Computational Intelligence Techniques for OES Data Analysis

By
Luca Puggini

A thesis presented on application for the degree of
Doctor of Philosophy

Department of Electronic Engineering
Maynooth University

Head of the Department: Dr. Ronan Farrell
Supervisors: Dr. Seán McLoone

January 7, 2017

Abstract

Semiconductor manufacturers are forced by market demand to continually deliver lower cost and faster devices. This results in complex industrial processes that, with continuous evolution, aim to improve quality and reduce costs. Plasma etching processes have been identified as a critical part of the production of semiconductor devices. It is therefore important to have good control over plasma etching but this is a challenging task due to the complex physics involved.

Optical Emission Spectroscopy (OES) measurements can be collected non-intrusively during wafer processing and are being used more and more in semiconductor manufacturing as they provide real time plasma chemical information. However, the use of OES measurements is challenging due to its complexity, high dimension and the presence of many redundant variables. The development of advanced analysis algorithms for virtual metrology, anomaly detection and variables selection is fundamental in order to effectively use OES measurements in a production process.

This thesis focuses on computational intelligence techniques for OES data analysis in semiconductor manufacturing presenting both theoretical results and industrial application studies. To begin with, a spectrum alignment algorithm is developed to align OES measurements from different sensors. Then supervised variables selection algorithms are developed. These are defined as improved versions of the LASSO estimator with the view to selecting a more stable set of variables and better prediction performance in virtual metrology applications. After this, the focus of the thesis moves to the unsupervised variables selection problem. The Forward Selection Component Analysis (FSCA) algorithm is improved with the introduction of computationally efficient implementations and different refinement procedures. Nonlinear extensions of FSCA are also proposed. Finally, the fundamental topic of anomaly detection is investigated and an unsupervised variables selection algorithm tailored to anomaly detection is developed. In addition, it is shown

how OES data can be effectively used for semi-supervised anomaly detection in a semiconductor manufacturing process.

The developed algorithms open up opportunities for the effective use of OES data for advanced process control. All the developed methodologies require minimal user intervention and provide easy to interpret models. This makes them practical for engineers to use during production for process monitoring and for in-line detection and diagnosis of process issues, thereby resulting in an overall improvement in production performance.

Acknowledgements

First and foremost I would like to thank my supervisor Seán McLoone that guided me throughout my PhD with wisdom and patience.

I would also like to thank Intel Ireland for the provided data and my industrial mentors Niall Macgearailt and Paul Sheehy for their assistance and technical guidance.

The financial support provided by Enterprise Ireland and Maynooth University is gratefully acknowledged.

Furthermore I would like to thank all my friends and colleagues from the Dynamics and Control Group and from the Center for Ocean Energy Resource for their friendship and support during my journey. Among these a special thanks goes to John Doyle that helped me a lot during the first year of my PhD.

A special thanks goes also to the fantastic people that I have met during these years in Dublin that made me feel like at home.

Finally I would like to thank my parents Benedetto Puggini, Diana Splendori and my sister Diana Puggini for their love and support over the years.

Contents

Chapter 1

Introduction

1.1 The Age of Data

In 1944 Fremont Rider estimated that American university libraries were doubling in size every sixteen years [1]. Rider said: “Yale Library in 2040 will have approximately 200,000,000 volumes, which will occupy over 6,000 miles of shelves”. The following years were characterised by rapid technological advancements and stored information slowly moved from books and paper to digital devices. In 1996, R.J.T. Morris and B.J. Truskowski showed how cheaply data could be digitally stored [2]. In the following years (1997-2000), technology advanced even faster during the Dot-Com Bubble. This lead to the spread of the internet, the diffusion of cheaper hard drives and the availability of more processing power making data storage and processing even easier. In 2010 the Economist published an article titled "The Data Deluge" showing how the amount of data collected was exponentially increasing. According to the article, mankind created 150 exabytes (10^{17} bytes) of data in 2005 and 1200 in 2010 and some estimates say that, in 2010, 90% of the world data was generated during the last year. Figure 1.1 shows the expected amount of data generated in the world by year.

“Big Data” is the term commonly used to refer to the large and complex datasets available. Big Data is commonly characterised by three characteristics reffered as the three Vs of Big Data [3]: Volume, Variety and Velocity. Volume refers to the large amount of data; Variety to the different formats of the data and the different sources of information; Velocity refers to the speed with which data is generated and collected.

The Big Data phenomena was originally related to web applications where an

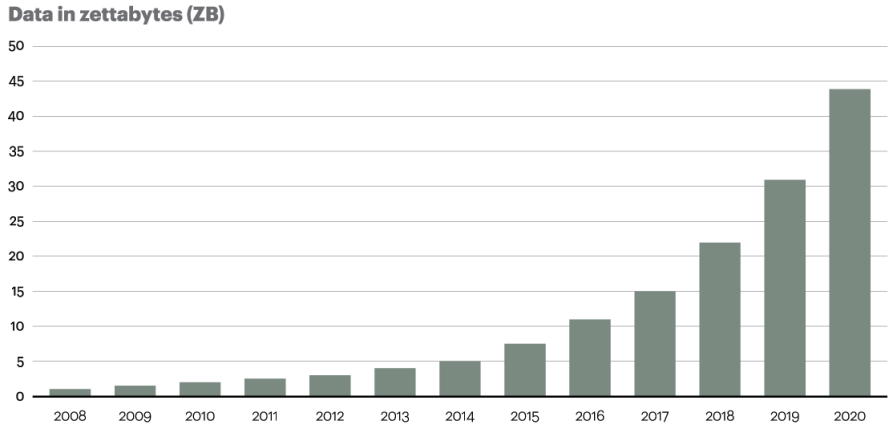


Figure 1.1: The expected amount of data generated in the world according to Oracle. (1 zettabyte = 10^{21} bytes).

incredible amount of data is continually generated. In recent years, thanks to the availability of cheaper sensors and measurement tools, the use of Big Data expanded to several sectors such as: health care, manufacturing, retail and security. This thesis focuses on applications in the semiconductor manufacturing industry, a field that is starting to rely heavily on data to improve production.

1.2 Semiconductor Manufacturing

Semiconductor manufacturing is one of the largest industries in the world, employing almost 250,000 people in the USA alone [4]. It posted sales globally totalling 335.2 billion dollars in 2015 [5] and sales are expected to continue to grow in the future as shown in Figure 1.2. Some big companies in the area, and their revenue in 2012, are reported in Table 1.1. Market demand and fierce competition has driven restless innovation in the sector. As a consequence, research and development (R&D) has always been at the forefront of semiconductor manufacturing.

Since the invention of integrated circuits in 1960, the number of transistors on an integrated circuit has doubled roughly every 2 years as predicted by Moore's law [6]. To maintain the pace of development set by Moore's law, production processes in semiconductor manufacturing are becoming more and more complex, consisting of several hundred processing steps. Better device performance and greater production throughput are achieved by re-

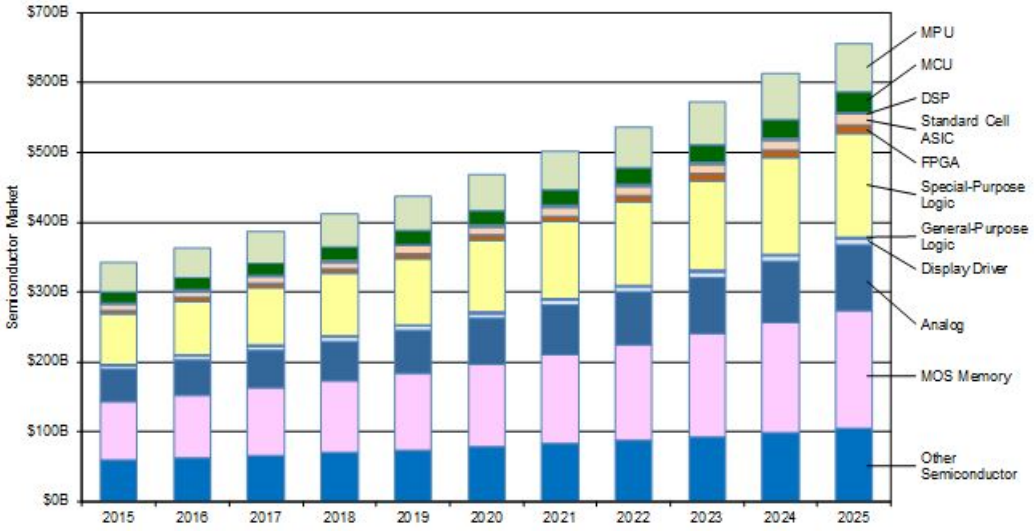


Figure 1.2: The global semiconductor market from 2015 through 2025. Figure and data from SEMI (`http://www.semi.org`)

Company Name	2012 Revenue in Million of Dollars
Intel	47,543
Samsung Electronics	30,474
Qualcomm	12,976

Table 1.1: Top three semiconductor manufacturing companies and their revenue in 2012.

ducing circuit critical dimensions, improving processor architecture, increasing wafer size and by improving product consistency and uniformity.

Every two years, the International Technology Roadmap for Semiconductors (ITRS) indicates research areas where more innovation is required and provides guidelines for the use of research funds. The 2007 ITRS roadmap [7] defines the wafer design parameters over an 8-year period. In order to meet the industry historical 30% cost-per-function reduction, and 50% cycle time improvement in manufacturing per decade, the wafer diameter is required to increase from 300 to 450mm while the critical dimensions are required to decrease from 80 to 22nm.

In order to mantain market share and remain competitive, it is essential to keep production costs low. In the past cost reductions were achieved via yield improvement. As yield improvements are more and more difficult to achieve, further improvement can be obtained by maximizing throughput of products with reduced setup and maintenance costs [8]. This has led to the adoption of statistical process control (SPC) techniques to monitor process faults and anomalies, with its use in fabs expanding rapidly between the 1980's and the 1990's. However, as production processes have became more complex, traditional SPC is no longer adequate leading to an increasing number of false alarms and undetected anomalies. In a modern manufacturing plant, a chip costs, on average, $40 and a wafer contains roughly 450 chips. The cost of producing a faulty wafer is therefore of the order of $22,500.

In order to improve on SPC, the use of advanced process control (APC) spread. APC refers to a broad range of techniques and technologies employed to maximise the use of available information about materials and processes. APC analyses diagnostic data and desired targets, selects model and control strategies, estimates the feasibility of the desired targets and generates the necessary alarms for process faults. APC is usually deployed optionally and, in addition to basic SPC, and developed over a period of time with the aim of solving specific problems [8].

1.2.1 Wafer Processing

Semiconductor device fabrication is a complex process. The main steps involved are represented in Figure 1.3. Among these, plasma etching has been recognized as one of the operations which has a decisive influence on product quality and has been the focus of many studies [9].

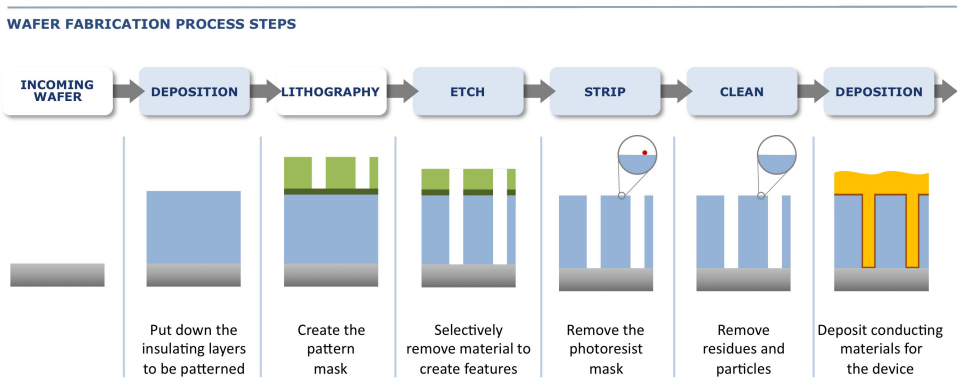


Figure 1.3: Wafer processing steps. Figure from Lam Research (`http://www.lamresearch.com/products/products-overview`).

1.2.2 Plasma Etching

In the late 1960s, plasma etching, a form of plasma processing used for integrated circuit (IC) manufacturing, emerged as an alternative to acid bath chemical etching (wet etching, [9]). In contrast to chemical etching, plasma etching can be directed. An highly directional etching process is desirable to guarantee product quality and avoid problems such as short circuits [10]. Figure 1.4 shows the differences between plasma etching and chemical etching; the former leads to a more precise and controlled etching and is able to shape the wafer surface as required.

Due to its physical structure plasma emits light and this light has been proven to be a reliable indicator of plasma chemistry. This can be collected non-intrusively during etching through the use of an Optical Emission Spectrometer [11]. The resulting Optical Emission Spectroscopy (OES) data allows real-time plasma monitoring and is a starting point for the application of APC to a plasma etching process [12]. Despite this, the application of APC to plasma etching processes remains challenging, due to the highly complex plasma physics and etching chemistry involved and the sensitivity of plasma to subtle process variations. In addition, OES data is generally characterized by high dimension (typically $>$2000 variables) and highly redundant variables, making its use and analysis challenging.

The OES data is generally represented as a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ where each column is a time series containing the intensity at a particular wavelength over time. In the rest of the thesis, with an abuse of notation, the term

"wavelength" is often used to indicate the intensity at a given wavelength or one of the columns of **X**. This will not lead to confusion as the meaning of wavelength will be clear from the context. A more detailed description of the OES data will be presented in Chapter 2.

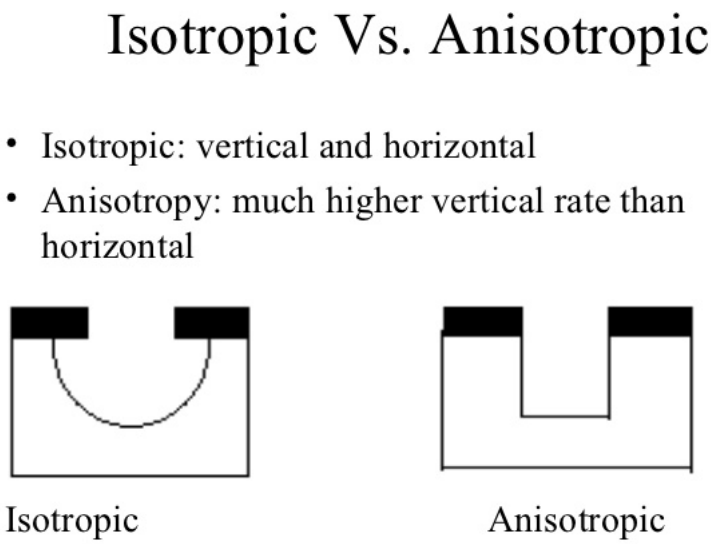


Figure 1.4: Difference between plasma (Anisotropy) and chemical (Isotropy) etching. Figure from Minh Anh Thi Nguyen.

1.3 High Dimensional Data and the Curse of Dimensionality

High dimensional datasets are difficult to deal with on several counts. If the number of variables is larger than the number of measurements, each variable can be obtained as a linear combination of the others making uncovering the true relationship between the different variables difficult. This is a common scenario in semiconductor manufacturing as the number of measurements often corresponds to the number of processed wafers and that is generally much smaller than the number of measured variables. Furthermore, high dimensional datasets are impacted by the so called "Curse of Dimensionality"

according to which distance measurements between samples become unreliable. In addition, high dimensional datasets are difficult to interpret, which in turn makes extracting useful process insight challenging. From a computational perspective, the curse of dimensionality can mean an exponential growth in computational complexity with dimension, leading to computation time problems. Addressing computational bottlenecks and developing computationally efficient algorithms is therefore critical when working with high dimensional data.

For any practical application, the dimensionality of the OES data needs to be reduced. Traditionally engineering knowledge of the underlining process chemistry can be used to extract the most relevant wavelengths from the OES data [13]. This process is problematic as it is time consuming, biased by the engineer's personal experience, and is limited to a particular process. Given changes in the process recipes or etching products, the effectiveness of the selected wavelengths can be reduced. As a consequence, more automated and unbiased techniques are required.

1.3.1 Dimensionality Reduction and Variable Selection

Highly dimensional datasets are not only present in semiconductor manufacturing, but are common in many fields such as image processing [14] and genetics [15]. As a consequence, a lot of interdisciplinary research is involved in the area and several dimensionality reduction algorithms have been developed.

Dimensionality reduction algorithms are, in general, based on feature extraction. This is a well known machine learning problem that was originally investigated in the field of pattern recognition and image processing. Feature extraction summarises the data with basic components that seek to extract all the information that is required for a given task. Feature extraction techniques are now widely applied in several fields in different forms. Due to this variety, it is impossible to provide an accurate definition of feature extraction. As Selfridge and Neisser [16] pointed out, feature extraction algorithms have to be designed individually to effectively tackle an unknown issue.

Feature extraction is, in general, performed trough the use of data driven black-box algorithms. The extraction methodologies are then widely applicable but the obtained features loose their physical meaning. This problem can be avoided with the use of variable selection algorithms. Variable se-

lection is equivalent to feature extraction, but the extracted features are constrained to be a subset of the original variables. This allows interpretable APC procedures to be developed and enables engineers to rapidly understand the root causes of faults or process variations.

Variable selection algorithms can be divided into three categories: supervised variable selection, semi-supervised and unsupervised variable selection [17]. In supervised variables selection the choice of the variables is guided by a target variable and the goodness of the selection is evaluated based on the obtained prediction performance. In the semi-supervised and unsupervised variable selection, instead, variables are selected in order to summarise the majority of the data looking for the right balance between number of selected variables and information loss, that, according to the application, can be measured with different metrics. The difference between an unsupervised and a semi-supervised analysis is that, in the first case, no information about the data is available while in the second case the data is known to contain only samples of a given category as. for example, only normal behaving wafers.

1.4 Aims and Scope of the Thesis

This thesis aims to develop techniques that enable OES measurements to be used effectively in a semiconductor manufacturing production process. A full industrial case study is discussed from the collection of the OES data to its application for virtual metrology and anomaly detection. Particular focus is on the development of supervised and unsupervised variable selection techniques, a fundamental step for any practical application of the OES data. In the unsupervised context, the aim of these algorithms is the definition of a set of variables that are able to summarise the full original data $\mathbf{X}$. In a supervised context the aim is to identify the smallest set of variables from $\mathbf{X}$ that allows an external signal $\mathbf{y}$ to be reconstructed.

1.5 Contributions

This thesis claims the following original contributions:

1. A spectral alignment procedure which aligns OES measurements from multiple sensors through a retrospective calibration process that corrects for inter-spectrometer variation in the mapping from wavelengths to spectrometer channels based on the Particle Swarm Optimization

algorithm [18]. The calibration process involves estimation of a correction function using Particle Swarm Optimization. PSO is needed because the cost function associated with the problem is non convex and multi-modal.

2. An investigation of the performance of the lasso algorithm with different types of cross-validation and the development of improved versions of lasso denoted as: High Frequency, High Mean, Monte Carlo High Frequency and Monte Carlo High Mean. The proposed methods stabilize the set of variables selected by lasso and have equivalent or lower prediction error to competing approaches.

3. A detailed presentation of the Forward Selection Component Analysis (FSCA) algorithm and the development an analysis of computationally efficient algorithm implementations [19].

4. A number of new variants of the FSCA algorithm that incorporate a refinement step to improve performance are proposed.

5. The development of nonlinear extensions of the Forward Selection Component Analysis and Forward Selection Variables algorithms. The newly proposed methods have roughly the same computational complexity as their linear counterparts but result in much better performance [20].

6. An unsupervised variables selection algorithm based on deep neural networks is proposed. This is competitive with and sometimes better than the linear-in-the-parameter nonlinear extension of Forward Selection Component Analysis (FSCA), but at the expense of higher computational complexity.

7. The effect of dimensionality reduction on anomaly detection is investigated. Forward Selection Independent Variables is proposed as a new unsupervised variables selection algorithm specifically designed for anomaly detection [21].

Other contributions in the thesis includes:

1. A review of the methodologies used to define a solution of the lasso estimator.

2. A review of unsupervised variables selection algorithms and a definition of data decomposition and reconstruction in an unsupervised variables selection context.

3. Extreme Learning Machines (ELM) neural networks are for the first time applied to a virtual metrology problem. Methodologies to automatically select the number of hidden nodes and weights initialization of ELMs are proposed [22].

4. A review of unsupervised anomaly detection algorithms for anomaly detection with OES data.

5. The Unsupervised Random Forest algorithm is for the first time applied for anomaly detection with OES resulting in good performance even for high dimensional datasets [23].

6. The Isolation Forest algorithm is for the first time applied to the OES data for anomaly detection and diagnosis using a newly proposed diagnosis procedure.

7. The Similarity Ratio algorithm for anomaly detection with OES time series is generalized and extended to multivariate anomaly detection.

8. A first application of Forward Selection Independent Variable Selection and One Class Support Vector Machine algorithms to OES time series data based anomaly detection [21].

1.5.1 List of Publications

- Puggini Luca, John Doyle and Seán McLoone. "Towards multi-sensor spectral alignment through post measurement calibration correction." *Irish Signals and Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014). 25th IET. IET, 2013.*

- Puggini Luca, John Doyle and Seán McLoone. "Fault Detection using Random Forest Similarity Distance." *IFAC-PapersOnLine 48.21 (2015): 583-588.*

- Puggini Luca and Seán McLoone. "Extreme learning machines for virtual metrology and etch rate prediction." *Signals and Systems Conference (ISSC), 2015 26th Irish. IEEE, 2015.*

- Puggini Luca and Seán McLoone. "Feature Selection for Anomaly Detection Using Optical Emission Spectroscopy." *4th IFAC International Conference on Intelligent Control and Automation Sciences (ICONS 2016).*

- Puggini Luca and Seán McLoone. "Nonlinear Forward Selection Component Analysis for Optical Emission Spectroscopy Wavelength Selection." *Signals and Systems Conference (ISSC), 2016 27th Irish. IEEE, 2015.*

- Puggini Luca and Seán McLoone. "Forward Selection Component Analysis: Algorithms and Applications." under review at *IEEE transactions on pattern analysis and machine intelligence.*

1.6 Thesis Structure

The reminder of the thesis is organised as follows:

Chapter 2 begins with an introduction to plasma, plasma etching and Optical Emission Spectroscopy (OES) plasma measurements. In the chapter a formal mathematical framework to describe OES time series is introduced, the PSI and J2M datasets are introduced as industrial case studies and the multi-chambers matching problem is discussed. Finally, a novel spectral alignment procedure that corrects for wavelength misalignment between spectrometers is developed.

Chapter 3 focuses on supervised variables selection and the lasso estimator. The stability of the lasso is investigated and algorithms to detect a stable set of variables are proposed. Evaluations are performed and results presented for a series of benchmark datasets.

Chapter 4 presents novel research on linear unsupervised variables selection. It starts with a review of the most popular unsupervised variables selection algorithms and then provides a detailed description of the Forward Selection Component Analysis algorithm. This is extended though the development of an alternative implementation and the introduction of different refinement steps. The extended methodology is compared with the basic ones and similar algorithms taken from the literature using a series of simulated and real-world case studies.

Chapter 5 extends the linear unsupervised variables selection algorithms described in the previous chapter through the introduction of multivariate non-linear models. The chapter starts with a description of neural networks and the Extreme Learning Machines algorithm. It is shown how the method can be improved and used with minimal user intervention. Extreme Learning

Machines Neural Networks are then used with other linear-in-the-parameters nonlinear multivariate algorithms for nonlinear unsupervised variables selection. The use of a multilayer neural network for unsupervised variables selection is also investigated. Comparison is performed between the new nonlinear algorithms and linear principal component analysis based unsupervised variable selection and linear dimensionality reduction.

Chapter 6 focuses on anomaly detection with application to OES data. The chapter starts with a survey of unsupervised anomaly detection algorithms that work well with highly dimensional datasets. In the second part of the chapter the effect of dimensionality reduction on anomaly detection is investigated. Here, an unsupervised variables selection algorithm specifically designed for anomaly detection is proposed. In the last part of the chapter, the proposed techniques are investigated for anomaly detection using an OES time series dataset as a case study.

Chapter 7 provides a concluding summary of the work done and of the proposed methodologies as well as possibilities for future research.

1.7 Notation

The mathematical notation and conventions used in the thesis are introduced here. This notation holds in most circumstances with a few exceptions where changes are appropriately introduced.

- Matrices are indicated with a bold capital letter as: $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{W}, \mathbf{\Lambda}, \ldots$. In general $\mathbf{X}$ denotes an input dataset and $\mathbf{Z}$ the matrix obtained with a subset of the columns of $\mathbf{X}$.

- In general, when referring to a dataset $\mathbf{X}$, n is the number of samples and p is the number of variables. Each column of $\mathbf{X}$ corresponds to a variable and each row to a sample. It follows that $\mathbf{X} \in \mathbb{R}^{n \times p}$. k is often used to indicate the number of features extracted from $\mathbf{X}$ and if $\mathbf{Z}$ is the matrix composed of those features it follows that $\mathbf{Z} \in \mathbb{R}^{n \times k}$.

- Column vectors containing the n measurements for a given variable are indicated with a bold lower case letter as: $\mathbf{x}, \mathbf{y}, \mathbf{z}, \cdots \in \mathbb{R}^n$. The elements of each vector are indicated with the same letter with an index determining the position in the vector e.g. $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.

- A row vector containing the measurements of different variables for a

given observation is represented as a bold lower case letter with an arrow on the top: $\vec{\mathbf{x}}, \vec{\mathbf{y}}, \vec{\mathbf{z}}, \cdots \in \mathbb{R}^p$

- Matrices can then be obtained by horizontally stacking column vectors or by vertically stacking row vectors, that is:

$$\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p) = \begin{pmatrix} \vec{\mathbf{x}}_1 \\ \vec{\mathbf{x}}_2 \\ \cdots \\ \vec{\mathbf{x}}_n \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} & \ldots & x_{1,p} \\ x_{2,1} & x_{2,2} & \ldots & x_{2,p} \\ & & \cdots & \\ x_{n,1} & x_{n,2} & \ldots & x_{n,p} \end{pmatrix} \in \mathbb{R}^{n \times p}$$

- Given a probability distribution $\mathcal{D}$ (for example $N(0, 1)$), p random variables following the distribution are indicated as $x_1, \ldots, x_p \sim \mathcal{D}$. There is often an implicit transformation between the random variable x_i and the vector containing the sampled values $\mathbf{x}_i$. For simplicity, in some cases, we will write $\mathbf{x} \sim \mathcal{D}$ with $\mathbf{x} \in \mathbb{R}^n$ indicating the vector contains n observations sampled from the $\mathcal{D}$ distribution.

Chapter 2

Optical Emission Spectroscopy: Collection and Representation

2.1 Introduction

The focus of this thesis is on the development of computational intelligence techniques for Optical Emission Spectroscopy (OES) data analysis. Dealing with OES data is, in general, challenging due to its size and complexity. Before performing any analysis on the data or using it for any industrial application it is important to have a good understanding of its characteristics and how it is collected. With this in mind, the aim of this chapter is to provide a detailed description of OES data, focusing in particular on problems related to data collection and data representation. In this sense, a mathematical formalism to describe OES time series data is reported and a data format that facilitates the analysis of differences between wafers in a time interval during the production is proposed. In an industrial environment, OES data is commonly collected with different sensors. Nonlinearities in the response of OES sensors and errors in their calibration lead to discrepancies in observed wavelength detector responses, with the result that wavelengths are misaligned when comparing data from different spectrometers. As the quality of the available measurements strongly influences the performance of any analytical method the multi-sensor matching problem is investigated and a procedure based on Particle Swarm Optimization (PSO, [24]) developed to retrospectively align OES data from multiple sensors.

The chapter is divided into three main sections. Section 2.2 contains a basic introduction to plasma, a description of the plasma etching chamber and of the optical emission spectrometer from which the OES measurements are col-

lected. In section 2.3 a mathematical description of the OES data is provided and some of the industrial case study datasets that will be used throughout the thesis are introduced. The $\mathbf{\Lambda}$ and $\mathbf{W}$ data formats are also introduced. They are particularly important for Chapter 6 where anomaly detection with OES data is investigated. Finally, section 2.5 describes the multi-sensor spectral alignment problem and the proposed retrospective alignment algorithm.

2.2 Plasma Etching and Optical Emission Spectroscopy

In this section the scientific principles of optical emissions from plasma and measurement techniques are introduced. In particular, a description of some basic plasma, plasma etching and optical emission spectrometer concepts are provided.

2.2.1 Plasma

Plasma is a form of matter in which many of the electrons wander around freely among the nuclei of the atoms. Normally the electrons stay with the same atomic nucleus but in a plasma, a significant number of electrons have such high energy levels that no nucleus can hold them. In a plasma the generation of the electrons and ions results from a series of collisions, which are referred to as electron impact ionization, excitation, relaxation and recombination [25].

Definition 2.2.1 (Ionization with electron)**.** When an incoming ion or electron with enough energy collides with an atom, the outermost electron of this atom can absorb energy to break the electric potential barrier that originally bound it to the atom. This results in a free moving electron and an equally charged ion. Defining A as the atom, the ionization of A can be expressed as:

$$e^- + A \rightarrow 2e^- + A^+ \tag{2.1}$$

Definition 2.2.2 (Excitation)**.** Excitation refers to the process of a plasma atom being activated to a higher energy level when colliding with a free moving electron, but where the absorbed energy is not enough, to break the electric potential barrier to form a free moving electron. The process can be summarised as:

$$e^- + A \rightarrow A^* + e^- \tag{2.2}$$

where A^* represents the excited atom.

In other words excitation results in the alteration of the state of an atom, ordinarily, from the condition of lowest energy (ground state) to one of higher energy (excited state). Excited atoms return to their original state by releasing energy through a phenomena called relaxation.

Definition 2.2.3 (Relaxation)**.** Relaxation refers to the process of the electron in an electronically excited atom transiting from a higher energy level to a lower energy level with excess energy released in the form of a photon.

$$A^* \rightarrow A + E \ \textit{(photon)} \tag{2.3}$$

A photon is a discrete packet of energy associated with electromagnetic radiation (light). The wavelength of the emitted light corresponds to exactly the energy difference between the two energy levels with

$$E = \frac{hc}{\lambda} \tag{2.4}$$

where λ denotes the wavelength of the photon, c denotes the speed of light and h is Planck's constant. Thus, an atom emits light at only certain discrete wavelengths. Each atomic and molecular species has its own unique spectral signature, hence by analysing the optical emission from a plasma its composition can be determined. This phenomenon leads to the characteristic light emission of a plasma.

Definition 2.2.4 (Recombination)**.** Recombination refers to the process of an electron being combined with an ion to form a neutral atom. However, a third body is required to take part in the process to allow the recombination to satisfy the conservation of energy and momentum requirements [25]. The recombination process can be expressed as:

$$e^- + A^+ + A \rightarrow A + A \tag{2.5}$$

2.2.2 Plasma Etching

In semiconductor manufacturing, plasma is, in general, generated though the application of microwave energy to a gas. Radio Frequency (RF) directs the ions toward the wafer surface where they interact both chemically and physically with the silicon wafer, etching away the exposed surface [26]. Figure 2.1 provides a graphical representation of the plasma etching process.

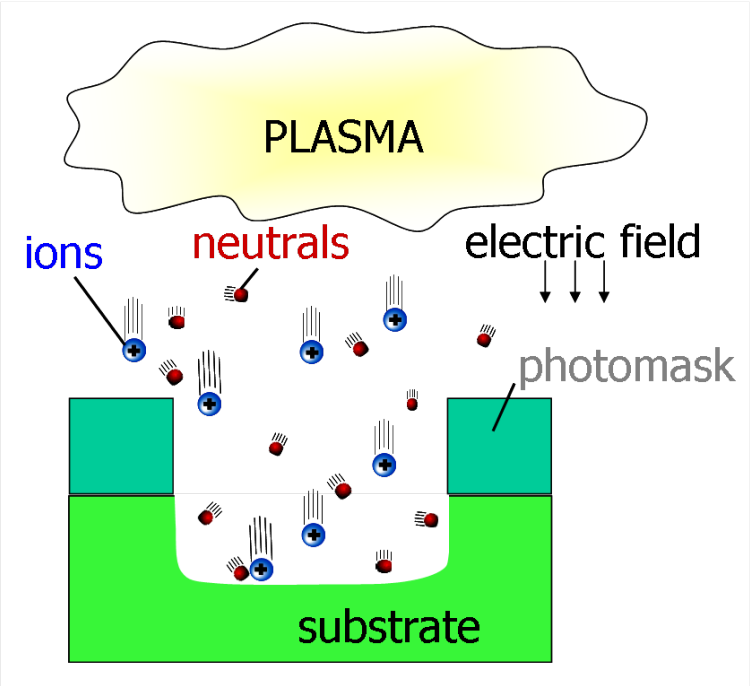


Figure 2.1: Plasma etching process. The ions bombard the exposed surface. Figure from `https://www.scorec.rpi.edu`.

2.2.2.1 Electron Cyclotron Resonance Plasma Etcher

Plasma etching takes place in a plasma etcher. In this section an Electron Cyclotron Resonance plasma etcher is considered.

Definition 2.2.5 (Electron Cyclotron Resonance)**.** Electron Cyclotron Resonance (ECR) refers to the phenomenon according to which a electron in a static and uniform magnetic field will move in a circle due to the Lorentz force.

The ECR plasma etcher makes use of microwave energy and a strong magnetic field to produce a low pressure and high density plasma and provides the necessities for achieving plasma etching [27]. The main components of an ECR etcher include a magnetic field generation system, a microwave oscillator, a gas supply system, a Radio Frequency (RF) generator and an etch chamber, as illustrated in Figure 2.2. A microwave power supply generates microwave which are oscillated by a magnetron, transmitted along a waveguide and injected into a quartz plate. The microwaves produce a dynamic electric field, which is perpendicular to the static magnetic field, which is generated as a DC current flowing through the solenoid coils. The interaction of these two fields generates a Lorentz Force, which causes the electrons to spiral in a helical motion. In this way, the microwaves transfer the energy to free electrons which in turn accelerate and collide with the atoms or molecules in the gas and produce ionization. The low gas pressure, which helps to reduce electron impact recombinations, is achieved by controlling the flow rates of the gases supplied to the chamber. A separate RF bias is applied to the wafer electrode to independently control ion energy at the wafer surface. The wafer temperature, which is an important factor influencing the uniformity of etch across the wafer surface, is reduced with the use of helium. An important feature of the ECR etcher is that ion energies can be controlled separately by the RF supply, allowing much greater control of etch rate. An OES (Optical Emission Spectroscopy) sensor and a PIM (Plasma impedance monitor) sensor are also shown in the figure. These can be used to monitor the optical and electrical characteristics of the plasma, respectively. In this thesis, we are concerned exclusively with analysing OES data.

2.2.3 Optical Emission Spectrometer

Analysis of plasma emission spectra can be used to estimate the instantaneous composition of a plasma over time. An Optical Emission Spectrometer is an optical device used to detect the optical emissions of plasma species,

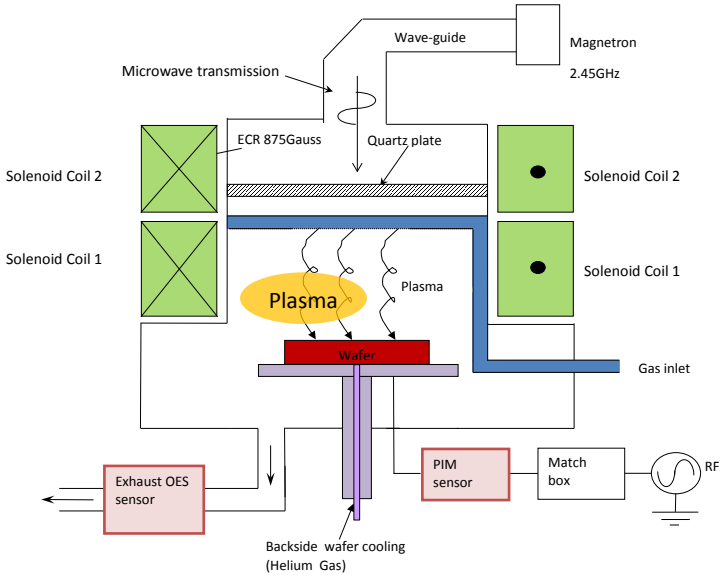


Figure 2.2: Illustration of the basic features of a plasma etching chamber.

providing direct information on plasma chemistry [11]. In Optical Emission Spectroscopy (OES), visible light is collected and redirected onto a Charged Coupled Device (CCD) detector with different wavelengths dispersed to different CCD [28]. The key component of a typical Optical Emission Spectrometer is the CCD detector. CCDs are a type of quantum detectors, which are used to measure the flux of photons. CCDs have been widely employed in modern optical detection devices for their fast response time and sensitivity to small photon fluxes.

2.2.3.1 Data Collection During Plasma Etch Processing

In a production process wafers are usually grouped in lots, with wafers in a lot arranged in slots on a cassette. Wafers in a lot are processed sequentially (according to slot number) undergoing several etching steps. Lots are are also processed sequentially through etch chambers, interspersed with cleaning and maintenance operations. Cleaning cycles are typically done between each lot to remove the by-products of plasma etching that build up on the chamber walls, and are detrimental to etching performance. This leads to a chamber seasoning effect during the first few wafers processed following each cleaning cycle, and consequently slot dependent differences in processed wafers. A consequence of this is shown later in Figure 2.11.

In this work the considered OES datasets are collected using Ocean Optics USB2000 spectrometers with CCD detectors consisting of a variable number of channels each one recording a single wavelength. Wavelength intensity measurements are taken without interruption with a fixed sampling rate during processing of wafers. This results in chronologically ordered values for a set of wafers. Figure 2.3 shows the graphical representation of the OES measurements for an example wafer. In this example wavelength intensity was measured during 40 seconds of plasma etching, at a sampling rate of 1 second for $p = 2000$ channels.

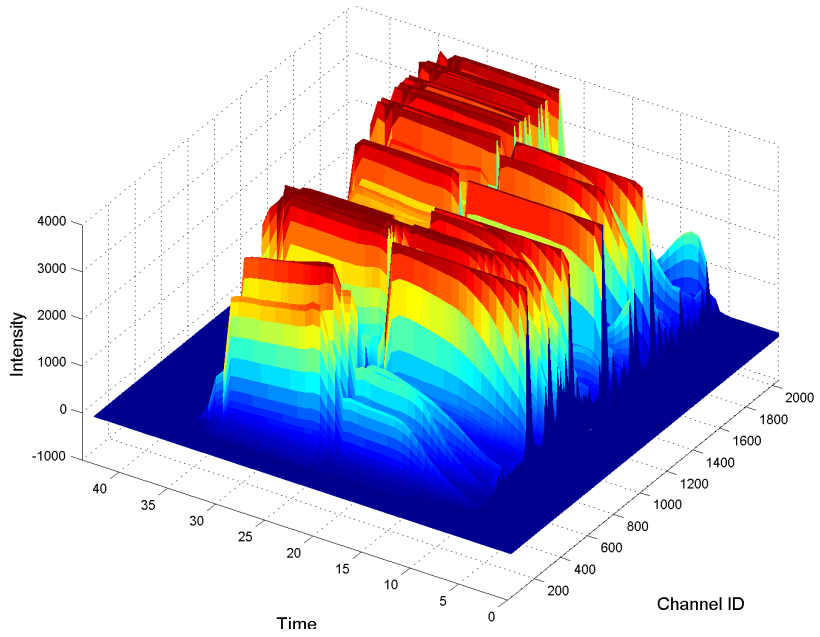


Figure 2.3: Plasma etch OES data for a single wafer ($\mathbf{X}_k$), recorded over two etching steps. The units on the time axis are in seconds.

2.3 Mathematical Representation of OES Data

In the previous section the physical concepts behind plasma and the tools that are used to collect the OES measurements are described. In a production environment the OES measurements are collected continually at a fixed sampling interval while the wafers are processed resulting in OES time series data. In this section the OES time series data is described and some datasets that will be used throughout the thesis are introduced.

2.3.1 Wafer Measurements

As noted in [12] plasma etch processing OES data is naturally organized in three dimensions: Wafer; Time; and Wavelength. With this in mind it is reasonable to represent a generic element of the OES data as:

$$x_i^{w_k}(t) \tag{2.6}$$

This represents the intensity of the i^{th} wavelength at time t for the k-th wafer. In this chapter K represents the total number of wafers and p the number of wavelengths. It is assumed that the whole production ends after T samples and that during production τ equally spaced measurements are taken for each wafer. While this is not entirely true, it is a good approximation of the reality and it is required in order to define a formal structure for analysis. Given these assumptions the OES spectrum for a single wafer w_k can be mathematically represented as a matrix $\mathbf{X}_k \in \mathbb{R}^{\tau \times p}$, where τ is the number of time samples and p is the number of wavelengths measured during etching, that is:

$$\mathbf{X}_k = \begin{pmatrix} x_1^{w_k}(t_{(k-1)\tau+1}) & x_2^{w_k}(t_{(k-1)\tau+1}) & \cdots & x_p^{w_k}(t_{(k-1)\tau+1}) \\ x_1^{w_k}(t_{(k-1)\tau+2}) & x_2^{w_k}(t_{(k-1)\tau+2}) & \cdots & x_p^{w_k}(t_{(k-1)\tau+2}) \\ & & \cdots & \\ x_1^{w_k}(t_{(k-1)\tau+\tau}) & x_2^{w_k}(t_{(k-1)\tau+\tau}) & \cdots & x_p^{w_k}(t_{(k-1)\tau+\tau}) \end{pmatrix} \in \mathbb{R}^{\tau \times p} \tag{2.7}$$

In some cases, in order to simplify the notation, $x_i^{w_k}(t_{(k-1)\tau+j})$ is denoted as $x_i^{w_k}(t_j)$. OES time series datasets thus st of as a set of matrices where each one contains the spectrum for a given wafer:

$$S = \left\{ \mathbf{X}_j \in \mathbb{R}^{\tau \times p} : \ j = 1, \ldots, K \right\} \tag{2.8}$$

Under the assumption that measurements are recorded at τ time points for each wafer, the data in the set S can then be represented as a three dimensional matrix.

$$\mathbf{X} \in \mathbb{R}^{K \times p \times \tau} \tag{2.9}$$

where K corresponds to the number of wafers, p to the number of wavelenghts and τ is the number of samples recorded for each wafer. Similar representations are reported in [29].

For practical applications this three dimensional matrix can be reshaped in a two dimensional matrix in three different ways:

- Aggregated by Time Point. The data is stored in a matrix $\mathbf{\Lambda} \in \mathbb{R}^{\tau K \times p}$

- Aggregated by Wafer. The data is stored in a matrix $\mathbf{W} \in \mathbb{R}^{K \times p\tau}$
- Aggregated by Wafer Processing Time. The data is stored in a matrix $\mathbf{F} \in \mathbb{R}^{\tau \times pK}$

A deeper description and an analysis of the data aggregated in the three matrices $\mathbf{\Lambda}, \mathbf{W}$ and $\mathbf{F}$ will follow in the next sections.

2.3.2 OES by Time Point

The data can be aggregated in a $\mathbf{\Lambda} \in \mathbb{R}^{\tau K \times p}$ matrix. Here each row contains a sample instant and each column corresponds to a wavelength. This format corresponds to the way the data is usually collected during production as the samples (rows) are chronologically ordered. $\mathbf{\Lambda}$ can be formed by vertically stacking the set of matrices S, i.e.

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \cdots \\ \mathbf{X}_K \end{pmatrix} \in \mathbb{R}^{K\tau \times p} \tag{2.10}$$

More specifically,

$$\mathbf{\Lambda} = \begin{pmatrix} x_1^{w_1}(t_1) & x_2^{w_1}(t_1) & \cdots & x_p^{w_1}(t_1) \\ x_1^{w_1}(t_2) & x_2^{w_1}(t_2) & \cdots & x_p^{w_1}(t_2) \\ \cdots & \cdots & \cdots & \cdots \\ x_1^{w_1}(t_\tau) & x_2^{w_1}(t_\tau) & \cdots & x_p^{w_1}(t_\tau) \\ x_1^{w_2}(t_{\tau+1}) & x_2^{w_2}(t_{\tau+1}) & \cdots & x_p^{w_2}(t_{\tau+1}) \\ \cdots & \cdots & \cdots & \cdots \\ x_1^{w_K}(T) & x_2^{w_K}(T) & \cdots & x_p^{w_K}(T) \end{pmatrix} \tag{2.11}$$

or equivalently

$$\mathbf{\Lambda} = \begin{pmatrix} x_1^{w_1}(t_1) & x_2^{w_1}(t_1) & \cdots & x_p^{w_1}(t_1) \\ x_1^{w_1}(t_2) & x_2^{w_1}(t_2) & \cdots & x_p^{w_1}(t_2) \\ \cdots & \cdots & \cdots & \cdots \\ x_1^{w_1}(t_\tau) & x_2^{w_1}(t_\tau) & \cdots & x_p^{w_1}(t_\tau) \\ x_1^{w_2}(t_1) & x_2^{w_2}(t_1) & \cdots & x_p^{w_2}(t_1) \\ \cdots & \cdots & \cdots & \cdots \\ x_1^{w_K}(t_\tau) & x_2^{w_K}(t_\tau) & \cdots & x_p^{w_K}(t_\tau) \end{pmatrix} \tag{2.12}$$

Each column of the $\mathbf{\Lambda}$ matrix is a time series representing a wavelength over K wafers. Two sample columns of the $\mathbf{\Lambda}$ matrix are plotted in Figure 2.4. The figure shows the periodic repetition of values that corresponds to the

different wafers. According to [12] this data format is particularly useful when performing wavelength selection. This simply follows from the fact that each column is a different wavelength. An example of this will be shown in Chapter 6 as an application of anomaly detection with OES data.

