, and the small number of support vectors reduces the computational costs of the algorithm.

As illustrated in Figure 4.3, the one-class problem is defined by two concentric hyperspheres, in between lie the support vectors. We fix two thresholds, namely $R_{detection}$ and R_{sparse}, in order to test new samples and detect the outliers in the online phase. The first threshold is fixed based on the predefined number of outliers, and R_{sparse} depends on the estimated number of support vectors. This new definition of the one-class problem reduces the computational costs of the proposed algorithm, and it allows to separate the outliers from the support vectors, which makes the algorithm more robust to outliers. This approach can be divided into two principal phases: an offline training phase and an online detecting/updating phase, where the offline phase allows for hot-start.

4.5.1.1 Offline training phase

In the offline phase, we learn the normal functioning modes of the studied system. Consider a training dataset $\boldsymbol{x}_i$, for $i = 1, 2, \ldots, n$, in an input space $\mathcal{X}$. Let $\boldsymbol{K}$ be the $n \times n$ kernel matrix with entries $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}$, for $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$, where $\phi(\boldsymbol{x})$ is the mapping function to the RKHS of some given kernel k. The mean of the mapped samples in the feature space, namely $\mathrm{E}[\phi(\boldsymbol{x})] = \int_{\mathcal{X}} \phi(\boldsymbol{x}) P(\boldsymbol{x}) d\boldsymbol{x}$, is estimated with the empirical mean, namely $\boldsymbol{c}_n = \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i)$. Since we are dealing with large volumes of data in the online mode and in order to minimize the computational complexity of the proposed algorithm, we approximate $\boldsymbol{c}_n$ with a sparse center $\boldsymbol{c}_{\mathcal{A}}$ using the support vectors of the description. The center $\boldsymbol{c}_{\mathcal{A}}$ depends only on the support vectors, and only these samples are taken into account in the decision function of the classifier. The set of support vectors $\mathcal{A}$ is given by:

$$\mathcal{A} = \left\{ i, R_{sparse} < \|\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n\|_{\boldsymbol{\Sigma}} \leq R_{detection} \right\}.$$

The sparse center is a linear combination of these samples as follows:

$$\boldsymbol{c}_{\mathcal{A}} = \sum_{i \in \mathcal{A}} \beta_i \phi(\boldsymbol{x}_i).$$

We minimize the error of approximating the center $\boldsymbol{c}_n$ with the sparse center $\boldsymbol{c}_{\mathcal{A}}$:

$$\arg\min_{\beta_i} \left\| \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) - \sum_{i \in \mathcal{A}} \beta_i \phi(\boldsymbol{x}_i) \right\|_{\boldsymbol{\Sigma}}^2,$$

and the coefficients β_i are computed as given in Section 4.2 as follows:

$$\boldsymbol{\beta} = \boldsymbol{K}_{\mathcal{A}}^{-1} \boldsymbol{k},$$

where the entries of the kernel matrix $\boldsymbol{K}_{\mathcal{A}}$ are $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $i, j \in \mathcal{A}$, and $\boldsymbol{k}$ is the column vector with entries $\frac{1}{n} \sum_{k \in \mathcal{A}} k(\boldsymbol{x}_i, \boldsymbol{x}_k)$, for $i = 1, \ldots, n$.

The squared Mahalanobis distance in the feature space between each sample $\phi(\boldsymbol{x})$ and the sparse center $\boldsymbol{c}_{\mathcal{A}}$ is computed as follows:

$$\begin{aligned} \left\| \phi(\boldsymbol{x}) - \boldsymbol{c}_{\mathcal{A}} \right\|_{\boldsymbol{\Sigma}}^2 = \sum_{k=1}^{m} \frac{1}{\lambda_k} \Big(& \sum_{i=1}^{n} \alpha_i^k k(\boldsymbol{x}_i, \boldsymbol{x}) - \sum_{i=1}^{n} \sum_{j \in \mathcal{A}} \alpha_i^k \beta_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ & - \sum_{i=1}^{n} \alpha_i^k \frac{1}{n} \sum_{j=1}^{n} k(\boldsymbol{x}_j, \boldsymbol{x}) + \sum_{i=1}^{n} \alpha_i^k \frac{1}{n} \sum_{j=1}^{n} \sum_{l \in \mathcal{A}} \beta_l k(\boldsymbol{x}_j, \boldsymbol{x}_l) \Big)^2. \end{aligned}$$

The novelty detection is defined by using this expression. Moreover, we propose also to approximate it by the truncated Mahalanobis distance, where only the eigenvectors associated to the largest eigenvalues of the covariance matrix are taken into consideration. See Section 3.3.2 for more details.

4.5.1.2 Online update phase

In the online phase, we have a new sample at each time step. The classifier tests each new sample $\boldsymbol{x}_t$, for any $t > n$, by computing its truncated Mahalanobis distance to $\boldsymbol{c}_{\mathcal{A}}$, namely $\|\phi(\boldsymbol{x}_t) - \boldsymbol{c}_{\mathcal{A}}\|_{\boldsymbol{\Sigma}}$. We can encounter three possible cases depending on this distance:

1. **First case:** $\|\phi(\boldsymbol{x}_t) - \boldsymbol{c}_{\mathcal{A}}\|_{\boldsymbol{\Sigma}} > R_{detection}$
   In this case, the new sample is considered as an outlier and an alarm is activated. The classifier must not be updated and this sample must not be included in the learning process; this prevents it from affecting the decision function of the classifier and leading to inaccurate results.

2. **Second case:** $R_{sparse} < \|\phi(\boldsymbol{x}_t) - \boldsymbol{c}_{\mathcal{A}}\|_{\boldsymbol{\Sigma}} \leq R_{detection}$
   In this case, the new sample $\boldsymbol{x}_t$ is considered as a support vector, and its index is included into the set of support vectors $\mathcal{A}$. The number of support vectors is incremented, and the new kernel matrix is updated from $\boldsymbol{K}_{\mathcal{A}}$ as follows:

$$\boldsymbol{K}_{\mathcal{A}new} = \begin{bmatrix} \boldsymbol{K}_{\mathcal{A}} & \boldsymbol{b} \\ \boldsymbol{b}^T & k(\boldsymbol{x}_t, \boldsymbol{x}_t) \end{bmatrix},$$

   where $\boldsymbol{b}$ is the column vector with entries $k(\boldsymbol{x}_i, \boldsymbol{x}_t)$ for all $i \in \mathcal{A}$. Also, $\boldsymbol{k}$ is updated to:

$$\boldsymbol{k}_{new} = \frac{1}{t} \begin{bmatrix} (t-1)\boldsymbol{k} + \boldsymbol{b} \\ k_t \end{bmatrix},$$

   having $k_t = \sum_{i=1}^{t} k(\boldsymbol{x}_i, \boldsymbol{x}_t)$. After updating $\boldsymbol{K}_{\mathcal{A}}$ and $\boldsymbol{k}$, the new coefficients $\boldsymbol{\beta}_t$ are given by:

$$\boldsymbol{\beta}_t = \boldsymbol{K}_{\mathcal{A}new}^{-1} \boldsymbol{k}_{new},$$

   where we apply the Woodbury matrix identity to obtain the inverse of the new Gram matrix from the inverse of the old one as follows:

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} & \boldsymbol{0} \\ \boldsymbol{0}^T & 0 \end{bmatrix} + \begin{bmatrix} -\boldsymbol{A}^{-1}\boldsymbol{B} \\ \mathbf{I} \end{bmatrix} (\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \begin{bmatrix} -\boldsymbol{C}\boldsymbol{A}^{-1} & \mathbf{I} \end{bmatrix},$$

and the new inverse of the new Gram matrix is given by:

$$\boldsymbol{K}_{\mathcal{A}new}^{-1} = \begin{bmatrix} \boldsymbol{K}_{\mathcal{A}}^{-1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \begin{bmatrix} -\boldsymbol{K}_{\mathcal{A}}^{-1}\boldsymbol{b} \\ 1 \end{bmatrix} (1 - \boldsymbol{b}^T \boldsymbol{K}_{\mathcal{A}}^{-1}\boldsymbol{b})^{-1} \begin{bmatrix} -\boldsymbol{b}^T \boldsymbol{K}_{\mathcal{A}}^{-1} & 1 \end{bmatrix},$$

having $\mathbf{0}$ a column vector of zeros, and $\mathbf{I}$ the identity matrix.

3. **Third case:** $\|\phi(\boldsymbol{x}_t) - \boldsymbol{c}_{\mathcal{A}}\|_{\boldsymbol{\Sigma}} \leq R_{sparse}$
   In this case, the new sample $\boldsymbol{x}_t$ is not a support vector, and its index is not included into the set $\mathcal{A}$. The number of support vectors remains unchanged, as well as the matrix $\boldsymbol{K}_{\mathcal{A}}$. Only the vector $\boldsymbol{k}$ is updated to

$$\boldsymbol{k}_{new} = \frac{1}{t}\big((t-1)\boldsymbol{k} + \boldsymbol{b}\big),$$

   where $\boldsymbol{b}$ is the column vector with entries $k(\boldsymbol{x}_i, \boldsymbol{x}_t)$ for all $i \in \mathcal{A}$. The new coefficients are given by:

$$\boldsymbol{\beta}_{new} = \frac{t-1}{t}\boldsymbol{\beta} + \frac{1}{t}\boldsymbol{K}_{\mathcal{A}}^{-1}\boldsymbol{b}.$$

4.6 Experimental Results

In this Section, we detail the results of the proposed one-class classification frameworks on two simulated datasets, and on the three real datasets described in Chapter 1. In the first framework, the coefficients β_j corresponding to the sparse samples are computed as given in equation (4.1), while the selection of these samples in the second framework is performed via the aforementioned modified shrinkage algorithms, namely LARS, LASSO and Elastic Net. In each case of these three subset selection approaches, the decision function of the classifier is defined using the Euclidean distance and the truncated Mahalanobis distance. The Gaussian kernel with the ℓ_2-norm is used, where the bandwidth parameter is computed with the heuristic proposed in Chapter 2. The results of the proposed online approach are discussed afterwards.

4.6.1 Simulated datasets

In order to visualize the impact of approximating the center with a sparse formulation, using a small fraction of the training dataset, we apply the proposed frameworks on two simulated 2-dimensional datasets, namely the sinusoidal and the square noise datasets. The sinusoidal dataset contains 95 samples without outliers, and the square noise has 450 samples including noisy samples that have to be detected as outliers. To provide

a comparative analysis, the sparse center of the proposed frameworks uses only 15% of the training samples as support vectors.

We compare the results of the proposed sparse one-class frameworks with three other approaches with sparse formulations, namely SVDD, slab SVM and robust SVM as shown in Figures 4.4 and 4.5. The fraction of the support vectors in these approaches is also fixed at 15% of the training samples. When it comes to the results on the sinusoidal dataset, the SVDD, slab SVM and robust SVM have loose boundaries. The use of the Euclidean distance in the decision function of the classifier in the proposed framework based on the shrinkage methods leads also to loose boundaries, which can be explained by the sensitivity of this distance to the scale in each feature direction. On the other hand, the use of the truncated Mahalanobis distance instead of the Euclidean distance gives better boundaries, and combining Elastic Net with the truncated Mahalanobis distance outperforms both LASSO and LARS, and it leads to a good result with a description boundary tighter than SVDD, slab SVM and robust SVM. The best result on the sinusoidal dataset is achieved with the first proposed framework, namely the sparse truncated Mahalanobis-based one-class, having the tightest decision boundary that outperforms all the other approaches. When it comes to the square noise dataset, the SVDD, slab SVM and robust SVM have loose boundaries that do not describe the distribution of the training samples, LARS and LASSO have good results with the truncated Mahalanobis distance, and the best results are achieved with the proposed sparse truncated Mahalanobis-based one-class framework, and when the truncated Mahalanobis distance is used with the modified Elastic Net algorithm of the second proposed framework. Therefore, the proposed frameworks have the best results and outperform the other one-class approaches.

4.6.2 Real datasets

The proposed sparse one-class classification frameworks are now tested on the three real datasets, namely the gas pipeline and the storage tank testbeds from the Mississipi State University SCADA Laboratory, and the water treatment plant dataset from the UCI Machine Learning Repository. See Chapter 1 for more details on the real datasets.

An example that highlights the differences in the solution paths of LARS, LASSO and Elastic Net algorithms is illustrated in Figure 4.6. This example on the gas pipeline real dataset shows the different behavior of each of these modified shrinkage algorithms. The results show that LARS solution paths are the most unstable, while Elastic Net has smoother solution paths that clearly show the "grouping effect" advantage of correlated variables over the LASSO.

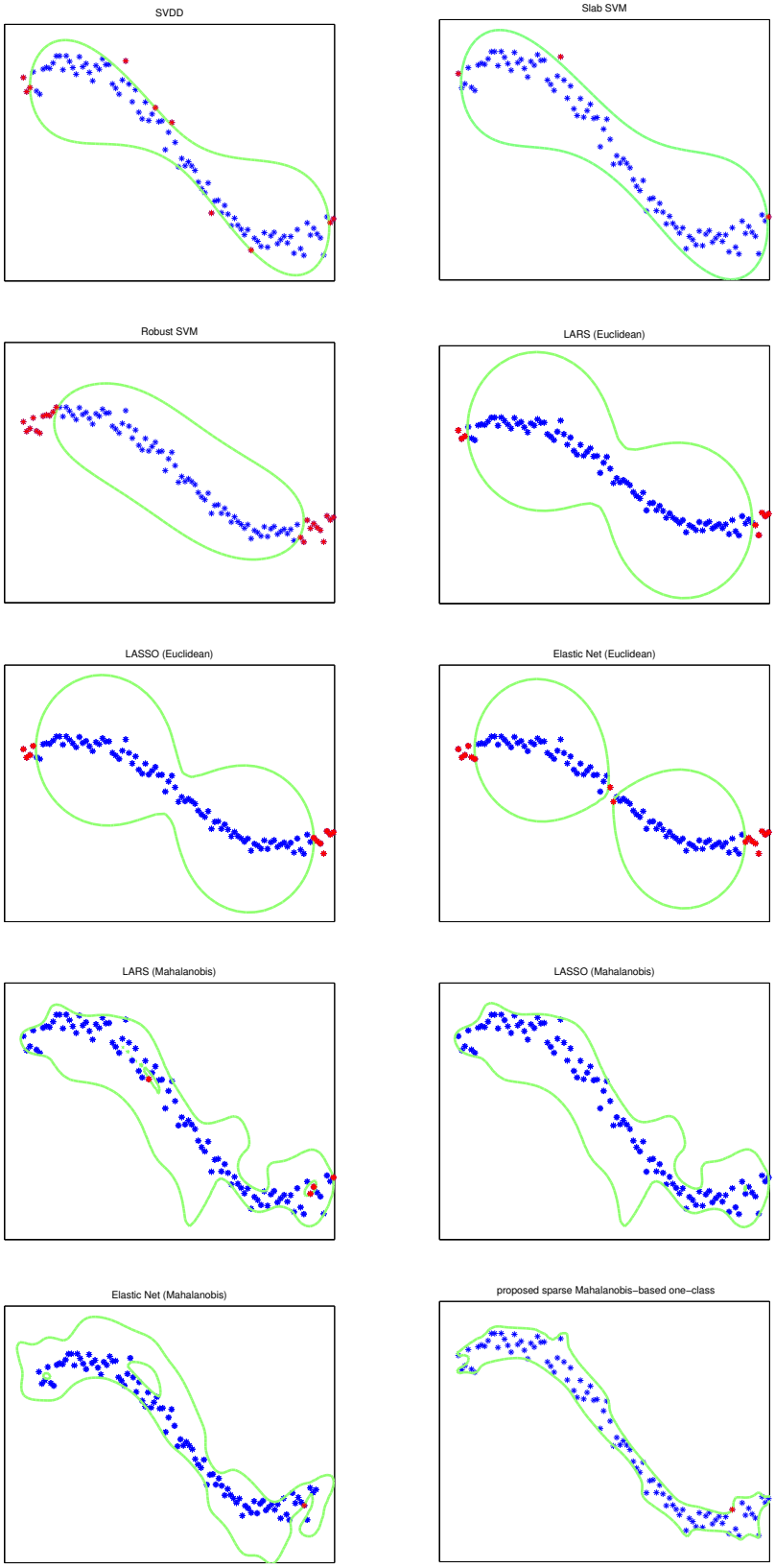

Figure 4.4: The decision boundaries (green lines) on the sinusoidal dataset for the studied sparse one-class algorithms. The red samples are the ones considered as outliers while the normal samples are in blue. The Elastic Net outperforms LARS and LASSO, and it gives a good decision boundary with the truncated Mahalanobis distance and outperforms SVDD, slab SVM and robust SVM. The best result is achieved with the proposed sparse truncated Mahalanobis-based one-class having the tightest decision boundary.

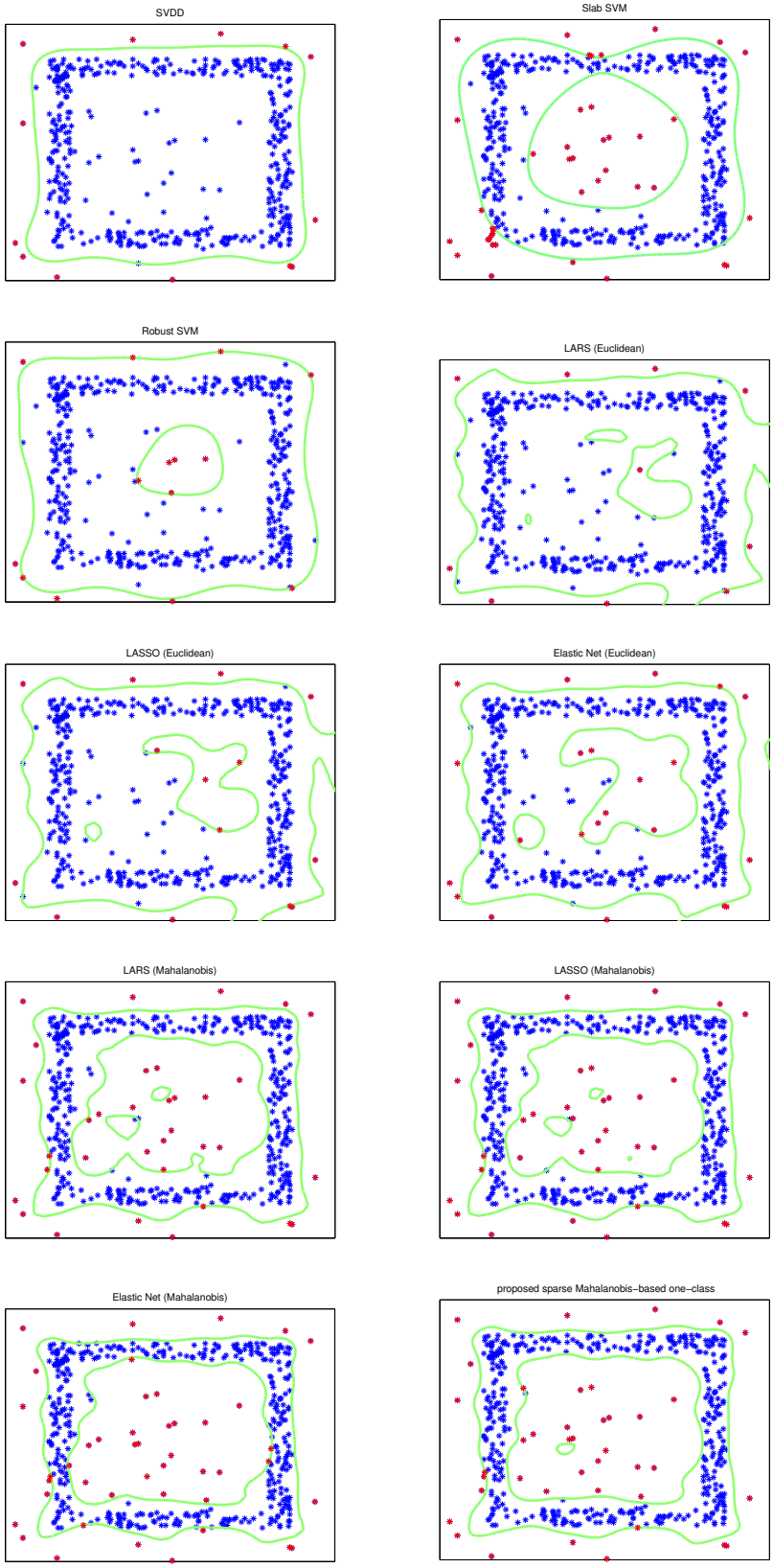

Figure 4.5: The decision boundaries (green lines) on the square-noise dataset for the studied sparse algorithms. The red samples are the ones considered as outliers while the normal samples are in blue. The best results are obtained with the proposed sparse truncated Mahalanobis-based one-class and with the modified Elastic Net with the truncated Mahalanobis distance, and outperform the other approaches.