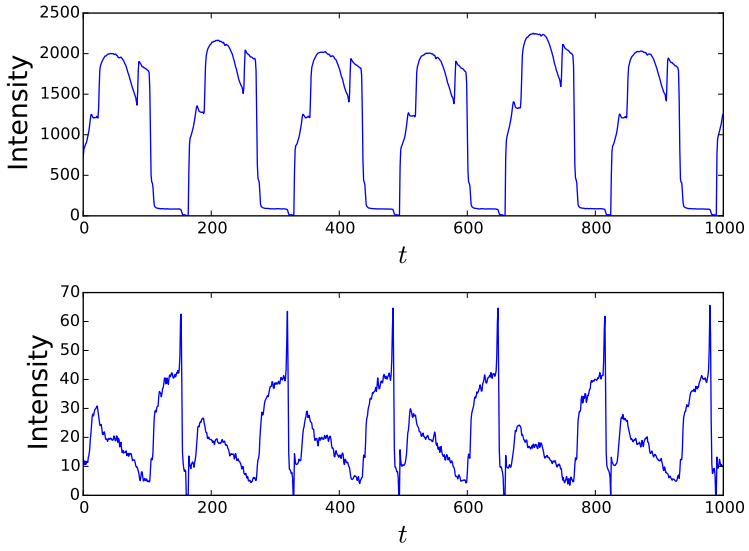


Figure 2.4: Two sample columns of the $\mathbf{\Lambda}$ matrix.

2.3.3 OES by Wafer

Alternatively the wafers in S can be aggregated in the wafer format as a matrix $\mathbf{W}$. In the $\mathbf{W}$ matrix each row corresponds to a wafer and each column contains the measurements of a given wavelength at a given time. $\mathbf{W}$ is obtained by transforming the elements of S into row vectors and stacking them vertically. Mathematically given a sample element of S:

$$\mathbf{X}_k = \begin{pmatrix} x_1^{w_k}(t_1) & x_2^{w_k}(t_1) & \cdots & x_p^{w_k}(t_1) \\ x_1^{w_k}(t_2) & x_2^{w_k}(t_2) & \cdots & x_p^{w_k}(t_2) \\ \cdots & \cdots & \cdots & \cdots \\ x_1^{w_k}(t_\tau) & x_2^{w_k}(t_\tau) & \cdots & x_p^{w_k}(t_\tau) \end{pmatrix} \in \mathbb{R}^{\tau \times p} \tag{2.13}$$

it is reshaped as a row vector

$$\tilde{\mathbf{X}}_k = (x_1^{w_k}(t_1), \ldots, x_p^{w_k}(t_\tau)) \in \mathbb{R}^{1 \times \tau p} \tag{2.14}$$

and the $\mathbf{W}$ representation is obtained by combining all the reshaped matrices in S to give:

$$\mathbf{W} = \begin{pmatrix} \tilde{\mathbf{X}}_1 \\ \cdots \\ \tilde{\mathbf{X}}_K \end{pmatrix} \in \mathbb{R}^{K \times \tau p} \tag{2.15}$$

or equivalently

$$\mathbf{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_p) \in \mathbb{R}^{K \times \tau p} \tag{2.16}$$

where

$$\mathbf{W}_i = \begin{pmatrix} x_i^{w_1}(t_1) & x_i^{w_1}(t_2) & \cdots & x_i^{w_1}(t_\tau) \\ x_i^{w_2}(t_1) & x_i^{w_2}(t_2) & \cdots & x_i^{w_2}(t_\tau) \\ \cdots & \cdots & \cdots & \cdots \\ x_i^{w_K}(t_1) & x_i^{w_K}(t_2) & \cdots & x_i^{w_K}(t_\tau) \end{pmatrix} \in \mathbb{R}^{K \times \tau} \quad i = 1, \ldots, p \tag{2.17}$$

When the data is stored in the $\mathbf{W}$ format each sample (row) represents a single wafer. This representation of the OES data is convenient for comparing the characteristics of different wafers. It was, for example, used in [30] for an anomaly detection task. A challenge with this representation is that the number of columns (or variables) increases dramatically.

2.3.3.1 Time Window

In some circumstances it is important to analyse the difference between wafers at a given time interval during the etching process. This is for example used in Chapter 6 to understand when during the production a fault occurred and can be used to perform on-line analysis during the etching process. With this in mind an alternative data representation based on the $\mathbf{W}$ format is proposed as follows.

Taking into account the fact that each column of $\mathbf{W}$ is associated with a different sample instant/wavelength combination, the variables can be grouped in matrices according to their t value. Sorting the columns of the $\mathbf{W}$ matrix (Equation 6.57) in chronological order it is possible to write:

$$\mathbf{W}_T = (\mathbf{W}^{t_1}, \mathbf{W}^{t_2}, \ldots, \mathbf{W}^{\tau}) \tag{2.18}$$

where

$$\mathbf{W}^{t_i} = \begin{pmatrix} x_1^{w_1}(t_i) & x_2^{w_1}(t_i) & \cdots & x_p^{w_1}(t_i) \\ x_1^{w_2}(t_i) & x_2^{w_2}(t_i) & \cdots & x_p^{w_2}(t_i) \\ & & \cdots & \\ x_1^{w_K}(t_i) & x_2^{w_K}(t_i) & \cdots & x_p^{w_K}(t_i) \end{pmatrix} \in \mathbb{R}^{K \times p} \quad i = 1, \ldots, \tau \tag{2.19}$$

The matrix

$$\mathbf{W}_{t_i}^{t_j} = (\mathbf{W}^{t_i}, \mathbf{W}^{t_{i+1}}, \ldots, \mathbf{W}^{t_j}) \in \mathbb{R}^{K \times (t_j - t_i)p} \tag{2.20}$$

contains the information required to analyse the process between time t_i and t_j. If the time interval $[t_i, t_j]$ is sufficiently small the dimension $\mathbf{W}_{t_i}^{t_j}$ will be small enough to avoid the need for further dimensionality reduction. Increasing the value of t_i and t_j it is possible to analyse the full process using the full information available for a given time interval and to use only $p(t_j - t_i)$ variables each time. In Chapter 6 this approach is used for anomaly detection and to understand when during a wafer processing the anomaly occurred.

2.3.4 OES by Time

The third and final way to aggregate the data is to stack the matrices in S horizontally, that is

$$\mathbf{F} = (\mathbf{X}_1, \ldots, \mathbf{X}_K) \in \mathbb{R}^{\tau \times pK} \tag{2.21}$$

where $\mathbf{X}_k$, as defined in equation 2.7, contains the OES measurements for the k^{th} wafer. The $\mathbf{F}$ matrix can be used to analyse the process in time. For example a PCA analysis of $\mathbf{F}^T$ can detect redundancy along the time dimension of the data, suggesting that it is sufficient to analyse the process at only few instants (sampling points). The $\mathbf{F}$ matrix is not used for any application in this thesis and it is reported only for completeness.

2.4 Industrial Case Studies

In this section two industrial datasets are introduced. These will be used as case studies throughout the thesis.

2.4.1 PSI Time Series

The first production dataset referred to as the PSI Time Series (PSI) dataset. This dataset contains all OES measurements collected from a single plasma etch chamber over several months.

Dataset 2.4.1 (PSI Time Series)**.** The PSI dataset contains the measurements of $p = 1747$ wavelengths related to the production of $K = 1006$ wafers divided in 83 lots (with up to 25 wafers per lot). All the wafers were processed according to the same recipe and processed in 7 steps for 3 different

products. The number of samples recorded for each wafer varied in the range $\tau \in [184, 192]$. Following an initial preprocessing step to remove irrelevant data the number of time samples per wafer was reduced to $\tau = 165$ with a sampling period of 0.5 seconds. The reasons for and the method used to achieve this reduction are described later in the thesis (Observation 2.4.1). After the preprocessing step the reference values for this datasets are $K = 1006$, $\tau = 165$ and $p = 1747$. A summary of the data is reported in Table 2.1 and Figure 2.5 shows the number of wafers per lot. The figure shows that most lots contain 12 or 13 wafers. This follows from the fact that a lot in general contains only odd or even slot numbers. A few lots contain less than 5 wafers. These slots were probably occupied by particular wafers that were used for monitoring and cleaning. These wafers were removed as they are not relevant for the considered analysis.

Feature	*Total Number*
Wavelengths	1747
Wafers	1006
Time points for each wafer	165
Total time points	190406
Lots	83
Slots	25
Recipes	1
Steps	7

Table 2.1: Summary of the PSI dataset (Dataset 2.4.1)

Figure 2.6 shows the variation in OES spectrum wavelength values during wafer processing. In the figure each coloured line corresponds to a row of the matrix $\mathbf{X}_k$. To aid visualisation only 5 selected rows of $\mathbf{X}_k$ are plotted.
The mean:

$$\bar{\mathbf{S}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{X}_k \in \mathbb{R}^{\tau \times p} \tag{2.22}$$

and the standard deviation:

$$Std(\mathbf{S}) = \frac{1}{K-1}\sum_{k=1}^{K}(\mathbf{X}_k - \bar{\mathbf{S}})^2 \tag{2.23}$$

obtained for the PSI data (Dataset 2.4.1) after some data cleaning steps are shown in Figures 2.7 and 2.8.

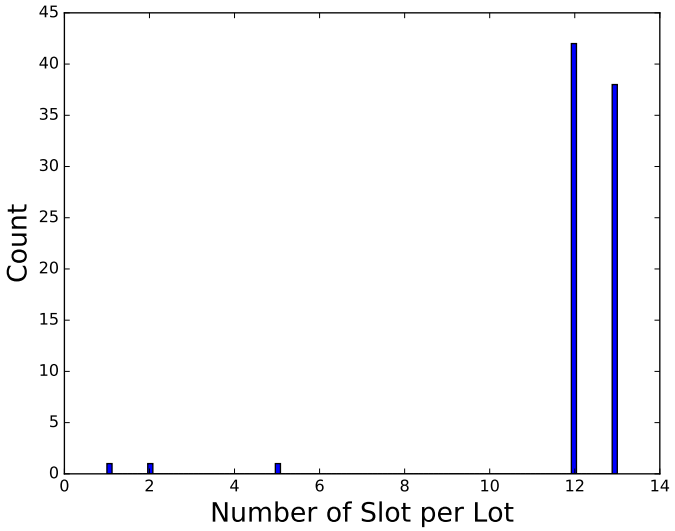


Figure 2.5: Number of slots in each lot for Dataset 2.4.1.

Observation 2.4.1. *The hypothesis that each wafer is processed during the same amount of time τ is required in order to create a one to one relationship between wafers at a given time:*

$$x_i^{w_v}(t_j) \longrightarrow x_i^{w_u}(t_j) \tag{2.24}$$

i. e.

$$x_i^{w_v}(t_{\tau(v-1)+j}) \longrightarrow x_i^{w_u}(t_{\tau(u-1)+j}) \tag{2.25}$$

In a real-time production scenario this is often not exactly true and the data may require a preliminary cleaning step. In the considered data, for example, during the processing of some wafers, the machine started to record wavelength values before wafer processing had actually started. As a consequence the time point measurements for some wafers are preceded by some zeros for each wavelength i.e.

$$\forall j \quad x_j^{w_k}(t_i) = 0 \ for \ i = 0, \ldots, z^{w_k} \tag{2.26}$$

where z^{w_k} is defined as:

$$z^{w_k} = min\left\{z : \ x_j^{w_k}(t_{z+1}) > 0\right\} \tag{2.27}$$

In order to align the start of etching for each wafer these initial zeros are removed, i.e.:

$$x_j^{w_k}(t_1) := x_j^{w_k}(t_{z^{w_k}+1}) \tag{2.28}$$

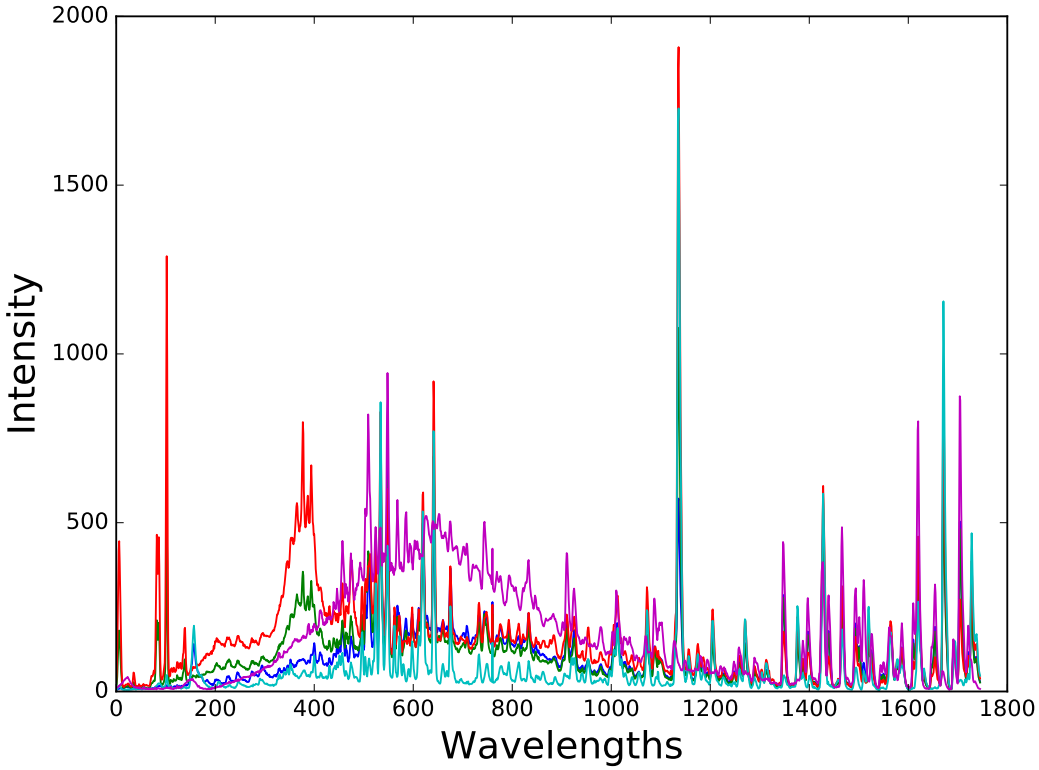


Figure 2.6: An illustration of the variation in OES spectrum wavelength intensities over time for a given wafer. Each line spectrum represents a different time instant (sample).

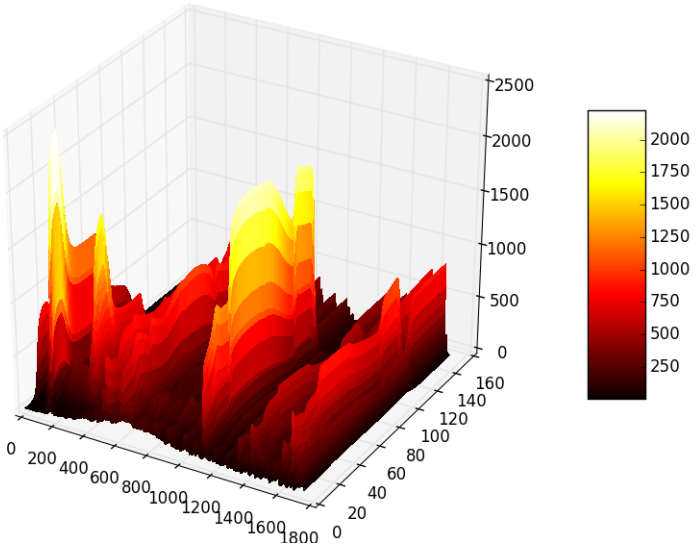


Figure 2.7: The 3-dimensional representation of the mean of the PSI dataset (Dataset 2.4.1) as defined in equation 2.22 for $t < 150$.

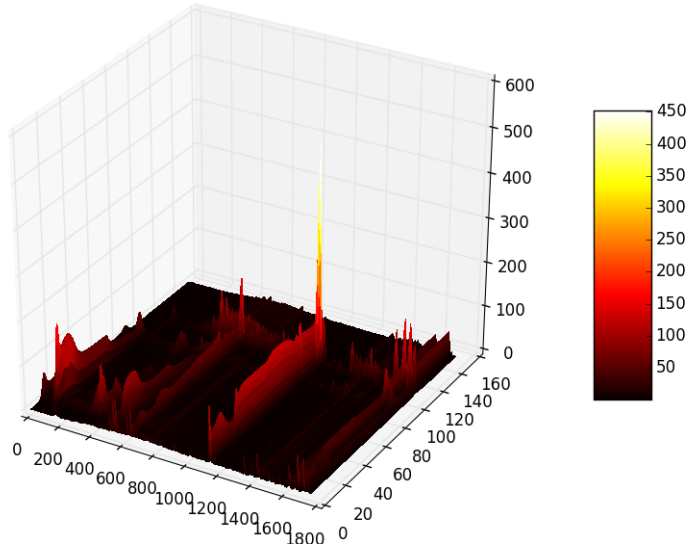


Figure 2.8: The 3-dimensional representation of the standard deviation of the PSI dataset (Dataset 2.4.1) as defined in equation 2.23 for $t < 150$.

Figure 2.9 shows the traces for a given wavelength of several wafers before and after alignment. In the first plot the wafer represented by a red line is slightly shifted with respect to the others while all the wafers are aligned in the second plot. Several problems may arise if the data is not properly aligned. In an anomaly detection context for example the misaligned wafers will be wrongly labelled as an anomaly. A similar problem is investigated in Chapter 6.

Henceforth, unless otherwise stated wafer $\{\mathbf{X}_k\}_{k=1,\dots,K}$ *will be assumed to be aligned and the same length (*τ *samples).*

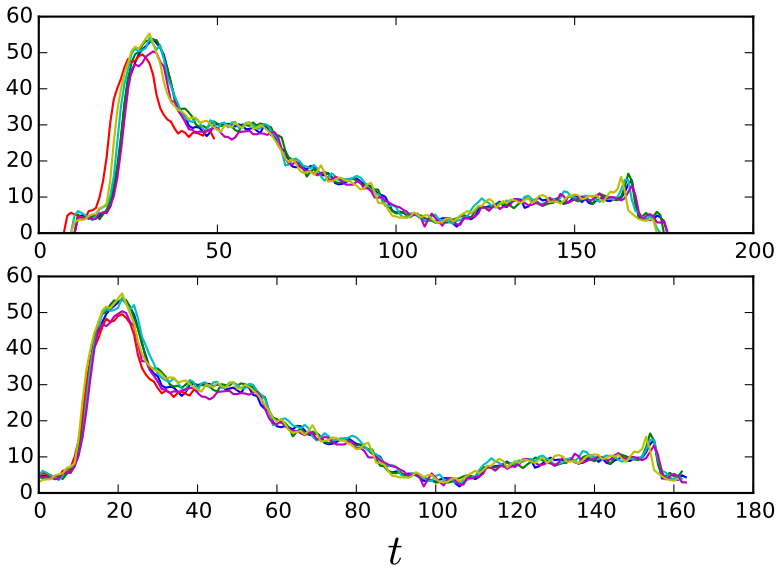


Figure 2.9: A given wavelength for 10 wafers before and after alignment.

2.4.1.1 PCA Analysis of the PSI Data

In order to better understand the structure of the PSI dataset a PCA analysis of the data presented in the $\mathbf{W}$ and $\mathbf{\Lambda}$ formats is performed.

W format PCA: The $\mathbf{W}$ matrix is scaled to unit variance and a PCA decomposition is performed. Due to the large size of the data and for computational efficiency an approximation of PCA is obtained with the Incremental PCA algorithm [31]. The percentage of explained variance is plotted as a function of the number of components in Figure 2.10. The figure shows that

the percentage of explained variance grows quickly with the first 20-30 principal components getting close to the 70%. The following components do not contribute much in terms of explained variance. This reflects the redundancy of the data that can be summarised by only 30 components. The remaining components are associated with variations that are not representative of the majority of the data and are likely to be capturing the noise in the data. Figure 2.11 shows the projection of the **W** matrix onto its first five principal components. In the figure the samples are coloured according to their slot number. Particularly interesting is the projection onto the first principal component. In this case the samples on the left are coloured blue. These correspond to the samples with the lowest slot number. Gradually the colour of the samples changes from blue to red as we move from left to right with red corresponding to samples with the largest slot number. This suggests that the **W** format is particularly useful for detecting inner wafer variations. Indeed, in Chapter 6 good anomaly detection performance is obtained using datasets derived from the **W** format of the data.

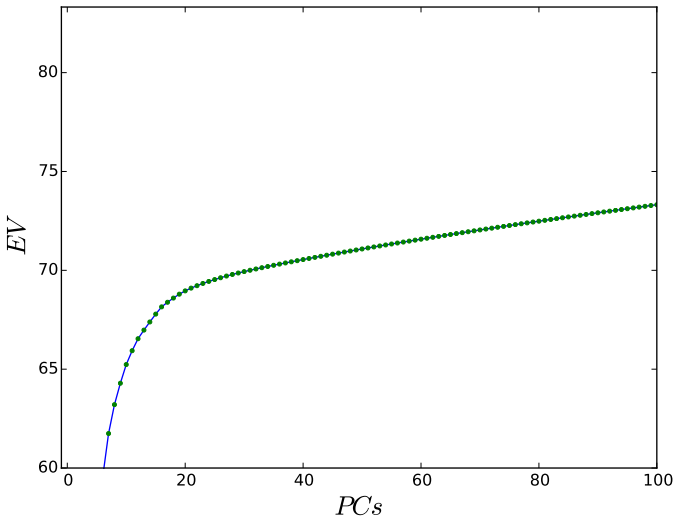


Figure 2.10: The percentage of explained variance by the first 100 *PCs* of the **W** matrix for Dataset 2.4.1. The matrix was scaled in order to have unit variance.

Λ format PCA: The **Λ** matrix is scaled to unit variance and a PCA decomposition is performed. Figure 2.13 shows the percentage of explained variance as a function of the number of principal components. Seven compo-

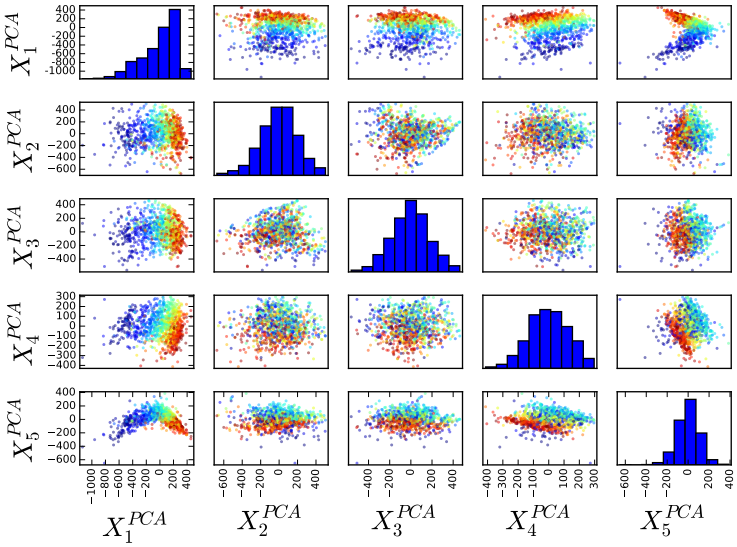


Figure 2.11: Dataset 2.4.1: The scores obtained projecting the **W** matrix on the subspace generated by its first 5 PCs. The data was scaled in order to have unit variance and the samples are coloured according to their slot number.

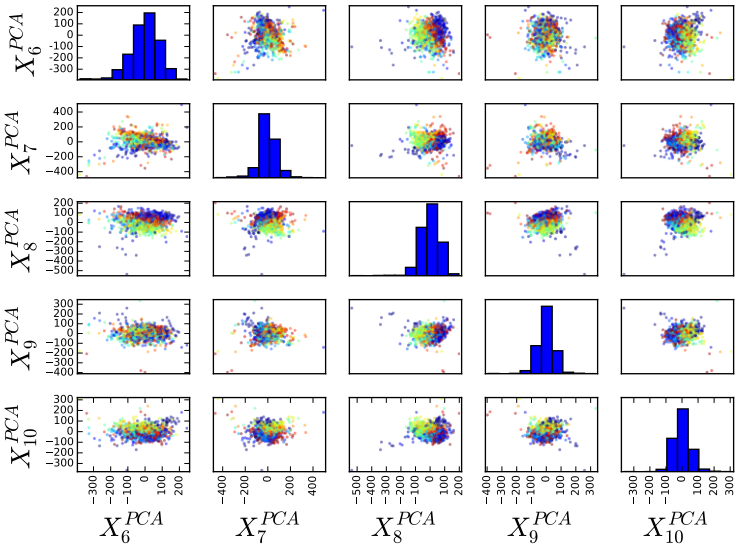


Figure 2.12: Dataset 2.4.1: The scores obtained by projecting the **W** matrix on the subspace generated by PCs 6 to 10. The data was scaled in order to have unit variance and the samples are coloured according to theirs slot number.

nents explain more than 99% of the total variance. This is a consequence of the high level of redundancy in the data. In this data format each column is a wavelength. These are very correlated for physical reasons as they are all measurements of light emitted from plasma as described in section 2.2. Also the rows are highly correlated as they contain the time points that repeat almost periodically for each wafer after τ samples (this was previously shown in Figure 2.4). Figures 2.14 and 2.15 show the projection of the rows of $\mathbf{\Lambda}$ corresponding to two wafers from a randomly selected lot (one from slot 1 and one from slot 25) respectively onto the PCA subspace. It is interesting to observe that the measurements corresponding to the two wafers behave similarly in the subspace defined by the first 3 PCs and start to behave differently onto the subspace generates by later PCs. This is a consequence of the fact that the two wafers have a similar main trend given by the periodical repetition of values characterising each wafer and some higher order differences probably related to the seasoning effect.

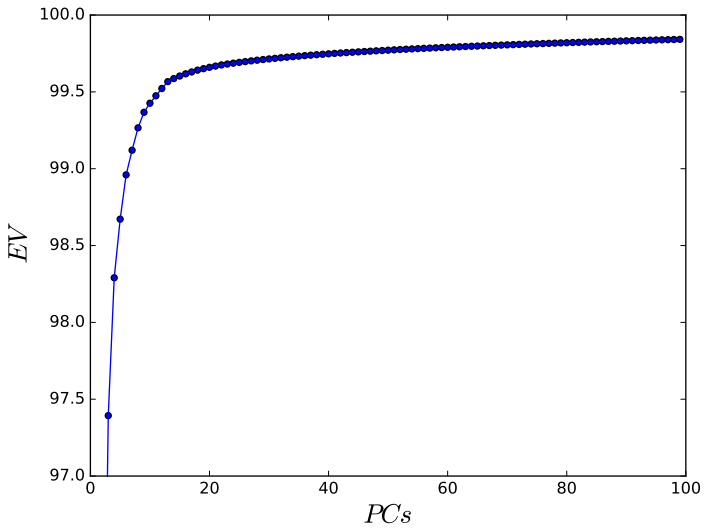


Figure 2.13: The percentage of explained variance by the first 100 PCs of the $\mathbf{\Lambda}$ matrix for Dataset 2.4.1. The matrix was scaled in order to have unit variance.

2.4.2 J2M Dataset

J2M is the second semiconductor manufacturing dataset introduced in this section. It was obtained from an OES time series represented in the $\mathbf{W}$ format through the use of summary statistics as defined in the next paragraph.

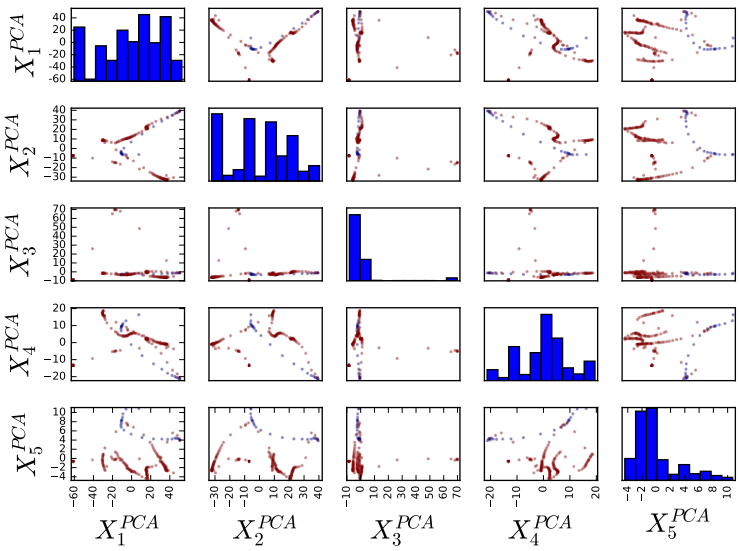


Figure 2.14: Dataset 2.4.1. The scores obtained projecting the $\mathbf{\Lambda}$ matrix on the subspace generated by its first 5 PCs. The data was scaled in order to have unit variance. Only samples corresponding to the measurements of a wafer in slot 1 (blue) and a wafer in slot 25 (red) from a randomly selected lot are plotted.

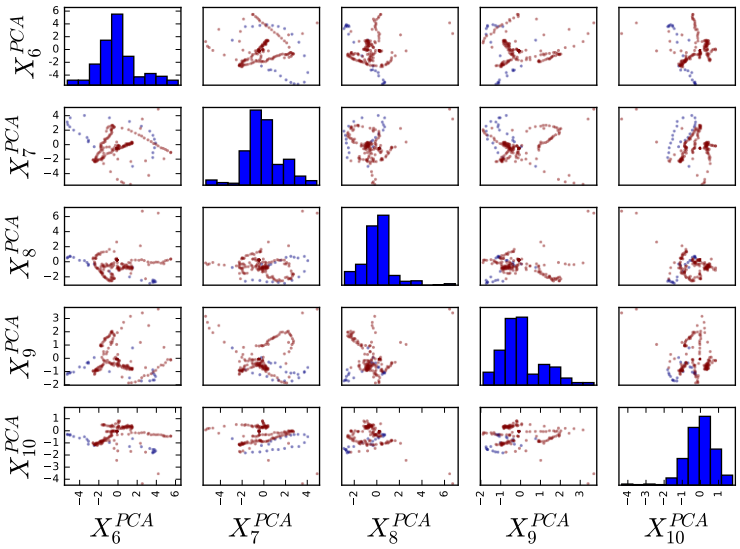


Figure 2.15: Dataset 2.4.1. The scores obtained projecting the $\mathbf{\Lambda}$ matrix on the subspace generated by its 6 to 10 PCs. The data was scaled in order to have unit variance. Only samples corresponding to the measurements of a wafer in slot 1 (blue) and a wafer in slot 25 (red) from a randomly selected lot are plotted.

Summary Statistics: The time interval over which a wafer is processed is split into intervals and the intensity values of each wavelength in each interval are summarised with summary statistics. Using the previously introduced notation the time interval $[t_1, t_\tau]$ is split into l roughly equal intervals $([t_1, t_{i_1}), [t_{i_1}, t_{i_2}), \ldots, [t_{i_l}, t_\tau])$ and the evolution of each wavelength over each interval is summarized using s summary statistics $\{m_i : i = 1, \ldots, s\}$. For simplicity assuming τ is a multiple of l this is mathematically expressed as:

$$(x_i^{w_k}(t_j), x_i^{w_k}(t_{j+1}), \ldots, x_i^{w_k}(t_{j+\tau/l})) \longrightarrow (m_1^i, \ldots, m_s^i)_k^{\left[t_j, t_{j+\tau/l}\right]} \tag{2.29}$$

Defining

$$\mathbf{M}_s^i(t_j, t_{j+\tau/l}) := \begin{pmatrix} (m_1^i, \ldots, m_s^i)_1^{\left[t_j, t_{j+\tau/l}\right]} \\ \ldots \\ (m_1^i, \ldots, m_s^i)_K^{\left[t_j, t_{j+\tau/l}\right]} \end{pmatrix} \tag{2.30}$$

the matrix $\mathbf{W}_i$ defined in equation 2.17 is reduced to:

$$\mathbf{W}_i^{s,l} = \left(\mathbf{M}_s^i(t_1, t_{1+\tau/l}), \ldots, \mathbf{M}_s^i(t_{\tau-\tau/l}, t_\tau)\right) \in \mathbb{R}^{K \times ls} \tag{2.31}$$

Finally, a lower dimensional version of $\mathbf{W}$ is then obtained as:

$$\mathbf{W}^{s,l} := (\mathbf{W}_1^{s,l}, \ldots, \mathbf{W}_p^{s,l}) \in \mathbb{R}^{K \times slp} \tag{2.32}$$

Observe that in general:

$$sl << \tau \tag{2.33}$$

This means that the dimension of $\mathbf{W}^{s,l} \in \mathbb{R}^{K \times slp}$ is much lower than the dimension of $\mathbf{W} \in \mathbb{R}^{K \times \tau p}$.

Dataset 2.4.2 (Joints Two Moments (J2M))**.** The J2M dataset consists of a matrix $\mathbf{X} \in \mathbb{R}^{1600 \times 12288}$ and a vector $\mathbf{y} \in \mathbb{R}^{1600}$. The matrix $\mathbf{X}$ was obtained by summarising time series OES measurements stored in $\mathbf{W}$ matrix format using 6 summary statistics, namely, mean, variance, skewness, kurtosis, min and max. Using the notation introduced earlier the data was reduced using summary statistics with $l = 1$ and $s = 6$. The vector $\mathbf{y}$ contains the Etch Rate (ER) measurement for each wafer. ER is a measure of how fast material is removed from a wafer surface during plasma etch. This measurement is not available during the manufacturing process, and typically is sparsely sampled from groups of processed wafer lots hours after the plasma etching process has finished. Within this dataset, the ER value of each wafer was measured and is plotted in Figure 2.16. In this dataset, ER under normal operation is

defined to be in the range $[66, 72]$. Within this dataset, a process fault was introduced that resulted in the etching rate of certain wafers falling outside permitted control limits. These faulty wafers, positioned near index 150, can be visually identified in Figure 2.16 as the group whose $ER < 60$. Similarly, a process shift was introduced between wafer 950 and wafer 1380.

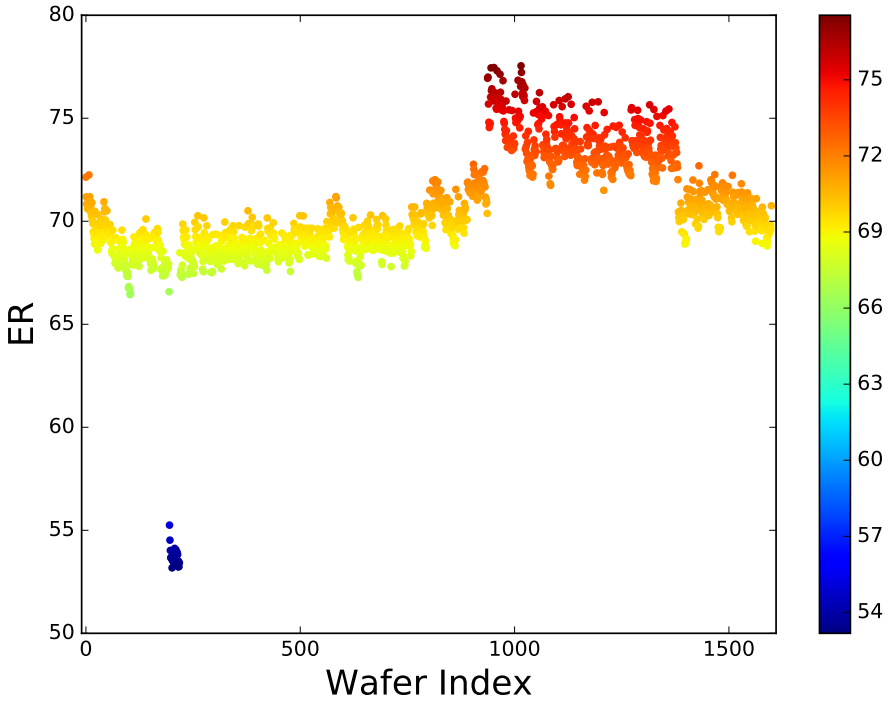


Figure 2.16: The normalized ER for the J2M data (Dataset 2.4.2).

The idea of reducing the dimension of the data by employing summary statistics has been extensively used in the industry and in research. In [32], for example, the dimension of a time series is reduced taking its mean value in a given time interval. In the previous notation this is equivalent to $s = 1$ and $m_1 = mean$. In other studies, for example [33], each time interval was summarized by six statistics. This justifies the use of this dataset as one of the main case studies in the thesis.

J2M PCA Analysis: A PCA analysis is performed on the J2M dataset (Dataset 2.4.2). Figure 2.17 shows the percentage of variance explained by the first 100 principal components. Also in this case the percentage of explained variance grows quickly with the first 20-30 components. These explain more than 90% of the total variance and are representative of the

majority of the information contained in the data. The other components contribute only slightly to the percentage of explained variance and are probably associated with noise. From the PCA analysis it follows that less than 10% of the variance in the data is noise. A low level of noise can be explained by the fact that the data was initially reduced with summary statistics a procedure that may reduce the impact of noise. From the PCA scores, plotted in Figures 2.18, 2.19, it is possible to observe the strong relationship between OES measurements and ER value. In most of the subspaces generated by the first 5 *PCs* the samples colour moves gradually from yellow to red corresponding to the change in ER. In addition the faulty samples associated with low values of ER and represented in blue are often well separated from the normal behaving samples.

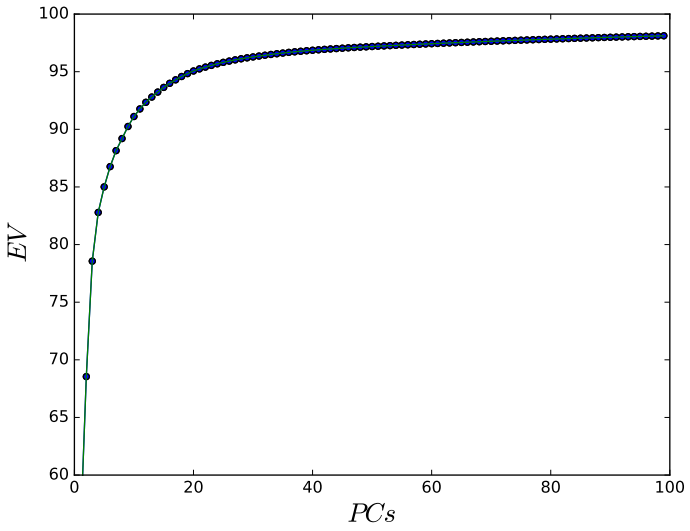


Figure 2.17: The percentage of explained variance by the first 100 *PCs* for Dataset 2.4.2. The matrix was scaled in order to have unit variance.

2.5 A Novel Multi-Sensor Spectral Alignment Procedure

In this section a novel multi-sensor spectral alignment procedure is proposed. In the previous section the OES measurements were stored in a three dimensional matrix $\mathbf{X}$ (equation 2.9). This and the derived mathematical formulations were based on the idea that the data was perfectly clean. As described

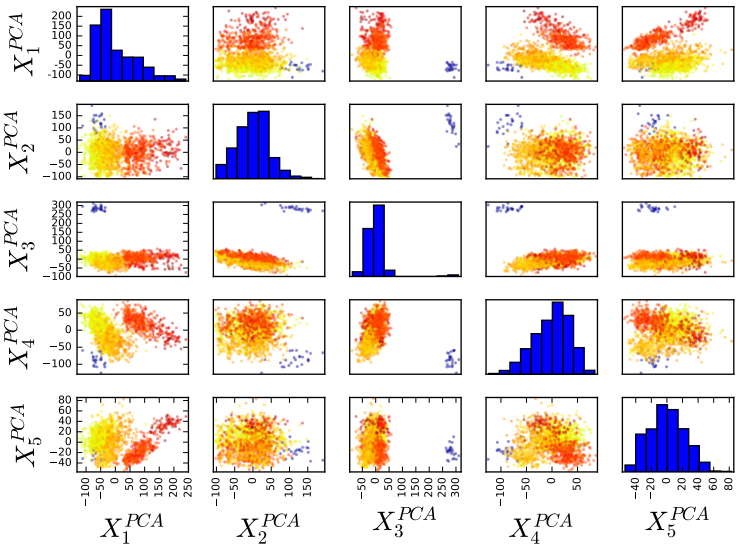


Figure 2.18: Dataset 2.4.2. The scores obtained projecting the data on the subspace generated by its first 5 PCs. The data was scaled in order to have unit variance and the samples are coloured according to their ER value as represented in Figure 2.16.

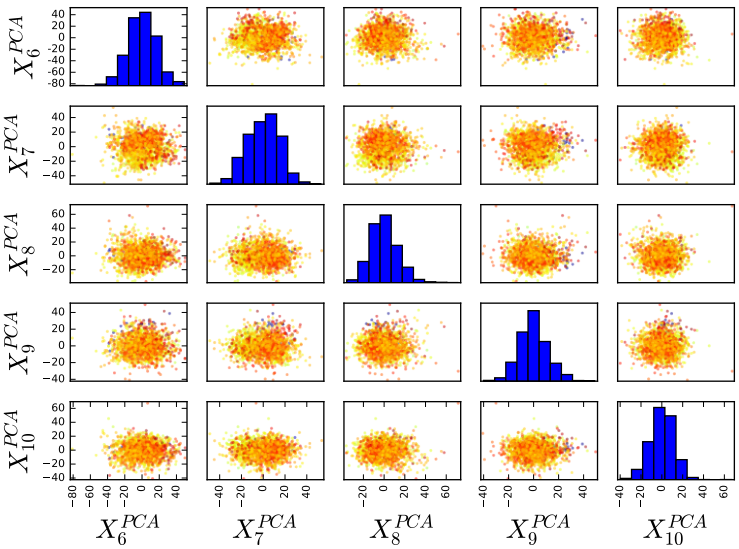


Figure 2.19: Dataset 2.4.2. The scores obtained projecting the data on the subspace generated by 6 to 10. The data was scaled in order to have unit variance and the samples are coloured according to their ER value as represented in Figure 2.16

in section 2.2, the collection of OES measurements is the result of complex chemical reactions and requires advanced and sensitive sensors. Often, for high volume production, wafers are processed on several chambers simultaneously. In order to ensure quality and consistency of production engineers attempt to match all chambers so that they operate identically, determined by measurements taken from each chamber. Currently, the use of OES measurements in process matching across multiple etch chambers presents difficulties due to the nonlinearities in detector response and errors in sensor calibration. These effects lead to variations between the observed intensity vectors at corresponding wavelengths between different OES detectors [34]. As a result, two identical etching processes acquired by different sensors can have different intensity values at the same corresponding wavelengths. In recent years, spectroscopy sensor calibration has been the topic of research in the fields of biology [35] and chemistry [36], [37] and [38]. Work by He *et al.* [39] demonstrated an alignment procedure for mass spectra alignment, in which a warping function is approximated from calibration peaks throughout the spectrum. A favourable comparison is made between the developed method and alignment techniques detailed by Monchamp *et al.* [34], Nielsen *et al.* [40], Tomasi *et al.* [41], Nederkassel *et al.* [42], Pravdova *et al.* [43], Eilers [44], Wong *et al.* [45] and Wong *et al.* [46].

This section presents work towards a spectral alignment methodology. A retrospective calibration process is proposed based on a minimisation of the difference in intensity between reference OES signals from different sensors. The key feature of the methodology is the use of Particle Swarm Optimisation (PSO, [24]) to estimate a calibration curve that is used to retrospectively apply a calibration correction to a set of reference signals from uncalibrated OES sensors. The resulting calibration curve estimation problem leads to a non convex optimization problem with multiple local minima, hence the need for PSO. Once estimated, this calibration curve can be used to align all OES recordings for each sensor.

2.5.1 Calibration Methodology

Given two discrete signals, f and g described by

$$\mathbf{f} = (f_1, \ldots, f_N) \quad \mathbf{t}^f = \left(t_1^f, \ldots, t_N^f\right) \tag{2.34}$$

$$\mathbf{g} = (g_1, \ldots, g_M) \quad \mathbf{t}^g = (t_1^g, \ldots, t_M^g) \tag{2.35}$$

where N and M are the number of discrete samples in $\mathbf{f}$ and $\mathbf{g}$, respectively, $\mathbf{t}^f$ is the vector of sample wavelengths related to points in $\mathbf{f}$ and $\mathbf{t}^g$ is the

vector of sample wavelengths related to points in **g**. The aim is to find a parametrisation $s(t)$ that aligns these signals

$$s(t) = \underset{s(t)}{\operatorname{argmin}} \left[\sum_{k=1}^{K} |f(s(t_k^f)) - g(t_k^g)|^2 \ + \Phi \right] \tag{2.36}$$

where K is the number of sample wavelengths compared and Φ is a limiting factor which constrains the selection of $s(t)$, such that temporal shifts outside the dynamic range of the sensor are penalised. Φ is given by

$$\Phi = \left| \frac{1}{K} \sum_{k=1}^{K} s(t_k^f) - \frac{1}{K} \sum_{k=1}^{K} t_k^g \right| . \tag{2.37}$$

The first step in this process is to transform the discrete signals **f** and **g** into the continuous functions $f(t)$ and $g(t)$. This may be achieved in several ways [47], here a cubic splines interpolation onto all real values of t is adopted, where all extrapolated values are set to zero. The details of this transform are given by equation 2.38 and equation 2.39 respectively.

$$\bar{f}(t) = \Big\{ f(t) \text{ if } t \in [t_1^f, t_N^f] \text{ else } 0 \Big\} \tag{2.38}$$

$$\bar{g}(t) = \{ g(t) \text{ if } t \in [t_1^g, t_M^g] \text{ else } 0 \} \tag{2.39}$$

Because OES recordings are taken from very similar sensors, it is reasonable to assume that the calibration function $s(t)$ should be very close to the identity function, as $f(t)$ and $g(t)$ should be similar. Therefore, it follows that we should be able to align each signal using a function that slightly modifies $f(t)$, which is mathematically expressed by a function similar to the identity. As a result, polynomials of the form

$$s(t) = \sum_{i=0}^{K} a_i t^i \text{ where } a_1 \approx 1 \text{ and } a_i \approx 0 \text{ if } i \neq 1 \ , \tag{2.40}$$

are considered and the family of functions $\mathcal{F}$ that we consider for our minimisation problem is defined as

$$\mathcal{F} = \{ s(t) : s(t) = \text{polynomials} \approx t \} \ . \tag{2.41}$$

This characterisation can be used to inform the initial condition of a minimisation procedure, as it indicates the general region of the optimal solution. From a computational point of view the cost function will be discrete. As a result, $\hat{s}(t)$ is given by

$$\hat{s}(t) = \underset{s\in\mathcal{F}}{\operatorname{argmin}} \left[\sum_{k=1}^{K} |f(s(t_k^f)) - g(t_k^g)|^2 + \Phi \right] \tag{2.42}$$

In general, the cost function defined by the expression in the [ ] in equation 2.42 is very irregular and contains multiple local minima. Therefore, the optimisation algorithm used must be able to escape from local minima and find the global optimum.

There are several techniques which have this ability [48], [49]. In particular, particle swarm optimisation (PSO) is well suited to minimising such a function as it aims to avoid local minima [50]. PSO is a computational method which iteratively searches a solution space while trying to improve a candidate solution with regard to a given measure of quality [24] and [51]. PSO optimises a given problem by having a population of candidate solutions, known as particles, and moves these particles stochastically within a search-space as a function of previous performance and neighbour performance [52].

2.5.2 PSO Calibration: Examples and Applications

Two simulated examples are presented in order to better illustrate the problem and the solution methodology. These two examples are designed in order to match as closely as possible the characteristics of real spectra. As such the generated signals are characterised by having several peaks. The misalignment of these peaks between two spectrometers generates a cost function with several local minima. In the first example two signals with linear distortion are considered:

Example 2.5.1 (Linear Distortion)**.** Consider two simulated signals $\mathbf{f}$ and $\mathbf{g}$, with $g(i) = f(s(i)) + \epsilon$, where ϵ is measurement noise and $s(i) = 1.01i + 1.1$, as depicted in Figure 2.20. We can construct $f(t)$ and $g(t)$, using the aforementioned spline interpolation procedure given by equations 2.38 and 2.39 respectively. Here, the associated cost function is given by

$$c(a_0, a_1) = \sum_k |f(a_1 t_k + a_0) - g(t_k)|^2 + \Phi. \tag{2.43}$$

By plotting the additive and multiplicative coefficients, as illustrated in Figure 2.21 and Figure 2.22, it is observed that both have numerous local minima. Using PSO [53], it is possible to estimate the calibration function, $\hat{s}(t)$, required to align $f(t)$ and $g(t)$, to a mean squared error (MSE) accuracy of

0.1058. This compares favourably to an initial error of 249.4048. The aligned signals are depicted in Figure 2.23.

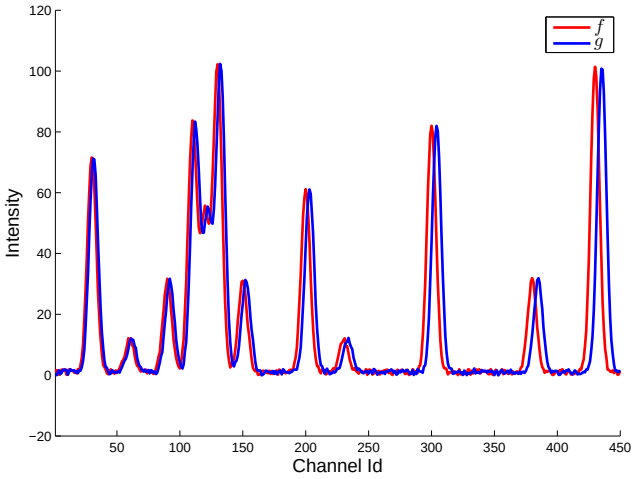


Figure 2.20: Simulated signals f and g, where signal offset is based on a linear function as described in Example 2.5.1.

Typically, the alignment characteristics between two non calibrated OES sensors exhibits a non-linear relationship. To model this behaviour two signals with quadratic distortion are used in the next example.

Example 2.5.2 (Quadratic Distortion)**.** Consider the two signals as described in Example 2.5.1. This signal $f(i)$ is altered such that $g(i) = f(s(i)) + \epsilon$ where $s(i) = 0.00001i^2 + 1.0008i + 3$. As before, $f(t)$ and $g(t)$ may be constructed using the spline interpolation procedure given by equation 2.38 and equation 2.39 respectively. $f(t)$ and $g(t)$ are illustrated in Figure 2.24. Applying PSO to solve $\hat{s}(t)$ as above, it is possible to align $f(t)$ and $g(t)$ with a MSE of 0.0879. This is compared with an initial MSE of 297.8234. The aligned signals are depicted in Figure 2.25, while Figure 2.26 illustrates the estimated calibration function.

To demonstrate the effectiveness of the proposed retrospective calibration methodology for OES, calibration is performed on a sample dataset from an industrial plasma etch chamber as a case study.

Example 2.5.3 (OES Calibration)**.** The dataset, which consists of 60 time samples $\times$ 2048 OES channels (wavelengths), was collected from the chamber exhaust from two etching tools. Before retrospective calibration can be

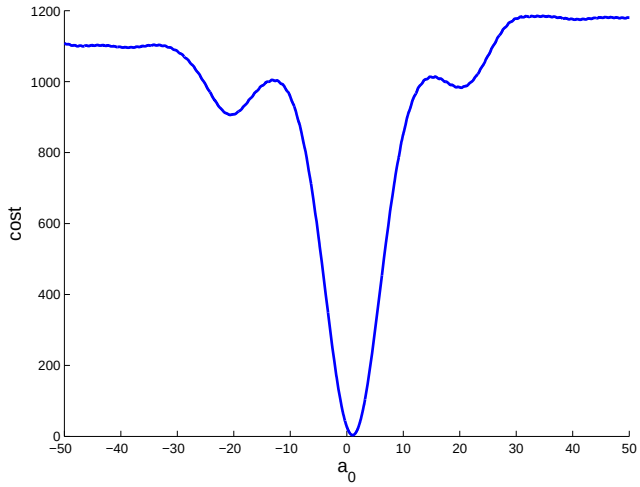


Figure 2.21: The cost function in Example 2.5.1 as a function of the additive coefficient. The multiplicative coefficient has been set equal to its true value.

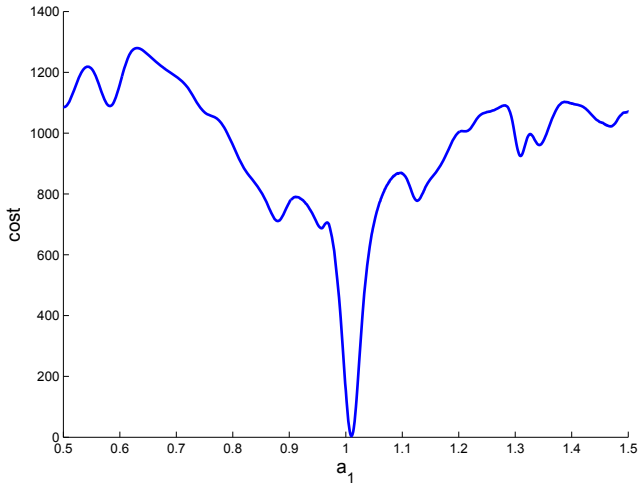


Figure 2.22: The cost function defined in Example 2.5.1 as a function of the multiplicative coefficient. The additive coefficient has been set equal to its true value.

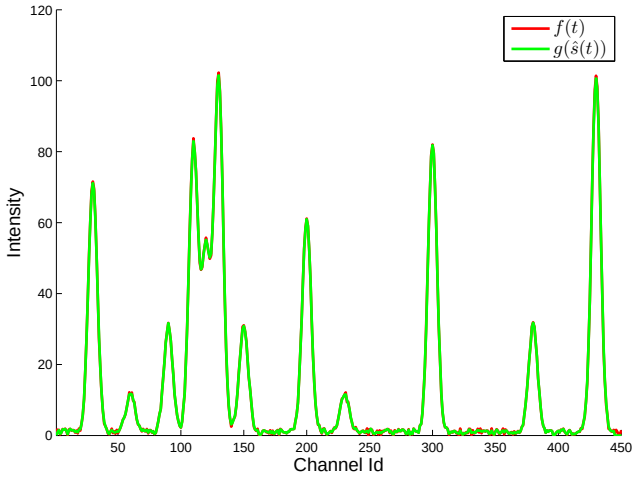


Figure 2.23: Calibrated $f(t)$ and $g(t)$ on the problem described in Example 2.5.1. The calibration function is estimated using PSO.

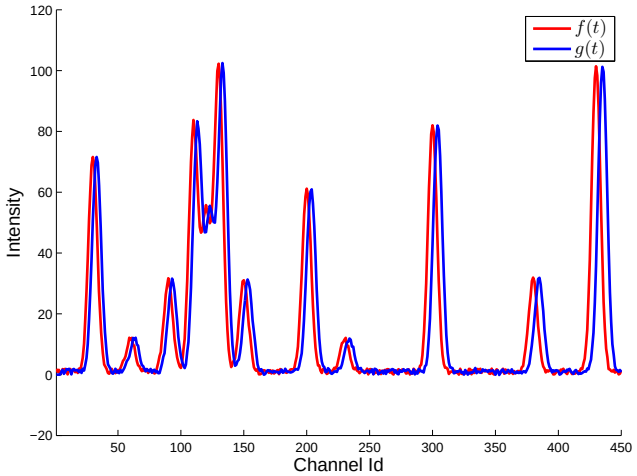


Figure 2.24: Simulated signals $f(t)$ and $g(t)$, where signal offset is based on a quadratic function as described in Example 2.5.2.

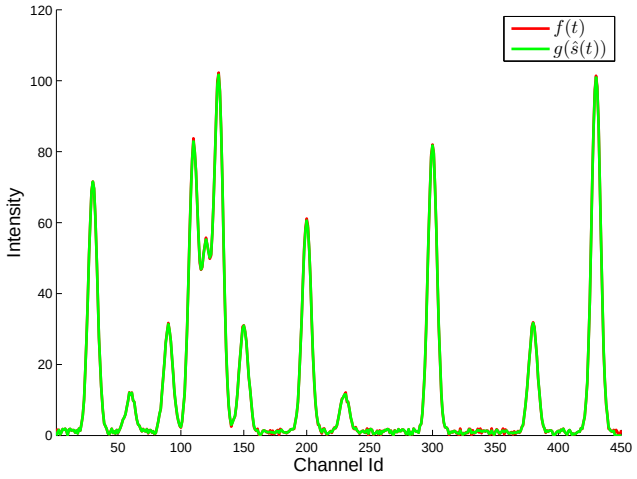


Figure 2.25: Calibrated $f(t)$ and $g(t)$ on the problem described in Example 2.5.2. The calibration function is estimated using PSO.

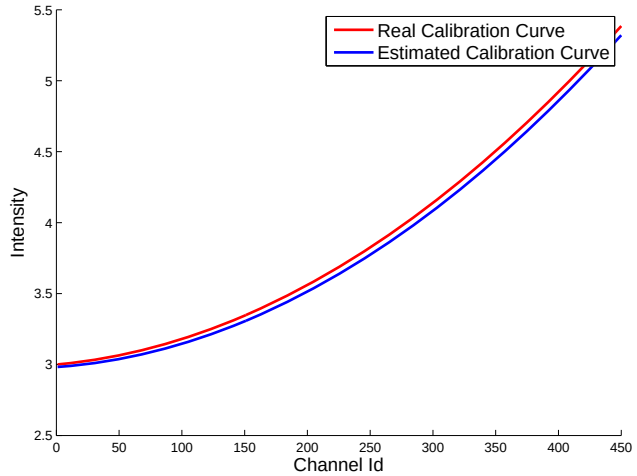


Figure 2.26: The estimated calibration curve $\hat{s}(t)$ obtained using PSO in Example 2.5.2.

applied between the two sensors, suitable time series signals need to be extracted for each etching step. Using one sensor as a reference, a suitable time series signal for an individual step is extracted based on the statistical distance of each temporal sample within $\mathbf{X}$, to the centre of all temporal samples of $\mathbf{X}$ for that particular step. This distance is calculated using the Mahalanobis distance [54]. The centrally distributed signal, ρ, corresponds to the minimum distance signal. This method is used to find the reference time series signal for each step, because the centrally distributed signal will correspond to the average behaving time series signal in that step. Once a centrally distributed signal, ρ, is found for a given step, α, the equivalent signal from the second sensor, σ, is given by

$$\sigma = \underset{\sigma \in X_\alpha}{\operatorname{argmin}} \sum_{q=1}^{p} |\rho(q) - \sigma(q)|^2 \ . \tag{2.44}$$

where X_α is the set of OES recordings for step α, on the second sensor. For each set of matched signals, ρ and σ, the calibration function $\hat{s}(t)$ is estimated using the methodology described in Section 2.5.1. The estimated calibration curve , $\hat{s}(t)$, for each process is depicted in Figure 2.27. When compared to the actual calibration curve, it is clear that the estimated calibration curve produced for step 2 is a better approximation compared to the one estimated for step 1. Applying the calibration curve produced for step 2, alignment across each step is achieved, as illustrated by Figure 2.28. The difference between $\hat{s}(t)$ for step 1 and 2 highlights the impact of etching species on calibration estimation. Therefore, in a multi-step etching process where the etching species varies, certain spectral regions for each step may be more suited for alignment proposes.

The proposed methodology consists of several stages summarised in the following algorithm:

Algorithm 2.5.1. OES alignment

1. Select one OES sensor as a reference sensor, and one as the target sensor.

2. For each process step, find the centrally distributed temporal OES recording from the reference sensor, ρ.

3. For each ρ, find the matching measurement from the target sensor, σ.

4. For each pair of matched signals, ρ and σ, construct a calibration curve $\hat{s}(t)$ using PSO to minimise the difference between ρ and σ.

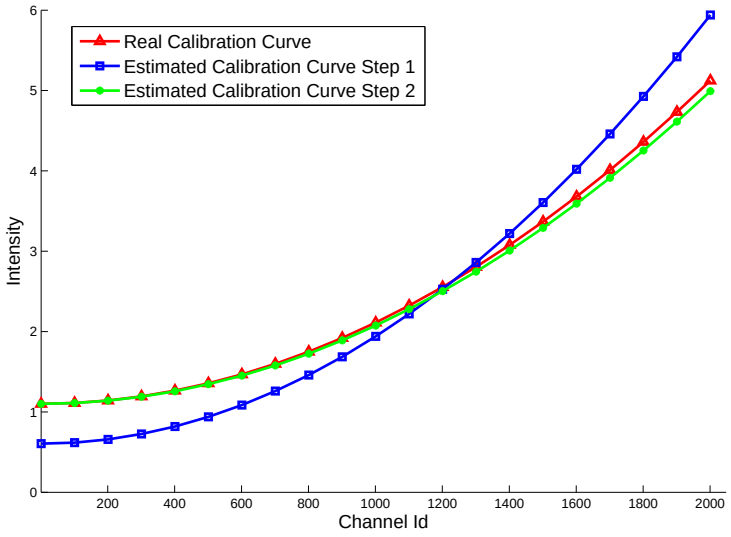


Figure 2.27: The estimated calibration curve $\hat{s}(t)$ obtained using PSO for each process step.

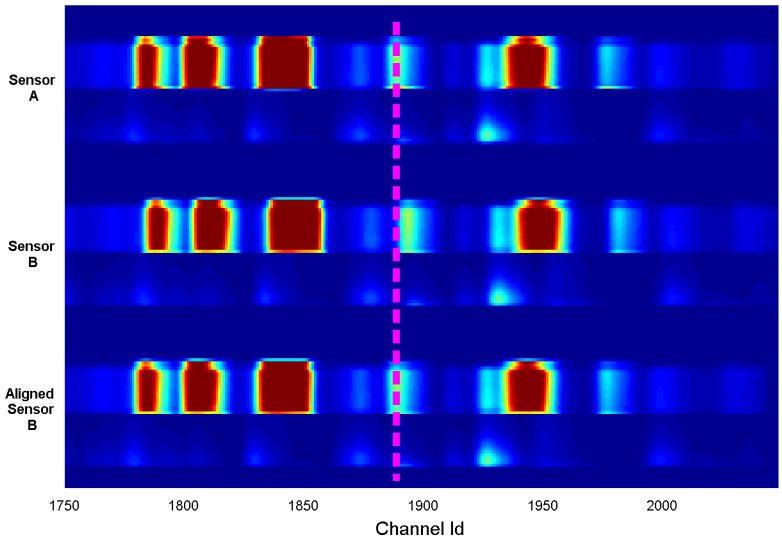


Figure 2.28: A portion of calibrated spectrum from two OES sensors, where the calibration function is estimated using PSO.

5. Construct a global calibration curve by combining suitable subsections from different process step calibration curves.

The etching species of a given step can have a significant impact on calibration performance. Consequently, to achieve alignment across each process step it may be necessary to construct a global calibration curve from subsections of individual step calibration curves.

2.6 Conclusion

This chapter provided the background material for the whole thesis. A basic introduction to the physics behind plasma etching is provided and the process behind OES measurements is described. It is shown how the raw data collected during production needs to be cleaned as wafer measurements are not properly aligned. After that the OES time series is formally described as a 3-dimensional matrix and various two dimensional representations are discussed. Some techniques commonly used in the industry for dimensionality reduction of OES time series are described and some of the datasets that will be used in the rest of the thesis are introduced. It is shown that slot variation can be tracked with the $\mathbf{W}$ matrix and the idea to partition the $\mathbf{W}$ matrix into time intervals is proposed. This will be very useful in the anomaly detection application in Chapter 6 as it allows an high dimensional matrix to be split into a set of matrices of lower dimension. In the last part of the chapter the multi-sensor matching problem is investigated. A matching procedure based on the Particle Swarm Optimization algorithm is proposed. Results indicate that good alignment is possible given suitably matched signals.

Chapter 3

Stable Supervised Feature Selection

3.1 Introduction

Variable selection in the context of regression and classification has received increasing attention in recent years due to the huge growth in data collected across many fields of science and engineering [17, 55, 56]. While the goal is often focused on developing models for accurate output prediction, with these large datasets estimation of model structure and identification of the few variables driving the output variation is also mandatory for understanding and interpreting the underlying processes that generated the data [57, 58]. Often, the numerical identification of such models is challenging because of the data presenting, either a high correlation among subsets of predictors, or because the number of measured variables is bigger than the number of available data points [59]. For this reason, several techniques have recently been developed for sparse model identification, i.e. where only a few important variables are selected. A first possibility is to look for the best subset of variables considering all the possible combinations. However, this strategy becomes infeasible as the number of candidate variables increases. Rather than searching through all possible combinations, we can seek a path through them. For example, *Forward-Stepwise selection* starts with no variables and includes in the model one variable at the time with an order that depends on the improvement in terms of fit (or other cost functions such as the Akaike information criterion [60]). An alternative strategy is the so called *Backward-Stepwise selection* that starts with all variables in the model and iteratively removes them according to their ability to explain the output value. Combinations of these methods can also be used to give improved performance,

for example sequential floating forward selection as proposed in [61] and the two-stage algorithm incorporating a backward refinement step proposed in [62].

Regularization based methods represent a different class of variable selection algorithm. These involve the estimation of the regression model by minimizing a cost function consisting of two terms, the first representing the model fit, and the second the complexity of the model. A classical approach is to consider the sum of squares of the model coefficients as a measure of complexity, leading to the so-called ridge regression model. The resulting model suppresses the influence of irrelevant variables by forcing them to have small coefficients relative to those assigned to the important variables. In the last decade, Tibshirani and co-workers [63] showed that employing the sum of the absolute values of the model coefficients as the regularization penalty has the desirable effect of shrinking the non-important variable coefficients to exactly zero implicitly performing variable selection. This technique is the well known Least Absolute Shrinkage and Selection Operator (Lasso), also known as basis pursuit in the signal processing community [64]. In recent years, several modifications of the Lasso have been proposed such as the elastic-net [65], and the group Lasso [66], [67] and new penalized regression methods have been proposed such as the Dantzig selector [68] and the Non-Negative Garrotte Estimator [69]. These methods have been designed in order to improve different aspects of the lasso estimator such its prediction performance and its handling of groups of similar variables.

In general sparsity and algorithmic stability are two desired properties of learning algorithms. This means that the output of the model should be a function of only a small subset of the input variables and that this subset should not change with small variations in the training data. Unfortunately a sparse algorithm cannot be stable and vice versa [70]. While this is a general problem of all sparse algorithms, this chapter focuses on the lasso estimator. The stability of the lasso is investigated and algorithms for detecting a stable set of variables are proposed. Four new algorithms are proposed: High Frequency Lasso (HF); High Mean (HM); Monte Carlo High Frequency (MCHF); and Monte Carlo High Mean (MCHM). The aim of these algorithms is to stabilize the lasso solution under CV variability taking into account both the K-fold Cross-Validation and the Monte Carlo Cross-Validation (bootstrap). These algorithms are easy to use and automate and at the same time provide competitive results with respect to competing approaches in the literature. In particular, we compare our algorithms with Stability Selection [71], Kappa Selection Criterion [72] and Lasso Percentile [73] for a range of simulated

and real datasets in order to highlight their strengths and drawbacks, both in terms of prediction accuracy, and in terms of their ability to recover the true model structure.

The outline of the chapter is as follow: Section 3.2 provides an introduction to linear models and penalized linear models. Section 3.3 reviews existing methods that attempt to obtain stable estimates from LASSO. Then section 3.4 introduces the four new algorithms, HF, HM, MCHF and MCHM. Section 3.5 describes some datasets used to benchmark the different methods and reports the results obtained for both the simulated and real datasets. Section 3.6 describes the computational complexity of the various algorithm and finally Section 3.7 concludes the chapter with some final remarks and suggestions.

3.2 Linear Regression and Penalized Models

The linear regression model is defined as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{3.1}$$

for a set of n observations of p variables where $\mathbf{y}$ is a $(n \times 1)$ vector, $\mathbf{X}$ is a $(n \times p)$ matrix collecting the data, $\boldsymbol{\beta}$ is a $(p \times 1)$ vector containing the linear model coefficients and $\boldsymbol{\epsilon}$ the $(n \times 1)$ vector describing the portion of the data not described by the linear model. An important value associated with the linear model is the Signal to Noise Ratio (SNR, [74]) measuring the relation between the strength of the signal and the noise:

$$SNR = \frac{Var(\mathbf{y})}{Var(\boldsymbol{\epsilon})} \tag{3.2}$$

The ordinary least squares estimator (OLS) of $\boldsymbol{\beta}$ in equation 3.1 is obtained by minimizing the least squares cost function $\sum_{i=1}^{n} \parallel y_i - \hat{y}_i \parallel_2^2$ given by the difference between the true output $\mathbf{y}$ and the one estimated by the model

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} \tag{3.3}$$

Estimating $\boldsymbol{\beta}$ by OLS is in general an ill-conditioned problem, unless the predictors are orthogonal. Usually, these numerical problems arise when dealing with high-dimensional datasets, i.e. where $p >> n$, and when there is high correlation between subsets of the input variables (i.e. columns of $\mathbf{X}$). We work under the assumption that the vector $\boldsymbol{\beta}$ is sparse in the sense that $s << p$ coefficients β_j are non-zero and that $s < n$. Here we denote the set

of non-zero coefficients as $S_0 = \{j : \beta_j \neq 0\}$ and our goal is to obtain a good estimate of S_0 that allows us to build a sparse model with good prediction performance using the Lasso estimator.

In order to describe the structure of the data some new notations are introduced. Define $\mathcal{D} = \{1, \ldots, n\}$ as the index set of the rows of $\mathbf{X}$ and $\mathbf{y}$ and $\phi \subset \mathcal{D}$ such that $\mathbf{X}_\phi$ and $\mathbf{y}_\phi$ are the submatrices of $\mathbf{X}$ and the subvector of $\mathbf{y}$, obtained using only the rows of $\mathbf{X}$ and $\mathbf{y}$ with indexes in ϕ, respectively. Moreover, given $A \subset \mathcal{D}$, the complementary set is defined as $\overline{A} = \mathcal{D} - A$.

The Lasso estimator is formally defined as the solution of a convex optimization problem:

$$\hat{\boldsymbol{\beta}}(\alpha) = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{argmin} \left\{ \| \mathbf{y} - \mathbf{X}\boldsymbol{\beta} \|_2^2 + \alpha \| \boldsymbol{\beta} \|_1 \right\} \tag{3.4}$$

where α is the regularization gain that controls the degree of sparsity in the model. For a given α the lasso estimator can be computed iteratively using various algorithms, e.g. LARS [75] or coordinate descent [76]. In this chapter the LARS algorithm is used. The optimum value of α can be estimated using a Cross-Validation (CV) procedure [77], [78]. Among these the most widely used is k-fold Cross-Validation as it provides a good balance between computational complexity and prediction error estimation accuracy. In order to perform k-fold Cross-Validation the data is randomly divided into K roughly equal-sized folds, with the index set for the k^{th} fold denoted as f^k. Then for a sequence of values for the tuning parameter α, penalized models are estimated using all but one of the folds as training data and the predictive performance of each model tested on the omitted "left-out" fold. This process is repeated until each fold has been left out. Thus

$$CV(\alpha) = \frac{1}{K} \sum_{k=1}^{K} \| \mathbf{y}_{f^k} - \mathbf{X}_{f^k} \hat{\boldsymbol{\beta}}_{\overline{f}^k}(\alpha) \|_2^2 \tag{3.5}$$

where $\hat{\boldsymbol{\beta}}_{\overline{f}^k}(\alpha)$ are the lasso coefficients estimated using all the samples except the ones in the fold f^k as training data and α as the regularization gain. The optimum value of α is then chosen to be the value in the sequence that minimizes the CV error, that is:

$$\alpha^* = \underset{\alpha}{argmin}\{CV(\alpha)\} \tag{3.6}$$

In the literature the number of folds K is often chosen as 5 or 10 [60]. In this chapter we will always use $K = 10$. The CV procedure is very popular

because it is intuitively appealing, easy to implement and can provide a good estimate of the expected prediction error [60]. However, the CV procedure is highly influenced by several factors such as the noise on the data and the split of the dataset into smaller subsets [73]. Problems arise when we are interested in identifying the underlying model structure S_0. In particular, different (random) data splits typically result in different α^* values and different subsets of variables being selected. Indeed even for a fixed α the selected variables will vary substantially with different data permutations, especially if the candidate variables are highly correlated as will be described in the next section.

3.2.1 Oracle Property and Irrepresentable Condition

In general it is important to be aware of the prediction capabilities of a penalized model. This can be expressed through the Oracle Property:

Definition 3.2.1. A penalized estimator is said to have the Oracle Property if it is asymptotically equivalent to the oracle estimator, which is defined as an ideal estimator obtained when using the true variables without penalization.

More formally let $\mathbf{X}$ and $\mathbf{y}$ be the input matrix and the output vector and let S_0 be the set of true variables. The Oracle estimator is then defined as:

$$\mathbf{y} = \mathbf{X}_{S_0}\hat{\boldsymbol{\beta}}_{S_0} \tag{3.7}$$

where $\mathbf{X}_{S_0}$ is the set of columns of $\mathbf{X}$ whose indices are contained in S_0 and

$$\hat{\boldsymbol{\beta}}_{S_0} = \underset{\boldsymbol{\beta}_{S_0}}{\operatorname{argmin}} \parallel \mathbf{y} - \mathbf{X}_{S_0}\boldsymbol{\beta}_{S_0} \parallel_2 \tag{3.8}$$

A penalized estimator is said to have the Oracle property if there is a sequence λ_n such that with $\lambda = \lambda_n$

$$P(\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}_{S_0}) \to 1 \tag{3.9}$$

A slightly weaker definition is

$$P(\hat{S} = S_0) \to 1 \tag{3.10}$$

In [79], [80] and [81] it is proven that under certain conditions the least square estimator with Smoothly Clipped Absolute Deviation (SCAD), Ridge or Lasso penalty has the oracle property. In this context for the Lasso estimator the Irrepresentable Condition, presented in the next definition, plays an important role.

Definition 3.2.2. The neighborhood stability condition, also known as the irrepresentable condition [82], [83], [84] is defined as:

$$\max_{k \in S_0^c} \left| sign(\boldsymbol{\beta}_{S_0})^T \left(\mathbf{X}_{S_0}^T \mathbf{X}_{S_0} \right)^{-1} \mathbf{X}_{S_0}^T \mathbf{X}_k \right| < 1 \tag{3.11}$$

where $\mathbf{X}_{S_0}$ is the set of true variables.

In [83] the authors prove that the Irrepresentable Condition (equation 3.11), except for a minor technicality (according to which a model is consistent if $sign(\hat{\beta}_i) = sign(\beta_i)\ \forall i$), is 'almost' necessary and sufficient for consistency of the lasso estimator (the word 'almost' refers to the fact that a necessary relationship uses $\leq$ instead of $<$). Consistency implies that, for each random realization there exists a correct amount of regularization that selects the true model. If this condition is violated, all that we can hope for is recovery of the regression vector $\boldsymbol{\beta}$ in an L^2-sense of convergence by achieving

$$\| \hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_S \|_2 \to 0 \quad for \quad n \to \infty \tag{3.12}$$

This type of L^2-convergence can be used to achieve consistent variable selection in a two-stage procedure by thresholding or, preferably, by employing the adaptive lasso [84]. The disadvantage of such a two-step procedure is the need to choose several tuning parameters without proper guidance on how these parameters can be chosen in practice. In conclusion if this condition is violated, the true $\boldsymbol{\beta}$ cannot be recovered unless some two-stage procedure based on thresholding or the adaptive LASSO are used [84], [85].

3.3 Review of Existing Approaches

In this section some methods from the literature are discussed. These methods can be split into two families: the first one is based on data resampling (similar to bootstrapping) and the second is based on repeated CV procedures with different choices of the folds. The standard Lasso model whose α is obtained by applying the standard CV procedure (3.5) is denoted as LCV and is used as baseline reference method.

To provide a graphical comparison between the various methods a simple dataset is introduced as follows:

Dataset 3.3.1. Set $n = 50$, $p = 100$ and $SNR = 1$ and generate $\mathbf{X} \in \mathbb{R}^{n \times p}$ as a random multivariate normal distribution such that $cov(\mathbf{x}_i, \mathbf{x}_j) = 0.6^{|i-j|}$. $\mathbf{y} \in \mathbb{R}^n$ is generated according to equation 3.1 where only ten randomly chosen regression coefficients are non-zero and their values have been randomly chosen between 0.1 and 1.

3.3.1 Resampling Based Method

3.3.1.1 Stability Selection

Stability Selection (SS) is not a new variable selection technique but rather it is an enhancement of the existing Lasso method for estimating $\boldsymbol{\beta}$. It is able to perform consistent variable selection even when the irrepresentability condition (equation 3.11) is violated [71].

Input: The input matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, the target variable $\mathbf{y} \in \mathbb{R}^n$, the number of simulations B , the threshold $\bar{\Pi}$, the Lasso LARS path $\mathcal{A}$, the weakness $0 < u \leq 1$

Output: $\mathbf{x}_1, \ldots, \mathbf{x}_s$ the variables chosen by the algorithm.

$\hat{X} = (\ )$ (the empty matrix)
for $\alpha \in \mathcal{A}$ **do**
  for $j = 1$ **to** B **do**
    Choose ϕ a random subset of $\mathcal{D}$ of size $\lfloor \frac{n}{2} \rfloor$ without replacement.
    From ϕ obtain $\mathbf{X}_\phi \in \mathbb{R}^{\lfloor \frac{n}{2} \rfloor \times p}$ and $\mathbf{y}_\phi \in \mathbb{R}^{\lfloor \frac{n}{2} \rfloor}$ taking the rows of $\mathbf{X}$ and $\mathbf{y}$ with index ϕ.
    $w_k \sim \mathcal{U}(u, 1)$ $k = 1, \ldots, p$
    $\hat{\boldsymbol{\beta}}^{\alpha,j} = \underset{\boldsymbol{\beta}}{argmin} \left\{ ||\mathbf{y}_\phi - \mathbf{X}_\phi \beta||_2^2 + \alpha \sum_{k=1}^{p} \frac{|\boldsymbol{\beta}_k|}{w_k} \right\}$
  end for
  for $i = 1, \ldots, p$ **do**
    $\pi_{\alpha,i} = \frac{1}{B} \sum_{j=1}^{B} 1_{\{\hat{\boldsymbol{\beta}}^{\alpha,j} \neq 0\}}$
    if $\pi_{\alpha,i} > \bar{\Pi}$ **and** $\mathbf{x}_i \notin \hat{\mathbf{X}}$ **then**
      $\hat{\mathbf{X}} = (\hat{\mathbf{X}}, \mathbf{x}_i)$.
    end if
  end for
end for

Pseudocode 3.3.1: Stability Selection

Stability Selection is based on a generalization of the Lasso, referred to as the randomized Lasso (see Pseudocode 3.3.1), where each penalty α_i in the path $\mathcal{A} = \{\alpha_1, \ldots, \alpha_c\}$ is changed to a randomly chosen value in the range $[\alpha, \alpha/u]$ where u is called the "weakness" $u \in (0, 1]$. A detailed explanation of SS is presented in the following algorithm:

Algorithm 3.3.1. Let $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_p)$ be the IID random variables in $[u, 1]$ for $k = 1, \ldots, p$. The random Lasso estimator for a given α_i is defined

as:

$$\hat{\boldsymbol{\beta}}^{\alpha,W} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{argmin} \left\{ ||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \alpha \sum_{k=1}^{p} \frac{|\boldsymbol{\beta}_k|}{w_k} \right\} \quad \forall \alpha \in A \tag{3.13}$$

The randomized Lasso is applied many times (indicated by B in Pseudocode 3.3.1) to a subset of the available data. The probability $\pi_{\alpha,i}$ that each variable is selected for each values of α is calculated and then each variable is assigned a score $\Pi_i = \underset{\alpha}{max}\ \pi_{\alpha,i}$. A variable is thus selected if its score is larger than a given threshold $\bar{\Pi}$. In the original paper [71] the authors suggest using as threshold a value between $0.6 \leq \bar{\Pi} \leq 0.9$.

We have observed that in numerous experiments this algorithm selects too many variables when $\alpha \to 0$. To solve this problem we have decided to use only the LARS path $\mathcal{A}$ larger than a certain value α_{min}. In particular we have decided to fix $\alpha_{min} = 0.1$. Another possible solution would be to choose the subset ϕ of size smaller than p as the LARS algorithm selects $min(n, p)$ variables when $\alpha = 0$ this ensures that not all the variables are selected even for small values of α. The general impression is that the choice of the α_{min} and u parameters strongly influences the results and that better results can be obtained by a visual analysis of the stability path. This has already been reported by Sylvia Richardson in the discussion at the bottom of [71]: "The authors seem to rely instead on a semi qualitative interpretation of the plots described in terms such as "variables standing out", "better separated", ... without giving quantitative a guidelines on how to judge such a separation". A comparison between the path generated by stability selection and the Lasso path is given in Figure 3.1 for Dataset 3.3.1.

3.3.1.2 Kappa Selection Criterion

The Kappa Selection Criteria (KSC) has recently been proposed in [72]. The main difference between SS and KSC is that while the former mainly focuses on selecting the informative variables, the latter aims at selecting the optimal tuning parameter. This method focuses on the estimation of the penalty value α such that the estimated active set is robust with respect to dataset changes, for example when the dataset is split for the CV procedure. In this algorithm an important role is played by Cohen's Kappa Coefficient $\mathcal{K}$ [86] calculated in Pseudocode 3.3.2. Given two sets of active variables $\mathcal{A}_1$ and $\mathcal{A}_2$ the value of k ranges between $-1 \leq \mathcal{K}(\mathcal{A}_1, \mathcal{A}_2) \leq 1$ and gives a measure of the agreement between the active sets in terms of selected variables [72]. The full algorithm is described in Pseudocode 3.3.3. Note that the treatment in Step 14 of the algorithm is necessary since some informative variables may have

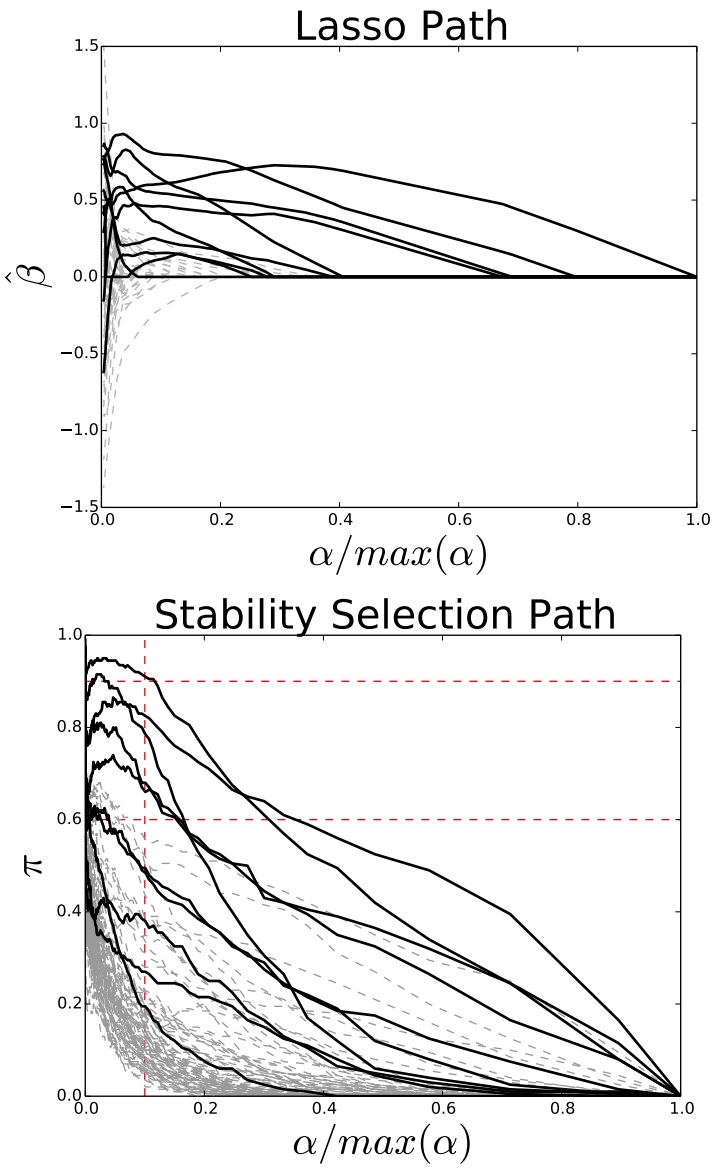


Figure 3.1: The lasso path (top) and the stability selection path (bottom) obtained using as weakness value $u = 0.5$. Both plots have been obtained using Dataset 3.3.1. The black lines are the true variables while the grey dashed lines are the noise variables. The dashed red lines on the Stability Selection path plot are the threshold values and the value of α_{min}.

relatively weak effect compared with others. A large value of α may produce an active set that consistently overlooks the weakly informative variables, which leads to an underfitted model with large variable selection stability.

Input: Two estimated sets of active variables A_1, A_2 and p the number of variables
Output: A score $-1 < k(A_1, A_2) < 1$
1: **if** $A_1 = A_2 = \emptyset$ or $A_1 = A_2 = \{1, \ldots, p\}$ **then**
2: return $k = -1$
3: **end if**
4: $n_{11} = |A_1 \cap A_2|$
5: $n_{12} = |A_1 \cap A_2^c|$
6: $n_{21} = |A_1^c \cap A_2|$
7: $n_{22} = |A_1^c \cap A_2^c|$
8: $Pr(a) = \dfrac{n_{11} + n_{22}}{p}$
9: $Pr(e) = (n_{11} + n_{12})(n_{11} + n_{21})/p^2$
10: **return** $\mathcal{K}(A_1, A_2) = \dfrac{Pr(a) - Pr(e)}{1 - Pr(e)}$

Pseudocode 3.3.2: Cohen's Kappa coefficient

3.3.1.3 Bolasso

Bolasso [87] is a method that can be used to improve the performance of lasso. We will describe this method as a possible way to increase the performance of some of the proposed algorithms but we will not include it in our study. The Bolasso method does not provide a way to select the penalty parameter α but the common method such as Cross-Validation can be easily used in order to set the α value. The Bolasso algorithm is described in Pseudocode 3.3.4. In general the idea of subsampling introduced by Bolasso has the same effect as the resampling based method introduced in Section 3.3.1 and the Monte Carlo approach that will be introduced in Section 3.4.3.

3.3.2 K-folds Cross-Validation Based Methods

The methods described in this section overcome the instability in CV by repeating the CV procedure r-times, changing at each iteration the split of the data into folds, and then applying some rules based on the distribution of the results. At each repetition, the optimum regularization value α is saved in order to obtain a sequence of ordered optimal values

Input: An input matrix $\mathbf{X}$ and an output vector $\mathbf{y}$.
Output: A penalty value α, a set of active variables A and the estimated linear model.

Set a path of penalty values $\mathcal{A} = \alpha_1, \ldots, \alpha_c$ and B the number of iterations
set $\hat{s}(\alpha) = 0$
set λ_n, (the authors suggest to use $\lambda_n = 0.1$)
for $i = 0$ **to** B **do**
    Randomly partition $\mathbf{X}$ and $\mathbf{y}$ in two distinct subsets of the same size $\mathbf{X}_{\phi_1},\ \mathbf{y}_{\phi_1}$ and $\mathbf{X}_{\phi_2},\ \mathbf{y}_{\phi_2}$
    for α **in** $\mathcal{A}$ **do**
        Obtain two sets of active variables $\hat{A}_{1\alpha}$ and $\hat{A}_{2\alpha}$ using the lasso estimator with penalty α on the two subsets $\mathbf{X}_{\phi_1},\ \mathbf{y}_{\phi_1}$ and $\mathbf{X}_{\phi_2},\ \mathbf{y}_{\phi_2}$
        $\hat{s}(\alpha) = \hat{s}(\alpha) + k(\hat{A}_{1\alpha}, \hat{A}_{2\alpha})$
    end for
end for
for α **in** $\mathcal{A}$ **do**
    $\hat{s}(\alpha) = \dfrac{\hat{s}(\alpha)}{B}$
end for
$\hat{\alpha} = min \left\{ \alpha : \dfrac{\hat{s}(\alpha)}{\underset{\alpha'}{max}\ \hat{s}(\alpha')} > 1 - \lambda_n \right\}$
Estimate the lasso using as penalty $\hat{\alpha}$

Pseudocode 3.3.3: Kappa Selection Criterion

Input: An input matrix $\mathbf{X}$, an output vector $\mathbf{y}$, a penalty value α and the number of bootstrap replicates r.
Output: A set of active variables.

for $i = 1$ **to** r **do**
    Generate bootstrap samples $\mathbf{X}_\phi, \mathbf{y}_\phi$
    Compute Lasso estimate $\hat{\boldsymbol{\beta}}_\phi(\alpha)$
    Compute the set of active variables $\mathcal{A}_i$
end for
Compute the set of active variables $S = \cap_{i=1}^{r} \mathcal{A}_i$

Pseudocode 3.3.4: Bolasso

$\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_r) : \ \hat{\alpha}_1 \leq \hat{\alpha}_2 \leq \cdots \leq \hat{\alpha}_r$. Denoting the true set of non-zero $\boldsymbol{\beta}_j$ as S_0 and $\hat{S}(\alpha)$ as the estimation of S_0 using Lasso with penalty α, CV based methods are based on the assumption that the true set of variables S_0 is contained within $\hat{S}(\alpha)$ [59], that is:

$$S_0 \subseteq \hat{S}(\hat{\alpha}_i) \ \ \forall i = 1, \ldots, r \tag{3.14}$$

or equivalently it is at the intersection of the variables selected at each iteration:

$$S_0 \subseteq \bigcap_{i=1}^{r} \hat{S}(\hat{\alpha}_i) \tag{3.15}$$

3.3.2.1 Lasso Percentile

Another recently proposed method, the Lasso Percentile (LP) [73], repeats the CV procedure for obtaining $\hat{\boldsymbol{\alpha}}$ r-times as can be seen in Pseudocode 3.3.5. Then, the penalty value $\hat{\alpha}_{perc}$ is chosen as a percentile θ of the $\hat{\boldsymbol{\alpha}}$ sequences. The value of θ can be chosen manually or estimated through CV. The model can then be estimated running Lasso with penalty value $\hat{\alpha}_{perc}$. In this algorithm the elements that have to be tuned are the number of simulations r and the value of θ.