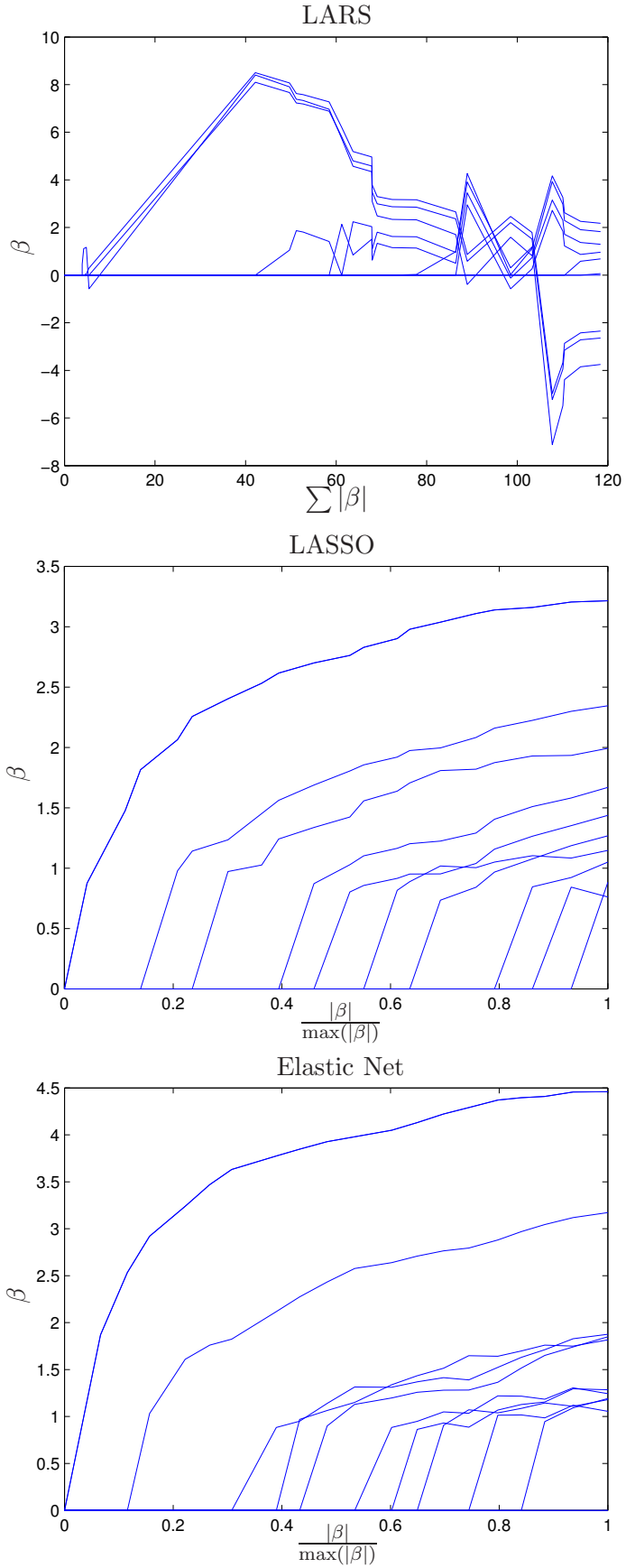

Figure 4.6: The solution paths of LARS, LASSO and Elastic Net algorithms. The LARS solution paths are the most unstable, while Elastic Net has smoother solution paths that clearly show the "grouping effect" advantage of correlated variables over the LASSO.

The first criterion for one-class classification algorithms is the detection rate on new samples not existing in the training dataset. These sparse one-class algorithms are tested on nearly 100 000 samples related to the attacks described in Chapter 1, and the detection rates are given in Tables 4.1, 4.2 and 4.3. The fraction of the support vectors in all the sparse one-class approaches is fixed at 10% of the training samples. When it comes to the first proposed framework based on shrinkages methods, the cases where the truncated Mahalanobis distance is used in the decision function of the classifier have better results than the cases with the Euclidean distance, due to the strong properties of the first one and to the scale sensitivity of the latter one. The modified LARS and LASSO algorithms have nearly the same good results, whereas Elastic Net outperforms both shrinkage algorithms with both Euclidean and truncated Mahalanobis distances. The best results for this framework are achieved when Elastic Net is used, and the decision function of the classifier uses the truncated Mahalanobis distance. The latter combination gives better detection rates than SVDD, slab SVM and robust SVM for the different types of the studied attacks. When it comes to the second proposed framework, namely the sparse truncated Mahalanobis-based one-class, the distance parameter is used as a sparsification rule. This framework has the best detection rates on the real datasets, and it outperforms the first framework and the other one-class methods for most of the attacks. When it comes to the false alarm rate, its value remains around 1% for all the sparse approaches.

The second criterion for sparse one-class approaches is the time consumption of the algorithms. Table 4.4 shows the estimated time for each approach. The modified subset selection algorithms in the proposed framework are faster than the other approaches, regardless of the shrinkage method used with either Euclidean or truncated Mahalanobis distances. The fastest algorithm is with the modified LARS, while the slowest algorithms are the ones in which a constrained quadratic programming problem has to be resolved, namely SVDD, slab SVM and robust SVM, having the slab SVM the slowest one. The use of the truncated Mahalanobis distance instead of the Euclidean distance in the decision function of the classifier slightly increases the time consumption, but it remains 4 times faster than robust SVM, 5 times faster than SVDD, and up to 25 times faster than slab SVM. Moreover, the sparse truncated Mahalanobis-based one-class is hardly as fast as the modified LARS. Therefore, the proposed frameworks lead to the best detection rates, and they are up to 25 times faster than the other one-class algorithms.

Table 4.1: Detection rates for the gas pipeline testbed.

				In this thesis						
				Framework based on shrinkage methods						Sparse Mahalanobis based framework
				Euclidean distance			Mahalanobis distance			
	SVDD	Slab SVM	Robust SVM	LARS	LASSO	Elastic net	LARS	LASSO	Elastic net	
NMRI	98.1	98.4	92.9	98.3	98.7	99.1	99.1	98.9	99.2	**99.3**
CMRI	99.5	99.5	98.5	98.1	98.3	99.2	98.7	98.8	99.5	**99.8**
MSCI	**89.1**	86.2	54.9	55.8	57.3	68.1	71.1	74.5	79.3	81.1
MPCI	98.2	96.9	98.1	97.1	96.7	97.8	98.2	97.6	98.9	**99.1**
MFCI	**89.9**	89.4	64.7	77.8	80.1	83.6	81.3	82.7	85.9	85.4
DOS	96.1	96.3	95.5	96.1	96.9	97.1	97.3	97.2	**97.5**	96.7
RA	**99.8**	**99.8**	99.7	99.1	99.5	**99.8**	99.6	99.7	**99.8**	**99.8**

Table 4.2: Detection rates for the storage testbed.

				In this thesis						
				Framework based on shrinkage methods						Sparse Mahalanobis based framework
				Euclidean distance			Mahalanobis distance			
	SVDD	Slab SVM	Robust SVM	LARS	LASSO	Elastic net	LARS	LASSO	Elastic net	
NMRI	95.1	92.2	94.1	93.4	91.7	94.7	97.4	94.1	98.1	**98.5**
CMRI	61.2	63.5	59.7	59.1	62.4	69.2	71.8	67.7	74.1	**80.1**
MSCI	97.3	96.9	98.1	97.1	97.4	97.9	98.1	98.1	98.3	**98.4**
MPCI	98.6	99.1	99.2	98.9	97.9	99.1	99.1	98.4	**99.7**	99.6
MFCI	97.9	98.7	98.1	97.1	98.4	99.1	99.1	99.3	99.8	**99.9**
DOS	71.7	73.4	59.8	72.3	71.2	74.7	81.1	79.1	**82.6**	80.6
RA	97.8	98.1	98.7	98.1	98.4	98.7	99.1	99.3	99.5	**99.7**

Table 4.3: Detection rates for the UCI water treatment testbed.

				In this thesis						
				Framework based on shrinkage methods						Sparse Mahalanobis based framework
				Euclidean distance			Mahalanobis distance			
	SVDD	Slab SVM	Robust SVM	LARS	LASSO	Elastic net	LARS	LASSO	Elastic net	
P_{ed}	78.6	81.6	74.7	71.4	71.4	78.6	85.7	85.7	**92.1**	**92.1**

4.6.3 Online results

The proposed online one-class classification approach is tested on the three aforementioned real datasets. The proposed online approach is compared to three other approaches, namely the online quarter-sphere SVM [Zhang et al., 2009], the online coherence-based one-class [Noumir et al., 2012a], and the online SVDD [Tax and Laskov, 2003]. The online quarter-sphere SVM has a linear optimization problem, and uses a fixed sliding time window to find the minimal radius at each instant and to update the classifier. The online SVDD uses also a fixed sliding time window, and needs to solve

Table 4.4: Estimated time (in seconds) of each approach.

				In this thesis						
				Framework based on shrinkage methods						Sparse Mahalanobis based framework
				Euclidean distance			Mahalanobis distance			
	SVDD	Slab SVM	Robust SVM	LARS	LASSO	Elastic net	LARS	LASSO	Elastic net	
gas	70.23	302.74	61.32	**9.12**	12.31	13.81	9.85	13.93	14.22	10.21
storage	123.72	557.23	102.28	**10.83**	13.62	14.27	11.79	14.10	15.72	12.02
UCI	12.91	78.95	14.78	**1.61**	2.27	2.51	2.11	2.83	2.94	1.65

a constrained quadratic programming problem at each instant. The online coherence-based one-class considers a least-squares optimization problem, and uses the coherence criterion for sparsification. We set the sparsity of the center in the proposed approach to 10% of the training samples, and the regularization parameters of the other methods are also fixed in order to get the same sparsity level. The size of the sliding window in the online quarter-sphere SVM and online SVDD is fixed at 1000 samples, which is the same number of training samples used in the offline phase of the proposed approach.

The first important criterion for online intrusion detection is the error detection rate. These algorithms are tested on nearly 100 000 samples related to the attacks described in Chapter 1. The resulting detection rates are given in Tables 4.5, 4.6 and 4.7. The results show that the proposed online approach gives better detection rates and outperforms the other approaches for all the studied attacks. Furthermore, in some cases, we have important gaps between the detection rates of the proposed online approach and the other approaches. These results can be explained by the advantages of the proposed online approach, in which the modified one-class formulation allows the isolation of the outliers without including them in the update step of the classifier. The second important criterion for online detection is the false alarm rate. The quarter-sphere SVM has a false alarm rate equal to 11% in average, the coherence-based one-class has 4%, the online SVDD has 5%, while the proposed online approach misclassified only 1% of the normal samples.

Another important criterion for online detection is the computational complexity. The estimated training time of the studied approaches is given in Table 4.9. The results show that the proposed online approach is the fastest one and, as expected, and the online SVDD is the slowest one since it needs to solve a constrained quadratic programming problem at each instant. Finally, the time for each approach for testing new samples is given in Table 4.10. The proposed approach has the best results with 0.0019 second for each new sample, the quarter-sphere SVM needs 0.0027 second, the coherence approach 0.0022 second and the online SVDD 0.0026 second.

Table 4.5: Detection rates for the gas pipeline testbed.

				In this thesis
	Quarter SVM	Online coherence	Online SVDD	proposed approach
NMRI	92.1	86.1	95.1	**99.3**
CMRI	98.4	92.4	99.3	**99.8**
MSCI	71.1	63.5	76.7	**81.4**
MPCI	98.1	92.4	98.7	**99.1**
MFCI	76.3	68.6	79.2	**83.3**
DOS	81.2	84.8	89.9	**95.6**
RA	99.7	91.76	**99.8**	**99.8**

Table 4.6: Detection rates for the storage tank testbed.

				In this thesis
	Quarter SVM	Online coherence	Online SVDD	proposed approach
NMRI	92.7	87.9	95.3	**98.4**
CMRI	70.1	74.3	75.7	**80.8**
MSCI	96.2	86.7	97.7	**98.4**
MPCI	99.1	90.3	99.4	**99.6**
MFCI	98.3	85.6	99.2	**99.8**
DOS	71.7	73.7	74.8	**82.1**
RA	94.2	88.4	96.9	**99.7**

Table 4.7: Detection rates for the water treatment testbed.

				In this thesis
	Quarter SVM	Online coherence	Online SVDD	proposed approach
P_{ed}	85.7	81.6	88.9	**92.1**

All these results are very interesting for online intrusion detection in real-world applications, where the proposed approach has the highest detection rates and the lowest false alarm rates, while it needs less than 0.002 second to detect the intrusion.

Table 4.8: False alarm rates of the online approaches.

				In this thesis
	Quarter SVM	Online coherence	Online SVDD	proposed approach
False alarm rate	0.11	0.04	0.05	**0.01**

Table 4.9: Estimated training time (in seconds) of each approach.

	Quarter SVM	Online coherence	Online SVDD	In this thesis proposed approach
gas	19.8	14.1	73.7	**11.7**
storage	21.3	16.3	104.1	**12.9**
UCI	3.9	2.1	13.9	**1.7**

Table 4.10: Estimated time (in seconds) to test a new sample.

	Quarter SVM	Online coherence	Online SVDD	In this thesis proposed approach
gas	0.0027	0.0022	0.0029	**0.0019**
storage	0.0025	0.0021	0.0026	**0.0018**
UCI	0.0028	0.0023	0.0030	**0.0019**

4.7 Conclusion

In this chapter, we investigated sparse formulations for one-class classification in order to reduce the computational complexity of the algorithms. We proposed two frameworks for sparse one-class classification, where the classifiers were defined by a hypersphere enclosing most of the samples, with its center defined using a sparse model that is expressed with only a small fraction of the training samples. The first framework was a sparse formulation of the truncated Mahalanobis-based one-class approach detailed in Chapter 3, and the selection of the support vectors was based on ad-hoc sparsification criteria, such as the distance criterion or the coherence criterion. The second framework was based on estimating simultaneously the sparse center and the support vectors. To this end, well-known shrinkage methods, namely LARS, LASSO and Elastic Net, were modified and adapted for the selection of the most relevant samples in the RKHS. In this framework, we used the Euclidean and the truncated Mahalanobis distances in the decision function of the classifier. We tested these frameworks on simulated and real data, and we compared the results with other sparse approaches. The results showed that the proposed frameworks outperformed the other sparse approaches, they had the best description boundaries on the 2-dimensional simulated data, and the best detection rates and the fastest algorithm on the high dimensional real data.

In addition, we investigated online one-class classification approaches for real time and sequential detection applications. We proposed an online approach that represented an extended version of the sparse one-class truncated Mahalanobis-based approach detailed in Section 4.2. In the proposed online approach, we modified the decision function of

the classifier, by defining two concentric hyperspheres enclosing the support vectors of the description. This new definition of the one-class problem allowed the separation of the outliers from the support vectors, which made the algorithm more robust to outliers, and it reduced the computational costs of the proposed algorithm. We compared the proposed online approach with other methods. The results showed that the proposed approach gave the highest detection rates, the lowest false alarm rates, and it was the fastest one when it comes to testing new unseen samples.

General Conclusion and Future Works

Supervisory Control and Data Acquisition (SCADA) systems have played an important role in monitoring and controlling the industrial systems and critical infrastructures. The security of these systems has been a hot topic in the past few years with the growth of cyberthreats and the increasing number of new and sophisticated cyberattacks generated on daily basis. This thesis has investigated new techniques in order to provide the necessary complementary help to the traditional intrusion detection systems (IDS) in detecting malicious intrusions and cyberattacks against critical infrastructures and industrial systems. The proposed techniques do not replace the existing IDS, but they are complementary to their work, and the primary objective of this thesis is to detect intrusions that have already bypassed traditional IDS and firewalls.

In the first place, we have presented the kernel methods in machine learning. These methods rely on mapping the training samples from the input space into a feature space, where linear algorithms are applied. In particular, we have investigated radial basis function kernels that are very common for classification techniques, namely the Gaussian and the Laplacian kernels. We have studied the influence of the metric in these kernels on the decision rule of the one-class classifier, and we have proposed a simple heuristic for choosing their bandwidth parameter.

In the second place, we have proposed a simple and fast one-class classification approach in which the classifier is defined by the hypersphere enclosing the training samples. We have estimated the center of this hypersphere without solving any constrained quadratic programming problem. We have used the truncated Mahalanobis distance in the feature space as a novelty measure, by projecting the samples onto the subspace spanned by the eigenvectors associated to the largest eigenvalues of the covariance matrix. We have applied this approach on simulated and real datasets, and we have compared the results with state-of-the-art one-class classification methods. The proposed approach has led

to the best description of the distribution of the training samples, the highest detection rates, and it has been the fastest approach.

In addition, we have proposed two frameworks for sparse one-class classification in order to decrease the computational complexity. The first framework is a sparse version of the Mahalanobis-based one-class approach detailed in Chapter 3, while the second framework is based on well-known shrinkage methods, namely Least Angle Regression, Least Absolute Shrinkage and Selection Operator, and Elastic Net. We have revisited these shrinkage methods and adapted their algorithms for estimating the sparse center of the one-class classifier. The tests have been conducted on simulated and real datasets, and the proposed frameworks have led to the best results compared with state-of-the-art sparse one-class classification methods.

Finally, we have proposed an online one-class classification framework for real-time detection applications. In the proposed online approach, which represents an extended version of the sparse one-class Mahalanobis-based approach detailed in Section 4.2, we have modified the decision function of the classifier by defining two concentric hyperspheres, with the support vectors lying in between them. This new definition of the one-class problem has allowed the separation of the outliers from the support vectors, which has made the algorithm more robust to outliers, and it has reduced the computational costs of the proposed algorithm. We have compared the proposed online one-class approach with state-of-the-art online methods. The results have showed that the proposed approach has given the highest detection rates, the lowest false alarm rates, and it has been the fastest one.