Input: Input matrix $\mathbf{X}$, output vector $\mathbf{y}$, r the number of simulations and θ a percentile value.
Output: $\mathbf{x}_1, \ldots, \mathbf{x}_k$ the variables chosen by the algorithm.
1: **for** $j = 1$ **to** r **do**
2: $\quad \hat{\alpha}_j := \underset{\alpha}{\texttt{argmin}}\, CV_j(\alpha)$
3: **end for**
4: Choose $\hat{\alpha}$ as the θ percentile of $(\hat{\alpha}_1, \ldots, \hat{\alpha}_r)$.
5: Compute Lasso with penalty value $\hat{\alpha}$

Pseudocode 3.3.5: Lasso Percentile

3.4 Novel Lasso Stabilization Algorithms

In this section four new lasso based algorithms are proposed: High Frequency Lasso (HF); High Mean (HM); Monte Carlo High Frequency (MCHF); and Monte Carlo High Mean (MCHM).

3.4.1 High Frequency Lasso

Here a new algorithm called High Frequency Lasso (HF) is proposed. In this algorithm the regular Lasso estimator (3.4) is calculated using $\hat{\alpha}_j$ obtained with the CV procedure (3.5) as the penalty value. According to Pseudocode 3.4.1, the procedure is repeated r times and at each iteration $\hat{\alpha}_j$ and $\hat{\boldsymbol{\beta}}(\hat{\alpha}_j)$ are saved. The $\boldsymbol{\beta}$ estimated by Lasso are sparse. The selection probability for each variable is obtained as the number of times a variable is selected (line 6 in Pseudocode 3.4.1) normalized by r. In particular, a variable is selected if its probability is greater than a given threshold $\bar{p}$ as can be seen from lines 7-8 of Pseudocode 3.4.1.

Input: Input matrix $\mathbf{X}$, output vector $\mathbf{y}$, r the number of simulations, the cut level $\bar{p}$
Output: $\mathbf{x}_1, \ldots, \mathbf{x}_k$ the variables chosen by the algorithm.
1: initialize an empty matrix $\tilde{\mathbf{X}} = ()$
2: **for** $j = 1$ **to** r **do**
3: $\quad \hat{\alpha}_j := \underset{\alpha}{argmin}\ CV^j_{\mathcal{D}}(\alpha)$
4: $\quad \hat{\boldsymbol{\beta}}^j_i := \hat{\boldsymbol{\beta}}^i_{\mathcal{D}}(\hat{\alpha}_j) \quad i = 1, \ldots, p$
5: **end for**
6: **for** $i = 1$ **to** p **do**
7: $\quad \hat{p}_i = \frac{1}{r} \sum_{j=1}^{r} 1_{\hat{\boldsymbol{\beta}}^j_i \neq 0}$
8: $\quad$ **if** $\hat{p}_i > \bar{p}$ **then**
9: $\quad\quad \tilde{\mathbf{X}} = (\tilde{\mathbf{X}}, \mathbf{x}_i)$
10: $\quad$ **end if**
11: **end for**

Pseudocode 3.4.1: High Frequency Lasso

This algorithm is easy to use and the results are easy to interpret. The values that have to be defined by the user are r (the number of simulations) and $\bar{p}$ (the probability threshold). When r is set to a value greater than 1000, we observed that good results occur using $\bar{p} = max_i\ \hat{p}_i$. Intuitively, selecting the threshold value correspondence of the coefficient with greatest probability, should return only the variable with highest probability $\hat{p}_i$. However, HF returns many coefficients with the same probability, thus several of them are selected. An alternative is to set $\bar{p} = c \max_i \hat{p}_i$ with $c \in [0, 1]$ chosen with a Cross-Validation procedure. The HF algorithm with c chosen a priori equal to 1 is denoted as HF(max). An illustration of how the number of correctly selected variables (True Positive TP) and wrongly selected variables (False Positive FP) change as a function of the threshold $\bar{p}$ is provided in Figure

3.2 for Dataset 3.3.1. In this example the number of iterations was set as $r = 100$.

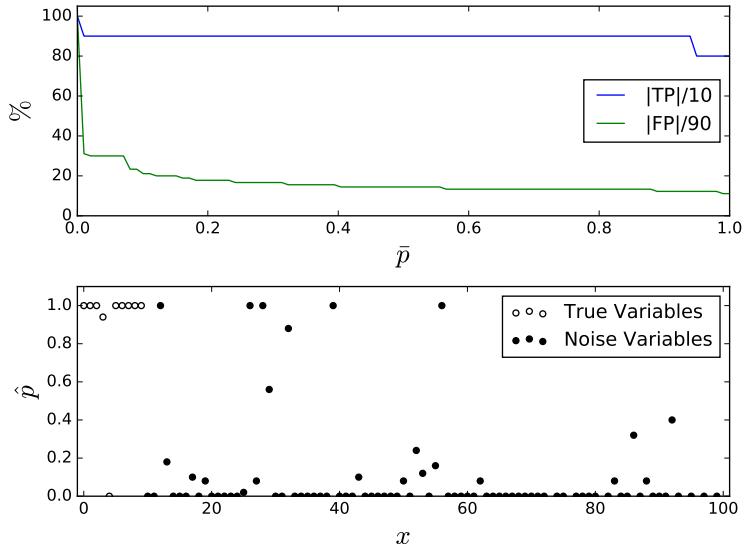


Figure 3.2: In the first figure the number of True Positives (TP) divided by the total number of true variables and the number of False Positives (FP) divided by the total number of variables with 0 regression coefficient are represented as a function of the threshold parameter $\bar{p}$ from the HF algorithm. The second figure shows the value of $\hat{p}$ assigned to each variable by the HF algorithm.

3.4.2 High Mean Lasso

High Mean Lasso (HM) has been developed observing that in many simulations the false positives, i.e. β_i coefficients estimated as non-zero when they actually are zero, have a small mean value $\bar{\beta}_i$ across the r CV repetitions:

$$\bar{\beta}_i := \frac{\sum_{j=1}^{r} \hat{\beta}_i(\hat{\alpha}_{cv}^j)}{r} \tag{3.16}$$

but a large standard deviation $\hat{\sigma}_i$ w.r.t. $\bar{\beta}_i$:

$$\hat{\sigma}_i := \sqrt{\frac{1}{r}\sum_{j=1}^{r}(\hat{\beta}_i(\hat{\alpha}_{cv}^j) - \bar{\beta}_i)^2} \tag{3.17}$$

HM is implemented in Pseudocode 3.4.2 with lines 1-4 the same as in HF, i.e. describing a CV procedure repeated r times. Then, for each coefficient $\hat{\beta}_i$ the mean and standard deviation are calculated as in (3.16) and (3.17) respectively. Moreover, the score Z is defined as:

$$Z_i = \sqrt{r}\bar{\beta}_i/\hat{\sigma}_i \tag{3.18}$$

Finally, the hypothesis test is performed on each $\hat{\beta}_i$:

$$H_0 : \mu_i = 0 \qquad H_1 : \mu_i \neq 0 \tag{3.19}$$

where $\mu_i = \mathbb{E}(\beta_i)$ and $\hat{\beta}_i^j$ is independent of $\hat{\beta}_i^k$ $j \neq k$ because their randomness depends only on the fold. So, under H_0, $Z_i \sim t(r-1)$. $\mathbf{x}_i$ is selected if H_0 is rejected. We reject H_0 if $\mathbb{P}(|z| \geq |Z_i|) < p_i$ where $z \sim t(r-1)$ as can be seen in lines 9-10 of Pseudocode 3.4.2. In this algorithm the two parameters that have to be tuned by the user are the p-value threshold and the number of repetitions.

Input: Input matrix $\mathbf{X}$, output vector $\mathbf{y}$, r the number of simulations, the p-value p
Output: $\mathbf{x}_1, \ldots, \mathbf{x}_k$ the variables chosen by the algorithm.
1: initialize an empty matrix $\tilde{\mathbf{X}} = [\,]$
2: **for** $j = 1$ **to** r **do**
3: $\quad \hat{\alpha}_{cv}^j := \underset{\alpha}{argmin}\ CV_j(\alpha)$
4: $\quad \hat{\beta}_i^j := \hat{\beta}_{\mathcal{D}}^i(\hat{\alpha}_{cv}^j) \quad i = 1, \ldots, p$
5: **end for**
6: **for** $i = 1$ **to** p **do** $\bar{\beta}_i = \dfrac{1}{r}\sum_{j=1}^r \hat{\beta}_i^j$
$\quad \hat{\sigma}_i = \sqrt{\dfrac{1}{r}\sum_{j=1}^r (\hat{\beta}_i^j - \bar{\beta}_i)^2}$
$\quad Z_i = \sqrt{r}\bar{\beta}_i/\hat{\sigma}_i$
7: $\quad$ **if** $\mathbb{P}(|z| \geq |Z_i|) < p_i$ where $z \sim t(r-1)$ **then**
8: $\quad\quad \tilde{\mathbf{X}} = (\tilde{\mathbf{X}}, \mathbf{X}_i)$
9: $\quad$ **end if**
10: **end for**

Pseudocode 3.4.2: High Mean Lasso

Observation 3.4.1. *HF(max) was defined as a particular case of HF where the c parameter is tuned at priori based on some statistical considerations. It may be tempting to similarly define HM(95%) or HM(99%) as the algorithms that select all the variables such that (referring to line 7 of Pseudocode 3.4.2)*

$\mathbb{P}(|z| \geq |Z_i|)$ *is smaller than* 0.05 *or* 0.01 *respectively. In general this does not lead to good results as the* Z_i *values estimated by HM are often quiet large requiring p-values much smaller than* 0.01 *in order for irrelevant variables to be discarded. This can for example be observed in Figure 3.3 where the* $\frac{Z_i}{\sqrt{r}} = \frac{\bar{\beta}_i}{\hat{\sigma}_i}$ *values, estimated using the data described in Dataset 3.3.1, are reported.*

High Mean Percentile (HMP) is a sparser version of HM which is obtained by performing the same first steps of HM as can be seen from lines 1-8 of Pseudocode 3.4.3. Once the score Z is obtained only the variables with relative higher score are selected. In particular, we select the θ percentile of Z. The value of θ can be chosen using CV or according to the number of variables that we want to select.

Input: Set r the number of simulations, θ a percentile value
Output: $\mathbf{x}_1, \ldots, \mathbf{x}_k$ the variables chosen by the algorithm.
1: initialize an empty vector $\mathbf{Z} = (\ )$
2: **for** $j = 1$ **to** r **do**
3: $\quad \hat{\alpha}^j_{cv} := argmin CV_j(\alpha)$
4: $\quad \hat{\beta}^j_i := \hat{\beta}^{\alpha}_i(\hat{\alpha}^j_{cv}) \quad i = 1, \ldots, p$
5: **end for**
6: **for** $i = 1$ **to** p **do**
7: $\quad \bar{\beta}_i = \frac{1}{r}\sum_{j=1}^{r} \hat{\beta}^j_i$
8: $\quad \hat{\sigma}_i = \sqrt{\frac{1}{r}\sum_{j=1}^{r}(\hat{\beta}^j_i - \bar{\beta}_i)^2}$
9: $\quad Z_i = \sqrt{r}\bar{\beta}_i/\hat{\sigma}_i.$
10: $\quad$ **if** $Z_i \neq 0$ **then**
11: $\quad\quad \mathbf{Z} = (\mathbf{Z}, Z_i)$
12: $\quad$ **end if**
13: **end for**
14: Set $\hat{\theta}$ the θ percentile of $\mathbf{Z}$.
15: Select X_i if $S_i \geq \hat{\theta}$

Pseudocode 3.4.3: High Mean Percentile

3.4.3 Monte Carlo methods

Monte Carlo Cross-Validation (MCCV) is an alternative to K-folds Cross-Validation that has been extensively used in literature. The $MCCV$ has two

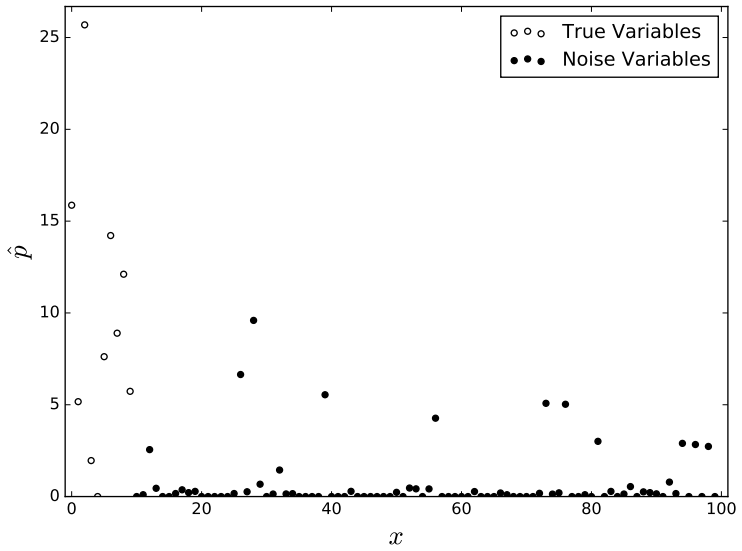


Figure 3.3: The value of $\frac{\bar{\beta}}{\hat{\sigma}}$ assigned to each variable by the HM algorithm. The true variables and the noise variables are represented by white and black circles, respectively.

user defined parameters v, q and is defined as

$$MCCV_{v,q}(\alpha) = \sum_{i=1}^{v} \parallel \mathbf{y}_{\phi(i)} - \mathbf{X}_{\phi(i)}\hat{\boldsymbol{\beta}}_{\bar{\phi}(i)}(\alpha) \parallel_2^2 \tag{3.20}$$

where v is the number of simulations, q is a number in $(0, 1)$ and represents the split between training and validation data, $\phi(i)$ is a random subset of $\mathcal{D} : |\phi(i)| = q|\mathcal{D}|$. As was the case for K-folds Cross-Validation, $MCCV_{v,q}$ can be computed r times but, in this case is simply equivalent to MCCV with different parameters i.e. $MCCV_{vr,q}$. This follows from the fact that:

$$\sum_{j=1}^{r}\sum_{i=1}^{v} \parallel \mathbf{y}_{\phi(i,j)} - \mathbf{X}_{\phi(i,j)}\hat{\boldsymbol{\beta}}_{\bar{\phi}(i,j)}(\alpha) \parallel_2^2 = \sum_{i=1}^{v\times r} \parallel \mathbf{y}_{\phi(i,j(i))} - \mathbf{X}_{\phi(i,j(i))}\hat{\boldsymbol{\beta}}_{\bar{\phi}(i,j(i))}(\alpha) \parallel_2^2 \tag{3.21}$$

Within the MCCV framework a number of different approaches can be considered.

- Global Alpha, Global Beta (GAGB): The regression coefficient are estimated as $\hat{\boldsymbol{\beta}}_{\mathcal{D}}(\hat{\alpha}_G)$ where $\hat{\alpha}_G$ is defined as:

$$\hat{\alpha}_G = \min_{\alpha} \quad MCCV_{v,q}(\alpha) \tag{3.22}$$

- Local Alpha, Global Beta (LAGB): A sequence of local penalty values, $\hat{\alpha}_L = (\hat{\alpha}_1^L, \ldots, \hat{\alpha}_v^L)$ is generated as described in the following equation.

$$\hat{\alpha}_i^L = \underset{\alpha}{\operatorname{argmin}} \parallel \mathbf{y}_{\phi(i)} - \mathbf{X}_{\phi(i)} \hat{\boldsymbol{\beta}}_{\bar{\phi}(i)}(\alpha) \parallel_2^2 \tag{3.23}$$

  A sequence of regression coefficients is then defined as $\{\hat{\boldsymbol{\beta}}_{\mathcal{D}}(\hat{\alpha}_i^L)\}_{i=1,\ldots,v}$ and techniques similar to the ones described for the Cross-Validation Methods in Section 3.3.2 can be used to estimate the set of active variables.

- Local Alpha, Local Beta (LALB): Starting from the sequence of penalty values obtained in equation 3.23 a sequence of regression coefficients is estimated as $\{\hat{\boldsymbol{\beta}}_{\phi(i)}(\hat{\alpha}_i^L)\}_{i=1,\ldots,v}$ and then techniques similar to the ones described for the Cross-Validation Methods in Section 3.3.2 can be used to estimate the set of active variables.

The distinction between LAGB and LALB is that in LALB $\boldsymbol{\beta}$ is computed for each $\phi(i)$ random subsample, while with LAGB the full dataset is used at all times. From [60] it can be easily deduced that the GAGB method will roughly behave like LCV with $K = 2$ folds. In particular, we can observe that if K is chosen such that $\frac{K-1}{K}|D| = q|D|$ then:

$$\parallel \mathbf{y}_{f^k} - \mathbf{X}_{f^k} \hat{\boldsymbol{\beta}}_{\bar{f}^k}(\alpha) \parallel_2^2 = \parallel \mathbf{y}_{\phi(i)} - \mathbf{X}_{\phi(i)} \hat{\boldsymbol{\beta}}_{\bar{\phi}(i)}(\alpha) \parallel_2^2 \quad if \quad \bar{f}^k = \bar{\phi}(i) \tag{3.24}$$

and so MCCV will behave very similarly to repeated K-folds CV [88]. The main difference is that while in MCCV the data is split totally randomly in repeated K-folds CV the data split is more structured. Therefore

$$\sum_{i=1}^{r} \sum_{k=1}^{K} \parallel \mathbf{y}_{f^k} - \mathbf{X}_{f^k} \hat{\boldsymbol{\beta}}_{\bar{f}^k}(\alpha) \parallel_2^2 = \sum_{i=1}^{r} \parallel \mathbf{y}_{\phi(i)} - \mathbf{X}_{\phi(i)} \hat{\boldsymbol{\beta}}_{\bar{\phi}(i)}(\alpha) \parallel_2^2 \quad if \quad K = 1 \tag{3.25}$$

In particular, if $r >> 0$ replacing $=$ with $\approx$ the previous equation holds also if $K \neq 1$. From this, it follows that the methods based on LAGB will return results very similar to those of the methods described in Section 3.3.2. The only Monte Carlo method that brings something new in our study is the LALB. We will define Monte Carlo High Frequency (MCHF) and Monte Carlo High Mean (MCHM) as two methods similar to HM and HF based on this MCCV variant.

3.5 Comparison of Methods

All the methods presented in the previous section estimate a subset $\hat{S}$ of the original variables with the aim of optimally predicting $\mathbf{y}$. It is important to obtain $\hat{S}$ that is as small as possible and at the same time to include all the variables that influence the output $\mathbf{y}$. The methods are therefore evaluated according to how close $\hat{S}$ is to the original set of variables S_0 and to how well a linear model built with the selected variables can approximate $\mathbf{y}$.

Before presenting the comparison some simulated datasets are described. These will be used as illustrative examples and in the simulation study in this section.

Dataset 3.5.1. Several datasets are generated using a linear model as in Equation 3.1. Here $\boldsymbol{\beta}$ is a vector with only 20 nonzero elements that are randomly generated uniformly between $(-2, 2)$ with the restriction $|\boldsymbol{\beta}| > 0.1$, and $\epsilon \sim N(0, \sigma_\epsilon^2)$ where σ_ϵ^2 has been chosen according to the value of the Signal-to-Noise ratio $SNR = \dfrac{Var(\mathbf{X}\boldsymbol{\beta})}{\sigma_\epsilon^2}$. In particular, different structures of $\mathbf{X}$ that have already been used in the literature [71] have been considered:

(A) Independent predictor variables. All predictor variables are i.i.d. standard normally distributed random variables. This dataset can be considered a simple one where the covariance between variables is not an issue.

(B) Factor model with 10 factors. Let $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{10}$ be ten latent variables following i.i.d. standard normal distributions. Each predictor variable $\mathbf{x}_k$, for $k = 1, \ldots, p$ is generated as $\mathbf{x}_k = f_{k,1}\boldsymbol{\phi}_1 + f_{k,2}\boldsymbol{\phi}_2 + \ldots + f_{k,10}\boldsymbol{\phi}_{10} + \boldsymbol{\eta}_k$ where $f_{k,1}, \ldots, f_{k,10}$, $\boldsymbol{\eta}_k$ have i.i.d. standard normal distribution for all $k = 1, \ldots, p$.

(C) The same as B but with 2 instead of 10 factors.

(D) Toeplitz design. The predictor variable follows a $\mathcal{N}(0, \boldsymbol{\Sigma})$ distribution, where $\Sigma_{k,m} = 0.7^{|k-m|}$.

The introduced datasets are characterized by different correlation structures. The next example investigates the likelihood of the irrepresentable condition holding in each dataset.

Example 3.5.1. Several instances of the data described in Dataset 3.5.1 are generated and for each one the value on the left term of equation 3.11

is computed in order to establish if the dataset satisfies the irrepresentable condition (IC). The results, which are reported in Table 3.1, show that the IC is strongly influenced by the values of n, p, the number of nonzero regression coefficients ($\| \boldsymbol{\beta}_{S_0} \|_0$) and by the correlation of the data. As expected, A is the dataset that is more likely to satisfy the IC due to the low correlation between its variables. It is interesting to observe that even though variables in B are less correlated than variables in C, B is less likely to satisfy IC than C. This happens because the false variables can be better approximated by a linear combination of the true variables in dataset B than in dataset C. The relation between p and n also has a strong influence. Even dataset A fails to satisfy IC if p is too large compared to n. Dataset A with $\| \boldsymbol{\beta}_{S_0} \|_0 = 10$ ranges from 100% to 0% for different values of n and p. It is also interesting to observe the effect of $\| \boldsymbol{\beta}_{S_0} \|_0$. Even when $n > p$ IC may not hold if $\| \boldsymbol{\beta}_{S_0} \|_0$ is large. In $A(n = 100, p = 50, \| \boldsymbol{\beta}_{S_0} \|_0 = 20)$, for example, the IC is satisfied only 6% of the time while it is satisfied 96% of the time in $A(n = 100, p = 50, \| \boldsymbol{\beta}_{S_0} \|_0 = 5)$.

3.5.1 Highly Correlated Variables

OES data is characterized by high correlation. Many of the methods presented break in the presence of highly correlated variables. It is then important to understand how they behave under this condition. The next example shows that when variables are estimated using different subsamples of the data, a relevant variables which is correlated with some others is not always selected.

Example 3.5.2. Let $\mathbf{x}_1, \mathbf{x}_{11}, \ldots, \mathbf{x}_{50} \sim N(0, 1)$, $\epsilon_1, \ldots, \epsilon_{10} \sim N(0, 0.0001)$ and $\epsilon \sim N(0, 2)$ be independent random variables. A set of variables correlated with $\mathbf{x}_1$ is defined as: $\mathbf{x}_i = \mathbf{x}_1 + \epsilon_i \; for \; i = 1, \ldots, 10$. The full data is then $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{50})$. The data is generated such that $\mathbf{X} \in \mathbb{R}^{50 \times 50}$. The vector containing the regression coefficients is all 0 except $\beta_1 = 1, \beta_{11} = 1, \beta_{12} = 1$. The output $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon$. In this dataset the first 10 variables are very correlated. Therefore in order to have a good recovery performance a model should select at least one variable from this group, $\mathbf{x}_{11}$ and $\mathbf{x}_{12}$. All the methods that assign a score to each variable among the ones described in Section 3.3 are applied to the dataset. The score assigned to each variable by each method is reported in Figure 3.4. While all methods are able to correctly assign a large score to $\mathbf{x}_{11}$ and $\mathbf{x}_{12}$ none recognize $\mathbf{x}_1$ as a relevant variable. This problem is mitigated by the selection of one of the variables correlated with $\mathbf{x}_1$ ($\tilde{\mathbf{x}}_1$). This ensures good prediction performances even if the identified model is not the right one. HM and MCHM are the only methods able to

Data	n	p	$\| \boldsymbol{\beta}_{S_0} \|_0$	IC	Data	n	p	$\| \boldsymbol{\beta}_{S_0} \|_0$	IC
A	100.0	50.0	5.0	96.0	B	100.0	50.0	5.0	3.0
A	100.0	50.0	10.0	56.0	B	100.0	50.0	10.0	0.0
A	100.0	50.0	20.0	6.0	B	100.0	50.0	20.0	0.0
A	100.0	100.0	5.0	91.0	B	100.0	100.0	5.0	4.0
A	100.0	100.0	10.0	33.0	B	100.0	100.0	10.0	0.0
A	100.0	100.0	20.0	0.0	B	100.0	100.0	20.0	0.0
A	100.0	500.0	5.0	73.0	B	100.0	500.0	5.0	1.0
A	100.0	500.0	10.0	4.0	B	100.0	500.0	10.0	0.0
A	100.0	500.0	20.0	0.0	B	100.0	500.0	20.0	0.0
A	100.0	1000.0	5.0	62.0	B	100.0	1000.0	5.0	1.0
A	100.0	1000.0	10.0	0.0	B	100.0	1000.0	10.0	0.0
A	100.0	1000.0	20.0	0.0	B	100.0	1000.0	20.0	0.0
A	300.0	50.0	5.0	100.0	B	300.0	50.0	5.0	1.0
A	300.0	50.0	10.0	100.0	B	300.0	50.0	10.0	0.0
A	300.0	50.0	20.0	86.0	B	300.0	50.0	20.0	0.0
A	300.0	100.0	5.0	100.0	B	300.0	100.0	5.0	5.0
A	300.0	100.0	10.0	98.0	B	300.0	100.0	10.0	0.0
A	300.0	100.0	20.0	81.0	B	300.0	100.0	20.0	0.0
A	300.0	500.0	5.0	100.0	B	300.0	500.0	5.0	2.0
A	300.0	500.0	10.0	98.0	B	300.0	500.0	10.0	0.0
A	300.0	500.0	20.0	34.0	B	300.0	500.0	20.0	0.0
A	300.0	1000.0	5.0	100.0	B	300.0	1000.0	5.0	1.0
A	300.0	1000.0	10.0	97.0	B	300.0	1000.0	10.0	0.0
A	300.0	1000.0	20.0	29.0	B	300.0	1000.0	20.0	0.0
C	100.0	50.0	5.0	38.0	D	100.0	50.0	5.0	12.0
C	100.0	50.0	10.0	38.0	D	100.0	50.0	10.0	3.0
C	100.0	50.0	20.0	37.0	D	100.0	50.0	20.0	0.0
C	100.0	100.0	5.0	32.0	D	100.0	100.0	5.0	11.0
C	100.0	100.0	10.0	29.0	D	100.0	100.0	10.0	0.0
C	100.0	100.0	20.0	17.000	D	100.0	100.0	20.0	0.0
C	100.0	500.0	5.0	31.0	D	100.0	500.0	5.0	14.0
C	100.0	500.0	10.0	25.0	D	100.0	500.0	10.0	1.0
C	100.0	500.0	20.0	1.0	D	100.0	500.0	20.0	0.0
C	100.0	1000.0	5.0	29.0	D	100.0	1000.0	5.0	9.0
C	100.0	1000.0	10.0	11.0	D	100.0	1000.0	10.0	0.0
C	100.0	1000.0	20.0	1.0	D	100.0	1000.0	20.0	0.0
C	300.0	50.0	5.0	46.0	D	300.0	50.0	5.0	12.0
C	300.0	50.0	10.0	65.0	D	300.0	50.0	10.0	4.0
C	300.0	50.0	20.0	74.0	D	300.0	50.0	20.0	1.0
C	300.0	100.0	5.0	39.0	D	300.0	100.0	5.0	14.0
C	300.0	100.0	10.0	55.0	D	300.0	100.0	10.0	3.0
C	300.0	100.0	20.0	71.0	D	300.0	100.0	20.0	0.0
C	300.0	500.0	5.0	38.0	D	300.0	500.0	5.0	12.0
C	300.0	500.0	10.0	41.0	D	300.0	500.0	10.0	1.0
C	300.0	500.0	20.0	45.0	D	300.0	500.0	20.0	0.0
C	300.0	1000.0	5.0	35.0	D	300.0	1000.0	5.0	16.0
C	300.0	1000.0	10.0	41.0	D	300.0	1000.0	10.0	1.0
C	300.0	1000.0	20.0	36.0	D	300.0	1000.0	20.0	0.0

Table 3.1: The percentage of time that a random realization of each dataset described in Dataset 3.5.1 satisfies the irrepresentable condition (IC) for different values of n and p and number of nonzero regression coefficients ($\| \boldsymbol{\beta}_{S_0} \|_0$).

separate $(\tilde{\mathbf{x}}_1, \mathbf{x}_{11}, \mathbf{x}_{12})$ from the noise variables. Also HF and MCHF are able to assign to the 3 variables an higher score than most of the other variables. SS is not able to strongly separate $\mathbf{x}_1$ or any of its correlated variables from the rest. This is a common problem of methods that select variables using different subsets of the data. Indeed at each iteration a different variable in $\mathbf{x}_1, \ldots, \mathbf{x}_{10}$ is selected. The same problem can affect the Monte Carlo based methods. In this case for example MCHM and MCHF assign to $\tilde{\mathbf{x}}_1$ a lower score than HM and HF.

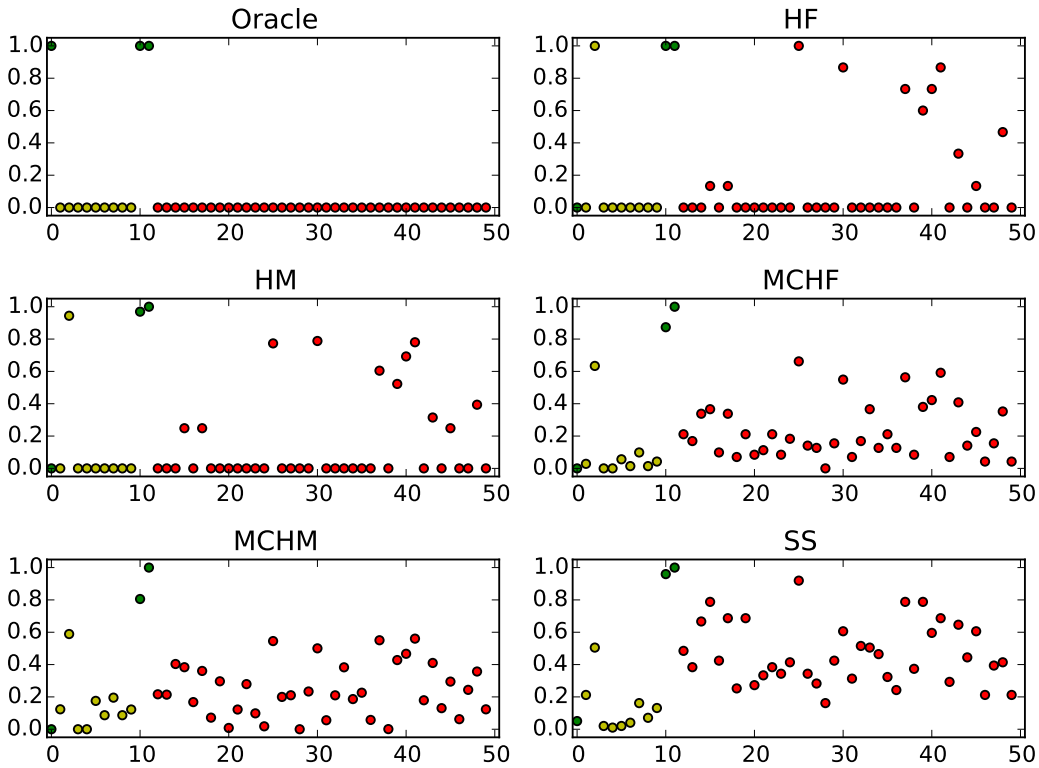


Figure 3.4: Score assigned to each variable by the various algorithms on the datasets described in Example 3.5.2

The previous example shows that bootstrap based methods are not reliable when used with highly correlated datasets. This problem can be solved by removing the groups of correlated variables. This may be achieved by pre-processing the data with algorithms such as Forward Selection Component Analysis [89] or Max Separation Clustering [90] which will be introduced in

Chapters 4, 5 and 6, respectively.

From the previous example it follows that Bootstrap based methods are weak when the data is composed of groups of highly correlated variables. On the other hand they seem to be the only algorithms able to correctly infer the set of active variables when the irrepresentable condition does not hold. This is shown in the next example.

Example 3.5.3. In this example we will discuss the performance of the different approaches on a dataset where the irrepresentable condition (equation 3.11) does not hold. The dataset considered is the same as the one used in [83]. To define this dataset we generate the variables $\mathbf{x}_1,\ \mathbf{x}_2,\ \mathbf{e}, \boldsymbol{\epsilon} \sim\ N(0,1)$. A third noise variable is defined as $\mathbf{x}_3 = \frac{2}{3}\mathbf{x}_1 + \frac{2}{3}\mathbf{x}_2 + \frac{1}{3}\mathbf{e}$ and finally the output is given by $\mathbf{y} = 2\mathbf{x}_1 + 3\mathbf{x}_2 + \boldsymbol{\epsilon}$. In this dataset we use $n = 1000$ samples. Observe that in this dataset the irrepresentable condition does not hold [83]. This leads to the presence of a false positive for each value of the penalty α as can be observed in Figure 3.5. It follows that all the algorithms based on Cross-Validation will not be able to correctly estimate $\hat{\beta}_3 = 0$. In the second plot we can observe that Stability Selection with weakness $u = 0.8$ is able to recognize $\mathbf{x}_3$ as a true negative. Also in this example we can observe that the result can be strongly influenced by the choice of u. Stability Selection with $u = 0.2$ is not able to correctly classify the variables. In practice with real datasets it would be difficult determine the correct value for u. The method MCHF estimates the selection probability of each variable as $\hat{\mathbf{p}} = (1, 1, 0.97)$ and hence is able to exclude the third variable. In this example MCHF is able to estimate the correct set of variables without the need for difficult parameter tuning. MCHM estimates $\mathbf{Z}$ as $\mathbf{Z} = (98.42, 141.87, 13.38)$ and hence in this case there is a good separation between $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$. Also in this case it can be difficult to determine a way to automatically remove $\mathbf{x}_3$ from the active set.

3.5.2 Not Convexity of KSC Score Function

KSC selects the penalty value $\hat{\alpha}$ in order to maximise the function $\hat{s}$. The following example shows a situation in which the $\hat{s}$ function has two maxima making it difficult to understand which values of α should be chosen.

Example 3.5.4. Consider the matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{103})$ where $\mathbf{x}_1, \ldots, \mathbf{x}_{103} \sim N(0,1)$ and $cor(\mathbf{x}_1, \mathbf{x}_2) = cor(\mathbf{x}_1, \mathbf{x}_3) = cor(\mathbf{x}_2, \mathbf{x}_3) = 0.9$ and $\mathbf{x}_4, \ldots, \mathbf{x}_{103}$ are independent. The output vector is defined as $\mathbf{y} = 2\mathbf{x}_1 + \mathbf{x}_3 + \mathbf{x}_{10} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim N(0, 0.5)$. The values of $\frac{\hat{s}(\alpha)}{max_\alpha(\hat{s}(\alpha))}$ for this example are reported in

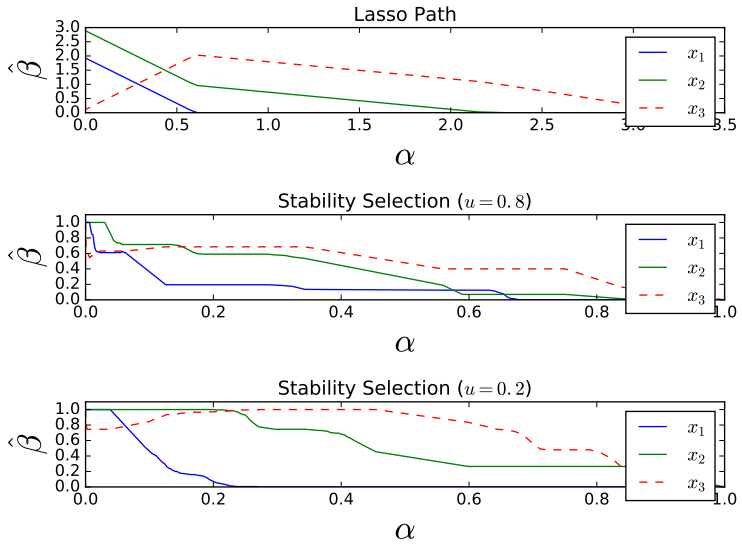


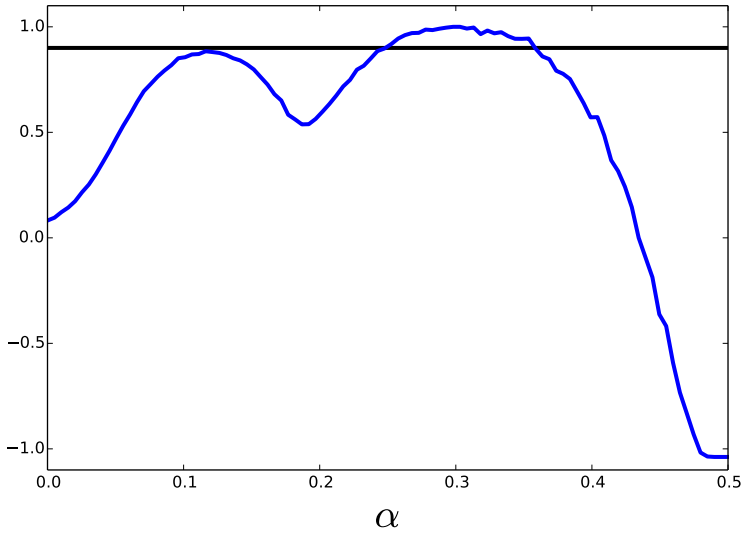
Figure 3.5: The Lasso path and the stability selection path with weakness $u = 0.2$ and $u = 0.8$ for the data described in Example 3.5.3. The dashed line is $\hat{\beta}_3(\alpha)$.

Figure 3.6. The figure shows two different maxima. It is difficult to decide if the first one should be ignored or considered. In this case for values of α around 0.1 (corresponding to the first maximum) the lasso selects 44 variables while for values of α around 0.25 (corresponding to the second maximum) it selects only one variable.

3.5.3 Performance Evaluation

The algorithms described in Section 3.3 are compared on the datasets defined in Dataset 3.5.1. For each type of data different datasets are generated varying the values of n, p and SNR. The experiment is conducted according to the following modeling rationale. Two versions of each dataset described in Dataset 3.5.1 are generated in order to have a training set and a testing set. The training set is used to select a set of variables with each method. Then, using the training set a model is estimated using the ridge estimator considering only the selected variables. The optimum ridge penalty is determined using 10 folds Cross-Validation. The testing set is only used to evaluate the prediction performance of the estimated models. The methods are evaluated based on the following criteria:

Figure 3.6: The function $\frac{\hat{s}(\alpha)}{max_\alpha(\hat{s}(\alpha))}$ obtained with KSC as described in Example 3.5.4



- Normalised Mean Square Error defined as:

$$MSE = \frac{\| \mathbf{y} - \hat{\mathbf{y}} \|_2^2}{\| \mathbf{y} \|_2^2} \tag{3.26}$$

- Number of Selected Variables
- Sensitivity
- Specificity

3.5.3.1 Data Preprocessing

In order to use lasso the data must be scaled as a preprocessing step. A dataset $\mathbf{X} = \{\tilde{x}_{i,j}\}_{i=1,\dots,n;\ j=1,\dots,p} \in \mathbb{R}^{n \times p}$, where each variable has mean μ_i and variance σ_i^2, is scaled to zero mean and unit variance as:

$$\tilde{\mathbf{X}} = \{\tilde{x}_{i,j}\}_{i=1,\dots,n;\ j=1,\dots,p} \tag{3.27}$$

where

$$\tilde{x}_{i,j} = \frac{x_{i,j} - \mu_i}{\sigma_i} \tag{3.28}$$

In practice the data will often be divided into a training dataset and a test dataset:

$$\mathbf{X}^{train} = \{x_{i,j}^{train}\}_{i=1,\ldots,n;\ j=1,\ldots,p} \ and \ \mathbf{X}^{test} = \{x_{i,j}^{test}\}_{i=1,\ldots,n;\ j=1,\ldots,p} \tag{3.29}$$

If $\boldsymbol{\mu}^{train}$, $\boldsymbol{\sigma}^{train}$ are the mean and the standard deviation of $\mathbf{X}^{train}$ respectively the two dataset can be scaled as:

$$\tilde{\mathbf{X}}^{train} = \{\tilde{x}_{i,j}^{train}\}_{i=1,\ldots,n;\ j=1,\ldots,p} \ and \ \tilde{\mathbf{X}}^{test} = \{\tilde{x}_{i,j}^{test}\}_{i=1,\ldots,n;\ j=1,\ldots,p} \tag{3.30}$$

where

$$\tilde{x}_{i,j}^{train} = \frac{x_{i,j}^{train} - \mu_j^{train}}{\sigma_j^{train}} \quad and \quad \tilde{x}_{i,j}^{test} = \frac{x_{i,j}^{test} - \mu_j^{train}}{\sigma_j^{train}} \tag{3.31}$$

In general the $\tilde{\mathbf{X}}^{test}$ dataset will not have zero mean or unit variance.

3.5.3.2 Discussion of the Results

The average of the relevant performance metrics, as defined in section 3.5.3 are reported in Tables 3.2, 3.3, 3.4, 3.5. (The standard deviation and a graphical representation of the results are reported in Appendix A). Some interesting conclusions can be reached from an analysis of the tables. Among the reported metrics the most interesting is the Normalised Mean Square Error (NMSE) obtained with the various algorithms on the test datasets. First of all it is clear that the performance of all the algorithms is strongly influenced by the SNR value and by the ratio between the number of samples (n) and the number of variables (p). The performance of each algorithm improves with larger values of SNR and with smaller values of p/n. This shows that the main influencing factors are the quality of the measurements and the number of collected samples and that the choice of the algorithm plays only a secondary role. That said, the choice of the algorithm is still an important influencing factor. As expected the Oracle estimator always yields the lowest prediction error. The difference between the Oracle estimator and the other methods is particularly marked when only $n = 100$ samples are used. In this case SNR does not play an important role. This may be explained by the presence of a larger number of correctly selected variables as MSE and Sensitivity seems to follow the same trend. From a prediction point of view the inclusion of several irrelevant variables does not significantly reduce the performance of the model. This can for example be observed in dataset A with $n = 100$, $p = 1000$ and $SNR = 1$. Here MCHM and MCHF select a large number of irrelevant variables as can be observed from their low specificity values. Despite this their prediction performances are roughly

equivalent to the ones of the other methods. This can be explained by the fact that when an irrelevant variable is selected by virtue of using the ridge estimator, its estimated regression coefficient is small and as such it does not have much influence on the predicted output.

Observation 3.5.1. *The lasso hyper-parameters are often estimated with Cross-Validation. This may not be the optimal procedure if the aim is to reconstruct the set S_0 as closely as possible. Cross-validation chooses the parameters in order to minimize the prediction error. In order to achieve good prediction performance a large sensitivity value is required and the result is not penalized by having a low specificity value. It follows that often when hyper-parameters are estimated with Cross-Validation many false positives can be included in the estimated set of active variables i.e.:*

$$S_0 \subset \hat{S} \tag{3.32}$$

Resampling based methods: Among the presented methods only SS and KSC are not based on Cross-Validation. Both these methods perform worse than the methods based on Cross-Validation in terms of NMSE. This was expected as the methods are designed in order to select a robust set of variables. This does not necessarily imply that the estimated set of active variable will lead to good prediction performance. In order to mitigate this problem in [91] the KSC parameter is tuned in order to maximise a function which combines the strength of both stability selection (measured with the Cohen's Kappa coefficient as in Pseudocode 3.3.2) and prediction error (measured with Cross-Validation). In terms of sensitivity and specificity SS and KSC tend to have different behaviours. SS has generally low sensitivity while it performs well in terms of specificity. In contrast KSC tends to select a large number of variables resulting in large sensitivity and low specificity. SS is very difficult to automate and better performance can be obtained by visually analysing the stability path. The same is true for KSC as manual control is required in order to avoid the problems described in example 3.5.4. In conclusion, we discourage the use of SS and KSC in automated industrial applications.

Cross-Validation Based Methods: All the other methods are based on bootstrap or Cross-Validation. These obtain generally better prediction performances than SS and KSC. All these methods tend to have the same NMSE performance. It is then reasonable to assume that they all select the most relevant variables and that differences in the sensitivity values may be due to the presence or the absence of variables with small regression coefficient.

Methods with similar NMSE performance can then be evaluated according to the number of selected variables. The number of selected variables depends on the choice of a threshold in HM, HF, MCHM and MCHF and by the chosen percentile in LP. These parameters are tuned with Cross-Validation. It is then clear that they will select a larger number of variables. On the other hand with HF(max) where the parameter $\bar{p}$ is not chosen via Cross-Validation the number of variables selected tends to be lower.

	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec
	n=100 p=100 SNR=1 Data=A				n=100 p=100 SNR=1.5 Data=A				n=100 p=100 SNR=2 Data=A			
Real	0.545	20.000	1.000	1.000	0.409	20.000	1.000	1.000	0.313	20.000	1.000	1.000
HM	0.687	37.111	0.685	0.695	0.538	38.333	0.750	0.699	0.416	38.111	0.802	0.712
HF	0.676	36.778	0.701	0.702	0.530	30.556	0.687	0.780	0.409	35.000	0.817	0.755
MCHM	0.656	21.556	0.599	0.865	0.518	24.000	0.633	0.847	0.406	28.000	0.739	0.823
MCHF	0.659	22.556	0.584	0.850	0.519	22.889	0.624	0.858	0.401	25.000	0.724	0.857
HF (max)	0.664	18.000	0.481	0.884	0.530	24.444	0.607	0.836	0.405	29.444	0.759	0.810
SS	0.726	33.000	0.444	0.692	0.604	29.667	0.404	0.725	0.541	25.889	0.385	0.770
LP	0.672	18.444	0.465	0.874	0.535	28.667	0.642	0.792	0.408	34.222	0.805	0.762
KSC	0.939	87.111	0.945	0.144	0.678	83.444	0.910	0.182	0.500	80.889	0.896	0.211
LCV	0.661	19.111	0.510	0.875	0.531	31.556	0.676	0.765	0.411	37.222	0.811	0.726
	n=100 p=500 SNR=1 Data=A				n=100 p=500 SNR=1.5 Data=A				n=100 p=500 SNR=2 Data=A			
Real	0.618	20.000	1.000	1.000	0.398	20.000	1.000	1.000	0.306	20.000	1.000	1.000
HM	0.799	57.444	0.425	0.898	0.590	56.556	0.580	0.906	0.497	59.000	0.684	0.905
HF	0.792	52.111	0.414	0.908	0.584	56.222	0.585	0.907	0.483	67.111	0.701	0.889
MCHM	0.824	81.556	0.469	0.849	0.596	60.889	0.587	0.897	0.508	54.444	0.626	0.912
MCHF	0.809	43.556	0.364	0.924	0.594	59.778	0.576	0.899	0.493	44.444	0.590	0.932
HF (max)	0.763	19.667	0.269	0.970	0.576	30.000	0.478	0.957	0.481	39.778	0.602	0.942
SS	0.814	73.667	0.294	0.859	0.681	19.556	0.173	0.966	0.589	18.333	0.201	0.970
LP	0.766	30.111	0.309	0.950	0.589	44.889	0.512	0.927	0.473	57.667	0.664	0.907
KSC	0.834	96.333	0.517	0.821	0.617	96.222	0.655	0.826	0.499	94.667	0.723	0.832
LCV	0.774	27.778	0.294	0.954	0.585	52.111	0.590	0.916	0.480	62.889	0.677	0.897
	n=100 p=1000 SNR=1 Data=A				n=100 p=1000 SNR=1.5 Data=A				n=100 p=1000 SNR=2 Data=A			
Real	0.573	20.000	1.000	1.000	0.427	20.000	1.000	1.0	0.334	20.000	1.000	1.000
HM	0.779	52.444	0.318	0.953	0.657	67.889	0.528	0.941	0.561	68.667	0.614	0.942
HF	0.781	50.667	0.312	0.955	0.650	65.222	0.523	0.944	0.568	70.444	0.629	0.941
MCHM	0.777	161.778	0.464	0.844	0.673	159.333	0.594	0.849	0.591	126.111	0.617	0.884
MCHF	0.773	173.556	0.448	0.832	0.674	115.444	0.556	0.893	0.586	97.444	0.612	0.913
HF (max)	0.781	16.889	0.178	0.986	0.667	21.444	0.317	0.984	0.562	31.333	0.485	0.978
SS	0.788	62.556	0.244	0.941	0.686	70.889	0.335	0.934	0.622	91.778	0.463	0.916
LP	0.768	18.333	0.194	0.985	0.659	42.667	0.456	0.966	0.568	59.778	0.585	0.951
KSC	0.784	96.444	0.335	0.908	0.658	96.667	0.557	0.913	0.553	96.000	0.641	0.915
LCV	0.775	24.111	0.223	0.980	0.654	47.000	0.451	0.961	0.556	54.000	0.540	0.956
	n=300 p=100 SNR=1 Data=A				n=300 p=100 SNR=1.5 Data=A				n=300 p=100 SNR=2 Data=A			
Real	0.324	20.000	1.000	1.000	0.217	20.000	1.000	1.000	0.164	20.000	1.000	1.000
HM	0.349	36.333	0.834	0.745	0.237	41.111	0.902	0.697	0.177	42.000	0.945	0.696
HF	0.353	37.111	0.841	0.736	0.236	40.222	0.921	0.713	0.179	42.222	0.952	0.696
MCHM	0.350	31.556	0.829	0.801	0.232	24.000	0.811	0.887	0.174	27.111	0.921	0.873
MCHF	0.348	25.778	0.787	0.862	0.231	27.000	0.865	0.862	0.174	24.556	0.890	0.898
HF (max)	0.352	33.222	0.823	0.780	0.235	36.889	0.902	0.750	0.178	40.222	0.952	0.720
SS	0.423	14.111	0.238	0.881	0.319	25.000	0.412	0.786	0.310	13.778	0.221	0.881
LP	0.353	34.444	0.823	0.765	0.235	38.444	0.914	0.734	0.179	41.667	0.952	0.703
KSC	0.360	58.222	0.945	0.500	0.242	55.444	0.946	0.532	0.181	52.667	0.976	0.573
LCV	0.354	38.889	0.848	0.716	0.236	40.556	0.927	0.711	0.179	42.889	0.958	0.689
	n=300 p=500 SNR=1 Data=A				n=300 p=500 SNR=1.5 Data=A				n=300 p=500 SNR=2 Data=A			
Real	0.328	20.000	1.000	1.000	0.222	20.000	1.000	1.000	0.167	20.000	1.000	1.000
HM	0.396	58.333	0.744	0.909	0.273	75.667	0.891	0.878	0.208	78.222	0.926	0.874
HF	0.392	51.444	0.727	0.922	0.271	68.000	0.885	0.894	0.207	74.667	0.926	0.882
MCHM	0.393	58.444	0.709	0.907	0.264	58.222	0.873	0.914	0.198	57.556	0.914	0.917
MCHF	0.397	65.889	0.743	0.893	0.270	61.889	0.861	0.906	0.200	61.222	0.914	0.909
HF (max)	0.382	40.556	0.698	0.944	0.261	49.889	0.862	0.931	0.197	54.222	0.908	0.924
SS	0.480	89.667	0.377	0.829	0.377	129.111	0.581	0.755	0.329	67.222	0.322	0.873
LP	0.388	44.556	0.704	0.936	0.263	54.444	0.867	0.921	0.200	57.778	0.908	0.916
KSC	0.508	229.000	0.830	0.557	0.331	212.889	0.925	0.594	0.253	209.667	0.960	0.602
LCV	0.389	47.333	0.727	0.931	0.265	58.333	0.879	0.914	0.203	65.000	0.914	0.901
	n=300 p=1000 SNR=1 Data=A				n=300 p=1000 SNR=1.5 Data=A				n=300 p=1000 SNR=2 Data=A			
Real	0.308	20.000	1.000	1.000	0.221	20.000	1.000	1.000	0.167	20.000	1.000	1.000
HM	0.388	89.000	0.658	0.922	0.277	94.889	0.837	0.920	0.211	102.000	0.899	0.914
HF	0.387	85.667	0.658	0.926	0.277	91.444	0.832	0.923	0.212	98.889	0.899	0.917
MCHM	0.378	78.222	0.653	0.933	0.274	83.333	0.803	0.931	0.202	65.444	0.877	0.951
MCHF	0.376	71.111	0.658	0.941	0.273	74.444	0.803	0.940	0.209	81.333	0.877	0.935
HF (max)	0.367	45.667	0.613	0.966	0.266	52.222	0.792	0.963	0.202	60.000	0.888	0.957
SS	0.427	227.000	0.455	0.777	0.367	87.222	0.338	0.918	0.317	140.556	0.521	0.867
LP	0.371	54.444	0.613	0.957	0.269	67.111	0.820	0.948	0.205	69.889	0.888	0.947
KSC	0.417	232.778	0.743	0.777	0.306	245.111	0.911	0.768	0.236	244.889	0.950	0.769
LCV	0.374	56.889	0.619	0.954	0.271	71.222	0.826	0.944	0.207	78.222	0.899	0.938

Table 3.2: The mean value of MSE, the number of selected variables, the sensitivity and the specificity of each method when applied to dataset A. The reported values are the mean over 10 Monte Carlo repetitions.

	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec
	n=100 p=100 SNR=1 Data=B				n=100 p=100 SNR=1.5 Data=B				n=100 p=100 SNR=2 Data=B			
Real	0.588	20.000	1.000	1.000	0.403	20.000	1.000	1.000	0.315	20.000	1.000	1.000
HM	0.629	19.333	0.276	0.826	0.437	25.556	0.42	0.781	0.345	28.444	0.511	0.764
HF	0.624	17.444	0.264	0.846	0.439	22.222	0.408	0.819	0.339	22.889	0.461	0.822
MCHM	0.63	18.000	0.283	0.844	0.446	20.333	0.346	0.829	0.347	23.556	0.476	0.818
MCHF	0.623	19.444	0.305	0.830	0.431	22.889	0.414	0.812	0.344	25.333	0.468	0.794
HF (max)	0.613	11.333	0.182	0.902	0.435	15.000	0.306	0.885	0.341	16.333	0.396	0.888
SS	0.635	13.556	0.198	0.879	0.483	15.889	0.271	0.866	0.416	12.111	0.253	0.907
LP	0.617	16.556	0.268	0.857	0.430	18.667	0.342	0.848	0.341	22.111	0.432	0.825
KSC	0.708	53.222	0.594	0.481	0.476	40.889	0.575	0.628	0.368	43.556	0.627	0.606
LCV	0.615	18.333	0.273	0.837	0.440	23.111	0.409	0.809	0.340	23.556	0.462	0.814
	n=100 p=500 SNR=1 Data=B				n=100 p=500 SNR=1.5 Data=B				n=100 p=500 SNR=2 Data=B			
Real	0.633	20.000	1.000	1.000	0.432	20.000	1.000	1.000	0.331	20.000	1.000	1.000
HM	0.731	32.889	0.145	0.937	0.504	36.333	0.208	0.933	0.404	41.444	0.269	0.925
HF	0.732	31.778	0.151	0.940	0.504	34.222	0.208	0.937	0.389	36.556	0.262	0.935
MCHM	0.756	113.667	0.348	0.777	0.498	70.333	0.314	0.866	0.401	106.444	0.441	0.796
MCHF	0.695	47.333	0.190	0.909	0.502	53.444	0.252	0.899	0.389	54.333	0.330	0.900
HF (max)	0.678	10.556	0.087	0.981	0.509	12.333	0.132	0.98	0.392	17.0	0.177	0.972
SS	0.709	37.000	0.189	0.931	0.493	25.444	0.178	0.954	0.402	26.111	0.194	0.953
LP	0.672	14.667	0.110	0.974	0.474	17.556	0.151	0.969	0.374	23.333	0.206	0.960
KSC	0.768	67.778	0.206	0.867	0.532	64.889	0.244	0.875	0.408	51.778	0.263	0.903
LCV	0.684	18.889	0.104	0.965	0.473	20.111	0.149	0.964	0.371	22.667	0.194	0.961
	n=100 p=1000 SNR=1 Data=B				n=100 p=1000 SNR=1.5 Data=B				n=100 p=1000 SNR=2 Data=B			
Real	0.584	20.000	1.000	1.000	0.401	20.000	1.000	1.000	0.313	20.000	1.000	1.000
HM	0.642	37.333	0.089	0.964	0.492	43.778	0.151	0.958	0.399	46.111	0.210	0.957
HF	0.641	36.778	0.089	0.964	0.493	46.444	0.151	0.956	0.400	49.333	0.227	0.954
MCHM	0.671	195.889	0.265	0.806	0.483	160.333	0.309	0.843	0.401	161.000	0.394	0.844
MCHF	0.686	124.667	0.197	0.877	0.479	165.333	0.325	0.838	0.389	122.889	0.338	0.881
HF (max)	0.616	11.444	0.033	0.989	0.449	13.444	0.078	0.988	0.377	14.667	0.106	0.987
SS	0.694	25.222	0.05	0.975	0.575	11.111	0.022	0.989	0.486	12.556	0.050	0.988
LP	0.617	14.667	0.039	0.986	0.442	20.778	0.106	0.981	0.367	24.667	0.146	0.978
KSC	0.726	88.333	0.156	0.913	0.512	85.111	0.241	0.918	0.405	80.778	0.297	0.924
LCV	0.620	14.444	0.044	0.986	0.454	25.444	0.117	0.976	0.373	30.778	0.180	0.972
	n=300 p=100 SNR=1 Data=B				n=300 p=100 SNR=1.5 Data=B				n=300 p=100 SNR=2 Data=B			
Real	0.313	20.000	1.000	1.000	0.210	20.000	1.000	1.000	0.159	20.000	1.000	1.000
HM	0.327	22.556	0.444	0.823	0.220	30.889	0.678	0.773	0.167	38.111	0.784	0.708
HF	0.325	23.667	0.455	0.812	0.220	32.000	0.689	0.762	0.167	36.333	0.797	0.732
MCHM	0.326	20.444	0.417	0.843	0.219	23.667	0.581	0.839	0.166	28.444	0.705	0.810
MCHF	0.324	19.667	0.416	0.852	0.220	28.444	0.625	0.791	0.166	26.778	0.699	0.828
HF (max)	0.324	18.000	0.370	0.863	0.223	22.222	0.536	0.848	0.167	28.222	0.688	0.808
SS	0.345	19.444	0.303	0.829	0.258	15.444	0.262	0.869	0.219	17.111	0.291	0.856
LP	0.323	21.000	0.414	0.836	0.219	27.222	0.624	0.806	0.166	33.222	0.761	0.762
KSC	0.326	28.333	0.485	0.762	0.220	30.556	0.640	0.769	0.167	32.111	0.767	0.777
LCV	0.324	21.444	0.419	0.832	0.219	27.889	0.623	0.797	0.167	34.889	0.779	0.746
	n=300 p=500 SNR=1 Data=B				n=300 p=500 SNR=1.5 Data=B				n=300 p=500 SNR=2 Data=B			
Real	0.294	20.000	1.000	1.000	0.198	20.000	1.000	1.000	0.150	20.000	1.000	1.000
HM	0.326	42.111	0.301	0.924	0.225	51.556	0.451	0.911	0.178	60.556	0.554	0.896
HF	0.324	40.889	0.301	0.927	0.224	49.556	0.445	0.915	0.176	55.889	0.542	0.905
MCHM	0.327	56.444	0.330	0.896	0.224	42.778	0.342	0.925	0.175	48.667	0.490	0.918
MCHF	0.322	38.444	0.251	0.930	0.223	42.333	0.393	0.928	0.173	47.889	0.495	0.920
HF (max)	0.313	16.667	0.209	0.974	0.217	25.556	0.329	0.960	0.169	30.222	0.457	0.955
SS	0.355	13.778	0.087	0.975	0.260	16.333	0.121	0.971	0.230	16.889	0.096	0.969
LP	0.312	20.778	0.221	0.966	0.218	31.000	0.358	0.95	0.171	38.667	0.485	0.939
KSC	0.359	121.778	0.424	0.764	0.236	83.444	0.480	0.846	0.179	66.889	0.573	0.884
LCV	0.316	28.333	0.272	0.952	0.219	33.000	0.352	0.945	0.172	42.667	0.508	0.932
	n=300 p=1000 SNR=1 Data=B				n=300 p=1000 SNR=1.5 Data=B				n=300 p=1000 SNR=2 Data=B			
Real	0.343	20.000	1.000	1.0	0.231	20.000	1.000	1.000	0.175	20.000	1.000	1.000
HM	0.383	59.222	0.225	0.944	0.266	71.556	0.343	0.934	0.204	83.111	0.472	0.925
HF	0.382	58.111	0.219	0.945	0.266	71.444	0.343	0.934	0.204	80.556	0.472	0.927
MCHM	0.395	129.333	0.307	0.874	0.264	105.444	0.398	0.900	0.207	128.222	0.500	0.879
MCHF	0.381	64.222	0.206	0.939	0.267	107.222	0.380	0.898	0.207	105.889	0.478	0.902
HF (max)	0.366	14.444	0.112	0.988	0.255	23.444	0.231	0.981	0.199	27.778	0.286	0.977
SS	0.393	30.889	0.136	0.971	0.308	28.778	0.144	0.974	0.264	27.889	0.156	0.975
LP	0.361	22.889	0.135	0.979	0.253	30.111	0.242	0.974	0.198	36.444	0.309	0.969
KSC	0.424	176.0	0.308	0.827	0.265	84.111	0.321	0.921	0.204	75.000	0.473	0.933
LCV	0.364	26.222	0.147	0.976	0.254	35.667	0.258	0.969	0.199	50.333	0.371	0.956

Table 3.3: The mean value of MSE, the number of selected variables, the sensitivity and the specificity of each method when applied to dataset B. The reported values are the mean over 10 Monte Carlo repetitions.

	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec
	n=100 p=100 SNR=1 Data=C				n=100 p=100 SNR=1.5 Data=C				n=100 p=100 SNR=2 Data=C			
Real	0.478	20.000	1.000	1.000	0.330	20.000	1.000	1.000	0.252	20.000	1.000	1.000
HM	0.577	31.000	0.484	0.728	0.403	34.556	0.661	0.724	0.323	35.000	0.703	0.728
HF	0.590	29.333	0.469	0.745	0.403	34.111	0.654	0.727	0.320	37.444	0.723	0.702
MCHM	0.560	26.889	0.46	0.773	0.397	25.778	0.573	0.811	0.309	28.556	0.652	0.795
MCHF	0.547	23.111	0.426	0.812	0.400	24.556	0.555	0.823	0.307	23.111	0.624	0.855
HF (max)	0.550	16.556	0.351	0.876	0.403	23.556	0.553	0.834	0.315	28.556	0.664	0.798
SS	0.592	26.333	0.333	0.752	0.477	12.556	0.206	0.891	0.418	19.333	0.283	0.826
LP	0.547	19.667	0.396	0.847	0.396	29.222	0.625	0.781	0.317	32.667	0.683	0.752
KSC	0.692	76.778	0.793	0.238	0.468	73.222	0.823	0.288	0.378	70.333	0.848	0.330
LCV	0.572	24.667	0.420	0.791	0.401	30.444	0.611	0.762	0.323	38.778	0.731	0.688
	n=100 p=500 SNR=1 Data=C				n=100 p=500 SNR=1.5 Data=C				n=100 p=500 SNR=2 Data=C			
Real	0.572	20.000	1.000	1.000	0.392	20.000	1.000	1.000	0.307	20.000	1.000	1.000
HM	0.683	33.667	0.251	0.940	0.503	40.889	0.348	0.929	0.417	52.000	0.452	0.910
HF	0.679	35.333	0.281	0.938	0.501	36.667	0.337	0.937	0.416	47.778	0.439	0.918
MCHM	0.693	44.333	0.286	0.919	0.495	26.667	0.310	0.957	0.430	51.778	0.391	0.908
MCHF	0.684	34.778	0.234	0.937	0.493	33.778	0.344	0.944	0.420	31.111	0.345	0.949
HF (max)	0.647	14.444	0.195	0.978	0.486	19.111	0.269	0.971	0.407	19.667	0.310	0.972
SS	0.729	58.667	0.251	0.888	0.540	58.889	0.305	0.890	0.473	55.444	0.334	0.898
LP	0.656	16.778	0.218	0.974	0.488	22.333	0.286	0.965	0.399	31.333	0.383	0.950
KSC	0.759	94.222	0.367	0.819	0.543	91.000	0.436	0.828	0.440	89.778	0.516	0.834
LCV	0.661	21.000	0.230	0.966	0.486	24.556	0.292	0.961	0.411	33.444	0.388	0.946
	n=100 p=1000 SNR=1 Data=C				n=100 p=1000 SNR=1.5 Data=C				n=100 p=1000 SNR=2 Data=C			
Real	0.616	20.000	1.000	1.000	0.419	20.000	1.000	1.000	0.323	20.000	1.000	1.000
HM	0.771	56.000	0.163	0.946	0.585	58.222	0.275	0.946	0.485	62.444	0.375	0.944
HF	0.770	55.889	0.157	0.946	0.585	51.667	0.263	0.953	0.486	62.222	0.375	0.944
MCHM	0.792	127.667	0.318	0.876	0.576	79.778	0.285	0.924	0.494	73.000	0.330	0.932
MCHF	0.785	98.556	0.245	0.904	0.590	112.667	0.335	0.892	0.496	84.778	0.347	0.921
HF (max)	0.738	8.000	0.044	0.993	0.559	14.111	0.123	0.988	0.476	22.111	0.213	0.982
SS	0.788	32.222	0.078	0.969	0.630	58.778	0.164	0.943	0.549	54.333	0.203	0.949
LP	0.736	20.556	0.067	0.98	0.575	42.444	0.218	0.961	0.477	41.222	0.280	0.964
KSC	0.813	95.556	0.207	0.907	0.600	92.778	0.320	0.912	0.494	93.556	0.392	0.912
LCV	0.748	26.556	0.083	0.975	0.56	42.667	0.218	0.961	0.470	48.889	0.342	0.957
	n=300 p=100 SNR=1 Data=C				n=300 p=100 SNR=1.5 Data=C				n=300 p=100 SNR=2 Data=C			
Real	0.303	20.000	1.000	1.000	0.204	20.000	1.000	1.000	0.154	20.000	1.000	1.000
HM	0.334	39.778	0.714	0.671	0.223	45.444	0.865	0.635	0.170	52.000	0.933	0.570
HF	0.326	34.000	0.706	0.739	0.220	40.667	0.859	0.692	0.167	42.778	0.897	0.674
MCHM	0.326	25.222	0.611	0.826	0.218	30.667	0.793	0.799	0.164	27.778	0.837	0.844
MCHF	0.325	27.667	0.643	0.803	0.218	25.667	0.790	0.859	0.165	29.556	0.846	0.824
HF (max)	0.325	29.111	0.667	0.791	0.220	38.222	0.853	0.720	0.167	39.556	0.884	0.711
SS	0.337	18.444	0.427	0.869	0.271	14.778	0.333	0.894	0.226	21.778	0.434	0.831
LP	0.326	32.556	0.693	0.754	0.221	40.889	0.853	0.688	0.168	43.778	0.903	0.664
KSC	0.336	61.889	0.872	0.437	0.226	59.889	0.939	0.475	0.171	54.000	0.946	0.548
LCV	0.330	37.556	0.711	0.697	0.222	43.889	0.865	0.654	0.169	47.000	0.909	0.625
	n=300 p=500 SNR=1 Data=C				n=300 p=500 SNR=1.5 Data=C				n=300 p=500 SNR=2 Data=C			
Real	0.308	20.000	1.000	1.000	0.208	20.000	1.000	1.000	0.158	20.000	1.000	1.000
HM	0.354	44.556	0.482	0.927	0.254	72.778	0.733	0.878	0.196	82.333	0.795	0.861
HF	0.351	42.667	0.471	0.930	0.252	68.556	0.727	0.887	0.195	77.111	0.784	0.871
MCHM	0.359	42.111	0.425	0.930	0.239	42.444	0.615	0.936	0.186	42.111	0.699	0.941
MCHF	0.346	32.000	0.408	0.950	0.243	48.111	0.632	0.925	0.186	48.000	0.722	0.929
HF (max)	0.346	30.000	0.414	0.954	0.240	41.667	0.655	0.940	0.188	54.889	0.750	0.916
SS	0.383	45.222	0.288	0.918	0.296	16.556	0.194	0.973	0.261	15.222	0.200	0.976
LP	0.348	34.222	0.437	0.946	0.242	50.778	0.677	0.922	0.189	60.444	0.762	0.905
KSC	0.444	218.000	0.749	0.577	0.297	187.222	0.811	0.643	0.223	165.111	0.846	0.691
LCV	0.348	37.333	0.464	0.941	0.241	48.000	0.677	0.927	0.190	63.000	0.767	0.900
	n=300 p=1000 SNR=1 Data=C				n=300 p=1000 SNR=1.5 Data=C				n=300 p=1000 SNR=2 Data=C			
Real	0.305	20.000	1.000	1.000	0.205	20.000	1.000	1.000	0.154	20.000	1.000	1.000
HM	0.378	70.667	0.522	0.939	0.262	82.556	0.628	0.929	0.205	107.000	0.739	0.906
HF	0.375	68.556	0.528	0.941	0.261	79.444	0.633	0.932	0.204	104.222	0.733	0.909
MCHM	0.364	50.889	0.494	0.958	0.257	62.667	0.589	0.948	0.194	57.444	0.650	0.955
MCHF	0.362	48.667	0.478	0.960	0.252	57.111	0.578	0.954	0.198	69.000	0.661	0.943
HF (max)	0.359	45.667	0.483	0.963	0.248	56.333	0.622	0.955	0.195	61.444	0.656	0.951
SS	0.411	117.333	0.383	0.888	0.310	80.667	0.356	0.925	0.251	41.889	0.300	0.963
LP	0.361	54.667	0.500	0.954	0.254	68.000	0.622	0.943	0.197	71.000	0.678	0.941
KSC	0.424	243.778	0.639	0.764	0.286	224.667	0.744	0.786	0.220	205.333	0.800	0.807
LCV	0.369	66.556	0.511	0.943	0.254	79.333	0.644	0.932	0.201	87.667	0.700	0.925

Table 3.4: The mean value of MSE, the number of selected variables, the sensitivity and the specificity of each method when applied to dataset C. The reported values are the mean over 10 Monte Carlo repetitions.

	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec
	n=100 p=100 SNR=1 Data=D				n=100 p=100 SNR=1.5 Data=D				n=100 p=100 SNR=2 Data=D			
Real	0.558	20.000	1.000	1.000	0.385	20.000	1.000	1.000	0.296	20.000	1.000	1.000
HM	0.682	38.778	0.610	0.660	0.440	37.444	0.718	0.699	0.361	41.000	0.794	0.673
HF	0.628	26.111	0.522	0.796	0.435	31.222	0.693	0.770	0.346	33.556	0.756	0.755
MCHM	0.613	19.444	0.485	0.869	0.428	20.333	0.600	0.882	0.335	22.333	0.668	0.873
MCHF	0.628	18.667	0.466	0.874	0.425	20.889	0.604	0.877	0.331	21.111	0.687	0.892
HF (max)	0.606	18.889	0.473	0.873	0.425	23.333	0.625	0.851	0.339	29.333	0.731	0.801
SS	0.685	31.667	0.448	0.712	0.494	28.889	0.517	0.760	0.408	26.778	0.549	0.793
LP	0.613	20.444	0.485	0.857	0.433	25.889	0.649	0.826	0.342	34.556	0.744	0.740
KSC	0.831	76.778	0.851	0.25	0.554	74.556	0.839	0.274	0.417	71.778	0.880	0.318
LCV	0.657	27.111	0.521	0.783	0.471	32.889	0.687	0.748	0.374	39.889	0.780	0.683
	n=100 p=500 SNR=1 Data=D				n=100 p=500 SNR=1.5 Data=D				n=100 p=500 SNR=2 Data=D			
Real	0.600	20.000	1.000	1.000	0.406	20.000	1.000	1.000	0.313	20.00	1.000	1.000
HM	0.792	54.111	0.368	0.902	0.583	69.333	0.576	0.879	0.495	79.222	0.685	0.863
HF	0.784	52.333	0.357	0.906	0.584	62.556	0.570	0.893	0.485	70.778	0.696	0.881
MCHM	0.833	56.333	0.407	0.899	0.587	50.556	0.515	0.916	0.477	48.778	0.594	0.923
MCHF	0.796	42.222	0.357	0.927	0.586	49.556	0.497	0.917	0.468	41.444	0.576	0.937
HF (max)	0.766	14.111	0.181	0.978	0.563	23.111	0.389	0.968	0.472	34.778	0.554	0.950
SS	0.821	79.778	0.283	0.845	0.641	122.333	0.448	0.764	0.592	89.222	0.355	0.829
LP	0.781	18.444	0.163	0.968	0.564	51.222	0.543	0.916	0.468	49.778	0.634	0.922
KSC	0.889	95.556	0.48	0.821	0.643	94.556	0.554	0.826	0.500	93.889	0.667	0.832
LCV	0.776	19.333	0.198	0.968	0.577	41.222	0.481	0.934	0.475	54.444	0.638	0.913
	n=100 p=1000 SNR=1 Data=D				n=100 p=1000 SNR=1.5 Data=D				n=100 p=1000 SNR=2 Data=D			
Real	0.606	20.000	1.000	1.000	0.415	20.000	1.000	1.000	0.320	20.000	1.000	1.000
HM	0.806	64.000	0.339	0.942	0.614	64.556	0.492	0.944	0.506	72.333	0.571	0.938
HF	0.796	56.556	0.339	0.949	0.603	70.222	0.526	0.939	0.510	66.222	0.560	0.944
MCHM	0.838	158.111	0.487	0.848	0.643	123.889	0.515	0.884	0.515	106.000	0.566	0.903
MCHF	0.814	138.444	0.451	0.868	0.618	83.889	0.493	0.924	0.507	79.889	0.525	0.929
HF (max)	0.768	19.000	0.192	0.984	0.593	24.778	0.322	0.981	0.491	36.222	0.492	0.973
SS	0.784	58.333	0.192	0.944	0.647	62.444	0.249	0.941	0.583	71.444	0.255	0.932
LP	0.769	39.000	0.255	0.965	0.591	49.333	0.407	0.958	0.500	66.778	0.571	0.943
KSC	0.824	96.333	0.356	0.909	0.608	96.556	0.497	0.911	0.510	96.778	0.577	0.913
LCV	0.780	35.222	0.244	0.969	0.598	55.222	0.469	0.953	0.497	57.667	0.549	0.952
	n=300 p=100 SNR=1 Data=D				n=300 p=100 SNR=1.5 Data=D				n=300 p=100 SNR=2 Data=D			
Real	0.297	20.000	1.000	1.000	0.200	20.000	1.000	1.000	0.152	20.000	1.000	1.000
HM	0.315	33.111	0.665	0.744	0.216	45.333	0.849	0.635	0.163	45.333	0.897	0.646
HF	0.313	30.444	0.682	0.780	0.214	40.889	0.848	0.690	0.163	42.556	0.867	0.673
MCHM	0.308	25.222	0.661	0.840	0.207	20.000	0.741	0.921	0.159	28.444	0.818	0.835
MCHF	0.311	19.333	0.612	0.901	0.208	22.667	0.758	0.893	0.159	26.000	0.825	0.868
HF (max)	0.313	27.000	0.645	0.814	0.211	36.556	0.848	0.743	0.161	37.444	0.849	0.732
SS	0.352	29.111	0.482	0.752	0.270	30.667	0.488	0.736	0.218	30.333	0.534	0.750
LP	0.312	28.667	0.663	0.798	0.212	38.556	0.848	0.719	0.162	39.111	0.861	0.715
KSC	0.327	52.778	0.798	0.533	0.218	54.222	0.89	0.535	0.166	55.667	0.921	0.525
LCV	0.315	34.444	0.705	0.736	0.214	42.556	0.86	0.672	0.163	44.444	0.891	0.656
	n=300 p=500 SNR=1 Data=D				n=300 p=500 SNR=1.5 Data=D				n=300 p=500 SNR=2 Data=D			
Real	0.351	20.000	1.000	1.000	0.234	20.000	1.000	1.000	0.176	20.000	1.000	1.000
HM	0.422	61.778	0.694	0.900	0.291	77.111	0.819	0.873	0.226	86.778	0.881	0.855
HF	0.419	57.000	0.683	0.909	0.286	67.000	0.796	0.893	0.219	73.778	0.864	0.882
MCHM	0.409	40.667	0.620	0.941	0.276	39.444	0.729	0.948	0.208	47.333	0.836	0.935
MCHF	0.409	37.556	0.615	0.947	0.276	40.444	0.734	0.946	0.209	46.556	0.830	0.937
HF (max)	0.408	42.556	0.660	0.938	0.279	53.889	0.785	0.920	0.213	59.889	0.853	0.910
SS	0.480	93.556	0.399	0.822	0.380	77.556	0.316	0.851	0.330	69.000	0.328	0.870
LP	0.413	46.333	0.666	0.931	0.280	57.000	0.785	0.913	0.216	66.778	0.859	0.896
KSC	0.512	189.000	0.789	0.639	0.346	194.111	0.874	0.632	0.261	186.000	0.910	0.650
LCV	0.415	51.444	0.677	0.920	0.284	64.444	0.802	0.898	0.216	70.333	0.870	0.889
	n=300 p=1000 SNR=1 Data=D				n=300 p=1000 SNR=1.5 Data=D				n=300 p=1000 SNR=2 Data=D			
Real	0.337	20.000	1.000	1.000	0.226	20.000	1.000	1.000	0.171	20.000	1.000	1.000
HM	0.418	92.889	0.715	0.92	0.288	108.444	0.865	0.907	0.219	107.667	0.887	0.908
HF	0.416	84.667	0.709	0.928	0.282	93.556	0.843	0.922	0.216	97.111	0.887	0.919
MCHM	0.405	64.444	0.658	0.948	0.269	56.111	0.754	0.958	0.202	57.111	0.837	0.959
MCHF	0.404	66.000	0.676	0.946	0.271	65.889	0.793	0.949	0.200	57.000	0.854	0.959
HF (max)	0.396	53.333	0.642	0.959	0.269	63.556	0.815	0.952	0.207	70.444	0.871	0.946
SS	0.503	187.444	0.391	0.817	0.357	77.111	0.381	0.929	0.300	135.667	0.545	0.873
LP	0.397	57.444	0.659	0.955	0.274	74.333	0.826	0.941	0.209	78.333	0.871	0.938
KSC	0.489	247.333	0.749	0.763	0.334	255.333	0.877	0.757	0.256	243.222	0.916	0.770
LCV	0.406	69.444	0.681	0.943	0.276	80.889	0.832	0.934	0.211	87.444	0.876	0.929

Table 3.5: The mean value of MSE, the number of selected variables, the sensitivity and the specificity of each method when applied to dataset D. The reported values are the mean over 10 Monte Carlo repetitions.

3.5.4 Virtual Metrology

As a final case study in this section the methods proposed in Section 3.3 are applied to a Virtual Metrology (VM) problem. VM aims to predict metrology values using sensor data from production equipment and physical metrology values of preceding samples. VM is a promising technology for the semiconductor manufacturing industry as it can reduce the frequency of in-line metrology operations and provide supportive information for other operations such as fault detection, predictive maintenance and run-to-run control. The J2M case study (Dataset 2.4.2) is used as an example. 800 samples are used as a training set and the remaining samples as a test set to compute the prediction error. Noise variables are eliminated by selecting only the 5000 wavelength statistics with the highest variation over the training samples. The data is then scaled to zero mean and unit variance. The models are evaluated according to their NMSE and the number of variables selected.

The prediction error and the number of variables used by each algorithm are reported in Table 3.6. The prediction error varies between 19.4% and 26.7%. The best performance is obtained with the Ridge model when all the variables are considered (19.4%). Among the sparse models SS and KSC have the worst performances (100% and 26.7%). SS does not select any variable and KSC only 19, which is not sufficient to obtain a good model. The data is highly correlated; this may be the reason for the low performance of SS and KSC which focus on selecting a stable set of variables. The other methods have all roughly the same prediction performances ($\approx$ 22% error). HM and HF select a larger number of variables than MCHM and MCHF. This is probably a consequence of the high correlation between the variables as in Monte Carlo based methods the score of groups of correlated variables is penalized and they are less likely to be selected. Optimal performance is obtained with HF(max) which selects the smallest number of variables among the algorithms and performs well in terms of prediction error. While the lasso seems to perform well both in terms of prediction error and model complexity, its main weakness is its instability. In this study LCV was repeated 100 times with random folds. Figure 3.7 shows how often each variable is selected and Figure 3.8 shows the frequency of the model complexity over the 100 repetitions. It is clear that the instability in the model leads also to an instability in model performance as shown in Figure 3.9.

	NMSE (Prediction)	NMSE (Training)	Number of Variables
Ridge	0.194	0.082	5000
LCV	0.225	0.125	192
SS	1.0	1.0	0
LP	0.221	0.13	133
KSC	0.267	0.22	19
HM	0.221	0.112	225
HF	0.221	0.115	206
MCHM	0.221	0.135	131
MCHF	0.225	0.13	164
HF(max)	0.222	0.134	110

Table 3.6: Prediction error, training error and number of variables selected by each algorithm for the problem described in Section 3.5.4

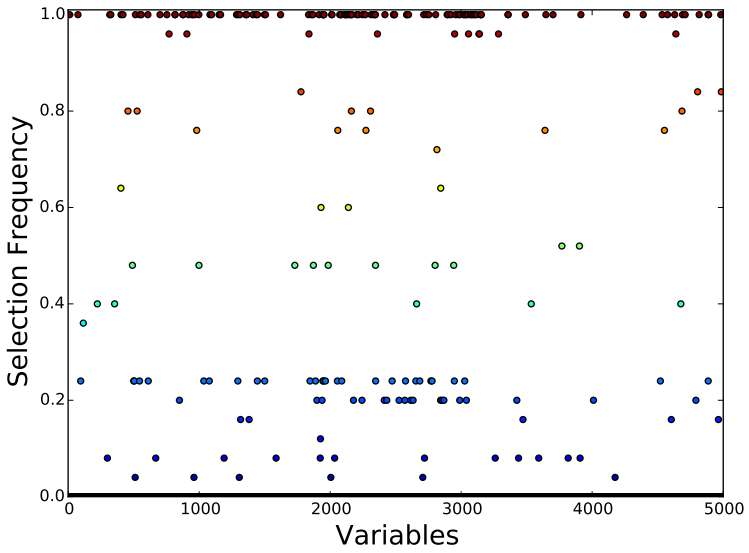


Figure 3.7: The frequency with which each variable is selected over 100 repetitions of LCV for the dataset described in Section 3.5.4

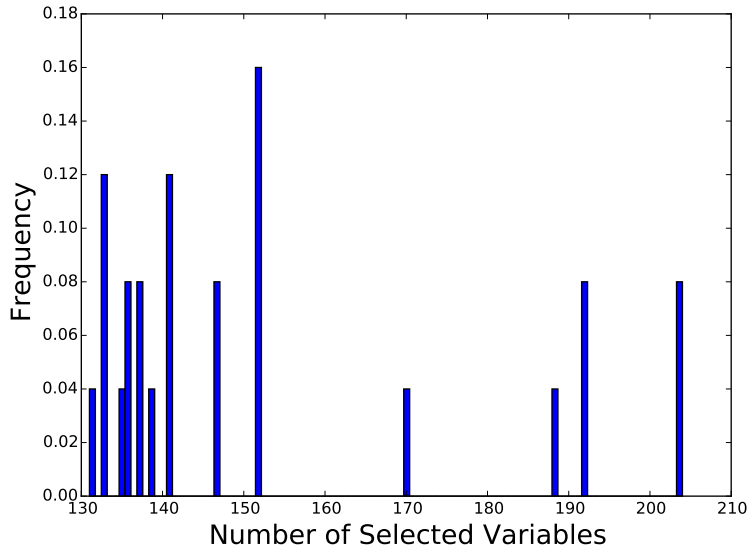


Figure 3.8: The frequency with which a given number of variables is selected over 100 repetitions of LCV for the dataset described in Section 3.5.4

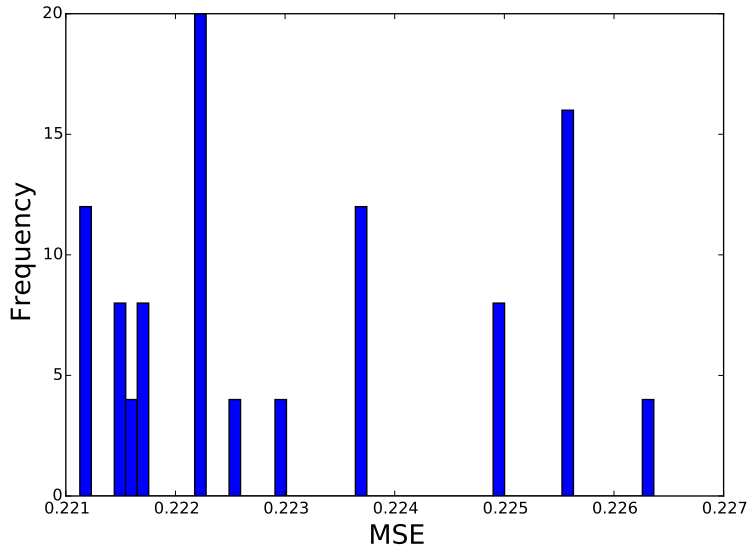


Figure 3.9: The NMSE obtained on the test set with the different models obtained with LCV for 100 repetitions. The case study is described in Section 3.5.4

3.6 Computational Time Evaluation

In this section the computational complexity of the various algorithms is evaluated. Define $C(n, p)$ as the cost required to compute the lasso estimator on a dataset $\mathbf{X} \in \mathbb{R}^{n \times p}$ and $\mathbf{y} \in \mathbb{R}^n$ using a fixed penalty value. This value can change according to the algorithm that is used to compute the lasso estimator. We define l the length of the penalty path $(\alpha_1, \ldots, \alpha_l)$. We assume that the cost to perform lasso is independent of the penalty value α. This does not affect our study because all the methods use the same penalty path and so the speed comparison will still be fair. We consider simple operations like finding the maximum in a vector or extracting two random subsamples of constant complexity. This is because their computational complexity is much lower than $C(n, p)$. Consider the Cross-Validation function defined in equation 3.5; then the cost of evaluating the Cross-Validation function is $C(n\frac{K-1}{K}, p) \times l \times K$. From this it follows that the complexity of the algorithms where the Cross-Validation function is computed r times (HF, LP, HM, HMP) is

$$cost_1 = r \times C(n\frac{K-1}{K}, p) \times l \times K. \tag{3.33}$$

On the other hand for the algorithms based on bootstrapping (KSC, SS), where for each value of the penalty path the lasso is computed B times, the complexity is

$$cost_2 = C(\frac{n}{2}, p) \times l \times B. \tag{3.34}$$

Under the assumption that $\frac{K-1}{K} > \frac{1}{2}$. We have that:

$$\frac{cost_1}{cost_2} = \frac{r \times C(n\frac{K-1}{K}, p) \times l \times K}{C(\frac{n}{2}, p) \times l \times B} > \frac{r \times C(\frac{n}{2}, p) \times K}{C(\frac{n}{2}, p) \times B} = \frac{r \times K}{B} \tag{3.35}$$

If we assume that both B and r are two large numbers of similar order we obtain that

$$cost_1 > cost_2 \frac{r \times K}{B} \approx cost_2 \times K \tag{3.36}$$

and hence $cost_1$ is K times larger than $cost_2$. The complexity of LCV is $cost_1$ when $r = 1$ and is therefore the fastest among all the methods. On the other hand we observe that all methods can be easily parallelized and the computational cost when using P processors became respectively $\frac{cost_1}{P}$

and $\frac{cost_2}{P}$. In particular in the extreme case when $P = r$ or $P = B$ the computational speed would be similar the one required to compute LCV.
The Monte Carlo methods with splitting parameter q and with r repetitions MCHF and MCHM have as complexity $cost_3 = C(q \times n, p) \times l \times r$ and so have similar complexity to the bootstrap based methods. To conclude this paragraph we observe that in stability selection the weakness parameter u has to be selected at the beginning of the algorithm and so the algorithm has to be restarted if we want to try a different weakness value. This can drastically increase the computational time required if a search of the optimal weakness parameter u has to be performed.

3.7 Conclusion

In this chapter the robustness of the lasso algorithm was investigated. Particular focus was given to datasets composed of more variables than samples ($p > n$). This is a common situation in semiconductor manufacturing as for example was observed in the case study considered (Dataset 2.4.2), where each sample is a wafer and thus n is generally small. While it is in general impossible to have a sparse and robust estimator, some algorithms that allow estimation of a set of variables that are consistent with the true model more than those obtained with lasso are proposed. The proposed algorithms compare favourably with competing approaches in the literature. In addition, it is demonstrated through the use of a simulation study that Cross-Validation based hyper-parameters selection is not suitable as the focus on prediction accuracy leads to many superfluous variables being included. In this sense the HF(max) algorithm is proposed as an enhancement of lasso that performs well both in real and simulated datasets.

Chapter 4

Linear Unsupervised Feature Selection

4.1 Introduction

The need to analyse large volumes of multivariate data is an increasingly common occurrence in many areas of science, engineering and business. In order to build more interpretable models, or to reduce the cost of data collection, it is important to discover good compact representations of high-dimensional datasets. This leads to the fundamental problem of dimensionality reduction. Many methods have been developed to perform supervised dimensionality reduction, for example the lasso [63] (discussed in Chapter 3) or [92], [93], [94]. Given an input matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ (containing n measurements of p variables) and an output value $\mathbf{y} \in \mathbb{R}^n$ these methods try to understand what subset of variables, or derived features of $\mathbf{X}$ optimally explain $\mathbf{y}$. Dimensionality reduction can also be defined as an unsupervised problem. In this case we look for the subset of variables/derived features that retain the maximum information content with respect to the original set of variables, in the sense of being able to reconstruct the full data matrix $\mathbf{X}$. Different unsupervised dimensionality reduction techniques have been proposed. Some of them, such as Unsupervised Feature Selection Using Feature Similarity[95], FSSEM [96], Spectral Feature Selection [97] and the techniques proposed in [98], have been developed with the goal of maximising performance when used as a pre-processing step in clustering or classification algorithms, while others, such as [99], [100], [101], [90], have been developed in order to obtain the optimal reconstruction of the full dataset. Among this latter group Principal Component Analysis (PCA, [101]) is the best known and most widely used technique. PCA provides the most efficient linear transformation of data to a

lower dimensional space and is relatively straight forward to compute. It has found many applications in chemometrics and other fields where datasets are encountered involving large numbers of variables with significant levels of inter-variable correlation and hence redundancy (see for example [102] and [103]). However, a weakness of PCA is that the resulting latent variables (principal components) are a linear combination of all original variables making it difficult to identify the significant variables within the data [104]. This is especially true in the case of highly correlated datasets due to the grouping effect, whereby the contribution of a group of highly correlated variables to a given principal component is distributed evenly across all variables in the group. While this characteristic is beneficial in terms of noise suppression, it means that the contribution of individual variables can be small making important variables difficult to identify. Hence, tasks such as identification of key variables, root-cause analysis and model interpretation can be challenging using PCA. Consequently, various approaches have been developed to obtain sparse approximations of PCA. The simplest strategy is to manually set to 0 the values of the principal components (PCs) that are smaller than a given threshold but this can lead to significant variables being missed if they are part of a group of highly correlated variables [105], [104]. More sophisticated approaches such as SCoTLASS [106], DSPCA [107], sparse PCA [108], SSPCA [109], sPCA-rSVD [110] and SOCA [111] use a lasso like L_1 or L_0 penalty or are formulated as constrained maximization problems in order to encourage sparsity in the PCA loadings. However, these methods are generally computationally intensive and difficult to use and interpret due to the need to establish the appropriate level of sparsity for each PC computed. These challenges motivated the development of various techniques with the aim of identifying a small number of key variables that are representative of the observed variance across all variables.