Future Works

Many enhancements can be made to improve the performance of the algorithms studied in this thesis. A further and more detailed study on the effect of using the truncated Mahalanobis distance is required. When it comes to sparse approximation, a detailed study on the other existing subset selection algorithms could be investigated, e.g., the adaptation of the dictionary elements as investigated in [Saide et al., 2013]. The authors considered the dictionary elements as adjustable model parameters, and they proposed an adaptation scheme in order to ensure a better performance and to minimize the instantaneous quadratic error. In addition, one can consider modifying the optimization problem of shrinkage methods, in a way to replace the Euclidean distance with the truncated Mahalanobis distance. It is worth noting that one can extend this work for multiclass classification in order to identify the type of the detected attacks. Solutions

should have low computational complexity, as studied for instance in [Honeine et al., 2013].

Furthermore, we should consider to integrate our approaches in the traditional intrusion detection systems in industrial infrastructures, since these approaches could play an important and complementary role to the IDS in detecting malicious attacks on physical systems, and they have a high processing performance (over 200 samples per second). Finally, online versions within the proposed framework could also be investigated, and they should be integrated in the SCADA systems to improve the real-time detection of machine faults and cyberattacks.

Appendix A

Résumé de la thèse

La sécurité des systèmes industriels et des infrastructures critiques a gagné l'attention des chercheurs au cours des dernières années avec l'augmentation du risque des cyber-attaques et des menaces terroristes contre ces systèmes. La majorité de ces infrastructures est contrôlée par les systèmes SCADA (Supervisory Control And Data Acquisition) qui permettent la surveillance et le contrôle à distance des processus industriels, comme les réseaux électriques, le transport de gaz, la distribution de l'eau potable, le traitement des eaux usées, les centrales nucléaires, etc. Les systèmes SCADA deviennent de plus en plus interconnectés avec le monde extérieur via les réseaux publics, ce qui a entrainé une augmentation du risque de cyber-attaques contre ces systèmes. Les systèmes traditionnels de détection d'intrusions sont incapables de détecter les nouvelles attaques qui ne figurent pas dans leurs bases de données, et par suite ils ne peuvent pas assurer une protection maximale pour les infrastructures critiques.

L'objectif principal de cette thèse est d'apporter une aide supplémentaire aux systèmes traditionnels de détection d'intrusions pour assurer une meilleure protection aux systèmes industriels contre les cyber-attaques et les intrusions. Afin d'atteindre cet objectif, nous utilisons les méthodes à noyaux, qui transforment les relations non-linéaires entre les données d'apprentissage en des relations linéaires dans l'espace transformé. Nous nous intéressons en particulier aux méthodes de classification mono-classe. Ces méthodes élaborent une fonction de décision à partir de données d'apprentissage, pour classer les nouveaux échantillons en données aberrantes ou données normales. La fonction de décision définie l'enveloppe d'une région de l'espace de données contenant la majeure partie des données d'apprentissage.

Dans ce manuscrit, les systèmes SCADA et leur différentes vulnérabilités sont présentés dans la Section A.1. La Section A.2 introduit les méthodes à noyaux, fournit une étude sur l'influence de la variation de la métrique des noyaux sur la fonction de décision du

classificateur, et propose une simple heuristique pour l'estimation de l'écart-type des noyaux à base radiale. La Section A.3 résume les méthodes de classification mono-classe existantes, et propose une approche simple et rapide pour l'estimation du centre du classificateur mono-classe. La Section A.4 se consacre sur les approches mono-classe qui reposent sur une représentation parcimonieuse du classificateur. Deux approches sont proposées dans cette section. La première est une formulation parcimonieuse de l'approche proposée dans la Section A.3. La deuxième approche est basée sur des méthodes connues de sélection de variables telles que LARS (Least Angle Regression), LASSO (Least Absolute Shrinkage and Selection Operator) et Elastic Net. Nous proposons aussi dans cette section une approche mono-classe en ligne, pour améliorer la détection en temps réel. La Section A.5 offre une conclusion et des travaux futurs.

A.1 Systèmes SCADA

Le rôle des systèmes SCADA (Supervisory Control And Data Acquisition) a augmenté dans les dernières décennies dans de nombreux domaines, en particulier dans les systèmes industriels et le secteur des infrastructures critiques. Les systèmes SCADA fournissent un accès à distance permettant la surveillance et le contrôle des infrastructures critiques telles que les réseaux électriques, les systèmes de distribution de gaz naturel, les usines de traitement des produits chimiques, les systèmes de distribution d'eau, les systèmes de traitement des eaux usées, les centrales nucléaires, etc [Stouffer et al., 2006; Bailey and Wright, 2003]. Les systèmes SCADA sont utilisés pour contrôler des installations dispersées géographiquement. Ils collectent des données relatives au système industriel supervisé, affichent ces informations pour les opérateurs, et les enregistrent dans les bases de données du système. Cela permet aux opérateurs de surveiller et de contrôler le système en temps réel, et d'envoyer des instructions de commande au système. Par exemple, une simple application SCADA serait de surveiller le niveau d'eau d'un réservoir; lorsque ce niveau dépasse un certain seuil, SCADA active le système de pompage pour transférer l'eau vers des réservoirs secondaires.

A.1.1 Architecture des systèmes SCADA

Les systèmes SCADA sont des systèmes comprenant des composants matériels et des logiciels. Les composants matériels permettent le transfert des données et des informations entre les composants des systèmes SCADA, tels que la radio, les lignes téléphoniques, et les capteurs distribués partout sur le site surveillé. Le logiciel permet d'indiquer au système les variables à surveiller, le paramétrage acceptable de ces variables dans les modes de fonctionnement normal du système, et les réponses adéquates du

système lorsque les paramètres vont en dehors des valeurs acceptables. Les composants principaux d'un système SCADA sont [Stouffer et al., 2011]:

- Une Interface Homme-Machine (IHM) : Elle affiche les informations relatives à l'état des différentes variables du processus sous forme d'une représentation schématique dy système physique. L'IHM permet aux opérateurs de surveiller l'état du processus physique et de modifier ces paramètres de contrôle.
- Une unité centrale de supervision et contrôle informatique : Cette unité est en charge du processus physique. Elle collecte les informations relatives au processus physique à partir des différents composants des systèmes SCADA, et stocke ces données reçues dans ses bases de données. Ensuite, elle émet les commandes de contrôle au systèm en se basant sur les informations reçues.
- Des unités terminales distantes (RTUs): Les RTUs sont des unités d'acquisition et de contrôle de données situées sur les sites distants du processus physique. Elles collectent les données des capteurs sur le terrain, et convertissent ces signaux en données numériques. Les RTUs gardent ces informations en mémoire jusqu'à ce que l'unité centrale les demande. Elles reçoivent également les signaux de commande de l'unité centrale.
- Les automates programmables industriels (PLCs): Ils sont de petits ordinateurs industriels conçus à l'origine pour effectuer les fonctions logiques du matériel électrique (relais, commutateurs, compteurs). Les PLCs sont reliés aux capteurs du processus physique, et ils ont des fonctionnalités plus sophistiquées que les RTUs. Les PLCs sont parfois utilisés pour servir de RTUs, car ils sont plus économiques, polyvalents, flexibles et configurables que les RTUs.

A.1.2 Vulnérabilités des systèmes SCADA

Les premières générations de systèmes SCADA étaient conçues pour opérer dans des environnements isolés, sans aucun échange ni connexion avec le monde extérieur au réseau. Les protocoles utilisés dans la communication entre les composants d'un système SCADA étaient propriétaires, donc très peu de personnes connaissaient le niveau de sécurité de ces installations. De nos jours, les protocoles de communication sont standardisés et en solutions ouvertes. En outre, les systèmes industriels et les infrastructures critiques sont de plus en plus interconnectés avec le monde extérieur via les réseaux publics, comme l'Internet. Ceci a introduit de nombreuses vulnérabilités à ces systèmes, ce qui a exposé ces infrastructures critiques à de nouvelles sources de menaces potentielles [Ten et al., 2008, 2011].

En plus des vulnérabilités en raison de la dépendance de leurs communications à l'Internet, les systèmes SCADA sont aujourd'hui confrontés à des menaces de cyber-attaques à cause des vulnérabilité des protocoles de communication implémentés dans leurs réseaux [Fovino et al., 2009; Morris and Pavurapu, 2010]. En fait, les protocoles les plus utilisés, tels que ModBus, Profibus et DNP3, présentent de nombreuses vulnérabilités concernant l'intégrité des informrmations transmises et les mécanismes d'authentification [Fovino et al., 2010b, 2012]. Ces protocoles ne vérifient pas l'intégrité des paquets transmis entre l'unité centrale et les autres composants du système, ne s'assurent pas de l'authentification de la source, et n'appliquent pas de mécanismes pour la non-répudiation.

A.1.3 Attaques contre SCADA

Les malfaiteurs ont profité des vulnérabilités des infrastructures critiques pour accéder aux réseaux SCADA, recueillir les informations échangées, et implanter des virus perturbant le fonctionnement normal du système. Les dernières décennies ont été témoins de plusieurs cyberattaques intentionnelles contre ces systèmes industriels, ce qui a causé de graves dégâts matériels et économiques. En 2000, un ex-employé en Australie a pris le contrôle des stations de pompage des eaux usées, et a libéré un million de litres d'eau non traitée dans les rivières et parcs locaux [Slay and Miller, 2007]. En 2003, un virus informatique a désactivé un système de surveillance de sécurité de la centrale nucléaire de Davis-Besse Ohio (Etats-Unis) pour près de cinq heures [Christiansson and Luiijf, 2007]. En 2009, des cyber-espions ont pénétré le réseau électrique américain et ont implanté des programmes qui pourraient être utilisés pour perturber le système [Gorman, 2008]. En 2010, le virus Stuxnet, découvert en Iran, visaient les automates connectés à des centrifugeuses nucléaires utilisées pour enrichir l'Uranium. Stuxnet implémentait des programmes malveillants d'une manière indétectable par l'opérateur [Chen and Abu-Nimeh, 2011]. En 2012, le virus flame a été découvert en Hongrie. Flame est un logiciel dédié à voler des informations. Il permet d'intercepter les e-mails, d'enregistrer les conversations en ligne, et envoyer ces informations via Bluetooth [Bencsáth et al., 2012b].

A.1.4 Méthodes de détection existantes

Les cyberattaques menaçant les systèmes industriels deviennent de plus en plus complexes, sophistiquées, et difficiles à détecter. De plus, de nouvelles attaques sont générées chaque jour. Pour ces raisons, les chercheurs se sont consacrés sur le développement des méthodes de détection pour limiter l'impact et les dégâts des cyberattaques sur les infrastructures critiques. Plusieurs méthodes de détection d'intrusions ont été proposées

dans [Gross et al., 2004; Fovino et al., 2010a; Yang et al., 2014; Carcano et al., 2011]. Une connaissance préalable sur le processus physique et sur ses différents états critiques est obligatoire pour construire les règles de détection pour certaines de ces méthodes, alors que les autres ne peuvent pas détecter les nouvelles attaques qui n'existent pas dans leurs bases de données. D'autres méthodes reposant sur des approches statistiques et paramétriques ont été proposées dans [Bigham et al., 2003; Knorn and Leith, 2008; Veracini et al., 2011; Ghadiri and Ghadiri, 2011]. Ces méthodes statistiques ne fonctionnent que sur des modèles prédéfinis ayant des relations linéaires entre les variables, nécessitent la connaissance statistique préalable sur les probabilités des réalisations de chaque échantillon, et exigent une connaissance préalable sur les différents types d'attaques pour assurer une détection précise.

Par conséquence, les méthodes traditionnelles de détection d'intrusions ont besoin d'une aide supplémentaire afin de détecter les nouvelles attaques générées chaque jour. La diversité des cyberattaques et de la complexité des systèmes étudiés ont rendu la modélisation des attaques très difficile, ce qui restreint le rôle des approches basées sur des modèles paramétriques. Ceci met en évidence le rôle potentiel des méthodes d'apprentissage statistique non paramétriques dans la détection des intrusions. L'objectif principal de cette thèse n'est pas de remplacer les IDS, mais d'apporter une aide complémentaire et nécessaire pour fournir une meilleure protection pour les systèmes industriels et des infrastructures critiques. Dans la section suivante, nous introduisons les méthodes à noyaux utilisées dans le domaine de l'apprentissage statistique.

A.2 Methodes à noyaux

Au cours des dernières décennies, les méthodes à noyaux sont devenues très populaires dans le domaine de l'apprentissage statistique, et ont fourni un moyen puissant pour détecter les relations cachées entre les échantillons [Hofmann et al., 2008; Shawe-Taylor and Cristianini, 2004]. Les méthodes à noyaux transforment les échantillons de l'espace initial à un espace de plus grande dimension. Des algorithmes linéaires sont appliqués sur les échantillons dans l'espace transformé, afin de détecter les relations cachées [Vert et al., 2004]. Les algorithmes utilisés peuvent être exprimés en fonction du produit scalaire entre les échantillons. Afin d'exploiter cette propriété, les méthodes à noyau utilisent des noyaux définis positifs pour l'injection dans l'espace transformé. Dans cette section, nous résumons les méthodes à noyaux et leurs propriétés principales. Nous étudions ensuite l'impact de la variation de la métrique dans ces noyaux, et nous proposons une heuristique simple pour le choix du paramètre de largeur de bande de ces noyaux.

A.2.1 Noyau défini positif et espace de Hilbert associé

Les méthodes à noyaux sont des techniques d'apprentissage statistique qui reposent sur l'injection des échantillons de l'espace initial $\mathcal{X}$ dans un espace de plus grande dimension $\mathcal{H}$. La fonction non-linéaire ϕ pour l'injection des échantillons est définie comme suit :

$$\begin{aligned} \phi\colon \mathcal{X} &\longrightarrow \mathcal{H} \\ \boldsymbol{x} &\longmapsto \phi(\boldsymbol{x}). \end{aligned}$$

L'injection des échantillons dans l'espace transformé permet de transformer le problème, initialement non-linéaire en $\boldsymbol{x}$, en un problème linéaire en $\phi(\boldsymbol{x})$. Les algorithmes utilisés dans l'espace transformé sont exprimés en fonction du produit scalaire entre les échantillons, sans avoir besoin d'une forme explicite de la fonction d'injection non-linéaire ϕ. Le produit scalaire est donc remplacé par une fonction dite noyau, définie de la manière suivante :

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \big\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \big\rangle_{\mathcal{H}}, \quad \text{pour} \quad \boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathcal{X},$$

où $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ désigne le produit scalaire dans l'espace transformé $\mathcal{H}$. Les fonctions noyaux utilisées dans ces méthodes d'apprentissage sont des fonctions définies positives. Une fonction k est appelée noyau défini positif si et seulement si elle est symétrique, c'est-à-dire $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_j, \boldsymbol{x}_i)$ pour n'importe quels échantillons $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$, et défini positif tel que :

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0,$$

pour tout $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$, et tout $c_1, \cdots, c_n \in \mathcal{R}$.

L'espace transformé est un espace de Hilbert défini par un ensemble de fonctions. Chacune de ces fonctions est une combinaison linéaire des échantillons de l'espace transformé. L'espace $\mathcal{H}$ a la forme suivante :

$$\mathcal{H} = \Big\{ \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i), \quad \boldsymbol{x}_i \in \mathcal{X}, \alpha_i \in \mathcal{R}, i = 1, \cdots, n \Big\}.$$

Considérons 2 fonctions f et g de l'espace transformé, notamment:

$$f = \sum_{i=1}^{l} \alpha_i \phi(\boldsymbol{x}_i) \quad \text{et} \quad g = \sum_{j=1}^{n} \beta_j \phi(\boldsymbol{x}_j).$$

Le produit scalaire entre f et g dans l'espace transformé est construit comme suit:

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{l} \sum_{j=1}^{n} \alpha_i \beta_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i=1}^{l} \alpha_i g(\boldsymbol{x}_i) = \sum_{j=1}^{n} \beta_j f(\boldsymbol{x}_j).$$

En prenant le cas particulier de $g = \phi(\boldsymbol{x})$, le produit scalaire entre f et g dans $\mathcal{H}$ conduit à la propriété suivante:

$$\langle f, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}} = \sum_{i=1}^{l} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) = f(\boldsymbol{x}).$$

Cette propriété est connue sous le nom de *propriété reproduisante* du noyau. L'espace transformé $\mathcal{H}$, qui correspond à la fonction k satisfaisant la propriété du noyau défini positif, sera désigné par *espace de Hilbert à noyau reproduisant (RKHS)*.

L'avantage d'utiliser ce genre de noyau est que ceci permet de construire des algorithmes de classification dans des espaces transformés, via une fonction noyau k qui s'écrit comme un produit scalaire entre les échantillons dans cet espace. Ces algorithmes ne requièrent pas le calcul des coordonnées des échantillons dans cet espace, donc sans la nécessité de connaitre explicitement l'expression de la fonction non-linéaire ϕ. Cette idée clé, connue sous le nom du *coup du noyau* ou *kernel trick* en anglais, est utilisée pour transformer les méthodes linéaires en des méthodes non-linéaires, sous réserve qu'elles puissent s'exprimer en fonction de produits scalaires des échantillons. Pour cela, le produit scalaire $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}$ est remplacé par le noyau $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. La Table 2.1 montre les noyaux reproduisants les plus utilisés dans la littérature.

Dans cette thèse, nous travaillons avec les noyaux à base radiale (RBF), comme étant les noyaux les plus communs et appropriés pour des problèmes de classification [Schölkopf et al., 2001b; Tax and Juszczak, 2002]. En particulier, nous utilisons le noyau Gaussien et le noyau exponentiel qui suit une distribution de Laplace :

$$\text{Noyau Gaussian :} \quad k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(- \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma^2} \right)$$

$$\text{Noyau exponentiel :} \quad k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(- \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2}{\frac{\sigma}{\sqrt{2}}} \right),$$

où $\boldsymbol{x}_i$ et $\boldsymbol{x}_j$ sont deux échantillons d'entrée, $\|\cdot\|_2$ est la distance Euclidienne, et σ représente le paramètre de largeur de bande de ces noyaux. Le paramètre σ doit être choisi de façon à éviter un sur-classement et sous-classement des données. La performance des méthodes à noyaux est fortement liée au choix de la métrique dans ces fonctions, ainsi qu'au choix du paramètre de largeur de bande. Dans la suite, nous

étudions l'impact de la variation de la norme dans ces noyaux, et nous proposons une heuristique pour choisir le paramètre de largeur de bande.

A.2.2 Variation de la norme des noyaux

Dans cette thèse, nous nous intéressons à des processus industriels pour mesurer la pression de gaz, la température d'un réacteur, les niveaux d'eau dans les réservoirs de stockage, etc. La variation de la valeur des variables est importante pour évaluer l'état du processus physique, afin de prévoir s'il passe d'un état de fonctionnement normal à des états critiques ou s'il est sous attaques malveillantes. Par conséquent, nous avons besoin de noyaux bien adaptés qui prennent en considération les petits changements simultanés de plusieurs attributs ainsi que les grandes variations d'un seul. Nous proposons de remplacer la norme ℓ_2 des noyaux RBF par d'autres normes afin d'étudier les effets de la variation de la norme sur la fonction de décision des classificateurs, donc sur le comportement des algorithmes d'apprentissage statistique.