The particular focus of this chapter is on unsupervised feature selection using the Forward Selection Components Analysis algorithm (FSCA). This was initially introduced in the context of Optical Emission Spectroscopy (OES) data analysis of plasma etch processes [112] where isolating a small number of wavelengths is important for understanding the underlying plasma chemistry. More recently, FSCA has been found to be a particularly effective tool for optimising measurement site selection for spatial wafer metrology in semiconductor manufacturing [89]. The method works by iteratively deriving a set of orthogonal components which are a function of only a subset of the original variables, and which sequentially maximize the explained variance. At one level FSCA can be regarded as the unsupervised counterpart of Forward Selection Regression in that it returns a set of Forward Selected

Variables (FSVs), but equally it retains some of the characteristics and utility of PCA in that it also returns a set of Forward Selection Components (FSCs) which form an orthogonal basis. This allows, for example, the contributions of individual components to be easily isolated.

In this chapter, we present, a complete framework for unsupervised dimensionality reduction showing that unsupervised feature selection algorithms share a similar structure to PCA. We show that several algorithms are based on principles similar to those of FSCA. FSCA is then used as an example methodology. It is described and efficient algorithm implementations developed. In addition, a number of backward refinement enhancements to FSCA are proposed and evaluated.

4.2 Background

A variety of variable selection methods have been developed based on making comparisons with or extracting information from a PCA decomposition of the data matrix (e.g. [113], [114], [115] and [116]). Other approaches employ clustering of features using a suitable feature similarity metric as the basis for variable selection (e.g. [95], [97] and [90]). Recently [117] proposed a novel L_1 regularised formulation for the unsupervised variable selection problem which has a similar philosophy to sparse PCA and can be thought of as the unsupervised counterpart of LASSO [63]. In addition to FSCA, two other techniques which can be considered as performing direct variable selection are the algorithms by Whitley et al. [118] and Wei and Billings [119], both of which employ orthogonalisation procedures. In the former, variables are selected based on sequentially finding the variables in the dataset that are most uncorrelated with linear combinations of the variables already selected, while in the latter the criterion used for variable selection is the maximum average squared correlation with all other variables in the dataset. Wei and Billings's algorithm, which they refer to as Forward Orthogonal Search (FOS), is similar in character to FSCA and, as will be discussed in Section 4.6, yields identical results to FSCA if the data is appropriately pre-scaled. Recently [120] introduced a kernel extension of variable selection that enables non-linear relationships between variables to be taken in account, while [121] developed an efficient parallel implementation for data parallel distributed computing that scales well for large problems. Both these algorithms are equivalent to FSCA in terms of the sequence of variables selected, but operate directly in the variable space rather than producing FSCs.

4.3 Data Decomposition and Reconstruction

Given a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, $\mathbf{S} \in \mathbb{R}^{n \times k}$ is said to be a good lower dimensional representation of $\mathbf{X}$ if $k << p$ and if it is possible to obtain a good reconstruction of $\mathbf{X}$ from $\mathbf{S}$. In other words:

$$\exists \Phi : \mathbb{R}^{n \times k} \longrightarrow \mathbb{R}^{n \times p} \tag{4.1}$$

such that:

$$\hat{\mathbf{X}} = \Phi(\mathbf{S}) \quad and \quad \| \mathbf{X} - \hat{\mathbf{X}} \|_\gamma \quad is\ small. \tag{4.2}$$

The idea of good reconstruction expressed in the previous equation is still vague as it may depend on the application or by the desired results. Indeed different metrics can be used to quantify the approximation error between the matrix $\mathbf{X}$ and its reconstruction $\hat{\mathbf{X}}$ (such as element-wise or induced L_1, L_2 and L_∞ norms of $\hat{\mathbf{X}} - \mathbf{X}$). The metric that is often considered when working with Principal Component Analysis, and the one adopted here, is the percentage of explained variance, that is, the percentage of the variance observed in $\mathbf{X}$ explained by $\hat{\mathbf{X}}$. Assuming, without lose of generality, that the columns of $\mathbf{X}$ have zero mean, this can be expressed in terms of the Frobenius norm ($\| \cdot \|_F$) as

$$V_{\mathbf{X}}(\hat{\mathbf{X}}) = 100 \times \left(1 - \frac{\| \hat{\mathbf{X}} - \mathbf{X} \|_F^2}{\| \mathbf{X} \|_F^2} \right). \tag{4.3}$$

Observation 4.3.1. *The value $V_{\mathbf{X}}(\hat{\mathbf{X}})$ is inversely proportional to the approximation error*

$$Err_{\mathbf{X}}(\hat{\mathbf{X}}) = \| \hat{\mathbf{X}} - \mathbf{X} \|_F^2 \tag{4.4}$$

Maximising $V_{\mathbf{X}}(\hat{\mathbf{X}})$ is therefore equivalent to minimising $Err_{\mathbf{X}}(\hat{\mathbf{X}})$. Observe in particular that:

$$\frac{Err_{\mathbf{X}}(\hat{\mathbf{X}})}{np} = MSE(\hat{\mathbf{X}}, \mathbf{X}) \tag{4.5}$$

Maximising the explained variance is therefore equivalent to minimise the mean square error (MSE) between the original matrix $\mathbf{X}$ and its approximation $\hat{\mathbf{X}}$.

In chapter 6 (Section 6.6) some limitations of $V_{\mathbf{X}}$ as a metric are shown and possible alternatives are investigated. It follows that the choice of the norm is strongly related to the field of application.

4.3.1 Linear Dimensionality Reduction

The simplest version of the dimensionality reduction problem is when the function Φ is linear. Given a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ representing a dataset with n measurements of p variables, the aim is to estimate a matrix $\mathbf{S} \in \mathbb{R}^{n \times k}$, where $k < rank(\mathbf{X}) \leq min(n, p)$, such that it is possible to obtain a good reconstruction of $\mathbf{X}$ by linear regression on $\mathbf{S}$:

$$\hat{\mathbf{X}} = \mathbf{S}\boldsymbol{\Theta} \text{ where } \mathbf{S} \in \mathbb{R}^{n \times k} \text{ and } \boldsymbol{\Theta} \in \mathbb{R}^{k \times p} \tag{4.6}$$

In general, given a regressor matrix $\mathbf{S}$, the optimal least square error linear reconstruction of the original signal $\mathbf{X}$ is given by

$$\boldsymbol{\Theta} = \underset{\tilde{\boldsymbol{\Theta}} \in \mathbb{R}^{k \times p}}{argmin} \| \mathbf{S}\tilde{\boldsymbol{\Theta}} - \mathbf{X} \|_F^2, \tag{4.7}$$

where $(\| \cdot \|_F)$ is the Frobenius norm. The solution to (4.7) is the well known least-squares solution:

$$\boldsymbol{\Theta} = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T\mathbf{X}, \quad \boldsymbol{\Theta} \in \mathbb{R}^{k \times p} \tag{4.8}$$

Hence, defining the projection matrix

$$\boldsymbol{\Phi}(\mathbf{S}) = \mathbf{S}(\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T, \quad \boldsymbol{\Phi}(\mathbf{S}) \in \mathbb{R}^{n \times n} \tag{4.9}$$

for a given matrix $\mathbf{S}$, the optimal linear reconstruction of $\mathbf{X}$ can be expressed as:

$$\hat{\mathbf{X}} = \boldsymbol{\Phi}(\mathbf{S})\mathbf{X}, \quad \hat{\mathbf{X}} \in \mathbb{R}^{n \times p} \tag{4.10}$$

Observation 4.3.2. *While $V_{\mathbf{X}}(\hat{\mathbf{X}})$ (defined in equation 4.3) is unbounded in the negative direction for arbitrary $\hat{\mathbf{X}}$, when $\hat{\mathbf{X}}$ is computed as a projection of $\mathbf{X}$ onto the subspace spanned by $\mathbf{S}$, as given by eqt. (4.10), $V_{\mathbf{X}}(\hat{\mathbf{X}}) \geq 0$ for arbitrary $\mathbf{S}$. (A proof is provided in Appendix B.)*

4.3.2 PCA

Principal component analysis (PCA) is probably the most popular unsupervised data reduction technique. PCA extracts explanatory or latent variables from a dataset using a matrix decomposition. The earliest descriptions of PCA-like algorithms were given by Pearson [122] in 1901 and Hotelling [123] in 1933. Both papers adopted different approaches. Pearson concentrated on finding lines and planes that best fit a set of points in p-dimensional space while Hotelling's motivation was to find a "fundamental set of independent variables".

4.3.2.1 Geometric Interpretation of PCA

A sample in a dataset is a row vector and its i^{th} element contains the measurement of the i^{th} variable. It follows that the matrix storing the data is represented according to a canonical base

$$\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_p\} \tag{4.11}$$

where $\mathbf{e}_i \in \mathbb{R}^p$ has all its elements equal 0 except the i^{th} element, which is equal to 1. The same matrix can be described using a new geometrical base. Under the assumption that $rank(\mathbf{X}) = p$ the covariance matrix

$$\mathbf{\Sigma} = \frac{\mathbf{X}^T\mathbf{X}}{n-1} \in \mathbb{R}^{p\times p} \tag{4.12}$$

is symmetric and positive definite. It is then possible to extract p eigenvectors $\{\vec{\mathbf{p}}_i\}_{i=1\ldots,p}$ that are used as a new base for the space, that is:

$$\mathcal{E}_{PCA} = \{\mathbf{p}_1, \ldots, \mathbf{p}_p\}, \ \mathbf{p}_i \in \mathbb{R}^p \ for \ i = 1, \ldots, p \tag{4.13}$$

where

$$\mathbf{\Sigma}\mathbf{p}_i = \lambda_i\mathbf{p}_i \tag{4.14}$$

and

$$\| \ \mathbf{p}_i \ \|_2 = 1 \quad \forall i \tag{4.15}$$

Since all the eigenvalues of $\mathbf{\Sigma}$ are positive they can be expressed as:

$$\lambda_i = \sigma_i^2 \ with \ \sigma_i \in \mathbb{R} \tag{4.16}$$

The PCA eigenvectors are called loadings and they are usually assumed to be sorted according to their eigenvalue i.e.

$$\sigma_{i+1}^2 \leq \sigma_i^2 \quad \forall i \tag{4.17}$$

4.3.2.2 PCA Decomposition

Two matrices $\mathbf{P} \in \mathbb{R}^{p\times p}$ and $\mathbf{T} \in \mathbb{R}^{n\times p}$ are particularly important in the PCA decomposition. $\mathbf{P}$ is the matrix whose columns are the p loadings i.e.

$$\mathbf{P} = (\mathbf{p}_1, \ldots, \mathbf{p}_p) \in \mathbb{R}^{p\times p} \tag{4.18}$$

The matrix $\mathbf{T} \in \mathbb{R}^{n\times p}$ is called the scores matrix. It is the representation of the data $\mathbf{X}$ when the columns of $\mathbf{P}$ are used to describe the space or

equivalently the projection of $\mathbf{X}$ onto the columns of $\mathbf{P}$. It follows that the two matrices are related by the change of base equation:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T \tag{4.19}$$

The columns of $\mathbf{P}$ are the eigenvectors of the matrix $\mathbf{\Sigma}$. $\mathbf{P}$ is then an orthonormal matrix:

$$\mathbf{P}^T\mathbf{P} = \mathbf{I} \tag{4.20}$$

From the two previous equations it follows that

$$\mathbf{T} = \mathbf{X}\mathbf{P} \tag{4.21}$$

Equations 4.19 and 4.21 are the two fundamental PCA equations. So far the score matrix $\mathbf{T}$ has still the same dimensionality of the original data $\mathbf{X}$. In the next section is shown how to use PCA for dimensionality reduction.

4.3.2.3 Dimensionality Reduction with PCA

The PCA decomposition of the data can be used for dimensionality reduction. The loadings $\{\mathbf{p}_i\}_{i=1\ldots,p}$ are assumed ranked according to their associated eigenvalue as in equation 4.17. A lower dimensional version of $\mathbf{P}$ is obtained using only the first k loadings i.e.

$$\mathbf{P}_k = (\mathbf{p}_1, \ldots, \mathbf{p}_k) \in \mathbb{R}^{n\times k} \tag{4.22}$$

The PCA decomposition of $\mathbf{X}$ defined in equation 4.19 becomes

$$\hat{\mathbf{X}}^k_{PCA} = \mathbf{T}_k\mathbf{P}_k^T = \sum_{i=1}^{k} \mathbf{t}_i\mathbf{p}_i^T, \tag{4.23}$$

where $\mathbf{P}_k \in \mathbb{R}^{p\times k}$ is then computed as the first k ordered eigenvectors of the data covariance matrix $\mathbf{X}^T\mathbf{X}$ (in descending eigenvalue order), and $\mathbf{T}_k \in \mathbb{R}^{n\times k}$ is the geometrical projection of $\mathbf{X}$ on the columns of $\mathbf{P}_k$, that is:

$$\mathbf{T}_k = \mathbf{X}\mathbf{P}_k. \tag{4.24}$$

Here, $\mathbf{p}_i$ and $\mathbf{t}_i$ are the i-th column's of $\mathbf{P}_k$ and $\mathbf{T}_k$, respectively. If $k = r = rank(\mathbf{X})$ then the PCA decomposition is exact and

$$\mathbf{X} = \hat{\mathbf{X}}^r_{PCA} = \mathbf{T}_r\mathbf{P}_r^T. \tag{4.25}$$

Otherwise, if $k < rank(\mathbf{X})$ PCA provides the best rank k approximation to $\mathbf{X}$ [101], that is, $\mathbf{P}_k$ is the solution to the optimisation problem

$$\underset{\mathbf{P}_k \in \mathbb{R}^{p\times k}}{argmax}\ V_{\mathbf{X}}(\mathbf{X}\mathbf{P}_k\mathbf{P}_k^T). \tag{4.26}$$

or equivalently

$$\mathbf{P}_k = \underset{\mathbf{M}_1,\mathbf{M}_2 \in \mathbb{R}^{p\times k}}{argmin} \parallel \mathbf{X} - \mathbf{X}\mathbf{M}_1\mathbf{M}_2^T \parallel_F \tag{4.27}$$

In particular, it can be shown that the approximation error is:

$$\parallel \mathbf{X} - \mathbf{X}\mathbf{P}_k\mathbf{P}_k^T \parallel_F = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_p^2} \tag{4.28}$$

Consequently, it follows that when $\mathbf{S}$ is restricted to k columns, the optimal choice is $\mathbf{S} = \mathbf{T}_k$, in which case $\mathbf{\Theta} = \mathbf{P}_k^T$.

Example 4.3.1 (PCA J2M)**.** J2M is a dataset derived from the OES spectra obtained from a plasma etch process as described in chapter 2 (Dataset 2.4.2) $\mathbf{X} \in \mathbb{R}^{2194\times 12277}$ and relative etch rate (ER) measurements $\mathbf{y} \in \mathbb{R}^{2194}$. A detailed description of the data is reported in dataset 2.4.2. The result of a PCA analysis on the J2M data is represented in Figure 4.1. In the first plot the $\mathbf{T}_2$ matrix (i.e. the PCA scores corresponding to the first 2 PCs) is represented. The samples are coloured according to their ER value. It can be observed that the lower dimensional approximation keeps the main structure of the data. Samples that have similar ER are close together in the two dimensional plot. The second plot shows the percentage of explained variance as a function of the number of PCA components. The OES data is characterized by high correlation between variables. It follows that only a few PCs are required to obtained a good approximation of the data. In this case $k = 10$ PCs explain more than 99% of the variance.

4.3.2.4 T^2 and Q Statistics

Hotelling's T^2 and the lack-of-fit Q are two important statistics associated with a PCA decomposition. Hotelling's T^2 statistic measures the variation of each sample $\vec{\mathbf{x}} \in \mathbb{R}^p$ within the PCA model.

$$T_k^2(\vec{\mathbf{x}}) = (\vec{\mathbf{x}}\mathbf{P}_k)\tilde{\mathbf{\Sigma}}_{k\times k}^{-1}(\vec{\mathbf{x}}\mathbf{P}_k)^T \tag{4.29}$$

where $\tilde{\mathbf{\Sigma}}_{k\times k}$ is the diagonal matrix whose diagonal contains the k largest eigenvalues of $\mathbf{\Sigma}$. Observe that $\vec{\mathbf{x}}\mathbf{P}_k$ is the projection of $\vec{\mathbf{x}}$ on the k-dimensional PCA subspace.

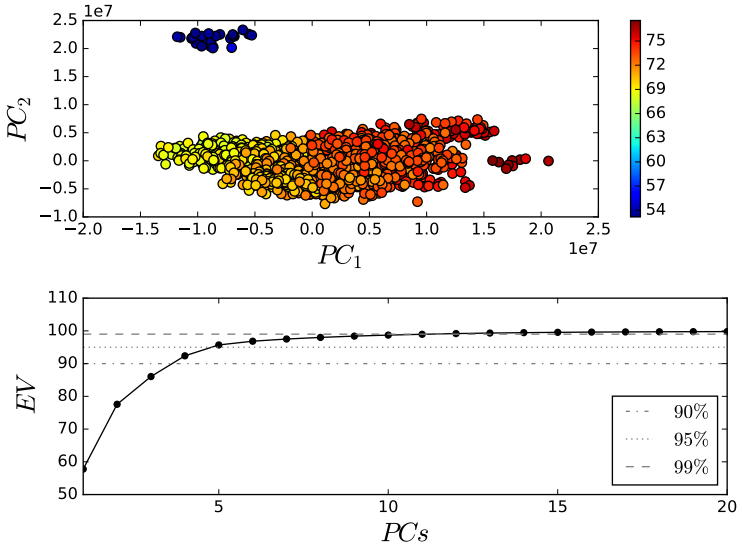


Figure 4.1: PCA analysis of the J2M dataset: The upper plot shows the two dimensional representation of the OES data obtained using PCA. The samples are coloured according to the ER value. The bottom plot shows the percentage of explained variance as a function of the number of PCs.

The Q-statistic, also known as the squared prediction error (SPE), is a measure of the degree of variation exhibited by a sample of the input variables (a row of $\mathbf{X}$) that is unexplained by the PCA model.

$$Q_k(\vec{\mathbf{x}}) = \vec{\mathbf{x}}(\mathbf{I} - \mathbf{P}_k\mathbf{P}_k^T)\vec{\mathbf{x}}^T \tag{4.30}$$

An application of the T and Q statistics to anomaly detection is described in chapter 6 (Section 6.3.1).

4.3.3 PCA Algorithms

Various algorithms exist for computing the PCA decomposition. Among these the most popular are the singular value decomposition (SVD, [124]) and the Nonlinear Iterative Partial Least Squares (NIPALS, [125]). While SVD is more numerically robust and efficient when a full PCA decomposition is required, NIPALS computes the PCA decomposition iteratively, one principal component (PC) at a time, in descending order. This makes it highly efficient in high dimensional problems where typically only a small number of PCs needs to be computed.

Definition 4.3.1 (SVD)**.** The Singular Values Decomposition (SVD) of a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ is its factorization into the product of three matrices:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{4.31}$$

where $\mathbf{D} \in \mathbb{R}^{n \times p}$ is a diagonal matrix of singular values while $\mathbf{U} \in \mathbb{R}^{n \times n}$ and $\mathbf{V} \in \mathbb{R}^{p \times p}$ are unitary matrix i.e.

$$\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I} \quad and \quad \mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I} \tag{4.32}$$

$\mathbf{U}$ and $\mathbf{V}$ are referred as left and right singular matrices.

PCA and SVD are strictly related as reported in [126]. Given a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, the columns of $\mathbf{V}$ obtained from its SVD decomposition as in equation 4.31, are the eigenvectors of its estimated covariance matrix $\mathbf{\Sigma} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X}$. Taking into account the notation used in the PCA section it easily follows that $\mathbf{V} = \mathbf{P}$ and $\mathbf{U}\mathbf{D} = \mathbf{T}$. As previously discussed the SVD decomposition is not an efficient algorithm if only $k << p$ PCs are required. In such a scenario the Nonlinear Iterative Partial Least Squares Algorithm (NIPALS) is much more efficient [127] as it recursively computes the principal components one component at a time. The algorithm calculates $\mathbf{p}_1$ and $\mathbf{t}_1$ from $\mathbf{X}$. $\mathbf{X}$ is then updated as

$$\mathbf{X} = \mathbf{X} - \mathbf{t}_1\mathbf{p}_1^T \tag{4.33}$$

and iteratively all the principal components are computed. This results in a dramatic reduction in computational time since calculation of the covariance matrix is avoided. A complete description of the algorithm is given in Pseudocode 4.3.1.

4.4 Unsupervised Feature Selection

As discussed in the previous sections PCA is an efficient method for reducing the dimensionality of the data. The main drawback of PCA is the fact that each principal component is a linear combination of all the original variables, thus it is often difficult to interpret the results. It is desirable not only to achieve a good lower dimensionality approximation but also to reduce the number of explicitly used variables. An ad hoc way is to artificially set the loadings with absolute values smaller than a threshold to zero. This informal thresholding approach is frequently used in practice but can be potentially misleading [105].

$[\mathbf{P_k}, \mathbf{T_k}]$=NIPALS($\mathbf{X}$, k)
Input: Data matrix $\mathbf{X}$, number of PCs k
Set $\mathbf{E} = \mathbf{X}$
Initialise $\mathbf{P}_0 = \mathbf{T}_0 = \emptyset$
Set $\epsilon = 10^{-6}$ (convergence threshold)
Initialise $\mathbf{t}$ to a non-zero column of $\mathbf{X}$
for $j = 1$ **to** k **do**
    Set $\mathbf{t}_{new} = \mathbf{t}$ and $\mathbf{t}_{old} = \mathbf{t} + 2\epsilon$
    while $\| \mathbf{t}_{old} - \mathbf{t}_{new} \|_2 \geq \epsilon$ **do**
        $\mathbf{t}_{old} = \mathbf{t}_{new}$
        $\mathbf{p} = \mathbf{E}^T\mathbf{t}/\mathbf{t}^T\mathbf{t}$
        $\mathbf{p} = \mathbf{p}/\sqrt{\mathbf{p}^T\mathbf{p}}$
        $\mathbf{t} = \mathbf{E}\mathbf{p}$
        $\mathbf{t}_{new} = \mathbf{t}$
    end while
    $\mathbf{P}_j = (\mathbf{P}_{j-1}, \mathbf{p})$
    $\mathbf{T}_j = (\mathbf{T}_{j-1}, \mathbf{t})$
    $\mathbf{E} = \mathbf{E} - \mathbf{t}\mathbf{p}^T$
end for
return $\mathbf{P}_k$, $\mathbf{T}_k$

Pseudocode 4.3.1: PCA NIPALS Algorithm

4.4.1 Sparse PCA

Sparse Principal Components Analysis (Sparse PCA [108]) was developed as a more interpretable PCA alternative. In Sparse PCA the loading matrix $\mathbf{P}$ is sparse (i.e. only few elements are nonzero). The score matrix $\mathbf{T}$ will then be obtained as a linear combination of only a few of the original variables. The relevant variables are the ones with nonzero coefficients in the loadings matrix. This allows dimensionality reduction and features selection to be performed at the same time. The next theorem shows that the PCA loadings can be obtained as the solution of a least square problem.

Theorem 4.4.1. *Let* $\mathbf{X} = \mathbf{UDV}^T$ *be the singular values decomposition of* $\mathbf{X}$ *and* $\mathbf{Y}_i = \mathbf{U}_i\mathbf{D}_i$ *the projection of* $\mathbf{X}$ *on* PC_i*. Given the solution of the Ridge regression problem*

$$\hat{\boldsymbol{\beta}}_{ridge} = argmin_{\boldsymbol{\beta}} \parallel \mathbf{Y}_i - \mathbf{X}\boldsymbol{\beta} \parallel_2^2 + \lambda \parallel \boldsymbol{\beta} \parallel_2^2 \tag{4.34}$$

and define

$$\hat{\mathbf{v}} = \frac{\hat{\boldsymbol{\beta}}_{ridge}}{\parallel \hat{\boldsymbol{\beta}}_{ridge} \parallel_2^2} \tag{4.35}$$

it follows that $\hat{\mathbf{v}} = \mathbf{V}_i$

The same theorem also holds without the ridge penalty but $\lambda \neq 0$ allows us to handle high dimensional data $p > n$ where a unique solution to the least square problem does not exist [108].

The ridge regression problem in equation 4.34 can be generalized to yield the elastic net problem.

$$\hat{\boldsymbol{\beta}} = argmin_{\boldsymbol{\beta}} \parallel \mathbf{Y}_i - \mathbf{X}\boldsymbol{\beta} \parallel_2^2 + \lambda \parallel \boldsymbol{\beta} \parallel_2^2 + \lambda_1 \parallel \boldsymbol{\beta} \parallel_1 \tag{4.36}$$

The L_1 penalty makes $\hat{\boldsymbol{\beta}}$ a sparse vector. Indeed this is essentially the lasso problem discussed in chapter 3. The vector $\hat{\mathbf{V}}_i = \frac{\hat{\boldsymbol{\beta}}}{\parallel \boldsymbol{\beta} \parallel_1}$ is the sparse approximation of $\mathbf{V}_i$. For consistency with the PCA notation we define

$$\mathbf{P}^{\lambda} = \hat{\mathbf{V}} \tag{4.37}$$

where λ is used to signify the dependency of $\mathbf{P}$ on the penalty weighting λ. The main PCA equation is then updated as:

$$\mathbf{XP}^{\lambda} = \mathbf{T}^{\lambda} \tag{4.38}$$

where $\mathbf{T}^{\lambda}$ is the projection of $\mathbf{X}$ in the space defined by $\mathbf{P}^{\lambda}$. The equation 4.19 does not hold in this case due to the non orthogonality of $\mathbf{P}^{\lambda}$. Instead the lower dimensional reconstruction of $\mathbf{X}$, corresponding to the projection of $\mathbf{X}$ onto the space spanned by $\mathbf{P}^{\lambda}$ can be obtained by linear regression as explained in Section 4.3.

4.4.1.1 Challenges with Variables Selection with SPCA

SPCA was initially motivated to obtain a more interpretable approximation of the PCA model and is commonly used as a benchmark for unsupervised features selection methodologies. The main issue with using SPCA for feature selection is the difficulty with determining what features to select. This problem was already reported in [117] and is illustrated in the following example.

Example 4.4.1 (SPCA on Glass data)**.** Glass is a popular datasets often used in machine learning. Its reference paper is [128]. The data has a total of $p = 9$ variables. 3 components are computed with SPCA and the result is reported in Table 4.1. From the table it is difficult to understand what variables should be selected. Variable $\mathbf{x}_5$ for example has a nonzero loading in the first component but variable $\mathbf{x}_6$ has a much larger weight on the third one. It is difficult to decide if variable $\mathbf{x}_5$ should be preferred to variable $\mathbf{x}_6$.

	PC_1	PC_2	PC_3
x_1	-9.164	0.0	0.0
x_2	0.0	0.0	0.0
x_3	0.0	6.872	0.0
x_4	0.0	-6.724	0.0
x_5	2.585	0.0	0.0
x_6	0.0	0.0	-9.595
x_7	-8.343	0.0	0.0
x_8	0.0	-6.86	0.0
x_9	0.0	0.0	0.0

Table 4.1: The first three PCs obtained with SPCA on the glass data.

In addition SPCA is not efficient for unsupervised variable selection. The reason for this is that in certain situations SPCA assigns similar loadings to similar variables. This follows from the fact that, even if it is not guaranteed, lasso can have the grouping effect [65]. This will be illustrated in Example 4.7.3, which will be presented later in the chapter.

4.4.1.2 SPCA Algorithms

Different algorithms have been developed in order to efficiently compute the sparse PCA decomposition. In the original paper [108] SPCA is computed with an iterative algorithm. This algorithm needs to compute the SVD decomposition of the data several times and it is consequently inefficient for high dimensional datasets. An alternative algorithm denoted as sPCA-rSVD is proposed in [110]. The sPCA-rSVD is described in Pseudocode 4.4.1. In step two of the original version of the algorithm the best rank one approximation of $\mathbf{X}$ is obtained with SVD. It is better to obtain it with the NIPALS algorithm as it is more efficient as only one PCA component is required.

Input: Input matrix $\mathbf{X}$, k the number of components, λ_1 the penalty value and the tolerance τ.

1: **for** $i = 1 : k$ **do**
2: Compute $\mathbf{u}_{new}, \mathbf{v}_{new}$: $\mathbf{u}_{new}\mathbf{v}_{new}^T$ is the best rank one approximation
3: $\mathbf{u}_{old} = 0,\ \mathbf{v}_{old} = 0$;
4: **while** $\| \mathbf{v}_{new} - \mathbf{v}_{old} \| \geq \mathbf{v}_{old}\tau$ **or** $\| \mathbf{u}_{new} - \mathbf{u}_{old} \| \geq \mathbf{u}_{old}\tau$ **do**
5: $\mathbf{v}_{old} = \mathbf{v}_{new},\ \mathbf{u}_{old} = \mathbf{u}_{new}$;
6: $\mathbf{v}_{new} = sign(\mathbf{X}^T\mathbf{u}_{old})(|\mathbf{X}^T\mathbf{u}_{old}| - \lambda_1)_+$;
7: $d = \| \mathbf{X}\mathbf{v}_{new} \|_2$;
8: $\mathbf{u}_{new} = \mathbf{X}\mathbf{v}_{new} / \| \mathbf{X}\mathbf{v}_{new} \|_2$;
9: **end while**
10: $\mathbf{X} = \mathbf{X} - \Phi(\mathbf{u}_{new})\mathbf{X}$;
11: **save** $\mathbf{v}_{new}$;
12: **end for**
13: **return** $\mathbf{T}$, $\mathbf{P}$

Pseudocode 4.4.1: sPCA-rSVD

4.5 More Effective Unsupervised Feature Selection Algorithms

As explained in the previous section it is difficult to perform unsupervised variable selection using the PCA and SPCA decompositions. In the literature several algorithms have been developed as better alternatives to PCA and SPCA. Many of these algorithms may be thought of as extensions to the unsupervised domain of algorithms like lasso [63], forward selection regression (FSR) and backward elimination regression (BER) that were originally designed for supervised regression [129]. The unsupervised version of lasso is

proposed in the paper convex principal features selection [117], CPFS and is described in the following algorithm.

Algorithm 4.5.1 (Convex Principal Features Selection)**.** CPFS is defined by the following minimization problem:

$$\hat{\mathbf{A}}_\lambda = \underset{\mathbf{A}\in\mathbb{R}^{p\times p}}{argmin} \parallel \mathbf{X} - \mathbf{XA} \parallel_F^2 + \lambda \sum_{i=1}^{p} \parallel \vec{\mathbf{a}}_i \parallel_\infty \tag{4.39}$$

where

$$A = \begin{pmatrix} \vec{\mathbf{a}}_1 \\ \vec{\mathbf{a}}_2 \\ \cdots \\ \vec{\mathbf{a}}_p \end{pmatrix} \tag{4.40}$$

The result of the minimization problem is a sparse matrix $\hat{\mathbf{A}}_\lambda$ that contains the regression coefficient of each variable. The rows of $\mathbf{A}$ whose values are all 0 or smaller than a certain threshold $\parallel \vec{\mathbf{a}}_j \parallel_2 \leq \epsilon$ correspond to the variables that are redundant and that can be discarded.

A more aggressive feature selection approach can be obtained with the unsupervised version of FSR and BER. Forward selection and backward elimination of variables are well known techniques extensively used in supervised learning. Given an input dataset $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p) \in \mathbb{R}^{n\times p}$, an output $\mathbf{y} \in \mathbb{R}^n$ and a function $\mathbf{\Phi}$ that maps $\mathbf{X}$ into $\mathbf{y}$, it is of interest to estimate the set of $k < p$ variables $(\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k})$ that can optimally reconstruct $\mathbf{y}$. Determining the optimal solution to the problem requires the evaluation of all $\binom{p}{k}$ combinations which become computationally intractable even for moderate values of p and k. An approximation of the solution is obtained using the forward selection [130], [131], [132], [133] or backward elimination procedure [129], [17]. Different algorithms have been proposed based on these principles for unsupervised features selection. These iteratively select or remove variables according to a maximization criteria.

4.5.1 Unsupervised Forward Selection and Backward Elimination of Variables

4.5.1.1 FOS-MOD

Among the unsupervised forward selection algorithms one of the most famous is FOS-MOD [119], Forward Orthogonal Search (FOS) algorithm by Maximizing the Overall Dependency (MOD), described in the following algorithm.

Algorithm 4.5.2 (FOS-MOD)**.** The similarity between two variables is defined as:

$$sc(\mathbf{x}, \mathbf{y}) = \frac{(\mathbf{x}^T\mathbf{y})^2}{(\mathbf{x}^T\mathbf{x})(\mathbf{y}^T\mathbf{y})} \tag{4.41}$$

and a matrix of similarity between variables is computed as:

$$\mathbf{C}^1 = \{c^1_{i,j}\}_{i,j=1,\ldots,p} = \{sc(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1,\ldots,p} \tag{4.42}$$

The first variable is then selected as $\mathbf{z}_1 = \mathbf{x}_{l_1}$ where l_1 is obtained as:

$$\bar{\mathbf{C}}^1_j = \frac{1}{p}\sum_{i=1}^{p} c^1_{i,j} \tag{4.43}$$

$$l_1 = \underset{1\leq j\leq n}{argmax}\ \bar{\mathbf{C}}^1_j \tag{4.44}$$

and the associated orthogonal variable is chosen as

$$\mathbf{q}_1 = \mathbf{z}_1 \tag{4.45}$$

At step $m-1$, $m-1$ variables have been selected

$$S_{m-1} = \{\mathbf{z}_1, \ldots, \mathbf{z}_{m-1}\} \tag{4.46}$$

and their respective $m-1$ associated orthogonal variables are

$$Q_{m-1} = \{\mathbf{q}_1, \ldots, \mathbf{q}_{m-1}\} \tag{4.47}$$

All the variables in the set complementary to S_{m-1} ($\mathcal{C}(S_{m-1})$) are transformed as:

$$\mathbf{q}^m_j = \boldsymbol{\alpha}_j - \frac{\boldsymbol{\alpha}^T_j\mathbf{q}_1}{\mathbf{q}^T_1\mathbf{q}_1} - \cdots - \frac{\boldsymbol{\alpha}^T_j\mathbf{q}_{m-1}}{\mathbf{q}^T_{m-1}\mathbf{q}_{m-1}} \quad \forall\ \boldsymbol{\alpha}_j \in \mathcal{C}(S_{m-1}) \tag{4.48}$$

The matrix $\mathbf{C}$ at step m is then defined as:

$$\mathbf{C}^m = \{c^m_{i,j}\}_{i,j=1,\ldots,p} = \{sc(\mathbf{x}_i, \mathbf{q}^m_j)\}_{i,j=1,\ldots,p} \tag{4.49}$$

and the m-th chosen variable is then $\mathbf{z}_m = \mathbf{x}_{l_m}$ where

$$\bar{\mathbf{C}}^m_j = \frac{1}{n}\sum_{i=1}^{n} c^m_{i,j} \tag{4.50}$$

$$l_m = \underset{1\leq j\leq n}{argmax}\ \bar{\mathbf{C}}^m_j \tag{4.51}$$

FOS-MOD bases all its computation on the covariance matrix $\mathbf{C}$. Other methods base the selection of variables on the reconstruction error or explained variance. This makes them more similar to PCA. Some of these are Orthogonal Feature Selection (OFS, [134]), Selection of Variables to Preserve Multivariate Data Structure (SV, [114]) and Forward Selection Component Analysis (FSCA, [89]). In the next section a high level overview of these algorithms is presented. It will be clear that they all share a similar structure. FSCA will then be used as an exemplar methodology and described in detail. It will be also shown that FOS-MOD is a particular case of FSCA.

4.5.1.2 Orthogonal Feature Selection

In Orthogonal Feature Selection (OFS) [134] the best rank one approximation of the data is computed with PCA ($\mathbf{t}_1\mathbf{p}_1^T$). The variable $\mathbf{z}_1$ is then selected as the one most correlated with $\mathbf{t}_1$. The data is then projected in the space orthogonal to $\mathbf{z}_1$ and the procedure is repeated until k variables are selected. The pseudocode for the algorithm is reported in 4.5.1. The main advantage of this algorithm is the low computational complexity when the number of variables p is large. Indeed $\mathbf{t}_1$ can be efficiently computed with the NIPALS Pseudocode 4.3.1 and all the other operations are relatively low complexity even if large datasets are used. On the other hand the algorithm relies on the fact that the variables are selected in order to approximate $\mathbf{t}_1$ which is itself an approximation of the original matrix $\mathbf{X}$. This means that the obtained reconstruction is, in general, less accurate than the one obtained with FOS-MOD or with the FSCA algorithm which will be presented in Section 4.6.

Input: Input matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p)$ and k the number of components.
1: **for** $r = 1, \ldots, k$ **do**
2: Compute $\mathbf{t}_1$ the projection of $\mathbf{X}$ on its first PC.
3: $\mathbf{z}_r = \text{argmax}_{\mathbf{x}_i}\, corr(\mathbf{x}_i, \mathbf{t}_1)$
4: Save $\mathbf{z}_r$ as the r^{th} variable
5: $\mathbf{X} = \mathbf{X} - \mathbf{\Phi}(\mathbf{z}_r)\mathbf{X}$ (Deflate $\mathbf{X}$)
6: **end for**
7: **return** $\mathbf{z}_1, \ldots, \mathbf{z}_k$

Pseudocode 4.5.1: Orthogonal Features Selection

4.5.1.3 Selection of Variables to Preserve Multivariate Data Structure

In [114] another feature selection method SV is proposed. At first all the variables are considered and then they are recursively discarded. The variables

are selected according to their ability to approximate the PCA scores. Indeed the algorithm is based on ideas similar to the ones used in OFS described in the previous section. A full description of the algorithm is reported in Pseudocode 4.5.2. The main drawback of this algorithm is its computational complexity. It is based on backward elimination, an algorithm that is more computationally demanding than forward selection. In addition PCA must be computed several times. In addition the backward elimination procedure does not work if the number of variables is larger than the number of samples as discussed in [135].

Input: Input matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p)$ and k the number of components.
while $\mathbf{X}$ has more than k variables **do**
  Compute $\mathbf{T}_v$ the projection of $\mathbf{X}$ on its first v PCs
  for $i = 1, \ldots, p$ **do**
    Obtain $\mathbf{X}_i$ removing $\mathbf{x}_i$ from $\mathbf{X}$.
    Compute $\mathbf{T}_v(i)$ the projection of $\mathbf{X}_i$ on its first v PCs
    $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is the SVD decomposition of $\mathbf{T}_v^T\mathbf{T}_v(i)$
    $\mathbf{D}_i = trace(\mathbf{T}_v^T\mathbf{T}_v - \mathbf{T}_v(i)^T\mathbf{T}_v(i) - 2\mathbf{\Sigma})$
  end for
  $v = \text{argmin}_i\, \mathbf{D}_i$
  Discard the variable $\mathbf{x}_v$ from $\mathbf{X}$
end while

Pseudocode 4.5.2: Selection of Variables to Preserve Multivariate Data Structure.

4.6 Forward Selection Component Analysis

In contrast to PCA, which produces a reduced set of new variables (PCs) that are linear combinations of all existing variables, Forward Selection Component Analysis (FSCA) derives a set of new variables (FSCs) that are a function of only a subset of the original variables that maximises the explained variance. This is achieved using the iterative procedure detailed in Pseudocode 4.6.1. The FSCA algorithm returns:

- A matrix $\mathbf{Z}_k$ composed of a subset of the columns of $\mathbf{X}$ (FSVs) ranked according to how well they contribute to the reconstruction of $\mathbf{X}$.
- A matrix of FSCA components (FSCs) $\mathbf{M}_k$. The first column of $\mathbf{M}_k$ is equivalent to the first column of $\mathbf{Z}_k$. The second column of $\mathbf{M}_k$ is a function of the first and the second column of $\mathbf{Z}_k$ and so on. In general

the k-th FSC will be defined as a function of itself and of the previous $k-1$ components and so is a function of the first k selected variables.

- A matrix of FSCA loadings $\mathbf{U}_k$.

FSCA leads to a decomposition of $\mathbf{X}$ of the form

$$\hat{\mathbf{X}}^k_{FSCA} = \mathbf{M}_k\mathbf{U}_k^T = \sum_{i=1}^{k} \mathbf{m}_i\mathbf{u}_i^T. \tag{4.52}$$

where $\mathbf{M}_k$ is an orthogonal matrix of Forward Selection Components (FSCs). Equivalently we can express the decomposition directly in terms of the Forward Selection Variables (FSVs), $\mathbf{Z}_k$ as

$$\hat{\mathbf{X}}^k_{FSV} = \mathbf{Z}_k\mathbf{B}_k^T = \sum_{i=1}^{k} \mathbf{z}_i\mathbf{b}_i^T. \tag{4.53}$$

Here $\mathbf{Z}_k = (\mathbf{z}_1, ..., \mathbf{z}_k) = (\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_k}) \subset \mathbf{X}$ and $\mathbf{B}_k^T$ are the corresponding least squares regression coefficients, that is

$$\mathbf{B}_k = \mathbf{X}^T\mathbf{Z}_k(\mathbf{Z}_k^T\mathbf{Z}_k)^{-1}. \tag{4.54}$$

Observation 4.6.1.

$$\hat{\mathbf{X}}_{FSCA} = \hat{\mathbf{X}}_{FSV} \tag{4.55}$$

In a similar fashion to NIPALS (Pseudocode 4.3.1), the FSCs generated by FSCA are ordered in descending order in terms of the variance of $\mathbf{X}$ explained. Furthermore, by virtue of their orthogonality the variance contribution of the i-th FSC is simply obtained as $(\mathbf{m}_i^T\mathbf{m}_i)(\mathbf{u}_i^T\mathbf{u}_i)$. A similar expression holds for PCA, but since $\mathbf{P}_k$ is an orthonormal matrix this reduces to $\mathbf{t}_i^T\mathbf{t}_i$.
The FSV decomposition could be computed directly, rather than as a by-product of the FSC computation by solving

$$i_k = \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}((\mathbf{Z}_{k-1}, \mathbf{x}_i))\mathbf{X}) \tag{4.56}$$

and setting $\mathbf{Z}_k = [\mathbf{Z}_{k-1}\ \mathbf{x}_{i_k}]$, with $\mathbf{Z}_0 = \emptyset$. However, the results are equivalent for a given number of components, that is $\hat{\mathbf{X}}^k_{FSC} = \hat{\mathbf{X}}^k_{FSV}$. In particular, once $\mathbf{Z}_k$ has been computed, the corresponding FSC matrix $\mathbf{M}_k$ can be obtained as the Gram-Schmidt orthogonalization of $\mathbf{Z}_k$. Thus an alternative FSV based implementation of FSCA is as given in Pseudocode 4.6.2. Note that steps 5 and 7 are place holders for the refinement step which will be introduced in Section 4.7 and are not part of the basic algorithm, which we

will refer to as the FSV algorithm (FSVA). If we are only interested in FSVs steps 8 and 9 can be omitted, in which case the algorithm corresponds to the feature selection methods presented in [120] and [121].

Since PCA provides the optimal representation in terms maximising the variance explained (eq. 4.27) it follows that $V_{\mathbf{X}}(\hat{\mathbf{X}}^k_{FSC}) \leq V_{\mathbf{X}}(\hat{\mathbf{X}}^k_{PCA})$. Hence PCA can be regarded as providing an upper bound on the performance achievable with FSCA with a given number of variables k, or equivalently a lower bound on the number of variables needed to achieve a desired reconstruction accuracy.

4.6.1 Computational Complexity of FSCA

The computation time of the FSCA algorithm (Pseudocode 4.6.1) is dominated by the combinatorial optimisation problem in step 4. It is relatively straight forward to show that maximising the explained variance is equivalent to maximising the Rayleigh Quotient of $\mathbf{X}\mathbf{X}^T$, that is

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{x}_i)\mathbf{X}) \equiv \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ \frac{\mathbf{x}_i^T \mathbf{X}\mathbf{X}^T \mathbf{x}_i}{\mathbf{x}_i^T \mathbf{x}_i}, \tag{4.57}$$

which can be computed efficiently as

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ \sum_{j=1}^{v} \frac{(\mathbf{x}_i^T \mathbf{x}_j)^2}{\mathbf{x}_i^T \mathbf{x}_i}. \tag{4.58}$$

This is proved in Theorem B.0.2 in Appendix B.

It is interesting to note that the corresponding expression for maximising the average squared correlation metric employed in the FOS algorithm proposed by Wei and Billing [119] is

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ \sum_{j=1}^{v} \frac{(\mathbf{x}_i^T \mathbf{x}_j)^2}{(\mathbf{x}_i^T \mathbf{x}_i)(\mathbf{x}_j^T \mathbf{x}_j)}. \tag{4.59}$$

Hence, while the FOS optimization objective has an additional scaling factor in the denominator, both algorithms will in fact yield identical results provided the columns of the data matrix $\mathbf{X}$ are normalised so that they are all the same length ($\mathbf{x}_j^T \mathbf{x}_j$ will then be invariant with respect to j).

It is also worth noting that if $\mathbf{x}_i$ is not constrained to be a column of $\mathbf{X}$ the solution to (4.57) is the largest eigenvector of $\mathbf{X}\mathbf{X}^\mathsf{T}$, but this is simply the

direction of the score vector $\mathbf{t}_1$ corresponding to the first PC of the data. This shows the similarity between FSCA and the OPFS whereby the first variable selected is the one that is most closely correlated with the first PC. OPFS is in general only an approximation to FSCA. This can be deduced as follows. Recalling the definition of the PCA decomposition in equations 4.23 and 4.19, the FSCA optimization objective (eqt. 4.57) can be expressed as

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \sum_{j=1}^{r} \frac{(\mathbf{x}_i^{\mathsf{T}} \mathbf{t}_j)^2}{\mathbf{x}_i^{\mathsf{T}} \mathbf{x}_i} \tag{4.60}$$

or equivalently as

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \sum_{j=1}^{r} \lambda_j corr(\mathbf{x}_i, \mathbf{t}_j)^2, \tag{4.61}$$

where $\lambda_j (= \mathbf{t}_j^{\mathsf{T}} \mathbf{t}_j)$ is the variance contribution of the j-th PC. In contrast the OPFS optimization objective corresponds to

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ corr(\mathbf{x}_i, \mathbf{t}_1)^2. \tag{4.62}$$

Thus, while OPFS selects variables based on their squared correlation with the first PC, FSCA selects them based on the variance-weighted average squared correlation with all PCs. Hence, the sequence of variables selected by OPFS will in general differ from, and explain less variance than, the variables selected by FSCA.

If FSCA is computed using the FSVA implementation (Pseudocode 4.6.2) an efficient solution can be obtained by noting that the combinatorial optimisation problem in equation (4.56) is equivalent to

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \sum_{j=1}^{v} \mathbf{x}_j^T \mathbf{\Phi}((\mathbf{Z}_{k-1}, \mathbf{x}_i)) \mathbf{x}_j. \tag{4.63}$$

Recalling the definition of $\mathbf{\Phi}$ (eqt. 4.9) this can be recast as

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \sum_{j=1}^{v} \mathbf{q}_{j(i)}^T (\mathbf{Z}_{(i)}^T \mathbf{Z}_{(i)})^{-1} \mathbf{q}_{j(i)}, \tag{4.64}$$

where $\mathbf{q}_{j(i)} = \mathbf{Z}_{(i)}^T \mathbf{x}_j$, and $\mathbf{Z}_{(i)} = (\mathbf{Z}_{k-1}, \mathbf{x}_i)$. Hence, determining the optimum $\mathbf{x}_i$ requires p^2 evaluations of the vector terms $\mathbf{q}_{j(i)}$ and p evaluations of the matrix inverse term $(\mathbf{Z}_{(i)}^T \mathbf{Z}_{(i)})^{-1}$. We can take two steps to substantially reduce the computation time for these terms. Firstly, as proposed in [120,

121], an $O(k^2)$ complexity recursive computation of the matrix inverse can be obtained by taking advantage of the fact that

$$\mathbf{Z}_{(i)}^T\mathbf{Z}_{(i)} = \begin{bmatrix} \mathbf{Z}_{k-1}^T\mathbf{Z}_{k-1} & \mathbf{r}_{(i)} \\ \mathbf{r}_{(i)}^T & a_{(i)} \end{bmatrix}, \tag{4.65}$$

where $\mathbf{r}_{(i)} = \mathbf{Z}_{k-1}\mathbf{x}_i$, and $a_{(i)} = \mathbf{x}_i^T\mathbf{x}_i$, and applying block matrix inversion algebra to obtain an expression for $(\mathbf{Z}_{(i)}^T\mathbf{Z}_{(i)})^{-1}$ in terms of $(\mathbf{Z}_{k-1}^T\mathbf{Z}_{k-1})^{-1}$ which has already been computed in the previous iteration, that is:

$$(\mathbf{Z}_{(i)}^T\mathbf{Z}_{(i)})^{-1} = \begin{bmatrix} (\mathbf{Z}_{k-1}^T\mathbf{Z}_{k-1})^{-1} + w\mathbf{b}\mathbf{b}^T & -w\mathbf{b} \\ -w\mathbf{b}^T & w \end{bmatrix}, \tag{4.66}$$

$$\mathbf{b} = (\mathbf{Z}_{k-1}^T\mathbf{Z}_{k-1})^{-1}\mathbf{r}_{(i)},\ w = (a_{(i)} - \mathbf{r}_{(i)}^T\mathbf{b})^{-1}.$$

In contrast direct calculation of the matrix inverse has $O(k^3)$ computational complexity.

Secondly, evaluating the terms $\mathbf{q}_{j(i)}$, $\mathbf{r}_{(i)}$ and $a_{(i)}$ all involve computing vector products $\mathbf{x}_i^T\mathbf{x}_j$ many times, with substantial repetition both within each variable selection iteration and between iterations. This repetition can be eliminated by precomputing the covariance matrix $\mathbf{C} = \mathbf{X}^T\mathbf{X}$, where $c_{ij} = \mathbf{x}_i^T\mathbf{x}_j$, at the cost of $\mathsf{O}(p^2n)$ floating point operations (flops) and $\mathsf{O}(p^2)$ additional memory. Table 4.2 shows the estimated complexity in terms of floating point operations for computing k FSCs with the FSCA and FSVA algorithms, with and without the covariance matrix precomputed, while Fig. 4.2 shows how complexity varies as a function of p and n for specific combinations of the other dimensions. As can be seen, the reduction in complexity is of the order $\mathsf{O}(k^2)$ for FSVA. Precomputing $\mathbf{C}$ is also beneficial for FSCA, but the impact is less significant since it has to be re-computed at each iteration due to the deflation step. That said, a factor of two reduction in computational complexity is achieved for FSCA. All algorithms scale quadratically with p and linearly with n, but differ in how they behave with respect to k, with FSCA implementations growing linearly and FSVA implementations growing cubically. If precomputing the covariance matrix is not an issue, the preferred algorithm is FSVA when $k < \sqrt{1.5n}$ (approx.) and FSCA otherwise. FSCA is substantially superior to FSVA when the covariance matrix is not precomputed and also outperforms precomputed FSVA when $k > \sqrt{3n}$. When the computational burden of FSCA becomes prohibitive OPFS may offer an attractive compromise due to its significantly lower computational complexity. In OPFS the PC needed at each step can be computed efficiently

using NIPALS yielding and algorithm with $\mathsf{O}((4\alpha + 8)np)$ complexity per selected variable, compared to order $\mathsf{O}(np^2)$ with FSCA. Here, α denotes the average number of iterations per selected variable for the NIPALS algorithm to converge. It is a function of the spread of the eigenvalues of the covariance matrix $\mathbf{C}$, and hence problem dependent.

Notation: $\bar{k} = k(k-1)/2$; $\bar{k}^2 = k(k+1)(2k+1)/6$; and PC denotes implementation with precomputed covariance matrix.

Method	Floating point operation count	Complexity $k << v$
FSCA	$\mathsf{O}(2p^2nk+6pnk+v^2k+2nk-pk-k)$	$\mathsf{O}(2p^2nk)$
FSCA (PC)	$\mathsf{O}(v^2nk + 2pnk + 2p^2k)$	$\mathsf{O}(p^2nk)$
FSVA	$\mathsf{O}(2p^2n\bar{k} + 2p^2\bar{k}^2 + 4pn\bar{k} + 4p\bar{k}^2 + 4pnk - 2pk + 2nk^2 + 4nk)$	$\mathsf{O}(p^2nk^2 + \frac{2}{3}p^2k^3)$
FSVA (PC)	$\mathsf{O}(p^2n+v^2(2\bar{k}^2+\bar{k})+2nk^2+2pnk+4nk-pk+p(4\bar{k}^2-6\bar{k}+3k))$	$\mathsf{O}(p^2n + \frac{2}{3}p^2k^3)$

Table 4.2: Flop count and asymptotic complexity for computing k FSCs of $\mathbf{X} \in \mathbb{R}^{m\times v}$ with different FSCA algorithm implementations

4.7 Backward Refinement Procedure for FSCA

In this section a series of novel backward refinement procedures that improve the FSCA algorithm are proposed. Ideally we would like to find the subset of k columns of $\mathbf{X}$ (variables) that can optimally reconstruct $\mathbf{X}$, that is

$$\underset{\mathbf{Z}_k \in \mathbf{X}}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{Z}_k)\mathbf{X}). \tag{4.67}$$

However, this is an NP hard combinatorial optimisation problem (requires the evaluation of ${}^vC_k = v!/((v-k)!k!)$ possible combinations of the variables). In general FSCA and other greedy local search approaches are sub optimal (i.e. they are not guaranteed to find the optimal subset of variables according to the defined optimization criteria), but they represent a pragmatic solution as searching over all possible subsets quickly becomes computationally

$[\mathbf{Z}_k, \mathbf{M}_k, \mathbf{U}_k]$=FSCA($\mathbf{X}$, k)

Input: Data matrix $\mathbf{X}$, number of FSCs k

Set $\mathbf{E} = \mathbf{X}$
Initialise $\mathbf{Z}_0 = \mathbf{M}_0 = \mathbf{U}_0 = \emptyset$
for $j = 1$ **to** k **do**
    $i = \underset{\mathbf{e}_i \in \mathbf{E}}{argmax}\ V_{\mathbf{E}}(\mathbf{\Phi}(\mathbf{e}_i)\mathbf{E})$
    $\mathbf{m} = \mathbf{e}_i$
    $\mathbf{z} = \mathbf{x}_i$
    $\mathbf{u} = \mathbf{E}^T\mathbf{e}_i/(\mathbf{e}_i^T\mathbf{e}_i)$
    $\mathbf{M}_j = (\mathbf{M}_{j-1}, \mathbf{m})$
    $\mathbf{Z}_j = (\mathbf{Z}_{j-1}, \mathbf{z})$
    $\mathbf{U}_j = (\mathbf{U}_{j-1}, \mathbf{u})$
    $\mathbf{E} = \mathbf{E} - \mathbf{\Phi}(\mathbf{m})\mathbf{E}$
end for
return $\mathbf{Z}_k,\ \mathbf{M}_k,\ \mathbf{U}_k$

Pseudocode 4.6.1: FSCA Algorithm

$[\mathbf{Z_k}, \mathbf{M_k}, \mathbf{U_k}]$=FSVA($\mathbf{X}$,$k$)

Input: Data matrix $\mathbf{X}$, number of FSVs k

$\mathbf{Z}_0 = \emptyset$
for $j = 1$ **to** k **do**
    $i_j = \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}((\mathbf{Z}_{j-1}, \mathbf{x}_i))\mathbf{X})$
    $\mathbf{Z}_j = (\mathbf{Z}_{j-1}, \mathbf{x}_{i_j})$
    Optional refinement step (recursive)
end for
Optional refinement step
$\mathbf{M}_k = \text{GramSchmidt}(\mathbf{Z}_k)$
$\mathbf{U}_k = \mathbf{X}^T\mathbf{M}_k(\mathbf{M}_k^T\mathbf{M}_k)^{-1}$
return $\mathbf{Z}_k,\ \mathbf{M}_k,\ \mathbf{U}_k$

Pseudocode 4.6.2: Direct FSV implementation of FSCA

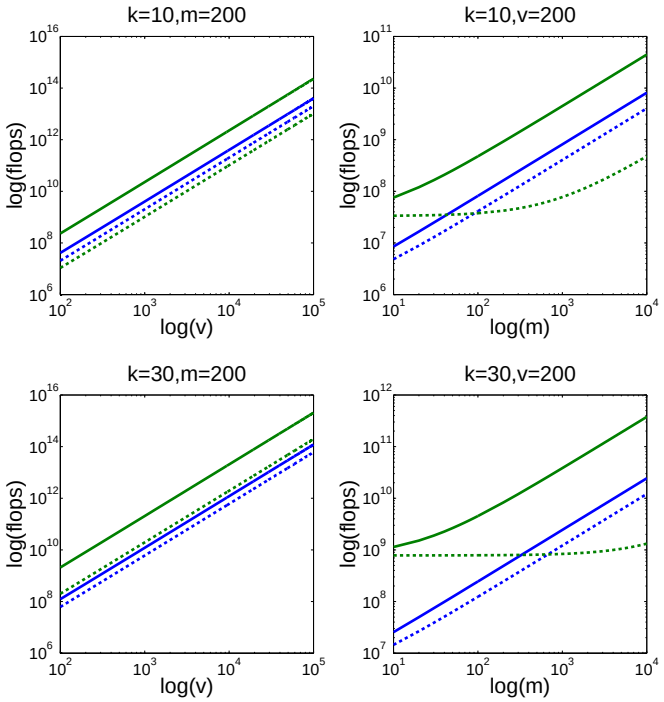


Figure 4.2: Complexity of FSCA algorithms as a function of v and m: Plots show FSCA (blue) and FSVA (green) implementations with precomputed covariance matrices (dashed lines) and without (solid lines).

intractable with increasing problem dimension.

As a consequence of the greedy strategy adopted by FSCA variables that are selected in early iterations of the algorithm can become redundant as other variables are included in later iterations. This can result in sub-optimal solutions and can also be detrimental to the performance of some applications, for example, the clustering application which will be presented in Section 4.7.6. To overcome this weakness, we propose introducing a backward refinement step similar to that presented in [62], [136] for forward selection regression applications, where, following completion of the forward selection process, selected variables are reviewed to see if they are still relevant and replaced if they are not.

Denoting $\mathbf{Z}_k^{(j)}(\mathbf{x}_i)$ as matrix $\mathbf{Z}_k$ with its j-th column replaced by $\mathbf{x}_i$, that is:

$$\mathbf{Z}_k^{(j)}(\mathbf{x}_i) = \mathbf{Z}_k + (\mathbf{x}_i - \mathbf{z}_j)\mathbf{e}_j^T, \tag{4.68}$$

where $\mathbf{e}_j$ is a vector with its j-th element equal to 1 and all others elements equal to zero, we define $\mathbf{z}_j \in \mathbf{Z}_k$ as relevant if

$$V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{Z}_k)\mathbf{X}) \geq \underset{\mathbf{x}_i \in \mathbf{X}/\mathbf{Z}_k}{max} V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{Z}_k^{(j)}(\mathbf{x}_i))\mathbf{X}). \tag{4.69}$$

This backward refinement step can be performed either at step 5 or step 7 of the FSV implementation of the FSCA algorithm, as highlighted Pseudocode 4.6.2. When placed at step 7 the refinement step is only undertaken once after the FSV algorithm has completed. In contrast, the refinement step is executed following the addition of each new variable if placed at step 5. We will refer to this latter implementation as recursive backward refinement.
There are also two flavours of the refinement step itself. In the first, referred to as Single-Pass Backward Refinement (SPBR) (summarised in Pseudocode 4.7.1), the relevance of each variable is evaluated in turn moving sequentially through the variables from the oldest to the newest. In the second, to take account of the fact that variables that are initially relevant may become irrelevant following refinements to variables later in the sequence, the process is repeated until a complete pass occurs without any refinements taking place. This version of the algorithm (summarised in Pseudocode 4.7.2) is referred to as Multi-Pass Backward Refinement (MPBR).
Note that by virtue of the sequencing of operations in each algorithm it follows that

$$V_{\mathbf{X}}(\hat{\mathbf{X}}_{FSCA}^k) \leq V_{\mathbf{X}}(\hat{\mathbf{X}}_{SPBR}^k) \leq V_{\mathbf{X}}(\hat{\mathbf{X}}_{MPBR}^k). \tag{4.70}$$

However, no such statement can be made with regard to R-SPBR or R-MPBR as they may follow different 'hill climbing' solution paths and hence it is possible for the solutions to be inferior to the non-recursive implementations when $k > 2$.

One of the side-effects of employing the backward refinement step is that it breaks the ordering of selected variables in terms of variance explained. If recovering this ordering is desirable, an additional modified FSV step can be performed on $\mathbf{Z}_k$ with respect to $\mathbf{X}$ after the refinement process has been completed (i.e. between Step 7 and 8 in Algorithm 3). As summarised in Algorithm 6, this involves recursively selecting the variables in $\mathbf{Z}_k$ based on how much of the variance of $\mathbf{X}$ that they explain.

4.7.1 Computational Complexity of Backward Refinement

The inclusion of backward refinement has major implications for the complexity of FSCA. The lowest complexity implementation is SPBR, which

involves a combinatorial search of similar complexity to the basic FSV algorithm (eqt. 4.64), the only difference being that $\mathbf{Z}$ is now a fixed size matrix, that is $\mathbf{Z}_{(i)} \rightarrow \mathbf{Z}_k^{(j)}(\mathbf{x}_i)$, where $\mathbf{Z}_k^{(j)}(\mathbf{x}_i)$ is as defined in eqt. (4.68). Since the covariance matrix, and hence the $\mathbf{q}_{j(i)}$ terms, will have already been precomputed for the forward selection step, the only concern is the development of an efficient recursive update procedure for the inverse matrix $(\mathbf{Z}_k^{(j)}(\mathbf{x}_i)^T\mathbf{Z}_k^{(j)}(\mathbf{x}_i))^{-1}$. This can be achieved by noting that

$$\mathbf{Z}_k^{(j)}(\mathbf{x}_i)^T\mathbf{Z}_k^{(j)}(\mathbf{x}_i) = \mathbf{Z}_k^T\mathbf{Z}_k + \mathbf{g}_{j(i)}\mathbf{e}_j^T + \mathbf{e}_j\mathbf{h}_{j(i)}^T, \tag{4.71}$$

where

$$\mathbf{g}_{j(i)} = \mathbf{Z}_k^T(\mathbf{x}_i - \mathbf{z}_j), \tag{4.72}$$

$$\mathbf{h}_{j(i)} = \mathbf{g}_{j(i)} + (\mathbf{x}_i - \mathbf{z}_j)^\mathsf{T}(\mathbf{x}_i - \mathbf{z}_j)\mathbf{e}_j. \tag{4.73}$$

It then follows, by application of the matrix inversion lemma [137], specifically the Sherman-Morrison formula [138], that

$$(\mathbf{Z}_k^{(j)}(\mathbf{x}_i)^T\mathbf{Z}_k^{(j)}(\mathbf{x}_i))^{-1} = \mathbf{A}_{j(i)} - \frac{\mathbf{A}_{j(i)}\mathbf{e}_j\mathbf{h}_{j(i)}^T\mathbf{A}_{j(i)}}{1 + \mathbf{h}_{j(i)}^T\mathbf{A}_{j(i)}\mathbf{e}_j}, \tag{4.74}$$

where

$$\mathbf{A}_{j(i)} = (\mathbf{Z}_k^T\mathbf{Z}_k)^{-1} - \frac{(\mathbf{Z}_k^T\mathbf{Z}_k)^{-1}\mathbf{g}_{j(i)}\mathbf{e}_j^T(\mathbf{Z}_k^T\mathbf{Z}_k)^{-1}}{1 + \mathbf{e}_j^T(\mathbf{Z}_k^T\mathbf{Z}_k)^{-1}\mathbf{g}_{j(i)}}. \tag{4.75}$$

This recursive inverse update can be computed in $\mathsf{O}(8k^2 + 4k + 6)$ flops and hence has $\mathsf{O}(8k^2)$ complexity, which compares favourably to the $\mathsf{O}(4k^2)$ complexity of the forward step inverse (eqt. 4.66). The overall additional complexity of executing SPBR is then $\mathsf{O}(2p^2k^3 + 8pk^3)$. In contrast, the recursive SPBR implementation contributes $\mathsf{O}(0.5p^2k^4 + 2pk^4)$ additional complexity. Since repetition of the MPBR loop is dependent on refinements taking place in the previous pass, the number of repetitions and hence overall algorithm complexity of the multi-pass implementations cannot be determined *a priori*. If we denote the average number of repetitions as λ then their complexity can be expressed as λ times the complexity of the corresponding SPBR and recursive SPBR implementations. The optional reordering step has $\mathsf{O}(2pk^3)$ complexity. Hence, the overall algorithm complexity of FSVA with Backward refinement is

$$\mathsf{O}(p^2nk^2 + (2\lambda + \frac{2}{3})p^2k^3) \tag{4.76}$$

for non-recursive implementations and

$$\mathsf{O}(p^2nk^2 + \frac{\lambda}{2}p^2k^4 + \frac{2}{3}p^2k^3 + 2\lambda pk^4) \tag{4.77}$$

for recursive implementations, where $\lambda = 1$ corresponds to SPBR and $\lambda > 1$ MPBR.

Empirically evaluated time performances for FSCA and its refined versions are reported in Example 4.7.5.

$[\mathbf{Z}_k, r_c]$=SPBR($\mathbf{X}$,$\mathbf{Z}_k$)
Input: Forward selected variables $\mathbf{Z}_k$, data matrix $\mathbf{X}$
1: $r_c = 0$ (refinement count)
2: **for** $j = 1$ **to** $k - 1$ **do**
3: $\quad i_j = \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{Z}_k^{(j)}(\mathbf{x}_i))\mathbf{X})$
4: $\quad$ **if** $\mathbf{z}_j \neq \mathbf{x}_{i_j}$ **then**
5: $\quad\quad r_c = r_c + 1$ (increment refinement count)
6: $\quad\quad \mathbf{Z}_k = \mathbf{Z}_k^{(j)}(\mathbf{x}_{i_j})$ (i.e. replace $\mathbf{z}_j$ with $\mathbf{x}_{i_j}$)
7: $\quad$ **end if**
8: **end for**
9: **if** $r_c > 0$ **then**
10: $\quad$ Repeat steps 3-7 for $j = k$
11: **end if**
12: **return** $\mathbf{Z}_k,\ r_c$

Pseudocode 4.7.1: Single-Pass Backward Refinement Algorithm

$[\mathbf{Z}_k]$=MPBR($\mathbf{X}$,$\mathbf{Z}_k$)
Input: Forward selected variables $\mathbf{Z}_k$, data matrix $\mathbf{X}$
1: $r_c = 1$ (refinement flag)
2: **while** $r_c > 0$ **do**
3: $\quad [\mathbf{Z}_k, r_c]$=SPBR($\mathbf{X}$,$\mathbf{Z}_k$)
4: **end while**
5: **return** $\mathbf{Z}_k$

Pseudocode 4.7.2: Multi-Pass Backward Refinement Algorithm

4.7.1.1 Refinement Step in the literature

The idea of improving greedy optimization algorithms with a refinement step is not new. A refinement step was used for supervised features selection in [139], [135] and [140]. In the first case the problem starts with K randomly selected variables and a refinement step is repeated T times. Each time l

$[\mathbf{Z}_k^o]$=ReOrder($\mathbf{X}$,$\mathbf{Z}_k$)
Input: Forward selected variables $\mathbf{Z}_k$, data matrix $\mathbf{X}$
1: $\mathbf{Z}_0^o = \emptyset$
2: **for** $j = 1$ **to** k **do**
3: $\quad i_j = \underset{\mathbf{z}_i \in \mathbf{Z}_k / \mathbf{Z}_{j-1}^o}{argmax} \; V_{\mathbf{X}}(\mathbf{\Phi}((\mathbf{Z}_{j-1}^o, \mathbf{z}_i))\mathbf{X})$
4: $\quad \mathbf{Z}_j^o = (\mathbf{Z}_{j-1}^o, \mathbf{z}_{i_j})$
5: **end for**
6: **return** $\mathbf{Z}_k^o$

Pseudocode 4.7.3: Modified FSV procedure for reordering variables following backward refinement

variables are replaced according to the magnitude of their regression coefficient. The algorithm is based on the idea that after T iterations the set of variables is stable. In the second case variables are recursively selected with a forward selection (FS) algorithm. Each time that a variable is added to the model a backward elimination (BE) step is performed to eliminate possible redundant variables. It is interesting to observe that both the FS and the BE steps are suboptimal and can both benefit from the application of a refinement step of the form discussed for FSCA. The third case uses similar ideas to the second one but it is based on an optimization procedure similar to the one proposed in [133].

4.7.2 Simulated Datasets

In this section various simulated datasets are used to highlight the differences between FSCA and FSCA with backward refinement. Comparisons are also made with PCA, Sparse PCA and OPFS where appropriate.

Example 4.7.1 (Four Distinct Variables)**.** As a first example we define four base variables $\mathbf{w}_0, \mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0 \sim N(0,1)$, 20 noise variables $\epsilon_1, \ldots, \epsilon_{20} \sim N(0, 0.1)$ and two larger noise variables $\epsilon_{21}, \epsilon_{22} \sim N(0, 0.4)$. These variables are used to generate a subset of variables similar to $\mathbf{w}_0$: $\{\mathbf{w}_i = \mathbf{w}_0 + \epsilon_i\}_{i=1,\ldots,5}$, a subset of variables similar to $\mathbf{x}_0$: $\{\mathbf{x}_i = \mathbf{x}_0 + \epsilon_{i+5}\}_{i=1,\ldots,5}$, a subset of variables similar to $\mathbf{y}_0$: $\{\mathbf{y}_i = \mathbf{y}_0 + \epsilon_{i+10}\}_{i=1,\ldots,5}$, a subset of variables similar to $\mathbf{z}_0$: $\{\mathbf{z}_i = \mathbf{z}_0 + \epsilon_{i+15}\}_{i=1,\ldots,5}$ and two additional redundant variables defined as $\mathbf{h}_1 = \mathbf{w}_0 + \mathbf{x}_0 + \epsilon_{21}$ and $\mathbf{h}_2 = \mathbf{y}_0 + \mathbf{z}_0 + \epsilon_{22}$. The complete dataset is then defined as $\mathbf{X} = [\mathbf{w}_0, \ldots, \mathbf{w}_5, \mathbf{x}_0, \ldots, \mathbf{x}_5, \mathbf{y}_0, \ldots, \mathbf{y}_5, \mathbf{z}_0, \ldots, \mathbf{z}_5, \mathbf{h}_1, \mathbf{h}_2]$, with $\mathbf{X} \in \mathbb{R}^{m \times 26}$. Hence, by design the dataset is highly redundant, with only 4 of the 26 variables independent. As such, the information it contains can be optimally summarized by the 4 base variables $(\mathbf{w}_0, \mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0)$. Table

4.3 shows the variance explained by PCA, OPFS, FSCA and the 4 backward refinement FSCA enhancements as a function of the number of selected variables k for an instance of the dataset with $m = 1000$, while Table 4.4 shows the sets of variables selected by FSCA, SPBR and OPFS for each value of k. Results for MPBR, R-SPBR and R-MPBR are omitted from Table 4.4 as they are identical to SPBR. When FSCA is applied to this dataset in general the first FSV will be $\mathbf{h}_1$ and the second will be $\mathbf{h}_2$, or viva versa, as dictated by the noise realization. Subsequent selections are then from among the base variables until at $k = 6$ all 4 based variables are selected. Hence, the initial selections become redundant as additional variables are added. As expected the backward refinement algorithms explain greater variance than FSCA when $k = 3, 4$ and 5. Note, that at $k = 4$ the refinement of the FSCA solution by SPBR identifies the optimum set of variables (i.e. the 4 base variables), hence there is no scope for further improvement by the more advanced refinement algorithms. For comparison purposes OPFS results are also presented in the tables. As can be seen, the variable selections for $k = 1$ and 2 are the same as FSCA, but thereafter OPFS takes a different path, and ends with a suboptimal solution at $k = 6$ (explained variance of 96% versus 99%). The performance of OPFS varies considerably for different instances of the dataset. This is illustrated in Figure 4.3, which shows the variation in performance of each method over 200 different dataset realizations with $m = 100$ when selecting $k = 4$ and 6 components. FSCA also shows considerable variation in performance but is in general superior to OPFS. A pairwise comparison of the variance explained by OPFS and FSCA over 1000 repetitions of the dataset shows that OPFS only outperforms FSCA 2% of the time. In contrast, SPBR and the other refinement algorithms are consistently superior to FSCA and OPFS and show little variation in performance over the different dataset realizations.

k	PCA	OPFS	FSCA	**SPBR**	**MPBR**	**R-SPBR**	**R-MPBR**
1	30.41	25.88	25.88	25.88	25.88	25.88	25.88
2	56.68	54.34	54.34	54.34	54.34	54.34	54.34
3	80.47	75.72	75.82	78.11	78.11	78.11	78.11
4	98.60	93.62	93.80	98.22	98.22	98.22	98.22
5	99.03	96.36	96.56	98.78	98.78	98.78	98.78
6	99.43	96.40	99.31	99.31	99.31	99.31	99.31

Table 4.3: Example 4.7.1: The percentage of explained variance achieved with PCA, OPFS, FSCA, and its backward refinement variants, for different numbers of selected components

k	FSCA	**SPBR**	OPFS
1	$\{\mathbf{h}_1\}$	$\{\mathbf{h}_1\}$	$\{\mathbf{h}_1\}$
2	$\{\mathbf{h}_1, \mathbf{h}_2\}$	$\{\mathbf{h}_1, \mathbf{h}_2\}$	$\{\mathbf{h}_1, \mathbf{h}_2\}$
3	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{x}_0\}$	$\{\mathbf{w}_0, \mathbf{h}_2, \mathbf{x}_0\}$	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{w}_0\}$
4	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{x}_0, \mathbf{z}_0\}$	$\{\mathbf{w}_0, \mathbf{y}_0, \mathbf{x}_0, \mathbf{z}_0\}$	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{w}_0, \mathbf{z}_0\}$
5	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{x}_0, \mathbf{z}_0, \mathbf{w}_0\}$	$\{\mathbf{y}_0, \mathbf{h}_1, \mathbf{x}_0, \mathbf{z}_0, \mathbf{w}_0\}$	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{w}_0, \mathbf{z}_0, \mathbf{y}_0\}$
6	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{x}_0, \mathbf{z}_0, \mathbf{w}_0, \mathbf{y}_0\}$	$\{\mathbf{x}_0, \mathbf{y}_0, \mathbf{w}_0, \mathbf{z}_0, \mathbf{h}_1, \mathbf{h}_2\}$	$\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{w}_0, \mathbf{z}_0, \mathbf{y}_0, \mathbf{w}_2\}$

Table 4.4: Example 4.7.1 Variables selected at each step by FSCA, SPBR and OPFS

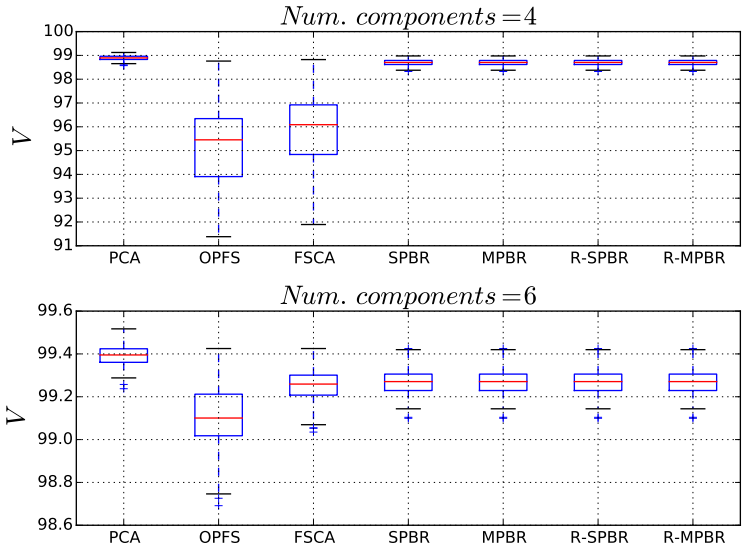


Figure 4.3: Example 4.7.1: Boxplots showing variation in performance of each method for $k = 4$ and 6 components, over 200 Monte Carlo repetitions

Example 4.7.2 (Block Redundancy)**.** In this example the dataset consists of a block of independent variables $\mathbf{X}^0$ augmented by a second block of noise perturbed redundant variables $\mathbf{X}^1$ generated as a linear combination of the variables in $\mathbf{X}^0$. In particular we define:

- $\mathbf{X}^0 \in \mathbb{R}^{n\times u}$: $\mathbf{X}^0_{i,j} \sim N(0,1)$
- $\phi \in \mathbb{R}^{u\times(p-u)}$: $\phi_{i,j} \sim N(0,1)$
- $\epsilon \in \mathbb{R}^{n\times(p-u)}$: $\epsilon_{i,j} \sim N(0,0.1)$
- $\mathbf{X}^1 = \mathbf{X}^0 \cdot \phi + \epsilon$
- $\mathbf{X} = [\mathbf{X}^0, \mathbf{X}^1]$

We generated 1000 instances of this dataset for different values of u and p for $n = 200$ and in each case computed a $k = u$ components FSCA, SPBR, R-SPBR, MPBR and R-MPBR. The variance explained by the k components selected by each algorithm averaged over the 1000 repetitions is reported in Table 4.5. The table also reports S_c, the percentage of true variables selected by each method, defined as

$$S_c = |\{\mathbf{z}_1, \ldots \mathbf{z}_u\} \cap \{\mathbf{x}_1, \ldots, \mathbf{x}_u\}|/u \times 100. \tag{4.78}$$

As expected, the introduction of a refinement step consistently increases the explained variance relative to FSCA with SPBR reducing unexplained variance by 50%-64% and MPBR reducing it by 58%-73%. There is no appreciable difference between the performance of the recursive and non-recursive implementations of each algorithm. A similar pattern is observed with respect to the number of true variables selected by each method, FSCA: 12%-22%, SPBR/R-SPBR: 35%-49% and MPBR/R-MPBR: 66%-82%.
Noting that PCA provides an upper bound on achievable explained variance for a given number of components, Figure 4.4 shows the variance explained by FSCA and the various backward refinement algorithms with $k = 1, 2, ..., 12$ selected components for the case where $u = 10$, $p = 30$ and $n = 1000$, expressed as a percentage of the variance explained by the equivalent number of PCs obtained using PCA. As can be seen, SPBR consistently provides improved performance over FSCA for $k > 1$. For values of k in the vicinity of the true dimensionality of the data, MPBR is marginally superior to SPBR (57.5% versus 49.2% reduction in unexplained variance at $k = 10$, 11.8% versus 9.7% at $k = 8$ and 27.0% versus 23.9% at $k = 12$). In general the improvement due to backward refinement decreases rapidly as k increases beyond the true dimensionality of the data.

Percentage of variance explained ($V_{\mathbf{X}}$)						
u	p	FSCA	**SPBR**	**MPBR**	**R-SPBR**	**R-MPBR**
10	30	99.75	99.87	99.89	99.88	99.89
15	50	99.77	99.89	99.92	99.90	99.92
20	75	99.78	99.90	99.94	99.90	99.94
25	100	99.78	99.91	99.94	99.91	99.94
Percentage of true variables selected (S_c)						
u	p	FSCA	**SPBR**	**MPBR**	**R-SPBR**	**R-MPBR**
10	30	22.38	48.80	70.11	47.73	66.30
15	50	16.03	43.73	74.20	41.98	72.06
20	75	14.70	41.60	81.66	45.11	79.82
25	100	12.85	34.74	71.46	38.21	71.95

Table 4.5: Example 4.7.2: Percentage variance explained ($V_{\mathbf{X}}$) and percentage of true variables selected (S_c) with FSCA and its backward refinement variants (averaged over 1000 repetitions)

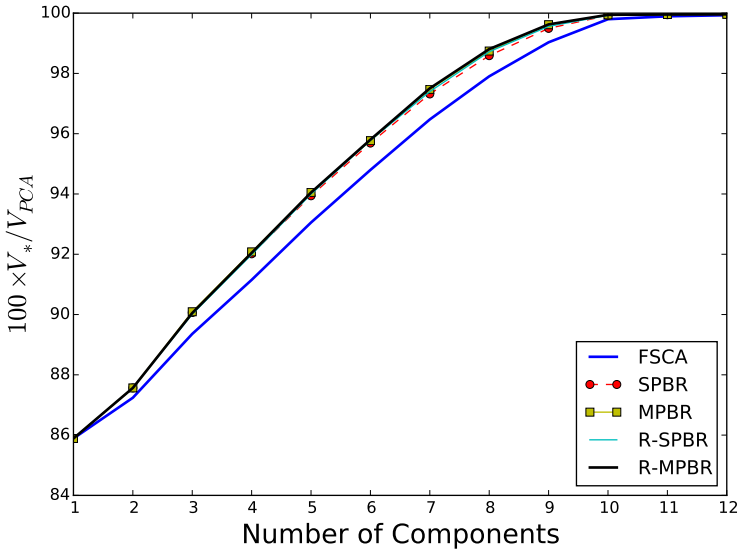


Figure 4.4: Example 4.7.2: The percentage of variance explained as a function of the number of selected components for $u = 10, p = 30$

i	PC_1	PC_2	SPC_1	SPC_2	FSC_1	FSC_2	$SPBR_1$	$SPBR_2$
1	-0.13	0.48		-80.00				
2	-0.13	0.48		-80.00				
3	-0.13	0.48		-80.00		1		1
4	-0.13	0.48		-80.00				
5	0.39	0.16	79.61				1	1
6	0.39	0.16	79.61					
7	0.39	0.16	79.61					
8	0.39	0.16	79.61					
9	0.41	0.01	77.43	3.09	1	1		
10	0.41	0.01	77.43	3.09				
$V_{\mathbf{X}}$	60.80	99.99	59.43	99.99	60.75	99.99	58.22	99.99

Table 4.6: Example 4.7.3: The 1st and 2nd loading generated by PCA and SPCA ($\lambda = 20$) and the 1st and 2nd FSC obtained with FSCA and SPBR

Example 4.7.3 (Sparse PCA Dataset)**.** This example is a simulated dataset used in [108] to assess the performance of the sparse PCA algorithm introduced therein. The dataset is generated by 3 hidden variables $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$

- $\mathbf{v}_1 \sim N(0, 290)$ $\mathbf{v}_2 \sim N(0, 300)$.
- $\mathbf{v}_3 = -0.3\mathbf{v}_1 + 0.952\mathbf{v}_2 + \epsilon$ where $\epsilon \sim N(0, 1)$.

and 10 observed variables

- $\mathbf{x}_i = \mathbf{v}_1 + \epsilon_i^1$ where $\epsilon_i^1 \sim N(0, 1)$ $for\ i = 1, \ldots, 4$
- $\mathbf{x}_i = \mathbf{v}_2 + \epsilon_i^2$ where $\epsilon_i^2 \sim N(0, 1)$ $for\ i = 5, \ldots, 8$
- $\mathbf{x}_i = \mathbf{v}_3 + \epsilon_i^3$ where $\epsilon_i^3 \sim N(0, 1)$ $for\ i = 9, 10$

The final data matrix $\mathbf{X} \in \mathbb{R}^{n \times 10}$ is then defined as $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_{10}]$, where $n = 1000$ is the number of samples. The results reported in Table 4.6, which are for a single realization of the dataset, show that both FSCA and SPBR with 2 components explain more than 99% of the total variance. In general, for other realizations FSCA will always select either $\mathbf{x}_9$ or $\mathbf{x}_{10}$ as one of the two variables. SPBR and MPBR will instead select one variable from the group generated by $\mathbf{v}_1$ and one of from the group generated by $\mathbf{v}_2$. PCA and SPCA assign similar values to similar variables due to the grouping effect. However, as a result they do not omit redundant variables. Thus, while SPCA yields sparse solutions it is not the optimal choice if the objective is to select a compact set of variables to represent the data.

4.7.3 Application Examples

In this section some applications of FSCA and its improved versions are presented.

Example 4.7.4 (Pitprops Dataset)**.** The pitprops dataset, originally introduced by [141] as a PCA case study, is a widely used benchmark problem for evaluating the performance of PCA and SPCA like methods. The dataset consists of 180 samples of 13 variables describing properties of timber and was used by the British Forestry Commission in a study to establish if homegrown timber had sufficient strength to be used to provide roof support struts 'Pitprops' for mines. Using the correlation matrix for the dataset provided in [141] we generated 180 samples of a multivariate normal distribution to synthesise an approximation of the original dataset. For the synthesised dataset six SPCA components were computed for a range of different values of the penalty λ. For each value of λ the number of uniquely selected variables was identified and then the corresponding number of FSVs computed with FSCA, SPBR, R-SPBR, MPBR and R-MPBR. The number of variables selected and the percentage of explained variance are reported in Table 4.7. In order to provide an upper bound for the percentage of explained variance that can be achieved we have also reported the corresponding values obtained with PCA. From the results it can be observed that SPCA is the method that explains the least variance for a given number of selected variables. In SPCA the number of selected variables is indirectly chosen trough a penalty parameter λ. In some situations it can be difficult to choose λ to select a specific number of variables. In our case, for example, it has not been possible to select $1, 2, 8$ or 13 variables using SPCA. In particular, observe that for 7 and 11 selected variables SPCA returns 2 different results. This is due to the fact that two different subsets of 7/11 variables were returned for two different values of λ. Table 4.8 lists the variables selected by a 6 components FSCA, SPBR, MPBR, R-SPBR and R-MPBR together with the set of 6 variables that maximize the percentage of explained variance (BEST). This set has been determined by evaluating all possible subsets of 6 variables. Observe that MPBR, R-SPBR and R-MPBR select the same variables as BEST while SPBR replaces the variable 'moist' with 'testsg'. In contrast, FSCA selects only 3 variables in common with BEST.

Example 4.7.5 (Plasma Etch OES Analysis)**.** In semiconductor manufacturing Optimal Emission Spectroscopy (OES) is increasingly used to monitor plasma etch processes. Due to the high dimensionality and correlated nature of OES data dimensionality reduction techniques such as PCA are usually employed as a pre-processing step (see for example [90], [112], [12] and [142]).

NCP	PCA	SPCA	FSCA	**SPBR**	**MPBR**	**R-SPBR**	**R-MPBR**
3	67.41	36.16	55.98	60.04	60.04	60.04	60.04
4	75.52	60.86	67.13	67.13	67.13	68.33	68.33
5	82.13	65.40	74.07	75.76	75.76	75.76	75.76
6	88.00	71.65	80.18	82.38	82.38	82.38	82.38
7	92.33	72.59	85.73	86.67	87.89	87.89	87.89
7	92.33	77.43	85.73	86.67	87.89	87.89	87.89
9	97.63	85.31	94.52	95.87	95.87	95.87	95.87
11	99.38	93.05	98.79	98.79	98.79	98.79	98.79
11	99.38	95.11	98.79	98.79	98.79	98.79	98.79
12	99.72	96.89	99.41	99.41	99.41	99.41	99.41
13	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 4.7: Example 4.7.4: Percentage of explained variance as a function of the number of variables selected for each algorithm

FSCA	**SPBR**	**MPBR**	**R-SPBR**	**R-MPBR**	BEST
whorls	ringbut	ringbut	ringbut	ringbut	ringbut
moist	moist	moist	moist	moist	moist
length	length	length	length	length	length
ringtop	clear	clear	clear	clear	clear
clear	bowmax	bowmax	bowmax	bowmax	bowmax
ovensg	ovensg	ovensg	ovensg	ovensg	ovensg
			$V_{\mathbf{X}}$		
80.18	82.38	82.38	82.38	82.38	82.38

Table 4.8: Example 4.7.4: The 6 variables selected by each algorithm and the optimum set of 6 variables (BEST) in terms of maximizing the percentage of explained variance in the dataset

Here we employ a sample OES dataset collected during the processing of a single wafer as a case study for comparing the orthogonal decompositions generated by FSCA and PCA. The OES spectrum in question, plotted in Fig. 4.5, consists of optical emission intensity time series data for each of the 2000 active spectrometer channels (each channel corresponds to a different optical wavelength in the range 192-875 nm). Each time series has 55 samples, hence the resulting dataset is a matrix $\mathbf{X} \in \mathbb{R}^{55\times 2000}$ of intensity values.