Afin de comprendre l'impact de la variation de la norme dans les noyaux, la variation du comportement des différentes normes ℓ_p dans un espace à 2 dimensions est illustré dans la Figure 2.3. Chaque échantillon a deux attributs, à savoir attribut 1 et attribut 2, et p prend une valeur parmi $\frac{3}{4}, 1, \frac{3}{2}, 2, 3, 4, 7$ et ∞. Chaque couleur représente des contours équidistants par rapport à l'origine O. Il est évident dans cette figure que chaque norme fonctionne différemment sur la variation simultanée de la valeur de plusieurs attributs. Par exemple, lorsque l'on considère la norme ℓ_2, connue également par la distance Euclidienne, une grande variation de la valeur d'un seul attribut a un effet beaucoup plus important que des petites variations simultanées des deux attributs. En effet, pour cette distance Euclidienne, les échantillons B et C sont à égale distance de l'origine O, les échantillons A et E sont équidistants de l'origine O, alors que A et E sont beaucoup plus loin que B et C. Cependant, pour la norme ℓ_1, C et D sont à égale distance de l'origine O et beaucoup plus proche de B, alors que ce même échantillon B est aussi loin de l'origine que le point C avec la norme ℓ_2. Ceci montre que pour le cas de la norme ℓ_1, un petit changement simultané de plusieurs attributs a la même importance que les grandes variations dans un seul. Par conséquent, les normes ayant une petite valeur de p sont particulièrement sensibles sur les petites variations simultanées de plusieurs attributs, tandis que ceux avec des valeurs de p plus grandes sont plus sensibles à de grandes variations dans un seul attribut.

Considérons la distance Euclidienne où $p = 2$ et les distance de Manhattan ayant $p = 1$. On peut facilement montrer que la norme ℓ_2 de tout vecteur est bornée par sa norme ℓ_1

comme suit:

$$\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2 \leq \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_1.$$

En utilisant les noyaux Gaussien et exponentiel, les échantillons dans l'espace transformé sont plus dispersés avec la norme ℓ_2 qu'avec la norme ℓ_1. Ceci montre que la variation de la norme dans les noyaux RBF affecte la distribution des échantillons dans l'espace transformé. Ce résultat peut être généralisable pour d'autres normes, et les relations pour les noyaux Gaussien et exponentiel deviennent :

$$\exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_p}{\frac{\sigma}{\sqrt{2}}}\right) \geq \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_r}{\frac{\sigma}{\sqrt{2}}}\right),$$

$$\exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_p^2}{2\sigma^2}\right) \geq \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_r^2}{2\sigma^2}\right),$$

pour tout $0 < r < p$.

A.2.3 Le paramètre de largeur de bande

La majorité des travaux de recherche dans la littérature a été concentrée sur les problèmes d'apprentissage supervisé, car il est plus facile de construire des théories et des algorithmes sur des échantillons étiquetés que sur des échantillons non étiquetés. Dans cette thèse, nous nous concentrons sur l'apprentissage non supervisé. Un des problèmes des noyaux RBF est le choix du paramètre de largeur de bande du noyau. Ce paramètre doit être choisi judicieusement car il joue un rôle crucial dans la définition de la fonction de décision du classificateur. D'une part, une petite valeur de σ conduit à une fonction de décision qui sur-classifie les échantillons d'apprentissage. D'une autre part, une grande valeur de σ provoque une sous-classification des échantillons d'apprentissage.

Plusieurs approches ont été proposées dans la littérature afin de choisir le paramètre de largeur de bande des noyaux RBF [Shi and Malik, 2000; Soares et al., 2004; Cherkassky and Ma, 2004; Evangelista et al., 2007; Gurram and Kwon, 2011]. L'inconvénient majeur de ces approches était la grande complexité des algorithmes menant à un coût de calcul très élevé, sans garantir la convergence vers une valeur optimale de ce paramètre. L'heuristique proposée dans cette thèse est inspiré par le travail de Haykin [1998]. Cette heuristique possède un temps de calcul relativement faible. En outre, les résultats empiriques qui apparaissent dans la suite montrent que l'heuristique proposée conduit à un très bon choix du paramètre de largeur de bande σ.

Le paramètre σ dépend de la distribution des échantillons d'apprentissage, du nombre des échantillons d'entrée ainsi que sur la fraction des échantillons considérés comme

vecteurs de support. Pour cela, l'estimation de σ devrait prendre en considération tous ces facteurs. En outre, on peut estimer, à partir de l'ensemble d'apprentissage, la fraction des échantillons considérés comme vecteurs de support. Nous proposons donc d'utiliser dans les algorithmes de classification de cette thèse l'expression suivante pour le calcul du paramètre de largeur de bande σ :

$$\sigma = \frac{d_{\max}}{\sqrt{2M}},$$

où d_{max} correspond à la distance maximale entre les échantillons d'apprentissage, et M représente l'estimation du nombre des vecteurs de support. La relation entre la distribution des échantillons et le nombre des vecteurs de support définit la largeur de bande σ du noyau. Nous notons que la norme utilisée pour le calcul de d_{max} est la même que celle dans les expressions des noyaux RBF. Cette expression garantit que les cas extrêmes (sur-classification et sous-classification des échantillons) sont évités, et que ce paramètre est obtenu avec un coût de calcul minimal. Dans ce quit suit, nous nous concentrons sur la classification mono-classe.

A.3 Classification mono-classe

Dans les problèmes de classification binaire et multi-classe, la fonction de décision du classificateur est soutenue par la présence des échantillons de chaque classe. Les algorithmes correspondants classifient les nouveaux échantillons parmi l'une de ces classes prédéfinies [Mathur and Foody, 2008; Rocha and Klein Goldenstein, 2014]. Dans de nombreuses applications, notamment dans le domaine industriel, seules les données de fonctionnement normal sont disponibles en quantité significative, alors que les données relatives aux modes anormaux ou aux états critiques sont difficiles à obtenir. De plus, le nombre des modes de défaillance et celui des nouvelles attaques générées tous les jours ne peuvent pas être limités [Ten et al., 2008]. Pour cette raison ces dernières années, des chercheurs se sont intéressés aux algorithmes de classification mono-classe, où les données disponibles correspondent à une classe unique [Hoffmann, 2007; Chandola et al., 2009]. Ces méthodes apprennent les modes de fonctionnement normal du système, et développent des fonctions de décision afin de tester des nouveaux échantillons et détecter les valeurs aberrantes [Khan and Madden, 2010]. Dans cette section, nous résumons les méthodes de classification mono-classe bien connues. Nous proposons ensuite une simple approche pour la classification mono-classe, et nous présentons les résultats sur des données simulées et réelles.

A.3.1 Méthodes de classification mono-classe existantes

Plusieurs formulations ont été proposées dans la littérature pour la classification mono-classe. L'approche des machines à vecteurs de support (SVM) a été proposé dans [Schölkopf et al., 1998b]. Cette méthode consiste à séparer dans l'espace de Hilbert les échantillons de l'origine à l'aide d'un hyperplan à marge maximale. Cette approche, se reposant sur une solution parcimonieuse, requiert la résolution d'un problème quadratique avec contraintes, et admet un coût de calcul élevé. L'approche SVDD (Support Vector Data Description) a été introduité dans [Tax and Duin, 1999]; elle estime l'hypersphère de rayon minimale contenant la plupart des données d'apprentissage. Cette approche exige aussi de résoudre un problème quadratique équivalent à celui des SVM. Afin d'avoir la même variance dans toutes les directions, une normalisation est préconisée dans [Tax and Juszczak, 2002]. Une approche rapide a été introduite dans [Noumir et al., 2012b] pour surmonter les inconvénients des algorithmes existants, mais elle s'est avérée sensible à la présence des d'échantillons aberrants. Le "Slab SVM" utilisé dans [Schölkopf et al., 2005; Tao et al., 2005] définit 2 hyperplans pour séparer les échantillons de l'origine au lieu d'un seul dans les SVM classiques. Amer et al. [2013] ont proposé le "Robust SVM" pour réduire l'influence des données aberrantes sur la fonction de décision du classificateur. Hoffmann [2007] a utilisé l'analyse en composantes principales à noyaux (KPCA) en étudiant l'erreur de reconstruction des échantillons par les vecteurs principaux les plus pertinents. Cette approche admet un faible coût de calcul, comparé aux SVM et SVDD, mais perd la parcimonie.

A.3.2 Méthode proposée

Dans cette thèse, nous proposons une approche de classification mono-classe. La classe normale est définie par l'hypersphère renfermant les échantillons dans l'espace de Hilbert. Nous estimons le centre de cette hypersphère sans résoudre aucun problème de programmation quadratique, en suivant le travail de [Noumir et al., 2012b]. En opposition à leur travail où la distance Euclidienne a été utilisée, nous proposons d'utiliser la distance de Mahalanobis tronquée dans l'espace transformé afin de fixer le seuil permettant de discriminer les nouveaux échantillons en données normales ou aberrantes. Cette distance tient compte de la dispersion des données [Mahalanobis, 1936].

Le centre empirique des échantillons d'apprentissage dans l'espace transformé est donné par

$$\boldsymbol{c}_n = \frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}_i).$$

Le centre $\boldsymbol{c}_n$ représente le centre de l'hypersphère dans l'espace transformé. La distance de Mahalanobis entre un échantillon $\phi(\boldsymbol{x})$ et $\boldsymbol{c}_n$ est définie comme suit :

$$\big\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\big\|_{\boldsymbol{\Sigma}}^2 = \big(\phi(\boldsymbol{x}) - \boldsymbol{c}_n\big)\boldsymbol{\Sigma}^{-1}\big(\phi(\boldsymbol{x}) - \boldsymbol{c}_n\big), \tag{A.1}$$

où $\boldsymbol{\Sigma}$ est la matrice de covariance dans l'espace transformé, ayant la forme suivante

$$\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n)(\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n)^T.$$

Sans aucune connaissance sur la fonction ϕ, la matrice de covariance ne peut pas être exprimée en fonction des échantillons dans l'espace transformé. Pour surmonter ce problème, on utilise la décomposition en valeurs singulières de la matrice de covariance $\boldsymbol{\Sigma}$ comme suit :

$$\boldsymbol{\Sigma} = \boldsymbol{V}^T\boldsymbol{D}\boldsymbol{V},$$

ayant $\boldsymbol{V}$ la matrice de vecteurs propres $\boldsymbol{v}^k$ de $\boldsymbol{\Sigma}$, et $\boldsymbol{D}$ la matrice diagonale avec les valeurs propres correspondantes λ^k pour $k = 1, 2, \ldots, n$. Chaque paire $(\boldsymbol{v}^k, \lambda^k)$ satisfait

$$\lambda^k\boldsymbol{v}^k = \boldsymbol{\Sigma}\boldsymbol{v}^k.$$

La définition de la matrice $\boldsymbol{\Sigma}$ montre que chaque vecteur propre est une combinaison linéaire des échantillons d'apprentissage $\phi(\boldsymbol{x}_i)$ dans l'espace transformé comme suit :

$$\boldsymbol{v}^k = \sum_{i=1}^{n}\alpha_i^k\big(\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n\big) = \sum_{i=1}^{n}\alpha_i^k\Big(\phi(\boldsymbol{x}_i) - \frac{1}{n}\sum_{j=1}^{n}\phi(\boldsymbol{x}_j)\Big).$$

En remplaçant $\boldsymbol{v}^k$ par cette expression dans la décomposition en valeurs singulières de la matrice de covariance $\boldsymbol{\Sigma}$, les coefficients α_i^k sont calculés en résolvant le problème de décomposition en valeurs singulières suivant :

$$n\lambda^k\boldsymbol{\alpha}^k = \widetilde{\boldsymbol{K}}\boldsymbol{\alpha}^k, \tag{A.2}$$

où la matrice $\widetilde{\boldsymbol{K}}$ est la version centrée de $\boldsymbol{K}$, et la fonction non lonéaire correspondante est donnée par $\widetilde{\phi}(\boldsymbol{x}_i) = \phi(\boldsymbol{x}_i) - \boldsymbol{c}_n$. Ensuite, l'équation (A.1) prend cette forme $\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\|_{\boldsymbol{\Sigma}}^2 = \boldsymbol{a}^T\boldsymbol{a}$ où

$$\boldsymbol{a} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{V}\Big(\phi(\boldsymbol{x}) - \frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}_i)\Big).$$

Chaque attribut a^k de $\boldsymbol{a}$, correspondant au vecteur propre $\boldsymbol{v}^k$, est calculé comme suit:

$$\begin{aligned} a^k &= (\lambda^k)^{-\frac{1}{2}} \Big(\sum_{i=1}^{n} \alpha_i^k k(\boldsymbol{x}_i, \boldsymbol{x}) - \frac{1}{n} \sum_{i,j=1}^{n} \alpha_i^k k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i^k \frac{1}{n} \sum_{j=1}^{n} k(\boldsymbol{x}_j, \boldsymbol{x}) + \sum_{i=1}^{n} \alpha_i^k \frac{1}{n^2} \sum_{j,j'=1}^{n} k(\boldsymbol{x}_j, \boldsymbol{x}_{j'}) \Big) \\ &= (\lambda^k)^{-\frac{1}{2}} \sum_{i=1}^{n} \alpha_i^k \widetilde{k}(\boldsymbol{x}_i, \boldsymbol{x}). \end{aligned}$$

Finalement, la distance de Mahalanobis de l'équation (A.1) est calculée dans l'espace transformé comme suit:

$$\big\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\big\|_{\boldsymbol{\Sigma}}^2 = \sum_{k=1}^{n} (\lambda^k)^{-1} \Big(\sum_{i=1}^{n} \alpha_i^k \widetilde{k}(\boldsymbol{x}_i, \boldsymbol{x}) \Big)^2. \tag{A.3}$$

Après avoir calculé la distance de Mahalanobis entre les échantillons d'apprentissage $\phi(\boldsymbol{x}_i)$ et le centre $\boldsymbol{c}_n$, et ayant fixé à l'avance le nombre de valeurs aberrantes n_{out} dans l'ensemble d'apprentissage, nous fixons un seuil R qui représente le rayon de l'hypersphère. La fonction de décision du classificateur considère un nouvel échantillon $\boldsymbol{x}$ comme une valeur aberrante si sa distance de Mahalanobis au centre dans l'espace transformé est supérieure à ce seuil :

$$\big\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\big\|_{\boldsymbol{\Sigma}} > R.$$

La distance de Mahalanobis utilise tous les vecteurs propres de la matrice de covariance. Nous proposons de "tronquer" cette distance, en sélectionnant un sous-ensemble de vecteurs propres. Cela correspond à la projection des échantillons dans le sous-espace engendré par ces vecteurs propres. Au lieu d'utiliser tous les vecteurs propres $\boldsymbol{v}^k$ pour l'opération de projection, nous profitons des avantages de la KPCA, où seuls les vecteurs propres associés aux plus grandes valeurs propres sont pris en considération. Le reste est considéré comme du bruit. Par conséquent, la distance de Mahalanobis est approximée par la distance de Mahalanobis tronquée dans l'espace transformé.

La performance des algorithmes de classification mono-classe dépend aussi de la dispersion hétérogène des données d'apprentissage dans l'espace transformé. Pour cela, nous adoptons également la normalisation des vecteurs propres de la matrice de covariance comme proposé dans [Tax and Juszczak, 2002], où la variance des échantillons est constante dans toutes les directions. Cette normalisation permet une transformation de l'échelle des échantillons d'apprentissage afin d'avoir une variance unitaire dans toutes les directions de l'espace transformé. Cette normalisation est obtenue comme suit:

$$(n\lambda^k)^2 \|\boldsymbol{\alpha}^k\|^2 = 1 \quad \Longrightarrow \quad \|\boldsymbol{\alpha}^k\| = \frac{1}{n\lambda^k} \quad \text{for all} \quad k = 1, 2, \ldots n.$$

A.3.3 Résultats expérimentaux

Nous allons détailler les résultats expérimentaux des méthodes de classification mono-classe sur des données simulées ainsi que sur des données réelles. Nous étudions deux noyaux RBF, le noyau Gaussien et le noyau exponentiel. En premier lieu, nous étudions l'influence de la métrique des noyaux sur la fonction de décision des classificateurs mono-classe. Nous détaillons ensuite les résultats de l'heuristique proposée pour choisir le paramètre σ. Enfin, nous présentons une comparaison entre l'approche proposée basée sur la distance de Mahalanobis tronquée et d'autres approches mono-classe bien connues, sur des données simulées et des données réelles.

A.3.3.1 Variation de la métrique des noyaux

Afin d'étudier l'influence de la métrique des noyaux sur les fonctions de décision des classificateurs, nous comparons deux méthodes de classification mono-classe bien connues, SVDD et KPCA. Nous varions la valeur du la métrique p des noyaux RBF entre $\frac{3}{4}$ et ∞. Nous appliquons ces algorithmes sur les données réelles d'un système de distribution de gaz de l'Université de Mississippi, et sur les données d'une usine de traitement d'eau de l'UCI "Machine Learning Repository". Pour plus de détails sur les données, voir Section 1.4. Les résultats sur les données du système de distribution de gaz sont illustrés dans les Figures 3.4 et 3.5. Pour toutes les valeurs de p, la fonction de décision définie l'enveloppe qui englobe les échantillons considérés comme données normales, alors que ceux considérés comme données aberrantes sont à l'extérieur de cette enveloppe.

L'utilisation de la norme ℓ_2 dans le noyau exponentiel avec l'approche SVDD (Figure 3.4) donne une enveloppe qui décrit bien l'ensemble de données d'apprentissage, tandis que la norme ℓ_∞ sur-classifie les données. Les normes ℓ_1 et $\ell_{\frac{3}{4}}$ conduisent à des enveloppes plus serrées que celle avec la norme ℓ_2. Nous avons des résultats similaires en utilisant le noyau exponentiel avec l'approche KPCA. Lorsque le noyau Gaussien est utilisé avec KPCA (Figure 3.5), les normes ℓ_2 et ℓ_1 donnent à peu près le même bon résultat, la norme $\ell_{\frac{3}{4}}$ sur-classifie les données, et les autres normes sous-classifient l'ensemble de données d'apprentissage. Nous avons aussi des résultats similaires en utilisant le noyau Gaussien avec SVDD. Nous notons que pour les valeurs de p supérieures à $p = 2$, par exemple $p = 3, 4, 7, \ldots$, les résultats de SVDD et de KPCA sont pires qu'avec les normes ℓ_1 et ℓ_2. En fait, le différent comportement des deux noyaux avec la norme infinie est illustré dans la Figure 3.6. Le premier niveau de contour du noyau Gaussien correspond aux quatre premiers niveaux de contour du noyau exponentiel. Par conséquent, quand la valeur de p augmente, les mêmes niveaux de contour deviennent plus "étendus" avec le noyau Gaussien et plus "serrés" avec le noyau exponentiel. C'est la raison pour laquelle

la norme infinie sur-classifie les données d'apprentissage avec le noyau exponentielle et les sous-classifie avec le noyau Gaussien.

Les probabilités de bonne détection des différents types de cyberattaques pour le noyau Gaussien en utilisant les normes ℓ_2 et ℓ_1 sont présentées dans les tableaux 3.1 et 3.2. La norme ℓ_1 surpasse la norme ℓ_2 dans plusieurs cas; c'est le cas des attaques "wave injection" et "slow injection" qui contiennent des petites variations simultanées des attributs, ce qui explique les meilleurs résultats de la norme ℓ_1. D'autre part, les données de l'attaque "burst injection" contiennent de grandes variations brusques des attributs, ce qui explique les meilleurs résultats de la norme ℓ_2. Les meilleurs résultats sont obtenus pour les attaques de type "single injection" et "slow injection" avec une probabilité de bonne détection de 99.25 %, alors que les résultats pour les attaques de type "wave injection" et "burst injection" ne sont pas acceptables pour les applications de sécurité avec des probabilités de bonne détection de 65% et 70% respectivement. Ces attaques ont été injectées dans les modes de fonctionnement normal du système afin d'imiter son comportement, ce qui rend leur détection très difficile. Nous obtiendrons dans la suite de meilleures probabilités de bonne détection avec l'approche proposée. Un exemple de la détection de valeurs aberrantes pour les différents types d'attaques est illustré dans la Figure 3.7.

A.3.3.2 Heuristique proposée pour choisir σ

Pour démontrer que l'heuristique proposée pour le calcul du paramètre de la bande passante σ conduit à un très bon résultat sans aucun coût calculatoire, nous comparons l'heuristique proposée à trois méthodes existantes dans la littérature:

- La validation croisée "5 fold" divise l'ensemble de données d'apprentissage en 5 sous-ensembles utilisés pour l'apprentissage et la validation. Les valeurs du paramètre σ suivent une suite géométrique de facteur 2 : $[0.5, 1, 2, \ldots, 1024]$.
- Le large intervalle proposé dans [Soares et al., 2004] qui choisit une valeur pour σ parmi une suite géométrique de facteur 4 : $[0.25, 1, 4, 16, 64, 256, 1000, 4000, 16000, 64000, 256000]$.
- L'intervalle restreint proposé dans [Cherkassky and Ma, 2004] qui dépend des échantillons de l'ensemble d'apprentissage. Ces échantillons sont normalisés, et le paramètre σ prend une des valeurs suivantes : $[0.1, 0.2, 0.3, 0.4, 0.5]$.

Nous étudions la variation de la probabilité de bonne détection et de la probabilité de fausse alarme en fonction de la valeur du paramètre de bande passante σ, en utilisant

le noyau Gaussien avec la norme ℓ_2. Les résultats sont illustrés dans la Figure 3.8. L'intervalle de bonnes valeurs pour ce paramètre est compris entre 0.9 et 1.5, ce qui représente le meilleur compromis entre un taux de bonne détection élevé et un taux de fausse alarme relativement faible. L'heuristique proposée conduit à $\sigma = 0,94$ qui appartient à cet intervalle. Ces bons résultats obtenus avec l'heuristique proposée confirment sa pertinence.

En outre, le temps estimé de chaque approche pour calculer le paramètre de la bande passante est donné dans le tableau 3.4. Ces résultats sont obtenus sur les données réelles du système de distribution de gaz. L'heuristique proposée est clairement centaines de fois plus rapide que les autres approches. En fait, le calcul de σ dans l'heuristique proposée dépend du nombre des échantillons de l'ensemble d'apprentissage, de la distribution de ces échantillons et de la fraction de vecteurs de support, mais il est indépendant de l'algorithme mono-classe utilisé. Pour les autres approches, SVDD nécessite plus de temps que KPCA pour calculer σ car elle nécessite la résolution d'un problème de programmation quadratique avec contraintes.

A.3.3.3 Les approches de classification mono-classe

Nous présentons une comparaison entre l'approche proposée basée sur la distance de Mahalanobis tronquée et d'autres approches mono-classe bien connues, comme SVDD [Tax and Duin, 2004], KPCA [Schölkopf et al., 1998b], simple one-class [Noumir et al., 2012b], slab SVM [Eigensatz et al., 2008] et robust SVM [Amer et al., 2013], sur des données simulées et des données réelles.. Nous utilisons dans les simulations le noyau Gaussien avec son expression standard, $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma^2})$.

Dans un premier temps, les algorithmes de classification mono-classe sont appliqués sur deux ensembles de données simulées, les données en forme d'une sinusoïde et les données en forme d'un carré [Hoffmann, 2007]. L'objectif principal est de définir une enveloppe qui suit la distribution de l'ensemble des échantillons d'apprentissage, tout en évitant les cas extrêmes de sur-classification et sous-classification des données. Nous notons que l'ensemble de données en forme d'une sinusoïde contient 95 échantillons, et celui en forme d'un carré 450 échantillons y compris les échantillons bruités. Les paramètres de régularisation de toutes les approches mono-classe sont fixés à 10% de l'ensemble des échantillons d'apprentissage. En outre, le nombre des vecteurs propres associés aux plus grandes valeurs propres de la matrice de covariance pris en compte pour l'opération de projection dans KPCA et dans l'approche proposée est fixé à 40. Les résultats pour les différentes approches sont présentés dans les Figures 3.9 et 3.10. Les meilleurs résultats sur les deux ensembles de données simulées sont obtenus avec

l'approche proposée, menant aux enveloppes les plus serrées qui suivent la distribution des échantillons d'apprentissage de la meilleure façon possible. L'approche KPCA conduit à un bon résultat, alors que SVDD, simple one-classe, slab SVM et robust SVM donnent des enveloppes lâches qui ne décrivent pas la distribution des échantillons d'apprentissage. Le bon résultat de l'algorithme proposé est dû aux fortes propriétés de la distance de Mahalanobis tronquée estimée par la projection dans le sous-espace engendré par les vecteurs propres associés aux plus grandes valeurs propres de la matrice de covariance.

Dans un deuxième temps, nous appliquons ces algorithmes sur les données réelles d'un système de distribution de gaz et d'un réservoir de stockage d'eau de l'Université de Mississippi, et sur les données d'une usine de traitement d'eau de l'UCI "Machine Learning Repository". Ces données contiennent des échantillons qui correspondent à plusieurs types d'attaques. Pour plus de détails sur les données, voir Section 1.4.

Le premier critère pour les algorithmes de classification mono-classe est leur capacité à détecter les échantillons aberrants qui n'existent pas dans l'ensemble d'apprentissage. Ces algorithmes sont testés sur près de 100 000 échantillons liés aux différents types d'attaques, et les taux de bonne détection sur les données réelles sont donnés dans les tableaux 3.5, 3.6 et 3.7. Les taux de détection les plus mauvais sont obtenus avec l'approche simple one-classe simple, ce qui peut être expliqué par sa sensibilité à la présence des échantillons aberrants parmi l'ensemble d'apprentissage. L'approche proposée surpasse toutes les autres approches, et donne les meilleurs taux de bonne détection pour les différents types d'attaques. Par rapport au tau de fausse alarme, toutes les approches ont un taux de fausse alarme autour de 1%.

Un autre critère pour les approches de classification mono-classe est la complexité des algorithmes. Le temps d'apprentissage estimé pour chaque approche est présenté dans le tableau 3.8, et le temps estimé pour tester chaque nouvel échantillon dans le tableau 3.9. Ces résultats montrent que l'approche simple one-classe est la plus rapide, mais avec les résultats les plus mauvais. L'approche proposée est deux fois plus rapide que KPCA, jusqu'à 10 fois plus rapide que SVDD et 50 fois plus rapide que slab SVM. En Plus, l'approche proposée est la plus rapide pour tester un nouvel échantillon, et elle est jusqu'à 4.5 fois plus rapide que SVDD. Ces résultats sont très importants ; l'approche proposée conduit aux meilleurs taux de bonne détection, et elle est jusqu'à 50 fois plus rapide que les autres approches mono-classe.

A.4 Representation parcimonieuse pour classification mono-classe

La complexité des méthodes de classification mono-classe existantes pour élaborer les fonctions de décision des classificateurs est souvent très élevée. Certaines approches ont besoin de résoudre un problème de programmation quadratique avec contraintes, tandis que d'autres nécessitent l'optimisation d'un problème de programmation conique du second ordre. La complexité de ces algorithmes croît avec le nombre des échantillons de l'ensemble d'apprentissage. D'où la nécessité de réduire la complexité des algorithmes, par exemple avec des représentations parcimonieuses [Tropp and Wright, 2010]. L'objectif principal de la théorie de l'approximation parcimonieuse des données est de décrire et d'approximer un signal à partir de quelques autres signaux élémentaires d'un ensemble fixe, connu sous le nom du dictionnaire [Elad, 2010]. En général, le choix d'un dictionnaire pertinent se fait soit par la construction d'un dictionnaire basé sur un modèle mathématique des données, soit par apprendre un dictionnaire à partir d'un ensemble d'apprentissage [Rubinstein et al., 2010].

Certaines méthodes de classification mono-classe reposent sur des représentations parcimonieuses. Dans l'approche SVDD, le centre de l'hypersphère contenant les échantillons est une combinaison linéaire des vecteurs de support. Ces vecteurs de support représentent une petite fraction de l'ensemble de données d'apprentissage. Dans le SVM mono-classe, l'hyperplan séparant les échantillons de l'origine avec une marge maximale dépend également des vecteurs de support. En outre, les hyperplans dans le slab SVM et robust SVM sont définis par des combinaisons linéaires des vecteurs de support. Bien que ces approches offrent des solutions parcimonieuses, leur complexité reste élevée, car elles doivent résoudre des problèmes de programmation quadratique avec contraintes.

Nous proposons dans cette Section deux approches de classification mono-classe qui reposent sur des représentations parcimonieuses. La première approche est une formulation parcimonieuse de l'approche basée sur la distance de Mahalanobis tronquée proposée dans la section précédente. Quant à la deuxième approche, nous revisitons les méthodes de sélection de variables bien connues, dont LARS (Least Angle Regression) [Efron et al., 2004], LASSO (Least Absolute Shrinkage and Selection Operator) [Tibshirani, 1996; Osborne et al., 1999], et Elastic Net [Zou and Hastie, 2005; Zhou, 2013], en adaptant leurs algorithmes pour l'estimation parcimonieuse du centre de l'hypersphère en classification mono-classe. L'objectif principal des approches proposées est d'obtenir un modèle parcimonieux pertinent en utilisant seulement une petite fraction de l'ensemble des échantillons d'apprentissage. Nous proposons également une approche de classification mono-classe en ligne qui repose sur un modèle parcimonieux. Le classificateur est

mis à jour d'une manière séquentielle à chaque instant, où chaque nouvel échantillon est pris en compte afin de mettre à jour sa fonction de décision.

A.4.1 Modèle parcimonieux

La moyenne des échantillons dans l'espace transformé a la forme suivante :

$$\mathrm{E}[\phi(\boldsymbol{x})] = \int_{\mathcal{X}} \phi(\boldsymbol{x}) P(\boldsymbol{x}) d\boldsymbol{x},$$

ayant $P(\boldsymbol{x})$ la probabilité de la distribution des échantillons d'apprentissage sur $\mathcal{X}$. Puisque la distribution $P(\boldsymbol{x})$ est généralement inconnue, nous pouvons estimer cette moyenne par le centre empirique de l'ensemble des échantillons d'apprentissage dans l'espace transformé comme suit :

$$\boldsymbol{c}_n = \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i).$$

Nous proposons d'approximer le centre empirique en utilisant une petite fraction des échantillons d'apprentissage, où seuls ces échantillons sont pris en compte dans les fonctions de décision des classificateurs. Ce centre parcimonieux est une combinaison linéaire d'un ensemble d'échantillons, à savoir:

$$\boldsymbol{c}_{\mathcal{A}} = \sum_{j=1}^{n} \beta_j \phi(\boldsymbol{x}_j),$$

où seule une petite fraction des coefficients β_j dans l'expression du centre est non nulle:

$$\boldsymbol{c}_{\mathcal{A}} = \sum_{j \in \mathcal{A}} \beta_j \phi(\boldsymbol{x}_j),$$

où $\mathcal{A} \subset \{1, 2, \ldots, n\}$. Le centre $\boldsymbol{c}_{\mathcal{A}}$ doit représenter la distribution de l'ensemble des échantillons d'apprentissage. Pour cela, nous définissons $\boldsymbol{c}_{\mathcal{A}}$ par l'approximation du centre empirique $\boldsymbol{c}_n$. Afin d'estimer le centre parcimonieux, nous proposons de minimiser l'erreur d'approximation du centre empiriques $\boldsymbol{c}_n$ avec $\boldsymbol{c}_{\mathcal{A}}$. Dans la suite, nous détaillons les approches de classification mono-classe basées sur des représentations parcimonieuses.

A.4.2 Approche basée sur la distance de Mahalanobis tronquée

Afin d'obtenir une approche parcimonieuse qui réduit la complexité de l'algorithme tout en maintenant une bonne enveloppe autour des échantillons de l'ensemble

d'apprentissage, nous considérons une formulation parcimonieuse de l'approche proposée dans la section précédente. La classe normale est définie par l'hypersphère renfermant les échantillons dans l'espace transformé. Pour estimer le centre parcimonieux de l'hypersphère, nous proposons d'approximer le centre estimé $\boldsymbol{c}_n$ en utilisant des vecteurs de support, par analogie à l'approche SVDD où les vecteurs de support sont les échantillons d'apprentissage qui se trouvent sur et à l'extérieur de l'hypersphère. Puisque le centre parcimonieux est une combinaison linéaire de ces vecteurs de support, seuls ces échantillons sont pris en compte dans l'estimation de la distance de Mahalanobis tronquée dans le RKHS. La sélection des vecteurs de support est basée sur des critères de parcimonie, tels que le critère de la distance ou le critère de cohérence. Nous présentons brièvement ces deux critères avant de décrire en détail la formulation parcimonieuse.

A.4.2.1 Choix des vecteurs de support

Plusieurs critères de parcimonie ont été proposés dans la littérature pour la sélection des vecteurs de support, afin de définir des dictionnaires pertinents tout en gardant une faible complexité de calcul. Dans ce qui suit, nous présentons deux critères de parcimonie bien connus, le critère de cohérence étudié dans [Honeine et al., 2007; Richard et al., 2009] pour le filtrage adaptatif non linéaire et le critère de distance [Noumir et al., 2012b; Noumir, 2012] inspiré de l'approche SVDD.

Le critère de cohérence d'un ensemble d'apprentissage est défini par la plus grande valeur absolue de leurs produits scalaires :

$$\mu = \max_{i \neq j} |\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}|,$$

ce qui correspond aux valeurs des éléments non-diagonaux les plus grandes de la matrice de Gram[1] :

$$\mu = \max_{i \neq j} |k(\boldsymbol{x}_i, \boldsymbol{x}_j)|.$$

Il est donc naturel de considérer l'ensemble avec la cohérence la plus faible comme étant l'ensemble pertinent de vecteurs de support.

Le critère de distance dépend de la distance Euclidienne dans l'espace transformé entre les échantillons et le centre estimé:

$$\big\|\phi(\boldsymbol{x}) - \boldsymbol{c}_n\big\|_{\mathcal{H}}^2 = k(\boldsymbol{x}, \boldsymbol{x}) - \frac{2}{n} \sum_{j=1}^{n} k(\boldsymbol{x}, \boldsymbol{x}_j) + \frac{1}{n^2} \sum_{i,j=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

[1] Cette définition correspond à un noyau de norme unité, c'est-à-dire $k(\boldsymbol{x}, \boldsymbol{x}) = 1$ pour tout $\boldsymbol{x} \in \mathcal{X}$; dans le cas général, il suffit de remplacer $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ par $\frac{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{k(\boldsymbol{x}_i, \boldsymbol{x}_j) k(\boldsymbol{x}_i, \boldsymbol{x}_j)}}$ dans cette définition.

L'ensemble $\mathcal{A}$ contenant les indices des échantillons les plus éloignés du centre est donné comme suit:

$$\mathcal{A} = \left\{ i, \quad \left\| \phi(\boldsymbol{x}_i) - \boldsymbol{c}_n \right\|_{\mathcal{H}}^2 > R^2 \right\},$$

ayant R le rayon/seuil de l'hypersphère. Cet ensemble définit l'ensemble des vecteurs de support. Nous notons que, indépendamment du critère de parcimonie utilisé pour la sélection des vecteurs de support, l'algorithme suivant reste inchangé.

A.4.2.2 Formulation parcimonieuse

Afin d'obtenir une formulation parcimonieuse de l'approche proposée dans la section précédente, l'approximation du centre $\boldsymbol{c}_n$ par le centre parcimonieux $\boldsymbol{c}_{\mathcal{A}}$ est réalisée par la minimisation de la distance de Mahalanobis entre $\boldsymbol{c}_n$ et $\boldsymbol{c}_{\mathcal{A}}$ comme suit :

$$\arg\min_{\beta_i} \left\| \frac{1}{n} \sum_{l=1}^{n} \phi(\boldsymbol{x}_l) - \sum_{i \in \mathcal{A}} \beta_i \phi(\boldsymbol{x}_i) \right\|_{\boldsymbol{\Sigma}}^2.$$

En annulant la dérivée partielle de cette fonction coût par rapport à chaque β_i, l'expression suivante est mise à zéro :

$$(\lambda^k)^{-\frac{1}{2}} \Big(\tfrac{1}{n} \sum_{l=1}^{n} \sum_{k \in \mathcal{A}} k(\boldsymbol{x}_l, \boldsymbol{x}_k) - \sum_{j,k \in \mathcal{A}} \beta_j k(\boldsymbol{x}_j, \boldsymbol{x}_k) \Big) \sum_{r=1}^{n} \alpha_r^k (\phi(\boldsymbol{x}_r) - \boldsymbol{c}_n).$$

Nous obtenons

$$\frac{1}{n} \sum_{l=1}^{n} \sum_{k \in \mathcal{A}} k(\boldsymbol{x}_l, \boldsymbol{x}_k) = \sum_{j,k \in \mathcal{A}} \beta_j k(\boldsymbol{x}_j, \boldsymbol{x}_k).$$

Les coefficients β_i sont calculés à partir de la notation matricielle suivante :

$$\boldsymbol{\beta} = \boldsymbol{K}_{\mathcal{A}}^{-1} \boldsymbol{k}, \tag{A.4}$$

où $\boldsymbol{K}_{\mathcal{A}}$ est la matrice noyau d'éléments $k(\boldsymbol{x}_j, \boldsymbol{x}_k)$ pour $j, k \in \mathcal{A}$, et $\boldsymbol{k}$ un vecteur colonne dont les éléments sont

$$\frac{1}{n} \sum_{k \in \mathcal{A}} k(\boldsymbol{x}_l, \boldsymbol{x}_k), \quad \text{pour} \quad l = 1, \ldots, n.$$

Nous fixons un seuil R en se basant sur le nombre prédéfini des échantillons aberrants n_{out}. La fonction de décision pour tout nouvel échantillon $\boldsymbol{x}$ est l'évaluation de la

distance de Mahalanobis entre $\phi(\boldsymbol{x})$ et le centre parcimonieux $\boldsymbol{c}_{\mathcal{A}}$ comme suit :

$$\left\|\phi(\boldsymbol{x})-\boldsymbol{c}_{\mathcal{A}}\right\|_{\boldsymbol{\Sigma}}^2=\sum_{k=1}^{m}\frac{1}{\lambda^k}\Big(\sum_{i=1}^{n}\alpha_i^k k(\boldsymbol{x}_i,\boldsymbol{x})-\sum_{i=1}^{n}\sum_{j\in\mathcal{A}}\alpha_i^k\beta_j k(\boldsymbol{x}_i,\boldsymbol{x}_j) - \sum_{i=1}^{n}\alpha_i^k\frac{1}{n}\sum_{j=1}^{n}k(\boldsymbol{x}_j,\boldsymbol{x})+\sum_{i=1}^{n}\alpha_i^k\frac{1}{n}\sum_{j=1}^{n}\sum_{l\in\mathcal{A}}\beta_l k(\boldsymbol{x}_j,\boldsymbol{x}_l)\Big)^2.$$

Si cette distance est supérieur au rayon R, l'échantillon est considéré comme une valeur aberrante; Dans le cas contraire, il est considéré comme un échantillon normal. De même que pour l'approche proposée dans la section précédente, la distance de Mahalanobis ci-dessus est tronquée en projetant les échantillons d'apprentissage dans le sous-espace engendré par les vecteurs propres associés aux plus grandes valeurs propres de la matrice de covariance.

A.4.3 Approche basée sur des méthodes de sélection de variables

L'approche précedente repose sur la séparation entre le critère de parcimonie et l'approximation du centre de l'hypersphère. La deuxième approche que nous proposons dans cette section considère la résolution de ces deux problèmes d'une manière simultanée, à savoir la sélection des vecteurs de support et l'approximation du centre parcimonieux. Le problème d'optimisation consiste à minimiser l'erreur d'approximation de $\boldsymbol{c}_n$ par $\boldsymbol{c}_{\mathcal{A}}$, selon

$$\arg\min_{\boldsymbol{\beta}}\Big\|\frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}_i)-\sum_{j=1}^{n}\beta_j\phi(\boldsymbol{x}_j)\Big\|_{\mathcal{H}}^2, \tag{A.5}$$

sous des contraintes de parcimonie. Pour résoudre ce problème d'optimisation, nous proposons de revisiter des méthodes de séléction de variables, comme LARS (Least Angle Regression), LASSO (Least Absolute Shrinkage and Selection Operator) et Elastic Net. Ces méthodes ont été utilisées pour résoudre des problèmes ayant la forme générale $\arg\min_{\boldsymbol{\beta}}\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}\|_2^2$ sous des contraintes de sparcimonie, où une fraction des coefficients restent non nulle. Dans la suite, nous modifions ces algorithmes pour estimer le centre parcimonieux $\boldsymbol{c}_{\mathcal{A}}$, en les adaptant pour la résolution du problème d'optimisation (A.5) afin de retenir seulement les échantillons les plus pertinents. Nous définissons la fonction de décision d'un échantillon $\phi(\boldsymbol{x})$ par sa distance Euclidenne au centre $\boldsymbol{c}_{\mathcal{A}}$, et nous fixons un seuil pour classer les nouvels échantillons comme données normales ou aberrantes. Cette distance dans l'espace de Hilbert est donnée par:

$$\|\phi(\boldsymbol{x})-\boldsymbol{c}_{\mathcal{A}}\|_2^2=k(\boldsymbol{x},\boldsymbol{x})-2\sum_{i=1}^{n}\beta_i k(\boldsymbol{x}_i,\boldsymbol{x})+\sum_{i,j=1}^{n}\beta_i\beta_j k(\boldsymbol{x}_i,\boldsymbol{x}_j).$$

Dans ce qui suit, nous détaillons les méthodes de sélection de variables modifiées, en revisitant les problèmes d'optimisation correspondants et les solutions résultantes.

A.4.3.1 LARS

LARS (Least Angle Regression) est un algorithme de sélection qui construit un modèle d'une manière séquentielle, en ajoutant à l'ensemble des échantillons les plus pertinents un échantillon à chaque itération. La modification de l'algorithme LARS afin de l'adapter à la classification mono-classe conduit à la résolution du problème d'optimisation suivant :

$$\arg\min_{\boldsymbol{\beta}} \|\frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}_i) - \sum_{j=1}^{n}\beta_j\phi(\boldsymbol{x}_j)\|_{\mathcal{H}}^2, \tag{A.6}$$

sous la contrainte $\sum|\beta| < t$, pour un certain paramètre t. Soient $\widehat{\boldsymbol{c}}_{\mathcal{A}_k}$ l'estimation du centre parcimonieux à l'étape k, $\mathcal{A}_k$ l'ensemble contenant les indices des échantillons les plus pertinents, et $(\boldsymbol{c}_n - \widehat{\boldsymbol{c}}_{\mathcal{A}_k})$ le résidu à l'étape k. Dans une formulation *à la* LARS, les échantillons sont projetés sur le plus corrélé avec le résidu $(\boldsymbol{c}_n - \widehat{\boldsymbol{c}}_{\mathcal{A}_k})$. Le processus est répété jusqu'à ce qu'un nouvel échantillon ait ce même niveau de corrélation avec le résidu. Ensuite, LARS poursuit l'estimé de $\boldsymbol{c}_{\mathcal{A}}$ dans une direction équiangulaire entre les échantillons les plus pertinents, et ainsi de suite.

Soient $\boldsymbol{X} = (\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \ldots, \phi(\boldsymbol{x}_n))$ et $\boldsymbol{X}_{\mathcal{A}}$ réprésente les échantillons les plus pertinents de l'ensemble $\mathcal{A}$. L'expression de l'estimé de $\boldsymbol{c}_{\mathcal{A}}$ est

$$\widehat{\boldsymbol{c}}_{\mathcal{A}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}},$$

où l'algorithme commence par $\widehat{\boldsymbol{c}}_{\mathcal{A}_0} = \boldsymbol{0}$. Le vecteur de corrélations est défini par :

$$\widehat{\boldsymbol{corr}} = \boldsymbol{X}^T(\boldsymbol{c}_n - \widehat{\boldsymbol{c}}_{\mathcal{A}}) = \frac{1}{n}\sum_{i,j=1}^{n}k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i,j=1}^{n}\widehat{\beta}_j k(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

et le vecteur assurant la direction équiangulaire possède la forme suivante:

$$\boldsymbol{u}_{\mathcal{A}} = \boldsymbol{X}_{\mathcal{A}}\boldsymbol{w}_{\mathcal{A}},$$

où $\boldsymbol{w}_{\mathcal{A}}$ est le vecteur des poids relatifs aux échantillons de $\mathcal{A}$. L'estimé du centre $\widehat{\boldsymbol{c}}_{\mathcal{A}}$ est mis à jour suivant le vecteur équiangulaire pour devenir:

$$\widehat{\boldsymbol{c}}_{\mathcal{A}+} = \widehat{\boldsymbol{c}}_{\mathcal{A}} + \widehat{\gamma}\boldsymbol{u}_{\mathcal{A}},$$

où $\widehat{\gamma}$ est le plus petit réel positif pour qu'un nouvel échantillon intègre l'ensemble $\mathcal{A}$. Finalement, les coefficients sont mis à jour de la manière suivante:

$$\boldsymbol{\beta}_{new} = \widehat{\boldsymbol{\beta}} + \widehat{\gamma}\boldsymbol{s}^T\boldsymbol{w}_{\mathcal{A}}, \tag{A.7}$$

ayant $\boldsymbol{s}$ le vecteur contenant les signes des corrélations. Cette méthode hérite un majeur inconvénient de LARS, qui est la présence d'échantillons à grande auto-corrélation. Les deux méthodes suivantes permettent de surmonter ce défaut.

A.4.3.2 LASSO

Le deuxième algorithme modifié et adapté à la classification mono-classe est le LASSO (Least Absolute Shrinkage and Selection Operator). En s'inspirant du LASSO, on minimise l'erreur entre $\boldsymbol{c}_n$ et $\boldsymbol{c}_{\mathcal{A}}$ avec une régularisation de type norme ℓ_1, selon:

$$\arg\min_{\beta_j} \Big\| \frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}_i) - \sum_{j=1}^{n}\beta_j\phi(\boldsymbol{x}_j)\Big\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{A.8}$$

pour un paramètre $\lambda > 0$. La norme ℓ_1 impose une certaine parcimonie à la solution du problème.

Une modification de l'algorithme de LARS peut aboutir à toutes les solutions de LASSO. En effet, contrairement à LARS, les coefficients dans LASSO ne peuvent pas changer de signe au cours de l'étape de la mise à jour car ils sont linéaires par morceaux. De plus, si à une étape k un coefficient change de signe pendant la mise à jour, cette condition n'est plus vérifiée. L'échantillon correspondant doit être retiré de l'ensemble des plus pertinents, à savoir

$$\mathcal{A}_{k+1} = \mathcal{A}_k \setminus \{j\},$$

et l'algorithme continue sa recherche de la direction équiangulaire suivante. Par conséquent, cette modification permet à l'ensemble $\mathcal{A}$ d'augmenter ou de diminuer un échantillon à la fois, jusqu'à ce que LARS conduit à toutes les solutions de LASSO. Cet algorithme hérite des inconvénients du LASSO classique. Le principal inconvénient reste avec les échantillons fortement corrélés, où LASSO tend à sélectionner arbitrairement un échantillon et ignore les autres.

A.4.3.3 Elastic Net

L'Elastic net est un algorithme dérivé du LARS, et il minimise l'erreur en pénalisant la norme ℓ_1 comme dans LASSO, mais aussi la norm ℓ_2 comme dans la regression ridge.

L'Elastic net permet de tenir en compte des groupes d'échantillons qui sont fortement corrélés, contrairement au LASSO. Le problème d'optimisation du *naïve* Elastic net est le suivant:

$$\arg\min_{\beta_j} \Big\| \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) - \sum_{j=1}^{n} \beta_j \phi(\boldsymbol{x}_j) \Big\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2. \tag{A.9}$$

Ce problème d'optimisation doublement pénalisé induit un biais supplémentaire inutile, en le comparant à celui du LASSO ou de la regression ridge. Afin d'améliorer les performances de prédiction et d'annuler ce biais, les coefficients de la version *naïve* de l'Elastic Net sont décalés pour obtenir les coefficients Elastic net comme suit:

$$\boldsymbol{\beta}_{(\text{Elastic Net})} = (1 + \lambda_2)\boldsymbol{\beta}_{(\text{naïve Elastic Net})}.$$

Nous notons que le problème LASSO peut être transformé en un problème Elastic Net en remplaçant le paramètre λ par [Zou and Hastie, 2005]

$$\frac{\lambda_1}{\sqrt{1 + \lambda_2}}.$$

Puisque les solutions de LASSO sont obtenues par la modification de LARS, comme détaillé dans le paragraphe précédant, LARS conduit à toutes les solutions de l'Elastic Net.

A.4.3.4 Extension à la norme de Mahalanobis

Dans les cas où les échantillons d'apprentissage contiennent des attributs hétérogènes, la norme Euclidienne devient sensible aux différentes échelles des attributs. Pour cela, nous proposons de remplacer la distance Euclidienne dans la fonction de décision du classificateur par la distance de Mahalanobis qui prend en considération la dispersion des échantillons de l'ensemble d'apprentissage. Le calcul de cette distance entre un échantillon $\phi(\boldsymbol{x})$ et le centre $\boldsymbol{c}_{\mathcal{A}}$ se fait comme suit :

$$\begin{aligned} \big\|\phi(\boldsymbol{x}) - \boldsymbol{c}_{\mathcal{A}}\big\|_{\boldsymbol{\Sigma}}^2 = \sum_{k=1}^{m} \frac{1}{\lambda^k} \Big(& \sum_{i=1}^{n} \alpha_i^k k(\boldsymbol{x}_i, \boldsymbol{x}) - \sum_{i=1}^{n} \sum_{j \in \mathcal{A}} \alpha_i^k \beta_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ & - \sum_{i=1}^{n} \alpha_i^k \frac{1}{n} \sum_{j=1}^{n} k(\boldsymbol{x}_j, \boldsymbol{x}) + \sum_{i=1}^{n} \alpha_i^k \frac{1}{n} \sum_{j=1}^{n} \sum_{l \in \mathcal{A}} \beta_l k(\boldsymbol{x}_j, \boldsymbol{x}_l) \Big)^2, \end{aligned}$$

où $n\lambda_k$ et $\boldsymbol{\alpha}^k$ sont les valeurs et vecteurs propres de la matrice de Gram centrée. De même que pour l'approche proposée dans la section précédente, la distance de Mahalanobis ci-dessus est tronquée en projetant les échantillons d'apprentissage dans le sous-espace engendré par les vecteurs propres associés aux plus grandes valeurs propres

de la matrice de covariance. Donc, la distance de Mahalanobis est estimée par la distance de Mahalanobis tronquée.

A.4.4 Classification mono-classe en ligne

Les méthodes de classification mono-classe proposées jusqu'ici dans cette thèse sont considérées comme des techniques d'apprentissage hors ligne, puisque tous les échantillons d'apprentissage sont disponibles dès le début. L'estimation de la règle de décision du classificateur ne change pas après la phase d'apprentissage initiale. De nombreuses applications de la vie réelle peuvent être considérées comme des problèmes d'apprentissage en ligne. La différence principale entre les techniques d'apprentissage hors ligne et celles en ligne est que, dans l'apprentissage en ligne, la règle de décision du classificateur est mise à jour après l'arrivée de chaque nouvel échantillon. Les chercheurs ont été confrontés à de nombreux défis pour élaborer des algorithmes de classification mono-classe en ligne, notamment pour réduire la complexité de l'algorithme, tout en améliorant la probabilité de bonne détection et en réduisant au minimum le taux de fausse alarme.

Afin de surmonter les inconvénients des méthodes existantes, nous proposons une approche de classification mono-classe en ligne, qui est une version étendue de l'approche parcimonieuse proposée dans la section précédente. Nous modifions la fonction de décision du classificateur pour l'adapter aux applications en ligne, en définissant deux hypersphères concentriques renfermant les vecteurs de support. Le premier avantage de l'utilisation de deux hypersphères au lieu d'une seule est l'isolation des échantillons aberrants en dehors de cette enveloppe, sans les prendre en compte dans l'étape de la mise à jour en ligne du classificateur, ce qui rend l'algorithme proposé plus robuste aux échantillons aberrants. Un autre avantage de cette approche est le petit nombre de vecteurs de support, ce qui réduit les coûts de calcul de l'algorithme. Une représentation des deux hypersphères concentriques est illustrée dans la Figure 4.3. Nous fixons deux seuils, $R_{detection}$ et R_{sparse}, afin de tester les nouveaux échantillons et de détecter les valeurs aberrantes dans la phase en ligne. Le premier seuil est fixé en fonction du nombre prédéfini des échantillons aberrants, et R_{sparse} dépend du nombre estimé de vecteurs de support. Cette approche peut être divisée en deux phases principales: une phase d'apprentissage hors ligne et une phase de détection/mise à jour en ligne.

A.4.4.1 Phase d'apprentissage hors ligne

Dans la phase d'apprentissage hors ligne, nous apprenons les modes de fonctionnement normal du système. Nous approximons le centre $\boldsymbol{c}_n$ par le centre parcimonieux $\boldsymbol{c}_{\mathcal{A}}$ qui dépend seulement d'une fraction des échantillons d'apprentissage, les vecteurs de

support. Seuls ces échantillons sont pris en compte dans la fonction de décision du classificateur. L'ensemble des vecteurs de support est donné par

$$\mathcal{A} = \left\{ i, R_{sparse} < \|\phi(\boldsymbol{x}_i) - \boldsymbol{c}_n\|_{\boldsymbol{\Sigma}} \leq R_{detection} \right\}.$$

Le centre parcimonieux est une combinaison linéaire des vecteurs de support, notamment $\boldsymbol{c}_{\mathcal{A}} = \sum_{i \in \mathcal{A}} \beta_i \phi(\boldsymbol{x}_i)$. Nous minimisons l'erreur d'approximation de $\boldsymbol{c}_n$ par $\boldsymbol{c}_{\mathcal{A}}$ selon

$$\arg\min_{\beta_j} \| \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) - \sum_{j=1}^{n} \beta_j \phi(\boldsymbol{x}_j) \|_2^2.$$

Les coefficients β_j sont claculés à partir de la notation matricielle $\boldsymbol{\beta} = \boldsymbol{K}_{\mathcal{A}}^{-1} \boldsymbol{k}$, où $\boldsymbol{K}_{\mathcal{A}}$ est la matrice noyau d'éléments $k(\boldsymbol{x}_j, \boldsymbol{x}_k)$ pour $j, k \in \mathcal{A}$, et $\boldsymbol{k}$ un vecteur colonne dont les éléments sont $\frac{1}{n} \sum_{k \in \mathcal{A}} k(\boldsymbol{x}_l, \boldsymbol{x}_k)$ pour $l = 1, \ldots, n$. La distance de Mahalanobis tronquée est utilisée dans la fonction de décision du classificateur.

A.4.4.2 Phase de détection/mise à jour en ligne

Dans la phase en ligne, nous avons un nouvel échantillon à chaque instant. Le classificateur teste chaque nouvel échantillon $\boldsymbol{x}_t$, pour tout $t > n$, en calculant sa distance de Mahalanobis tronquée au centre $\boldsymbol{c}_{\mathcal{A}}$. Nous pouvons rencontrer trois cas possibles:

1. **Cas 1:** $\|\phi(\boldsymbol{x}_t) - \boldsymbol{c}_{\mathcal{A}}\|_{\boldsymbol{\Sigma}} > R_{detection}$
   Dans ce cas, le nouvel échantillon est considéré comme une valeur aberrante et une alarme est activée. Le classificateur ne doit pas mettre à jour sa fonction de décision.

2. **Cas 2:** $R_{sparse} < \|\phi(\boldsymbol{x}_t) - \boldsymbol{c}_{\mathcal{A}}\|_{\boldsymbol{\Sigma}} \leq R_{detection}$
   Dans ce cas, cet échantillon est considéré comme un vecteur de support, et son indice est inclus dans l'ensemble $\mathcal{A}$. Le nombre de vecteurs de support est incrémenté, et la nouvelle matrice du Gram est mise à jour à partir de $\boldsymbol{K}_{\mathcal{A}}$ comme suit:

   $$\boldsymbol{K}_{\mathcal{A}new} = \begin{bmatrix} \boldsymbol{K}_{\mathcal{A}} & \boldsymbol{b} \\ \boldsymbol{b}^T & k(\boldsymbol{x}_t, \boldsymbol{x}_t) \end{bmatrix},$$

   où $\boldsymbol{b}$ est le vecteur colonne dont les éléments sont $k(\boldsymbol{x}_i, \boldsymbol{x}_t)$ pour tout $i \in \mathcal{A}$. Le vecteur $\boldsymbol{k}$ est mis à jour selon

   $$\boldsymbol{k}_{new} = \frac{1}{t} \begin{bmatrix} (t-1)\boldsymbol{k} + \boldsymbol{b} \\ k_t \end{bmatrix},$$

ayant $k_t = \sum_{i=1}^{t} k(\boldsymbol{x}_i, \boldsymbol{x}_t)$, et les coefficients $\boldsymbol{\beta}_t$ selon $\boldsymbol{\beta}_t = \boldsymbol{K}_{\mathcal{A}new}^{-1}\boldsymbol{k}_{new}$.

3. **Cas 3:** $\|\phi(\boldsymbol{x}_t) - \boldsymbol{c}_{\mathcal{A}}\|_{\boldsymbol{\Sigma}} \leq R_{sparse}$
   Dans ce cas, le nouvel échantillon n'est pas un vecteur de support. La matrice $\boldsymbol{K}_{\mathcal{A}}$ reste inchangée. Seuls le vecteur $\boldsymbol{k}$ est mis à jour selon

$$\boldsymbol{k}_{new} = \frac{1}{t}\big((t-1)\boldsymbol{k} + \boldsymbol{b}\big),$$

   et les coefficients $\boldsymbol{\beta}_t$ selon

$$\boldsymbol{\beta}_{new} = \frac{t-1}{t}\boldsymbol{\beta} + \frac{1}{t}\boldsymbol{K}_{\mathcal{A}}^{-1}\boldsymbol{b}.$$

A.4.5 Résultats expérimentaux

Dans cette section, nous présentons les résultats expérimentaux des méthodes de classification mono-classe parcimonieuses. Dans un premier temps, nous présentons les résultats sur des données simulées, ensuite les résultats sur des données réelles, et enfin les résultats des approches mono-classe en ligne.

A.4.5.1 Données simulées

Afin de visualiser l'impact de l'estimation du centre avec une formulation parcimonieuse en utilisant une petite fraction de l'ensemble des échantillons d'apprentissage, nous appliquons les approches proposées sur deux ensembles de données simulées, les données en forme d'une sinusoïde et les données en forme d'un carré. Le centre parcimonieux des approches proposées utilise seulement 15% des échantillons d'apprentissage comme vecteurs de support.

Nous comparons les résultats des approches proposées avec trois autres approches ayant des formulations parcimonieuses, à savoir SVDD, slab SVM et robust SVM. Les résultats sur les deux ensembles de données simulées sont illustrés dans les figures 4.4 et 4.5. La fraction des vecteurs de support dans ces approches est également fixée à 15 % des échantillons d'apprentissage. Les approches SVDD, slab SVM et robust SVM donnent des enveloppes lâches qui ne décrivent pas la distribution des échantillons d'apprentissage. Les meilleurs résultats sont obtenus avec la première approche proposée menant à des enveloppes serrées qui suivent la distribution des échantillons d'apprentissage de la meilleure façon possible. Quant à la deuxième approche basée sur les méthodes de sélection de variables, l'utilisation de la distance de Mahalanobis

tronquée dans la fonction de décision du classificateur a mené aussi à de bons résultats qui surpassent SVDD, slab SVM et robust SVM, surtout avec Elastic Net.

A.4.5.2 Données réelles

Dans un deuxième temps, nous appliquons ces algorithmes sur les données réelles d'un système de distribution de gaz et d'un réservoir de stockage d'eau de l'Université de Mississippi, et sur les données d'une usine de traitement d'eau de l'UCI "Machine Learning Repository". Pour plus de détails sur les données, voir Section 1.4.

Les algorithmes sont testés sur près de 100 000 échantillons liés aux différents types d'attaques, et les taux de bonne détection sur les données réelles sont donnés dans les tableaux 4.1, 4.2 and 4.3. LARS et LASSO ont à peu près les mêmes résultats, alors que Elastic Net surpasse ces deux algorithmes. Les meilleurs résultats sont obtenus quand Elastic Net est utlisé pour sélectionner les échantillons les plus pertinents, et quand la norme de la fonction de décision du classifieur est celle de Mahalanobis tronquée. Cette combinaison surpasse les autres approches, surtout SVDD, slab SVM et Robust SVM, pour plusieurs types d'attaques. Quant à la première approche proposée, elle surpasse toutes les autres approches mono-classe parcimonieuses pour tous les types d'attaques. Un autre critère pour les approches de classification mono-classe parcimonieuses est le temps d'apprentissage des algorithmes. Le Tableau 4.4 affiche le temps estimé pour chaque approche. Les approches proposées sont les plus rapides ; elles sont 4 fois plus rapide que robust SVM, 5 fois plus rapide que SVDD, et jusqu'à 25 fois plus rapide que la slab SVM. Par conséquent, les approches proposées conduisent aux meilleurs taux de bonne détection, et elles sont jusqu'à 25 fois plus rapide que les autres approche mono-classe.

A.4.5.3 Classification mono-classe en ligne

L'approche de classification mono-classe en ligne proposée est testée sur les données réels mentionnés ci-dessus. L'approche proposée est comparé à trois autres approches, à savoir le quarter-sphere SVM [Zhang et al., 2009], coherence-based one-class [Noumir et al., 2012a], et online SVDD [Tax and Laskov, 2003]. Nous avons fixé le degré de parcimonie dans toutes les approches à 10% des échantillons d'apprentissage.

Le premier critère important pour la détection en ligne est le taux de bonne détection. Ces algorithmes sont testés sur près de 100 000 échantillons liés aux attaques décrites dans le Chapitre 1. Les taux de bonne détection sont présentés dans les tableaux 4.5, 4.6 et 4.7. Les résultats montrent que l'approche en ligne proposée donne les meilleurs taux

de bonne détection et surpasse les autres approches pour toutes les attaques étudiées. En outre, dans certains cas, nous avons des différences importantes entre les taux de bonne détection de l'approche proposée et les autres approches. Ces résultats peuvent être expliqués par les avantages de l'approche en ligne proposée, dans laquel la modification de la fonction de décision permet l'isolation des valeurs aberrantes sans les inclure dans l'étape de la mise à jour du classificateur. Le deuxième critère important pour la détection en ligne est le taux de fausses alarmes. Quarter-sphere SVM a un taux de fausse alarme égal à 11% en moyenne, coherence-based one-class a ce taux égal à 4%, online SVDD a 5%, tandis que l'approche en ligne proposée a mal-classé seulement 1% des échantillons normaux.

Un autre critère important pour la détection en ligne est le temps d'apprentissage des algorithmes. Le temps d'apprentissage estimé des approches étudiées est donné dans le tableau 4.9. Les résultats montrent que la méthode proposée est la plus rapide et, comme prévu, SVDD est la plus lente car elle doit résoudre un problème de programmation quadratique à chaque instant. Enfin, le temps de chaque approche pour tester un nouvel échantillon est donné dans le tableau 4.10. L'approche proposée est la plus rapide avec 0.0019 seconde pour chaque nouvel échantillon, quarter-sphere SVM met 0.0027 seconde, coherence-based met 0,0022 seconde et online SVDD 0.0026 seconde.

Tous ces résultats sont très intéressants pour la détection d'intrusion en ligne dans les applications du monde réel, où l'approche proposée a les taux de bonne détection les plus élevés et les taux de fausses alarmes les plus bas, tout en ayant besoin de moins de 0,002 seconde pour détecter l'intrusion.

A.5 Conclusion et perspectives

Les systèmes SCADA (Supervisory Control And Data Acquisition) permettent la surveillance et le contrôle à distance de la majorité des systèmes industriels et des infrastructures critiques. Le risque des cyber-attaques et des menaces terroristes contre les systèmes industriels a augmenté ces dernières années. Les systèmes traditionnels de détection d'intrusions (IDS) ne peuvent pas détecter les nouvelles attaques, devenues de plus en plus sophistiquées, ne figurant pas dans leurs bases de données. L'objectif principal de cette thèse était d'apporter une aide supplémentaire et complémentaire aux IDS pour assurer une meilleure protection contre les cyber-attaques et les intrusions.

Dans un premier temps, nous avons introduit les méthodes à noyaux et leurs principales propriétés. Nous avons étudié l'influence de la variation de la métrique des noyaux à base radiale sur la fonction de décision du classificateur, et nous avons proposé une

simple heuristique pour l'estimation de l'écart-type de ces noyaux. Nous avons proposé également une approche de classification mono-classe, dans laquelle la classe normale est définie par l'hypersphère renfermant les échantillons dans l'espace de Hilbert. Le centre de cette hypersphère est estimé sans résoudre aucun problème de programmation quadratique. Nous avons proposé d'utiliser la distance de Mahalanobis tronquée dans l'espace transformé en projetant les échantillons dans le sous-espace engendré par les vecteurs propres associés aux plus grandes valeurs propres de la matrice de covariance. Les résultats sur des données simulées et des données réelles ont montré que l'approche proposée a conduit aux meilleures enveloppes autour des échantillons d'apprentissage, aux meilleurs taux de bonne détection, et elle a été plus rapide que les autres approches mono-classe.

Dans un deuxième temps, nous avons proposé deux approches de classification mono-classe qui se basent sur un modèle parcimonieux. La première approche est une formulation parcimonieuse de l'approche mono-classe basée sur la distance de Mahalanobis tronquée, tandis que la deuxième approche est basée sur de méthodes de selection de variables biens connues, comme LARS, LASSO, et Elastic Net. Nous avons revisité ces méthodes en adaptant leurs algorithmes pour l'estimation parcimonieuse du centre de l'hypersphère en classification mono-classe. Les résultats sur des données simulées et des données réelles ont montré que les approches proposées ont conduit aussi aux meilleures enveloppes autour des échantillons d'apprentissage, aux meilleurs taux de bonne détection, et elle ont été plus rapide que les autres approches de l'état de l'art.

Finalement, nous avons proposé une approche de classification mono-classe en ligne qui se base sur un modèle parcimonieux. Nous avons modifié la fonction de décision du classificateur pour l'adapter aux applications en ligne, en définissant deux hypersphères concentriques renfermant les vecteurs de support. Cette nouvelle définition a permis la séparation des échantillons aberrants des vecteurs de support, ce qui a rendu l'algorithme proposé plus robuste aux échantillons aberrants. Nous avons comparé notre approche à d'autres approches dans la littérature, et les résultats ont montré que l'approche proposée a conduit aux meilleurs taux de bonne détection, aux taux de fausse alarme les plus bas, et elle a été la plus rapide.

A.5.1 Travaux futurs

De nombreuses modifications peuvent être faites pour améliorer la performance des approches présentées dans cette thèse. Une étude plus détaillée sur la distance de Mahalanobis tronquée est nécessaire. Quant à l'approximation parcimonieuse, d'autres méthodes de sélection d'un sous-ensemble pertinent existent, comme l'adaptation des

éléments du dictionnaire étudié dans [Saide et al., 2013]. Les auteurs ont considéré les éléments du dictionnaire comme étant des paramètres ajustables, et ils ont proposé un schéma d'adaptation afin d'assurer une meilleure performance en minimisant l'erreur quadratique instantanée. En outre, le problème d'optimisation des méthodes de sélection de variables peut être modifié de façon à remplacer la distance Euclidienne par la distance de Mahalanobis tronquée. Les approches proposées peuvent aussi être adaptées à la classification multiclasse pour classer les attaques détectées. Enfin, les approches proposées dans cette thèse doivent être intégrées dans les systèmes traditionnels de détection d'intrusion des infrastructures critiques, puisqu'elles pourraient jouer un rôle important et complémentaire aux IDS dans la détection des attaques malveillantes et améliorer la détection en temps réel.

Bibliography

M. Abrams and J. Weiss. *Bellingham, Washington, Control System Cyber Security Case Study.* Bellingham, WA, 2008. 22

T. Ahmed, B. Oreshkin, and M. Coates. Machine learning approaches to network anomaly detection. In *Proceedings of the 2nd Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, SYSML'07, pages 7:1–7:6, Berkeley, CA, USA, 2007. USENIX Association. 52

M. A. Aizerman, E. A. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control,*, pages 821–837, 1964. 32

M. Amer, M. Goldstein, and S. Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description (ODD), August 11-14,*, pages 8–15. New York, USA, 2013. ISBN 978-1-4503-2335-2. 52, 59, 131, 136

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68(3):337 – 404, 1950. 31, 33

M. E. Azami, C. Lartizien, and S. Canu. Robust outlier detection with L0-SVDD. In *22th European Symposium on Artificial Neural Networks (ESANN), Bruges, Belgium, April 23-25, 2014*, 2014. 51

K. Bache and M. Lichman. University of California (UCI) Machine Learning Repository, 2013. 22, 28, 77

D. Bailey and E. Wright. *Practical SCADA for Industry.* Elsevier Science, 2003. ISBN 9780080473901. 8, 122

G. E. A. P. A. Batista and M. C. Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, pages 519–533, 2003. 28

B. Bencsáth, L. Buttyán, M. Felegyhazi, and G. Pék. Duqu: Analysis, detection, and lessons learned. In *ACM European Workshop on System Security (EuroSec)*. ACM, 10-11 Sept 2012a. 19

B. Bencsáth, G. Pék, L. Buttyán, and M. Felegyhazi. The cousins of Stuxnet: Duqu, Flame, and Gauss. *Future Internet*, 4(4):971–1003, 2012b. 19, 124

J. Bigham, D. Gamez, and N. Lu. Safeguarding SCADA systems with anomaly detection. In V. Gorodetsky, L. J. Popyack, and V. A. Skormin, editors, *MMM-ACNS*, volume 2776 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2003. 21, 125

R. Blog. The real-time innovations (RTI) blog, tag archives: SCADA system, http://blogs.rti.com/tag/scada-system/, 2014. 13

A. Blum. On-line algorithms in machine learning. In *Developments from a June 1996 Seminar on Online Algorithms: The State of the Art*, pages 306–325, London, UK, 1998. Springer-Verlag. ISBN 3-540-64917-4. 100

O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. In *O. Bousquet, U.v. Luxburg, and G. Ratsch (Editors)*, pages 169–207. Springer, 2004. 31

S. A. Boyer. *SCADA: Supervisory Control And Data Acquisition*. International Society of Automation, USA, 4th edition, 2009. ISBN 1936007096, 9781936007097. 7

R. Brause, T. Langsdorf, and M. Hepp. Neural data mining for credit card fraud detection. In *Tools with Artificial Intelligence. Proceedings 11th IEEE International Conference on*, pages 103–106, 1999. 32

N. Bredeche, Z. Shi, and J.-D. Zucker. Perceptual learning and abstraction in machine learning: an application to autonomous robotics. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 36(2):172–181, March 2006. 32

M. L. Brown and J. F. Kros. Data mining and the impact of missing data. *Industrial Management and Data Systems*, 103(8):611–621, 2003. 28

A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. Fovino, and A. Trombetta. A multidimensional critical state analysis for detecting intrusions in SCADA systems. *Industrial Informatics, IEEE Transactions on*, 7(2):179 –186, May 2011. 20, 125

A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry. Attacks against process control systems: risk assessment, detection, and response. In

Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS Í1, pages 355–366, New York, USA, 2011. ACM. ISBN 978-1-4503-0564-8. 18

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 13, 2001. 100

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. 50, 130

T. Chen and S. Abu-Nimeh. Lessons from Stuxnet. *Computer*, 44(4):91–93, 2011. 18, 124

V. Cherkassky and Y. Ma. Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Netw.*, 17(1):113–126, Jan. 2004. 46, 75, 129, 135

H. Christiansson and E. Luiijf. Creating a european SCADA security testbed. In E. Goetz and S. Shenoi, editors, *Critical Infrastructure Protection*, volume 253 of *IFIP International Federation for Information Processing*, pages 237–247. Springer US, 2007. ISBN 978-0-387-75461-1. 18, 124

P. J. Company. The master operation room, http://parsjahd.com/solutions/scada-rtu, 2015. 11

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39:1–49, 2002. 38

M. Davy, F. Desobry, A. Gretton, and C. Doncarli. An online support vector machine for abnormal events detection. *Signal Process.*, 86(8):2009–2025, Aug. 2006. 101

D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1-3):161–190, 2002. 55

F. Desobry, M. Davy, and C. Doncarli. An online kernel change detection algorithm. *Signal Processing, IEEE Transactions on*, 53(8):2961–2974, Aug 2005. 100, 101

V. L. Do, L. Fillatre, and I. V. Nikiforov. A statistical method for detecting cyber/-physical attacks on SCADA systems. In *IEEE Conference on Control Applications, CCA, Juan Les Antibes, France, October 8-10, 2014*, pages 364–369. 2

V. L. Do, L. Fillatre, and I. Nikiforov. Two sub-optimal algorithms for detecting cyber/-physical attacks on SCADA systems. In *Proceedings of the X International Conference on System Identification and Control Problems (SICPRO'15)*, 2015. 2

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004. 84, 85, 93, 138

M. Eigensatz, J. Giesen, and M. Manjunath. The solution path of the slab support vector machine. In *The 20th Canadian Conference on Computational Geometry, Mcgill University*, pages 211–214. CCCG, 2008. 52, 57, 136

M. Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 144197010X, 9781441970107. 84, 138

P. F. Evangelista, M. J. Embrechts, and B. K. Szymanski. Some properties of the Gaussian kernel for one class learning. In *Proceedings of the 17th international conference on Artificial neural networks*, ICANN'07, pages 269–278, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-74689-7, 978-3-540-74689-8. 46, 129

K. L. Expert. Gauss: Abnormal distribution. Kaspersky, Tech. Rep., August 2012. 19

I. Fovino, A. Carcano, T. De Lacheze Murel, A. Trombetta, and M. Masera. Modbus/DNP3 state-based intrusion detection system. In *Advanced Information Networking and Applications (AINA), 24th IEEE International Conference on*, pages 729–736, April 2010a. 20, 125

I. Fovino, M. Masera, L. Guidi, and G. Carpi. An experimental platform for assessing SCADA vulnerabilities and countermeasures in power plants. In *Human System Interactions (HSI), 3rd Conference on*, pages 679 –686, May 2010b. 16, 124

I. Fovino, A. Coletta, A. Carcano, and M. Masera. Critical state-based filtering system for securing SCADA network protocols. *Industrial Electronics, IEEE Transactions on*, 59(10):3943–3950, Oct 2012. 16, 124

I. N. Fovino, A. Carcano, M. Masera, and A. Trombetta. An experimental investigation of malware attacks on SCADA systems. *International Journal of Critical Infrastructure Protection*, 2(4):139 – 145, 2009. 16, 124

R. Fujimaki. Anomaly detection support vector machine and its application to fault diagnosis. In *Data Mining. ICDM '08. Eighth IEEE International Conference on*, pages 797–802, Dec 2008. 32

W. Gao, T. Morris, B. Reaves, and D. Richey. On SCADA control system command and response injection and intrusion detection. In *eCrime Researchers Summit (eCrime)*, pages 1–9, Oct 2010. 21, 26, 27

A. B. Gardner, A. M. Krieger, G. Vachtsevanos, and B. Litt. One-class novelty detection for seizure analysis from intracranial EEG. *Journal of Machine Learning Research*, 7: 1025–1044, 2006. 50

C. Gentile. A new approximate maximal margin classification algorithm. *J. Mach. Learn. Res.*, 2:213–242, Mar. 2002. 100

A. Ghadiri and N. Ghadiri. An adaptive hybrid architecture for intrusion detection based on fuzzy clustering and RBF neural networks. In *Communication Networks and Services Research Conference (CNSR), Ninth Annual*, pages 123–129, May 2011. 21, 125

V. Golovko, P. Kachurka, and L. Vaitsekhovich. Neural network ensembles for intrusion detection. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. IDAACS. 4th IEEE Workshop on*, pages 578–583, Sept 2007. 21

V. Gomez-Verdejo, J. Arenas-Garcia, M. Lazaro-Gredilla, and A. Navia-Vazquez. Adaptive one-class support vector machine. *Signal Processing, IEEE Transactions on*, 59 (6):2975–2981, June 2011. 101

S. Gorman. Electricity Grid in U.S. Penetrated By Spies. *The Wall Street Journal*, Apr. 2008. 18, 124

R. Greiner, B. Silver, S. Becker, and M. Gruninger. A review of machine learning at aaai-87. *Machine Learning*, 3(1):79–92, 1988. 32

P. Gross, J. Parekh, and G. Kaiser. Secure selecticast for collaborative intrusion detection systems. In *3rd International Workshop on Distributed Event-Based Systems (DEBS'04)*, Edinburgh, Scotland, UK, 2004. 20, 125

J. W. Grzymala-Busse, L. K. Goodwin, W. J. Grzymala-Busse, and X. Zheng. Handling missing attribute values in preterm birth data sets. In D. Slezak, J. Yao, J. F. Peters, W. Ziarko, and X. Hu, editors, *RSFDGrC (2)*, volume 3642 of *Lecture Notes in Computer Science*, pages 342–351. Springer, 2005. 28

P. Gurram and H. Kwon. Support-vector-based hyperspectral anomaly detection using optimized kernel parameters. *Geoscience and Remote Sensing Letters, IEEE*, 8(6): 1060–1064, 2011. 46, 129

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, USA, 2001. 84

S. Haykin. *Neural Networks: A Comprehensive Foundation (2nd Edition)*, chapter 5. Prentice Hall, 2 edition, July 1998. ISBN 0132733501. 46, 129

R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, Cambridge, MA, USA, 2001. ISBN 026208306X. 32

H. Hoffmann. Kernel PCA for novelty detection. *Pattern Recognition*, 40(3):863 – 874, 2007. 50, 52, 61, 64, 76, 77, 130, 131, 136

T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36:1171–1220, 2008. 32, 125

P. Honeine. Analyzing sparse dictionaries for online learning with kernels. *IEEE Transactions on Signal Processing*, (in press) 2015. 86

P. Honeine, C. Richard, and J. C. M. Bermudez. On-line nonlinear sparse approximation of functions. In *Proc. IEEE International Symposium on Information Theory*, pages 956–960, Nice, France, June 2007. 86, 140

P. Honeine, Z. Noumir, and C. Richard. Multiclass classification machines with the complexity of a single binary classifier. *Signal Processing*, 93(5):1013–1026, May 2013. 119

D.-J. Kang, J.-J. Lee, S.-J. Kim, and J.-H. Park. Analysis on cyber threats to SCADA systems. In *Transmission Distribution Conference Exposition: Asia and Pacific*, pages 1–4, Oct 2009. 7

M. Karasuyama and I. Takeuchi. Multiple incremental decremental learning of support vector machines. *Neural Networks, IEEE Transactions on*, 21(7):1048–1059, July 2010. 100

L. Kennedy, Y. Irvin-Erickson, and A. Kennedy. *Translational Criminology and Counterterrorism: Global Threats and Local Responses*. SpringerBriefs in Criminology / SpringerBriefs in Translational Criminology. Springer New York, 2014. ISBN 9781461455554. 16

S. S. Khan and M. G. Madden. A survey of recent trends in one class classification. In *Proceedings of the 20th Irish conference on Artificial intelligence and cognitive science*, AICS'09, pages 188–197, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-17079-X, 978-3-642-17079-9. 50, 130

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82 – 95, 1971. 38

J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *Signal Processing, IEEE Transactions on*, 52(8):2165–2176, Aug. 2004. 100

F. Knorn and D. Leith. Adaptive Kalman filtering for anomaly detection in software appliances. In *INFOCOM Workshops, IEEE*, pages 1–6, April 2008. 21, 125

B. Krekel, G. Bakos, and C. Barnett. Capability of the People's Republic of China to conduct cyber warfare and computer network exploitation. Research report, The US–China Economic and Security Review Commission, Washington, DC, Oct. 2009. 15

V. Kůrková and M. Sanguineti. Learning with generalization capability by kernel methods of bounded complexity. *J. Complex.*, 21(3):350–367, June 2005. 38

R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *Security Privacy, IEEE*, 9(3): 49–51, 2011. 18

P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller. Incremental support vector learning: Analysis, implementation and applications. *J. Mach. Learn. Res.*, 7:1909–1936, Dec. 2006. 101

D. Li, J. S. Deogun, W. Spaulding, and B. Shuart. Towards missing data imputation: A study of fuzzy k-means clustering method. In S. Tsumoto, R. Slowinski, H. J. Komorowski, and J. W. Grzymala-Busse, editors, *Rough Sets and Current Trends in Computing*, volume 3066 of *Lecture Notes in Computer Science*, pages 573–579. Springer, 2004. 28

Y. Li and P. M. Long. The relaxed online maximum margin algorithm. *Mach. Learn.*, 46(1-3):361–387, Jan. 2002. 100

J. Ma and S. Perkins. Time-series novelty detection using one-class support vector machines. In *Neural Networks. Proceedings of the International Joint Conference on*, volume 3, pages 1741–1745 vol.3, July 2003. 50

P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, Apr. 1936. 64, 131

S. Mahfouz, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi. Target tracking using machine learning and Kalman filter in wireless sensor networks. *Sensors Journal, IEEE*, 14(10):3715–3725, Oct 2014. 32

A. Mathur and G. Foody. Multiclass and binary SVM classification: Implications for training and classification users. *Geoscience and Remote Sensing Letters, IEEE*, 5(2): 241–245, April 2008. 50, 130

O. Mazhelis. One-class classifiers : a review and analysis of suitability in the context of mobile-masquerader detection. *South African Computer Journal*, 36:29–48, 2006. 50

T. Morris and K. Pavurapu. A retrofit network transaction data logger and intrusion detection system for transmission and distribution substations. In *Power and Energy (PECon), IEEE International Conference on*, pages 958–963, Nov 2010. 16, 124

T. Morris, A. Srivastava, B. Reaves, W. Gao, K. Pavurapu, and R. Reddi. A control system testbed to validate critical infrastructure protection concepts. *International Journal of Critical Infrastructure Protection*, 4(2):88 – 103, 2011a. 20, 22, 23, 77

T. Morris, R. Vaughn, and Y. S. Dandass. A testbed for SCADA control system cybersecurity research and pedagogy. In *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, CSIIRW '11, pages 27:1–27:1, New York, USA, 2011b. ACM. ISBN 978-1-4503-0945-5. 20, 22

A. Munoz and J. Moguerza. Estimation of high-density regions using one-class neighbor machines. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(3): 476–480, March 2006. 52

P. Nader, P. Honeine, and P. Beauseroy. Intrusion detection in SCADA systems using one-class classification. In *Proc. 21th European Conference on Signal Processing (EUSIPCO)*, Marrakech, Morocco, 9–13 September 2013. 5, 50

P. Nader, P. Honeine, and P. Beauseroy. The role of one-class classification in detecting cyberattacks in critical infrastructures. In *Proc. 9th International Conference on Critical Information Infrastructures Security (CRITIS)*, Limassol, Cyprus, 13–15 October 2014a. 5

P. Nader, P. Honeine, and P. Beauseroy. Mahalanobis-based one-class classification. In *Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Reims, France, 21–24 September 2014b. 5, 50

P. Nader, P. Honeine, and P. Beauseroy. l_p-norms in one-class classification for intrusion detection in SCADA systems. *Industrial Informatics, IEEE Transactions on*, 10(4): 2308–2317, Nov 2014c. 5, 50

P. Nader, P. Honeine, and P. Beauseroy. Online one-class classification for intrusion detection based on the Mahalanobis distance. In *Proc. 23th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, Bruges, Belgium, 22–24 April 2015a. 5

P. Nader, P. Honeine, and P. Beauseroy. Shrinkage methods for one-class classification. In *Proc. 23th European Conference on Signal Processing (EUSIPCO)*, Nice, France, 31 August – 4 September 2015b. 5

P. Nader, P. Honeine, and P. Beauseroy. One-class classification framework based on shrinkage methods. *Machine Learning*, submitted 2015. 5

Z. Noumir. *Etude de la qualité de l'eau potable dans un réseau de distribution par apprentissage statistique*. PhD thesis, Université de technologie de Troyes, 2012. 86, 140

Z. Noumir, P. Honeine, and C. Richard. Online one-class machines based on the coherence criterion. In *Proc. 20th European Conference on Signal Processing*, Bucharest, Romania, 27–31 August 2012a. 101, 111, 149

Z. Noumir, P. Honeine, and C. Richard. On simple one-class classification methods. In *Proc. IEEE International Symposium on Information Theory*, MIT, Cambridge (MA), USA, 1–6 July 2012b. 52, 63, 64, 86, 131, 136, 140

P. W. Oman and M. Phillips. Intrusion detection and event monitoring in SCADA networks. In *Critical Infrastructure Protection*, pages 161–173, 2007. 20

M. R. Osborne, B. Presnell, and B. A. Turlach. On the LASSO and its dual. *Journal of Computational and Graphical Statistics*, 9:319–337, 1999. 84, 85, 138

C. Richard, J. Bermudez, and P. Honeine. Online prediction of time series data with kernels. *Signal Processing, IEEE Transactions on*, 57(3):1058–1067, March 2009. 86, 140

A. Rocha and S. Klein Goldenstein. Multiclass from binary: Expanding one-versus-all, one-versus-one and ecoc-based approaches. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(2):289–302, Feb 2014. 50, 130

V. Roth. Kernel Fisher Discriminants for Outlier Detection. *Neural Comput.*, 18(4): 942–960, Apr. 2006. 52

R. Rubinstein, A. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, June 2010. 84, 138

C. Saide, R. Lengelle, P. Honeine, and R. Achkar. Online kernel adaptive algorithms with dictionary adaptation for mimo models. *Signal Processing Letters, IEEE*, 20(5): 535–538, May 2013. 118, 152

T. Schneider. Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values, 2001. 28

B. Schölkopf, C. Burges, and A. Smola. Introduction to support vector learning. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 1–22. MIT Press, 1998a. 51, 55

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319, July 1998b. 52, 61, 131, 136

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized Representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, pages 416–426, London, UK, 2001a. Springer-Verlag. ISBN 3-540-42343-5. 38

B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7): 1443–1471, July 2001b. 42, 51, 55, 127

B. Schölkopf, J. Giesen, and S. Spalinger. Kernel methods for implicit surface modeling. In *Advances in Neural Information Processing Systems 17*, pages 1193–1200. MIT Press, 2005. 52, 57, 131

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, USA, 2004. ISBN 0521813972. 2, 32, 96, 98, 125

J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000. 46, 129

J. Slay and M. Miller. Lessons learned from the Maroochy water breach. In *Critical Infrastructure Protection*, pages 73–82, 2007. 18, 124

C. Soares, P. B. Brazdil, and P. Kuba. A Meta-Learning Method to Select the Kernel Width in Support Vector Regression. *Machine Learning*, 54(3):195–209, 2004. 46, 75, 129, 135

Q. Song, W. Hu, and W. Xie. Robust support vector machine with bullet hole image classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32(4):440–448, Nov 2002. 52

S. J. Stolfo, F. Apap, E. Eskin, K. Heller, S. Hershkop, A. Honig, and K. Svore. A comparative evaluation of two algorithms for windows registry anomaly detection. *J. Comput. Secur.*, 13(4):659–693, July 2005. 50

K. Stouffer, J. Falco, and K. Kent. Guide to supervisory control and data acquisition (SCADA) and industrial control systems security. Technical report, National Institute of Standards and Technology (NIST), September 2006. 8, 122

K. Stouffer, J. Falco, and K. Scarfone. Sp 800-82. guide to industrial control systems (ICS) security: Supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and other control system configurations such as

programmable logic controllers (PLC). Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2011. 7, 10, 123

D. Strauss, W. Delb, J. Jung, and P. Plinkert. Adapted filter banks in machine learning: applications in biomedical signal processing. In *Acoustics, Speech, and Signal Processing. Proceedings. (ICASSP '03). IEEE International Conference on*, volume 6, pages VI–425–8 vol.6, April 2003. 32

Q. Tao, G.-w. Wu, and J. Wang. A new maximum margin algorithm for one-class problems and its boosting implementation. *Pattern Recogn.*, 38(7):1071–1077, July 2005. 52, 57, 131

A. Tartakovsky, I. Nikiforov, and M. Basseville. *Sequential Analysis: Hypothesis Testing and Changepoint Detection*. Taylor & Francis, 2014. ISBN 9781439838204. 2

D. M. J. Tax and R. P. W. Duin. Data domain description using support vectors. In *Proceedings of the European Symposium on Artificial Neural Networks*, pages 251–256, 1999. 51, 53, 131

D. M. J. Tax and R. P. W. Duin. Support vector data description. *Mach. Learn.*, 54(1): 45–66, Jan. 2004. 51, 53, 136

D. M. J. Tax and P. Juszczak. Kernel whitening for one-class classification. In *Pattern Recognition with Support Vector Machines, First International Workshop, Niagara Falls, Canada, August 10*, pages 40–52, 2002. 42, 51, 64, 66, 127, 131, 133

D. M. J. Tax and P. Laskov. Online SVM learning: from classification to data description and back. *Neural Networks for Signal Processing (NNSP'03), IEEE 13th Workshop on*, pages 499–508, 2003. 101, 111, 149

C.-W. Ten, C.-C. Liu, and G. Manimaran. Vulnerability assessment of cybersecurity for SCADA systems. *Power Systems, IEEE Transactions on*, 23(4):1836–1846, 2008. 16, 50, 123, 130

C.-W. Ten, J. Hong, and C.-C. Liu. Anomaly detection for cybersecurity of the substations. *Smart Grid, IEEE Transactions on*, 2(4):865–873, Dec 2011. 16, 123

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996. 84, 85, 138

A. N. Tikhonov and V. Y. Arsenin. *Solutions of ill-posed problems*. V. H. Winston & Sons, Washington, D.C.: John Wiley & Sons, New York, 1977. Translated from the Russian, Preface by translation editor Fritz John, Scripta Series in Mathematics. 31, 33

J. Tropp and S. Wright. Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE*, 98(6):948–958, June 2010. 84, 138

O. G. Troyanskaya, M. Cantor, G. Sherlock, P. O. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001. 28

I. Tsang, J. Kwok, and S. Li. Learning the kernel in Mahalanobis one-class support vector machines. In *Neural Networks. IJCNN '06. International Joint Conference on*, pages 1169–1175, 2006. 52

E. UK. SCADA system process, http://www.ukessays.com/essays/information-systems/scada-system-process.php, November, 2013. 14

V. Urias, B. Van Leeuwen, and B. Richardson. Supervisory command and data acquisition (SCADA) system cyber security analysis using a live, virtual, and constructive (lvc) testbed. In *Military Communication Conference - MILCOM*, pages 1–8, 2012. 17

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, USA, 1995. ISBN 0-387-94559-8. 31, 55

T. Veracini, S. Matteoli, M. Diani, and G. Corsini. An anomaly detection architecture based on a data-adaptive density estimation. In *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 3rd Workshop on*, pages 1–4, June 2011. 21, 125

J. P. Vert, K. Tsuda, and B. Scholkopf. A primer on kernel methods. *Kernel Methods in Computational Biology*, pages 35–70, 2004. 32, 34, 125

G. Wahba. *Spline models for observational data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1990. 38

D. Wang, D. Yeung, and E. C. C. Tsang. Structured one-class classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(6):1283–1295, Dec 2006. 52

Q. Wang, L. S. Lopes, and D. M. J. Tax. Visual object recognition through one-class learning. In *ICIAR (1)*, pages 463–470, 2004. 50

A. K. C. Wong and D. K. Y. Chiu. Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(6):796–805, 1987. 28

J. Yan, C.-C. Liu, and M. Govindarasu. Cyber intrusion of wind farm SCADA system and its impact analysis. In *Power Systems Conference and Exposition (PSCE), IEEE/PES*, pages 1–6, March 2011. 20

Y. Yang, K. McLaughlin, T. Littler, S. Sezer, B. Pranggono, and H. Wang. Intrusion detection system for iec 60870-5-104 based SCADA networks. In *Power and Energy Society General Meeting (PES), IEEE*, pages 1–5, July 2013. 20

Y. Yang, K. McLaughlin, S. Sezer, T. Littler, E. Im, B. Pranggono, and H. Wang. Multiattribute SCADA-specific intrusion detection system for power networks. *Power Delivery, IEEE Transactions on*, 29(3):1092–1102, June 2014. 20, 125

N. Ye, Q. Chen, and C. Borror. Ewma forecast of normal system activity for computer intrusion detection. *Reliability, IEEE Transactions on*, 53(4):557–566, Dec 2004. 21

Y. Yu, Y. Wei, G. Fu-Xiang, and Y. Yu. Anomaly intrusion detection approach using hybrid MLP/CNN neural network. In *Intelligent Systems Design and Applications. ISDA. Sixth International Conference on*, volume 2, pages 1095–1102, Oct 2006. 21

Z. Zeng, Y. Fu, G. Roisman, Z. Wen, Y. Hu, and T. Huang. One-class classification for spontaneous facial expression analysis. In *Automatic Face and Gesture Recognition (FGR). 7th International Conference on*, pages 281–286, April 2006. 50

R. Zhang, S. Zhang, Y. Lan, and J. Jiang. Network anomaly detection using one class support vector machine. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume I, Hong Kong, March 2008. 50

Y. Zhang, N. Meratnia, and P. Havinga. Adaptive and online one-class support vector machine-based outlier detection techniques for wireless sensor networks. In *Advanced Information Networking and Applications Workshops. WAINA. International Conference on*, pages 990–995, May 2009. 101, 111, 149

S. Zhioua. The middle east under malware attack dissecting cyber weapons. In *Distributed Computing Systems Workshops (ICDCSW), IEEE 33rd International Conference on*, pages 11–16, July 2013. 19

D.-X. Zhou. On grouping effect of Elastic net. *Statistics & Probability Letters*, 83: 2108–2112, 2013. 84, 85, 138

H. Zou and T. Hastie. Regularization and variable selection via the Elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005. 84, 85, 94, 138, 145

Patric NADER

Doctorat : Optimisation et Sûreté des Systèmes

Année 2015

Classification mono-classe pour la détection des cyber-intrusions dans les systèmes industriels

La sécurité des infrastructures critiques a suscité l'attention des chercheurs récemment avec l'augmentation du risque des cyber-attaques et des menaces terroristes contre ces systèmes. La majorité des infrastructures est contrôlée par des systèmes SCADA (Supervisory Control And Data Acquisition) permettant le contrôle à distance des processus industriels, comme les réseaux électriques, le transport de gaz, la distribution d'eau potable, les centrales nucléaires, etc. Les systèmes traditionnels de détection d'intrusions sont incapables de détecter les nouvelles attaques ne figurant pas dans leurs bases de données. L'objectif de cette thèse est d'apporter une aide supplémentaire à ces systèmes pour assurer une meilleure protection contre les cyber-attaques.

La complexité et la diversité des attaques rendent leur modélisation difficile. Pour surmonter cet obstacle, nous utilisons des méthodes d'apprentissage statistique mono-classes. Ces méthodes élaborent une fonction de décision à partir de données d'apprentissage, pour classer les nouveaux échantillons en données aberrantes ou données normales. La fonction de décision définie l'enveloppe d'une région de l'espace de données contenant la majeure partie des données d'apprentissage. Cette thèse propose des méthodes de classification mono-classe, des formulations parcimonieuses de ces méthodes, et une méthode en ligne pour la détection temps réel. Les performances de ces méthodes sont montrées sur des données benchmark de différents types d'infrastructures critiques.

Mots clés : apprentissage automatique - traitement du signal - reconnaissance des formes (informatique) - détection du signal – représentation parcimonieuse.

One-class Classification for Cyber Intrusion Detection in Industrial Systems

The security of critical infrastructures has been an interesting topic recently with the increasing risk of cyber-attacks and terrorist threats against these systems. The majority of these infrastructures is controlled via SCADA (Supervisory Control And Data Acquisition) systems, which allow remote monitoring of industrial processes such as electrical power grids, gas pipelines, water distribution systems, wastewater collection systems, nuclear power plants, etc. Traditional intrusion detection systems (IDS) cannot detect new types of attacks not listed in their databases, so they cannot ensure maximum protection for these infrastructures.

The objective of this thesis is to provide additional help to IDS to ensure better protection for industrial systems against cyber-attacks and intrusions. The complexity of studied systems and the diversity of attacks make modeling these attacks very difficult. To overcome this difficulty, we use machine learning, especially one-class classification. Based on training samples, these methods develop decision rules to classify new samples as outliers or normal ones. This dissertation proposes specific one-class classification approaches, sparse formulations of these approaches, and an online approach to improve the real-time detection. The relevance of these approaches is illustrated on benchmark data from three different types of critical infrastructures.

Keywords: machine learning - signal processing - pattern recognition systems - signal detection – sparse representation.

Thèse réalisée en partenariat entre :

Agence Nationale de la Recherche ANR

utt université de technologie Troyes

Ecole Doctorale "Sciences et Technologies"