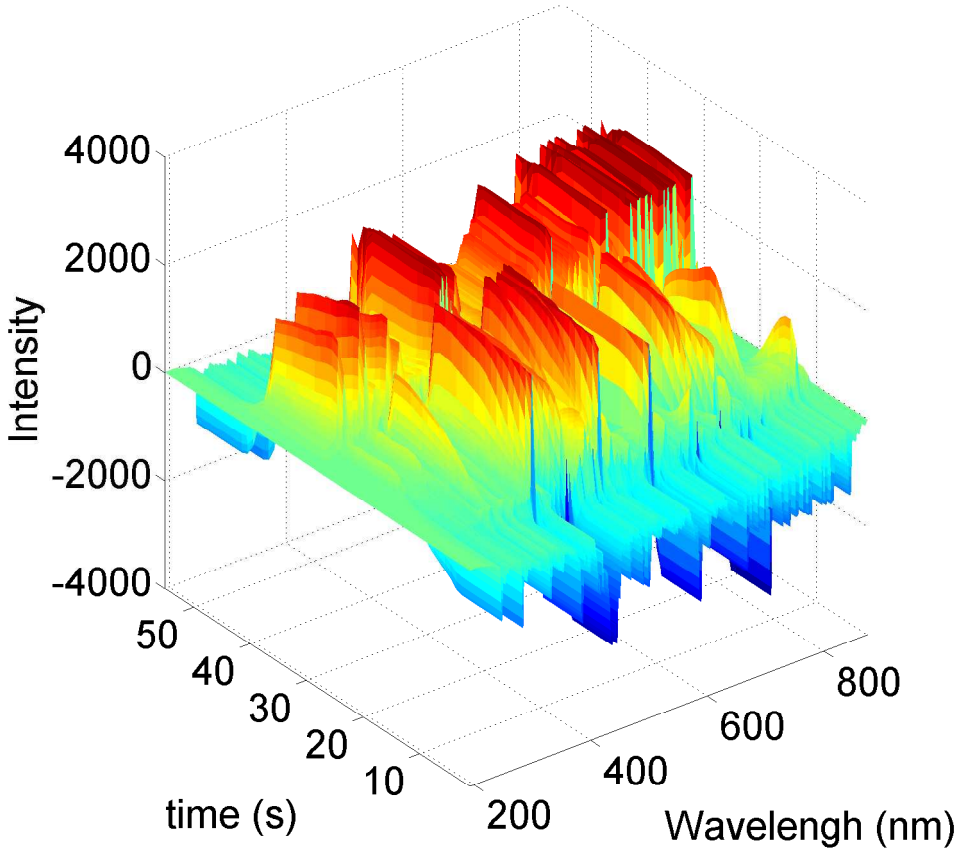


Figure 4.5: Plasma Etch Process OES Spectrum

The highly correlated nature of the data is evident from Table 4.9, which shows that the first 4 PCs and the first 4 variables selected by FSCA and its backward refinement variants explain more than 99% of the variation in data. In Fig. 4.6 the PCA loadings and scores are compared with the FSVs and

k	PCA	FSVA	**SPBR**	**MPBR**	**R-SPBR**	**R-MPBR**
1	77.04	76.63	76.63	76.63	76.63	76.63
2	96.53	96.00	96.37	96.37	96.37	96.37
3	98.20	98.00	98.14	98.14	98.14	98.14
4	99.47	99.27	99.42	99.42	99.42	99.42
5	99.69	99.50	99.65	99.65	99.65	99.65
6	99.81	99.66	99.75	99.79	99.79	99.79
7	99.88	99.79	99.85	99.86	99.87	99.86
8	99.93	99.89	99.91	99.92	99.92	99.92
9	99.95	99.93	99.94	99.95	99.95	99.95

Table 4.9: Example 4.7.5: Accumulative variance explained by PCA, FSCA and the four backward refinement variants of FSCA for different values of k

FSCs obtained with FSCA. This reveals that the etch process can be analysed using either PCA scores or FSCA components. In particular, observe that the FSCA components and PCA scores tend to have similar trends. As noted previously, the PCA scores are obtained as a linear combination of all 2000 original variables (as defined by the PCA loadings), while the four FSCs can be expressed as a linear combination of just 4 original variables (the FSVs). The benefit of being able to trace process variably back to a small number of OES wavelengths is that individual wavelengths map to specific chemical species present in the plasma, enabling process engineers to gain insight into the underlying drivers of process variability. The computation time in seconds for each algorithm in reported in Table 4.10 for different numbers of computed components/variables selected. As expected, computational time grows rapidly with increasing k for the more complex refinement algorithms. For example, while SPBR is only twice as computationally intensive as FSVA at $k = 9$ R-MPBR is more than 20 times more computationally intensive.

Example 4.7.6 (Wafer Site Optimisation)**.** As a final application example, we evaluate the performance of SPBR, MPBR, R-SPBR and R-MPBR as alternatives to FSCA for the semiconductor wafer metrology site optimisation methodology developed in [89]. The pertinent details are as follows. The objective of the methodology is to use historical metrology data for a set of candidate measurement sites to determine the minimum set of sites that need to be measured in order to accurately reconstruct wafer profiles. The case study dataset consists of production metrology data for a deposition process used in the manufacture of read-write heads, a key component of hard disk drives. The dataset, which was collected over several weeks from a single production tool for the process, contains measurements of 50 candidate sites

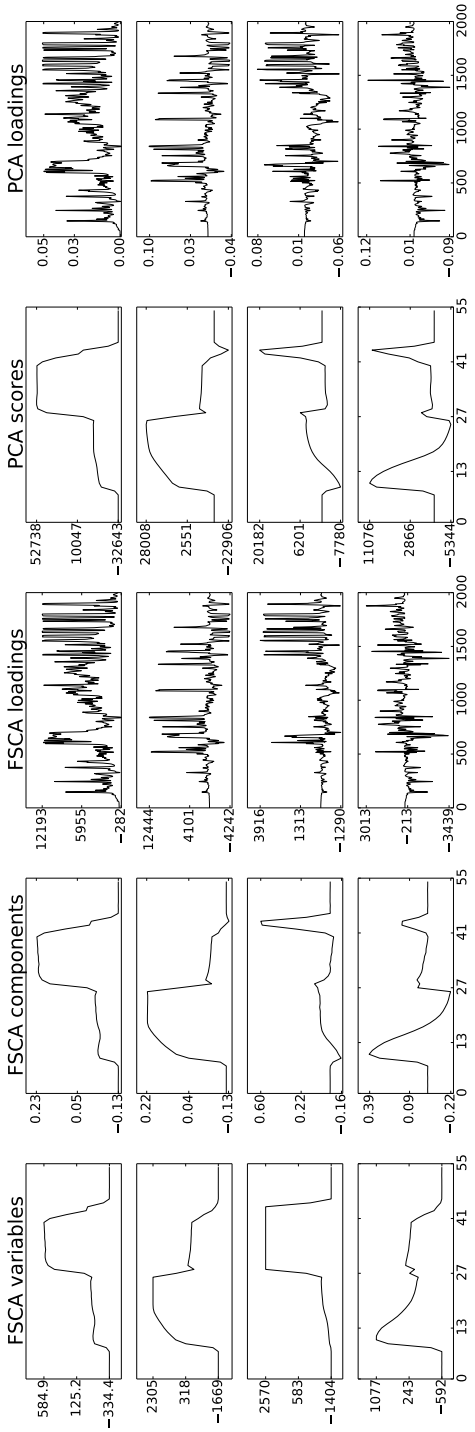


Figure 4.6: Example 4.7.5: The first 4 PCA and FSCA components and scores

k	PCA	FSVA	**SPBR**	**MPBR**	**R-SPBR**	**R-MPBR**
1	0.03	0.05	0.10	0.10	0.10	0.10
2	0.04	0.10	0.20	0.31	0.25	0.36
3	0.05	0.16	0.32	0.50	0.47	0.75
4	0.07	0.22	0.46	0.96	0.80	1.32
5	0.09	0.28	0.60	1.25	1.16	2.32
6	0.10	0.36	0.76	2.40	1.67	3.25
7	0.13	0.42	0.93	3.54	2.22	5.85
8	0.14	0.49	1.09	2.91	2.90	8.35
9	0.15	0.56	1.29	3.47	3.63	11.86

Table 4.10: Example 4.7.5. Computation time for PCA, FSVA and the four backward refinement variants of FSCA for different values of k

for 316 wafers. Hence, $\mathbf{X} \in \mathbb{R}^{316\times 50}$ and the site selection problem equates to selecting the subset of columns of $\mathbf{X}$ that best describe $\mathbf{X}$. For a detailed description of the problem statement, solution methodology and case study dataset the reader is referred to [89].

Here, FSCA and the newly proposed backward refinement variants are employed to determine the optimum subset of wafer sites. The percentage of variance explained by each method for different numbers of selected sites (k) is reported in Table 4.11. Defining 99% variance explained as the minimum reconstruction accuracy threshold, it follows that 7 sites are needed when using FSC, while 6 sites are sufficient when using SPBR, MPBR, R-SPBR and R-MPBR. The PCA results, which are also recorded in Table 4.11, show that the lower bound on the number of sites required is 5. Again in this example we observe that, as expected, SPBR out performs FSCA and MPRB outperforms SBBR (to a lesser extent), but that unlike the previous examples, R-SPBR and R-MPBR are sometimes marginally inferior to their non-recursive counterparts (i.e. for $k \geq 5$).
It is also interesting to observe how representative the FSCA selected sites are of the full wafer surface. This can be visualised by clustering the unmeasured sites in k clusters $\mathcal{C}_{\mathbf{z}_1}, \ldots, \mathcal{C}_{\mathbf{z}_k}$ according to their similarity to the k FSCA selected sites. Here, the similarly between an FSCA site, $\mathbf{z}_i$, and an unmeasured site, $\mathbf{x}_j$, is defined in terms of the impact on reconstruction accuracy of replacing $\mathbf{z}_i$ with $\mathbf{x}_j$. Specifically, denoting the k selected variables as $\mathbf{Z}_k = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$, and noting the definition of $\mathbf{Z}_k^{(i)}(\mathbf{x}_j)$ given in eqt.

	PCA	FSCA	**SPBR**	**MPBR**	**R-SPBR**	**R-MPBR**
1	42.69	38.84	38.84	38.84	38.84	38.84
2	68.69	64.44	67.01	67.01	67.01	67.01
3	85.72	82.59	84.57	85.08	84.79	85.08
4	98.48	96.37	97.40	97.58	97.58	97.58
5	**99.12**	97.59	98.63	98.74	98.72	98.73
6	99.47	98.76	**99.19**	**99.20**	**99.17**	**99.16**
7	99.67	**99.22**	99.45	99.46	99.42	99.39
8	99.75	99.47	99.60	99.60	99.59	99.58
9	99.81	99.64	99.71	99.71	99.71	99.69
10	99.86	99.72	99.77	99.80	99.78	99.79

Table 4.11: Example 4.7.6: The percentage of variance explained by the various methods for different values of k, the number of selected variables

(4.68), sites are assigned to clusters according to the rule

$$\mathbf{x}_j \in \mathcal{C}_{\mathbf{z}_i} \text{ if } i = \underset{p=1\ldots k}{argmax}\ V_{\mathbf{X}}(\mathbf{\Phi}(\mathbf{Z}_k^{(p)}(\mathbf{x}_j))\mathbf{X}) \tag{4.79}$$

Fig. 4.7 shows the clusters obtained with each of the FSCA algorithms for $k = 4$ and $k = 8$. The clusters are represented by markers of different colour and/or shape. Of particular note is the variation in spatial consistency of clusters. It is apparent that the refinement steps yield much better spatial consistency of clusters than FSCA, especially when $k = 8$. Contrast, for example, the FSCA and R-MPBR plots. With FSCA a number of clusters have sites which are distributed in a disjoint fashion across the whole wafer surface, while with R-MPBR the clusters are localised to particular regions of the wafer surface. Note, in particular, that in the FSCA plot the 'black star' site is clustered with only one other site which is in a spatially unrelated area of the wafer. These anomalies are a consequence of the sites initially selected by FSCA becoming redundant as additional sites are selected, as discussed in Section 4.7. This issue, which detracts from the interpretability of clusters, is addressed through the introduction of the backward refinement step. As expected, the most consistent clusters are obtained when using MPBR and R-MPBR.

Example 4.7.7 (ER Prediction)**.** In this example FSCA and its refined version are used as preprocessing step to select regressors for a regression model to predict a target output. The dataset is a reduced version of the J2M data consisting of an input matrix $\mathbf{X} \in \mathbb{R}^{2194\times 200}$ and an output vector

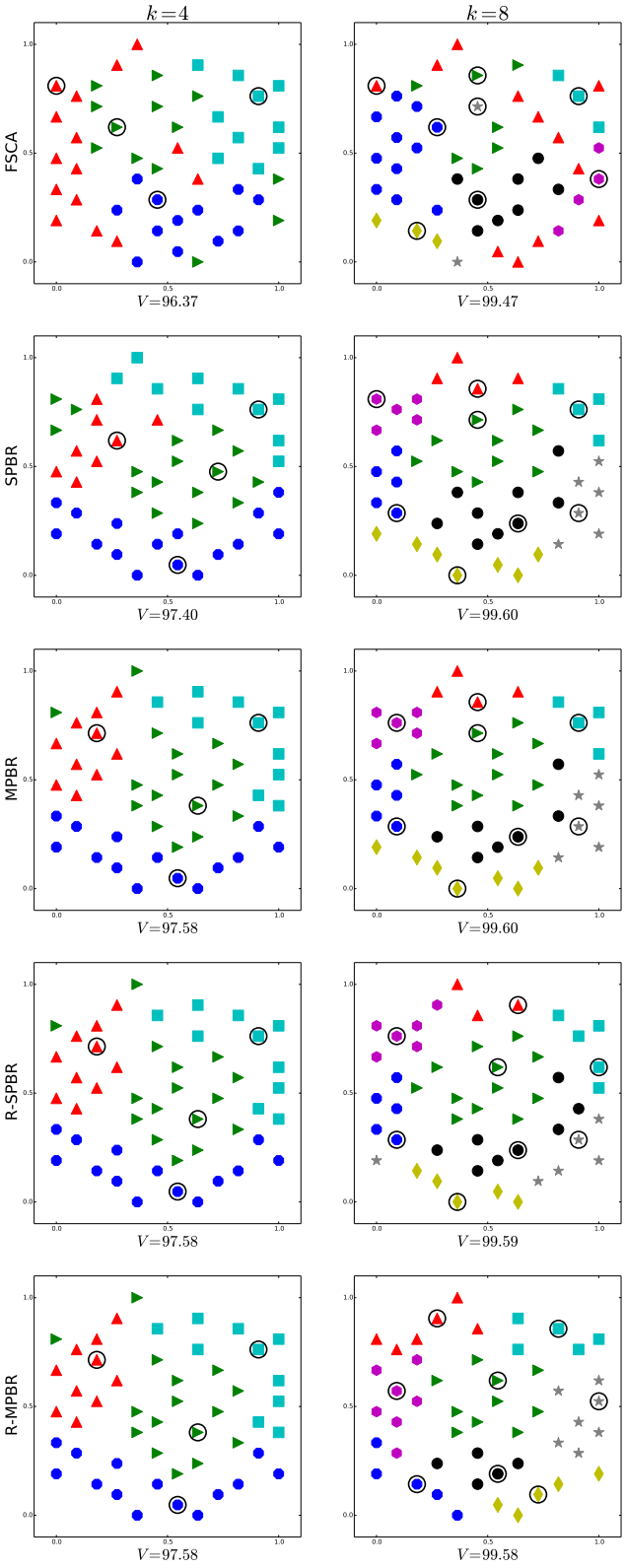


Figure 4.7: Example 4.7.6: The FSCA clusters obtained with FSCA, SPBR, MPBR, R-SPBR and R-MPBR for $k = 4$ and $k = 8$. In each case the FSCA selected sites are indicated by circles and the associated clusters by markers of different colour and/or shape. The percentage variance explained by the different algorithms is reported under each plot.

$\mathbf{y} \in \mathbb{R}^{2194}$. The data is randomly split into a training dataset and in a test set dataset corresponding respectively to 70% and 30% of the data. The data is scaled in order to have zero mean and unit variance on the training data. Then K components are selected using the training dataset with FSCA and its refined versions. These components are then used to build a linear regression model also estimated using the training data. The prediction performance of the models are then evaluated on the test dataset. The process is repeated 20 times. The mean and the standard deviation of the prediction error and the mean of the percentage of explained variance on the training data is reported in Table 4.12. It is easy to observe that all the refined methods have a lower prediction error than FSCA when $K = 5$ components are used. FSCA and SPBR have the same performances when $K = 10$ while the more complex refinement methods have a lower prediction error. All the methods have roughly the same performances when $K = 15$. This seems to be compatible with the fact that all the methods express roughly the same amount of variance.

	FSCA	**SPBR**	**R-SPBR**	**MPBR**	**R-MPBR**
K			Mean		
5	0.341	0.315	0.315	0.315	0.315
10	0.289	0.291	0.282	0.291	0.277
15	0.260	0.265	0.269	0.265	0.271
K			Std		
5	0.025	0.022	0.022	0.022	0.021
10	0.023	0.029	0.030	0.031	0.028
15	0.017	0.019	0.018	0.018	0.018
K			Explained Variance		
5	96.16	96.74	96.74	96.74	96.74
10	98.80	99.04	99.06	99.07	99.07
15	99.57	99.68	99.70	99.71	99.70

Table 4.12: Example 4.7.7: The mean and the standard deviation of the prediction error and the mean of the percentage of explained variance on the train set is reported. These values were estimated with 20 bootstrap iterations for the problem described in Example 4.7.7

4.7.4 Discussion

This section has sought to provide a comprehensive presentation of Forward Selection Component Analysis, as the unsupervised counterpart of Forward Selection Regression and an alternative to PCA for dimensionality reduction and variable selection in large highly correlated datasets. A number of alternative FSCA algorithm implementations have been proposed, namely FSCA and FSVA, with and without pre-computation of the covariance matrix, and their computational complexity analysed. In particular, this analysis reveals that:

- All algorithms scale linearly with the number of measurements m and quadratically with the number of variables v.
- FSCA implementations grow linearly with the number of selected variables k, while FSVA implementations grow cubically with k.
- The optimum choice of implementation is dependent on the ratio $k/\sqrt{n}$.

In general, it is computationally advantageous to pre-compute and store the covariance matrix when using either FSCA or FSVA, with FSVA computationally the most efficient implementation provided $k/\sqrt{n} < \sqrt{1.5}$. FSCA without pre-computation of the covariance matrix is the superior implementation when $k/\sqrt{n} > \sqrt{3}$.

A number of novel backward refinement variants of FSCA have also been proposed and efficient algorithm implementations developed. Results from simulated and application case studies confirm that the refinements yield improvements in performance relative to FSCA in terms of variance explained for a given number of components/variables selected, better variable selection and, in the case of the wafer site optimisation problem, more coherent FSCA clusters. Overall the key observations are that MPBR is superior to SPBR, which is, in turn, superior to FSCA, and that there is little, if any, benefit to be gained from employing the recursive formulations (R-SPBR and R-MPBR) over their non-recursive counterparts. Indeed, in some instances the recursive implementations can yield poorer results.

In terms of computational complexity the ordering is FSCA $<$ SPBR $<$ MPBR $<$ R-SPBR $<$ R-MPBR, with SPBR having the same asymptotic complexity as FSCA. It is also noteworthy that the largest relative improvement in performance in terms of variance explained occurs with the change from FSCA to SPBR. As such, for most practical applications SPBR is rec-

ommended as it provides a good balance between complexity and quality of results.

4.8 Conclusion

In this chapter the unsupervised variable selection problem is introduced. It is shown that feature selection and dimensionality reduction, share the same structure. Indeed variable selection is equivalent to dimensionality reduction with the restriction that the lower dimensional approximation of the data must be obtained using only a subset of the original variables. In the chapter several variable selection algorithms are presented and FSCA is described in detail and used as an example methodology. It is in particular shown that the performance of the FSCA algorithm improves if a refinement step is introduced. On the application side it is shown that unsupervised features selection algorithms become particularly helpful when used on high dimensional datasets composed of highly correlated variables, for example OES data. Indeed these algorithms can be used to obtained a more compact and easy to understand representation of the data that is a good approximation of the original one.

Chapter 5

Nonlinear Unsupervised Feature Selection

5.1 Introduction

In chapters 3 and 4 the problem of supervised and unsupervised feature selection is introduced. In these chapters the relationship between the inputs and outputs or between the data and a subset of its variables is considered to be linear as respectively defined in equations 3.1 and 4.10. A natural extension of linear variable selection is the nonlinear variable selection (NVS) problem. Supervised NVS is equivalent to the model identification problem. In this sense several algorithms have already been proposed. Some examples are [143] where variables that are not related to the output are detected through the use of random forest or [144] where the number of nodes of a neural network is reduced with a pruning strategy. On the unsupervised side many nonlinear dimensionality reduction approaches exist. For example, several non-linear extensions of PCA have been proposed in the literature, such as [145] and [146]. In the former, the dimensionality of the data is reduced by using a neural network autoencoder, while in the latter a nonlinear kernel is employed to translate nonlinear PCA in the feature space into linear PCA in a high dimensional kernel space. Despite this, unsupervised nonlinear variable selection (UNVS) is a relatively new problem. Some recent contributions are [147], where variables are selected based on a multivariate version of the CART algorithm and [148] where variables are selected according to an evolutionary search. Both these algorithms are designed as a pre-processing step in order to optimize an unsupervised procedure such as clustering. In this thesis, instead, features are selected with the aim of obtaining a good reconstruction of the original data. Hence, the focus of this chapter is on investigating

UNVS methodologies that can lead to more compact data representations than conventional FSCA or PCA which assume linear relationships between variables. While several nonlinear data-driven algorithms exist, in this chapter particular focus is on linear-in-the-parameters models as they can be used for multi-output regression.

In the first sections of the chapter an introduction to neural networks and extreme learning machine is provided. A methodology for training ELMs is described which enables ELM models to be estimated with minimal user intervention. In particular, a new strategy for determining the distribution of the random hidden layer weights in ELMs is proposed that, for the industrial case study considered, enables improved and more consistent prediction performance when compared with lasso and ridge estimators. As such, ELMs offer a promising alternative to conventional linear regression for ER prediction and similar VM applications. Then FSCA and FSV algorithms are extended to nonlinear scenarios with ELMs used to perform the nonlinear mappings. It is shown that if nonlinearities are taken into account the two methods are not equivalent anymore. A data decomposition and reconstruction paradigm is introduced as a nonlinear extension of the one proposed in Section 4.3 of Chapter 4. Based on this several nonlinear variables selection algorithms are proposed and compared. The proposed methods outperform the linear FSCA and FSV and are often better than PCA.

5.2 Neural Network

Artificial Neural Networks (ANN) are a family of models inspired by biological neural networks. They are generally presented as systems of interconnected "neurons" (node) which exchange messages between each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning. Often an ANN can be represented as a function

$$\vec{\mathbf{y}} = f(\vec{\mathbf{x}}) \tag{5.1}$$

that maps the input variables $\vec{\mathbf{x}} \in \mathbb{R}^p$ into the output $\vec{\mathbf{y}} \in \mathbb{R}^o$. In an ANN the p nodes containing the input variables are called input nodes and the nodes containing the o outputs are called the output nodes. All the other nodes are called hidden nodes. In Figure 5.1 a graphical representation of a neural network is shown. Usually the nodes in an ANN are grouped in layers, as illustrated in Figure 5.2. The layer containing the input nodes is called the

input layer, the ones containing the output nodes is the output layer and the ones in the middle are the hidden layers.

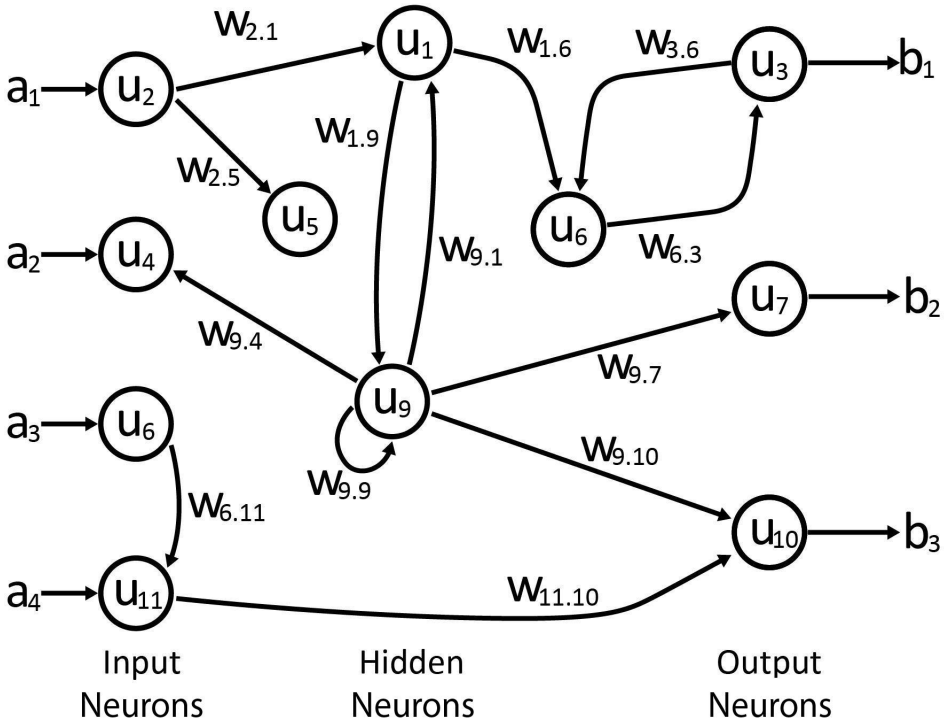


Figure 5.1: A graphical representation showing the typical structural features and information flow in neural networks.

5.2.1 Feedforward Neural Network

The feedforward neural network (FNN, [149]) was the first and simplest type of ANN devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) to the output nodes. There are no cycles or loops in the network. In a neural network the value of a neuron u is defined by the input that it receives from the connected neurons $v_1, \ldots, v_k$. The value of u is then defined as:

$$u = \sigma\left(\sum_{i=1}^{k} v_i w_i\right) \tag{5.2}$$

where w_i is the weight of the link between node u and node v_i and σ is called the activation function. The structure of an ANN is then defined by the

number of nodes, their connections and the activation function of each node. The weights $W = \{w_i\}_{i=1,\dots,n_c}$, where n_c is the number of connections in the network, are the free parameters of the neural network and they are tuned in order to obtain a good approximation of the data.
Indeed, given a training dataset

$$(\mathbf{X}, \mathbf{y}) \in \mathbb{R}^{n \times p} \times \mathbb{R}^{n \times o} \tag{5.3}$$

the weights are estimated as:

$$\hat{W} = \underset{W}{\operatorname{argmin}} \sum_{i=1}^{n} \parallel \vec{\mathbf{y}}_i - f_W(\vec{\mathbf{x}}_i) \parallel_2^2 \tag{5.4}$$

In general training a neural network is a very complex task. This is due to the fact that the cost function $\sum_{i=1}^{n} \parallel \vec{\mathbf{y}}_i - f_W(\vec{\mathbf{x}}_i) \parallel_2^2$ is not convex as a function of W. Indeed neural networks have often numerous local minima and saddle points.

5.2.1.1 Single Hidden Layer Feedforward Network

One of the the simplest and most popular versions of FNN is the Single Hidden Layer Feedforward Network (SLFN). The SLFN is composed of an input layer, a single hidden layer and an output layer. In SLFNs each node is connected with all the nodes in the previous layer. A SLFN with p inputs, l hidden nodes and k outputs, is mathematically represented by a function:

$$f(\vec{\mathbf{x}}) = \mathbf{g}_2(\mathbf{b}_2 + \mathbf{W}_2^T \mathbf{g}_1(\mathbf{b}_1 + \mathbf{W}_1^T \vec{\mathbf{x}})) \tag{5.5}$$

where $\mathbf{W}_1 \in \mathbb{R}^{p \times l}$ is the matrix of the weights connecting the inputs and the hidden layers, $\mathbf{g}_1$ is the activation function of the neurons in the hidden layer, $\mathbf{W}_2 \in \mathbb{R}^{l \times k}$ is the matrix of the weights between the hidden and output layers and $\mathbf{g}_2$ is the activation function of the output layer. In general the weight matrix between layer r and $r+1$ is defined as $\mathbf{W}_r = \{w_{i,j}^r\}$ where $w_{i,j}^r$ is the weight between the i-th neuron in layer r and j-th node in the layer $r+1$. An example of an SLFN with $p = 4$, $l = 5$ and $k = 1$ is illustrated in Figure 5.2. The performance of neural networks are strictly related to their weights $\hat{W}$ defined as the solution of the optimization problem reported in equation 5.4. This optimization problem is very challenging for neural networks with several hidden layers with the result that most applications have focused on SLFN. Recently new methods for training multi-layer ANNs have been proposed [145], [150] making research on this topic more popular again. In [145] the focus was on ANNs with several hidden layers and resulted in the birth of a new field called deep learning. In deep learning a huge amount of data and

high computing power is often required. In semiconductor manufacturing the data is usually scarce and our computational resources are very limited. For these reasons only a few examples of deep learning will be presented. In [150] the Extreme Learning Machine (ELM) is proposed as a new method for training single hidden layer neural networks which makes training easy and efficient. ELMs are described in detail in Section 5.2.2 and some applications to semiconductor manufacturing are presented.

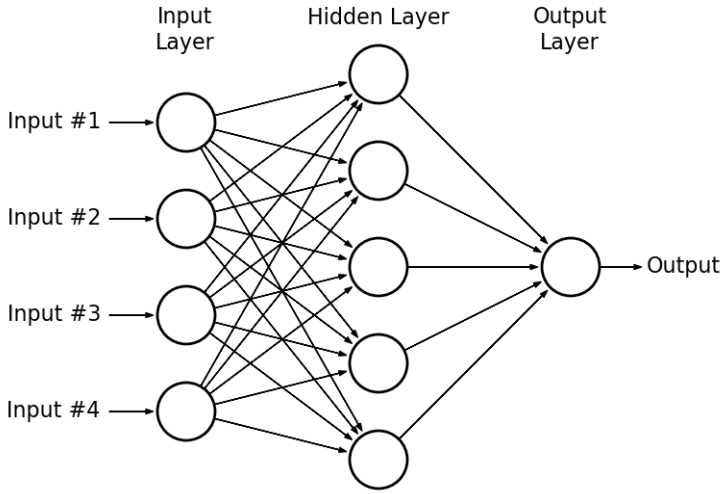


Figure 5.2: An example of SLFN with four input variables and one output.

5.2.1.2 Features Selection with Neural Network

Neural networks are much more complex structures than linear models. In linear models variable selection is based around a few simple ideas as described in chapters 3 and 4. Several algorithms for variables selection with neural network are presented in the literature. The majority of these algorithms are designed for supervised learning i.e. given an input matrix $\mathbf{X}$ and an output $\mathbf{y}$ the aim is to find the variables in $\mathbf{X}$ that can best estimate $\mathbf{y}$. Some examples are: [151] where both the structure of the network and its weights are optimized with a genetic algorithm. This automatically leads to a variable selection system as there is a strict connection between the structure of the neural network and the input variables considered; [152] where the variables are ranked according to the sensitivity of the outputs for each input; [153] where features are selected penalizing the weights between the input

layer and the first hidden layer in a lasso like fashion; [154] where features are recursively added according to the minimization error on a validation set and each time the network is only partially trained; and [155] where the ant algorithm is used to select the optimal set of variables.

Neural networks can easily handle multi-output regression problems. This is obtained using an output layer composed of a different neuron for each target instead of a single output node. Some examples of multi-output neural networks are reported in [156] and [157]. It follows that all these methods can be used for unsupervised feature selection if the data $\mathbf{X}$ is used both as input data and target output data. In general training a neural network is a difficult task. The training problem becomes particularly difficult if, in addition to the weights, the set of input variables also needs to be optimized. In semiconductor manufacturing it is important to have systems that can run with minimal user intervention. With this in mind the focus here is on algorithms where variables are iteratively added or removed to the model as in the linear models described in chapter 4. In addition, the problem of training neural networks is simplified through the use of Extreme Learning Machine neural networks (ELM, [158]) described in the next section.

5.2.2 Extreme Learning Machines

In a SLFN weighted connections between the layers propagate information through the network, and through appropriate training of these weights the neural networks can learn desired input-output mappings. Various approaches have been proposed to train feedforward neural networks, most of which take advantage of the natural formulation of training as an optimisation problem, for example the classical backpropagation [159], Levenberg-Marquardt [160] and hybrid BFGS training algorithms [161]. However, in general even with the most advanced algorithms, training is computationally time-consuming and not guaranteed to converge to suitable values, due primarily to the challenges posed by the nonlinear hidden layer weights. Alternatively in [162], a method that was recently renamed Extreme Learning Machine (ELM, [150]), has been proposed as a new algorithm to train single hidden layer feed forward neural networks (SLFN), where only the linear output layer weights are optimised. In contrast to the common belief that all network weights need to be optimized, with ELMs it has been shown that the hidden layer weights can be chosen randomly, leaving only the output layer weights to be optimized.

Algorithm 5.2.1. Consider an input matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ and an output vector

$\mathbf{y} \in \mathbb{R}^{n \times o}$ defined by n distinct samples $(\vec{\mathbf{x}}_i, \vec{\mathbf{y}}_i) \in \mathbb{R}^p \times \mathbb{R}^o$, each one composed of p input and o target variables. A SLFN with L hidden nodes and a linear output layer activation function is mathematically represented as:

$$f_L(\vec{\mathbf{x}}_j) = \sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \cdot \vec{\mathbf{x}}_j + b_i) \tag{5.6}$$

where $g(x)$ is the activation function and $\mathbf{w}_j$ is the j^{th} column of the weights matrix $\mathbf{W} = \{w_{i,j}\}_{i=1\ldots,n\ j=1,\ldots,L} \in \mathbb{R}^{p \times L}$ and $w_{i,j}$ is the weight of the connection between node i of the input layer and node j of the hidden layer in the SLFN. The previous equation can be expressed in matrix form with an abuse of notation as:

$$f_L(\mathbf{X}) = \mathbf{H}\beta \tag{5.7}$$

where $\mathbf{H}$ is defined as:

$$\mathbf{H} = \mathbf{H}(\mathbf{W}, \mathbf{b}, \mathbf{X}) \in \mathbb{R}^{n \times L} : \ \mathbf{H}_{i,j} = g(\mathbf{w}_j \cdot \vec{\mathbf{x}}_i + b_j) \tag{5.8}$$

and $\mathbf{b}$ is defined as $\mathbf{b} = \{b_i\}_{i=1,\ldots,L}$. In [150] the authors demonstrate that universal approximation capabilities can be achieved by randomly choosing the weights of the hidden layer $\mathbf{W}$ leaving only the output layer weights to be optimised. Hence, training reduces to estimating the linear model:

$$\hat{\boldsymbol{\beta}} = \underset{\beta}{\operatorname{argmin}} \parallel \mathbf{y} - \mathbf{H}\boldsymbol{\beta} \parallel_2 \tag{5.9}$$

This has a closed form solution given by the Moore-Penrose generalized inverse [163]:

$$\hat{\boldsymbol{\beta}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y} \tag{5.10}$$

and hence

$$\hat{\mathbf{y}} = \mathbf{H}\hat{\boldsymbol{\beta}} \tag{5.11}$$

From the algorithm it is easy to observe that an ELM is simple to implement and computationally efficient. In addition in [164], [165] and [166] provide proof of its modelling capabilities. While in this thesis only the original version of the ELM is considered, some generalizations are reported in [167] where locally connected networks are considered and [168] and [169] where pruning is incorporated to improve network performance.

Example 5.2.1. In order to show the properties of the ELM algorithm the method is applied to a simple example. Let $\mathbf{x}$ be a one dimensional vector, $\mathbf{x} = (x_1, \ldots x_{1000})$, of equally spaced points, where $x_1 = -10$, $x_{1000} = 10$ and $x_k = x_1 + 0.02k$, and define

$$y_i^0 = \frac{sin(x_i)}{x_i} \ if \ x_i \neq 0, \quad y_0 = 1 \ if \ x_i = 0 \tag{5.12}$$

and

$$y_i = y_i^0 + \epsilon \; where \; \epsilon \sim N(0, 0.5), \quad \mathbf{y} = (y_1, \ldots, y_{1000}) \tag{5.13}$$

The number of nodes in the hidden layer is set to $L = 20$ and random values are assigned to the matrices $\mathbf{W}$ and $\mathbf{b}$.

$$b_j, w_{i,j} \sim Unif(-0.12, 0.12) \; i.i.d. \tag{5.14}$$

The non-linear activation function g is defined as

$$g(x) = \frac{1}{1 + e^{-x}} \tag{5.15}$$

From the results reported in Figure 5.3 it is clear that the estimated ELM model is able to capture the non-linear relationship between the input and output and provides a good reconstruction of $\mathbf{y}$.

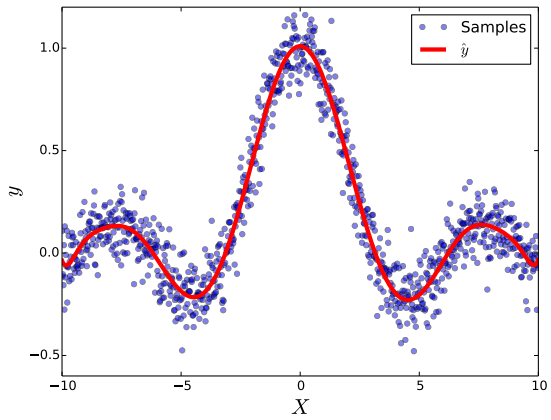


Figure 5.3: The input data (samples) and the estimated output function $\hat{\mathbf{y}}$ obtained with an ELM model with 20 neurons

5.2.2.1 Parameter tuning

In practical applications in manufacturing and other time-critical environments there is a need for data analysis techniques that require minimum user intervention. Considering this requirement, Lasso and Ridge regression are very good choices because they only require the tuning of a single penalty value and this can easily be achieved through an automated Cross-Validation procedure [60]. Neural networks generally required tuning of multiple parameters. Indeed, even with ELMs various parameters such as the number

of hidden nodes in the hidden layer (L), the activation function g, and the random distribution of the weights matrix $\mathbf{W}$ need to be selected. In this section we will show that it is possible to set these parameters in order to obtain a totally automated estimator.

5.2.2.2 Number of Hidden Nodes and Network Structure

In general the performances of a neural network and the family of functions that it is able to approximate are strongly dependent on the structure of the network. The structure of the SLFN, defined in equation 5.6, is only dependent on the number of nodes in the hidden layer L and on the activation function g.

Activation Function Various activation functions have been proposed in the literature. Among these, the most popular are radial basis functions (RBF) and the sigmoid function [170]. Here we consider ELMs with sigmoid activation function i.e.

$$g(x) = \frac{1}{1 + e^{-x}} \tag{5.16}$$

Number of hidden nodes In ELMs, once the activation function g has been chosen the only missing parameter is L: the number of hidden nodes. In [171] the authors show that it is possible to increase the prediction performance of ELMs if the number of nodes in the hidden layer L is set to a random large number and the linear model in equation 5.9 is replaced with the lasso estimator

$$\underset{\beta}{\operatorname{argmin}} \parallel \mathbf{y} - \mathbf{H}\boldsymbol{\beta} \parallel_2 + \lambda \parallel \boldsymbol{\beta} \parallel_1 . \tag{5.17}$$

As previously highlighted the lasso estimator is a sparse estimator, i.e. it selects only a subset of the columns of $\mathbf{H}$ when generating the regression model. Observing that each node in the hidden layer corresponds to a column of the matrix $\mathbf{H}$, this is equivalent to automatically selecting the number of nodes in the hidden layer. Again to achieve an automated solution we will tune the regularization parameter λ with a 10-fold Cross-Validation procedure as described in equation 3.5 (Chapter 3).

5.2.2.3 Choosing the weights

While in ELMs the weights are usually chosen randomly without any indication as to the range of the uniform distribution it is shown in [172] that the

size of the weights has a great influence on the performance of a neural network. The data that we will considered in this thesis is usually characterized by high dimension $p >> 0$ and so it is very likely that

$$\sum_{k=1}^{p} x_{i,k} w_{j,k} + b_j >> 0 \tag{5.18}$$

At the same time the sigmoid and Radial Basis Function (RBF, [173]) activation functions show variation with respect to the input only in a compact interval i.e.

$$\forall\ \epsilon > 0\ \ \exists\ c_\epsilon\ \ \max_x g(x) - \min_x g(x) < \epsilon\ \ if\ |x| > c_\epsilon \tag{5.19}$$

as can be observed in Figure 5.4 for the sigmoid and RBF function. We will now propose a method to ensure that the norm of the matrix $\mathbf{H}$ is not too small and we will show that this leads to an improvement in the performance of the ELM model. Let $\vec{\mathbf{x}}_i = (x_{i,1}, \ldots, x_{i,p})$ be an input sample. A generic element of $\mathbf{H}$ associated with this sample is then defined as:

$$\mathbf{H}_{i,j} = g(\sum_{k=1}^{p} x_{i,k} w_{j,k} + b_j) \tag{5.20}$$

and $\forall \epsilon > 0$:

$$\sum_{j=1}^{L} Var(\mathbf{H}_{.,j}) \geq L\epsilon \Leftrightarrow \left| \sum_{k=1}^{p} x_{i,k} w_{j,k} + b_j \right| < c_\epsilon \tag{5.21}$$

It follows that the matrix $\mathbf{H}$ is strongly influenced by the norm of the weights $\mathbf{W}$. If $\mathbf{W}$ and $\mathbf{b}$ are both drawn from a uniform distribution $Unif(-c, c)$, then the value c can be used to control $\parallel \mathbf{W} \parallel_2$ and hence the norm of $\mathbf{H}$. Observe that if c is large the elements of $\mathbf{H}$ tend to be small while if c is small the variance of $\parallel \mathbf{W} \parallel_2$ is small as well. In both cases the approximation capability of the ANN is reduced, as shown in Figure 5.5. Only when the appropriate value of c is selected does the neural network achieve a good approximation of the data. Renaming the neural network $f_L(\vec{\mathbf{x}}_j)$ in equation (5.6) as $f_L^c(\vec{\mathbf{x}}_j)$ to reflect the dependency on c, the optimum choice of c can be determined as

$$\hat{c} = \underset{c}{\operatorname{argmin}} \sum_{j=1}^{n} \parallel y_j - f_L^c(\vec{\mathbf{x}}_j) \parallel_2 \tag{5.22}$$

where the minimization is performed over the training data. Note, that this should not lead to over fitting because the weights are still chosen randomly.

Figure 5.6 shows the training error and the prediction error for various values of c computed on two independent subsets of the J2M data as described in the next example (Example 5.2.2). We can observe that training and prediction errors have similar trends if defined as a function of c. In the next example

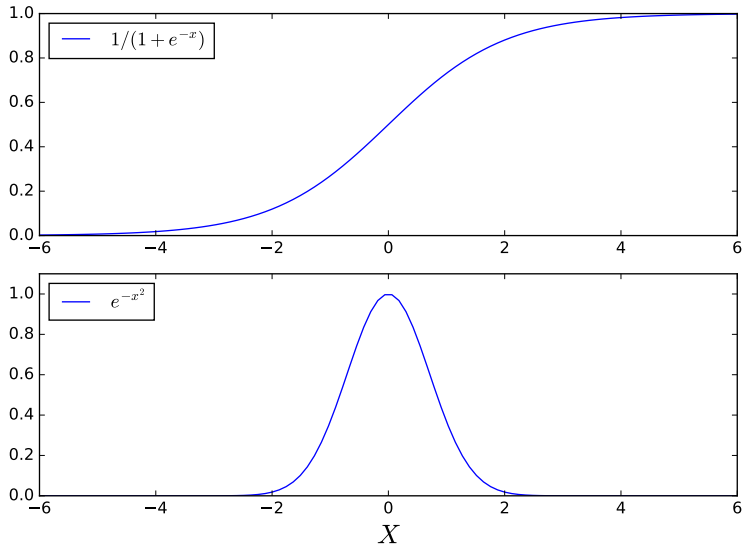


Figure 5.4: A Sigmoid (top) and Radial Basis Function (RBF) (Bottom). These are commonly used as activation functions in neural networks.

ELM is used for virtual metrology and its performance compared with the ones obtained with Lasso and Ridge regression.

Example 5.2.2. In this example the J2M data (Dataset 2.4.2) is used as a benchmark dataset for etch rate prediction. The size of the OES spectrum is reduced keeping only the 100 variables with the highest variation. The data is then reduced to $(\mathbf{X}, \mathbf{y}) \in \mathbb{R}^{2194\times 100} \times \mathbb{R}^{2194}$. An ELM with $L = 1000$ nodes and the weights chosen according to the procedure described in Section 5.2.2.3 was used to estimate a VM model for this dataset and its prediction performance compared to the models obtained with lasso and ridge regression. Initially, the lasso, ridge and ELM models were trained using the full OES dataset $(\mathbf{X}, \mathbf{y})$ and their reconstruction accuracy evaluated on the same data. The results obtained are reported in Figure 5.7 and clearly show that, while all methods are able to achieve a reasonable approximation to the ER over most of the wafers, the ELM model is the only one able to give a reasonable approximation for values of $\mathbf{X}$ around 200 where there is a clear discontinuity in the data. The prediction capability of the models was also evaluated using bootstrapping [60]. Here, the data $(\mathbf{X}, \mathbf{y})$ is split into two independent

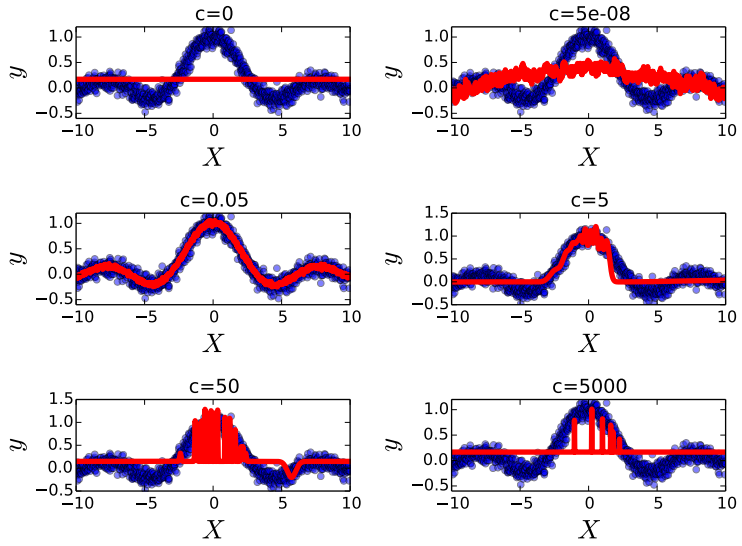


Figure 5.5: The approximation of the function $\frac{sin(x)}{x}$ for different values of the parameter c as described in Section 5.2.2.3. The data is described in Section 5.2.1.

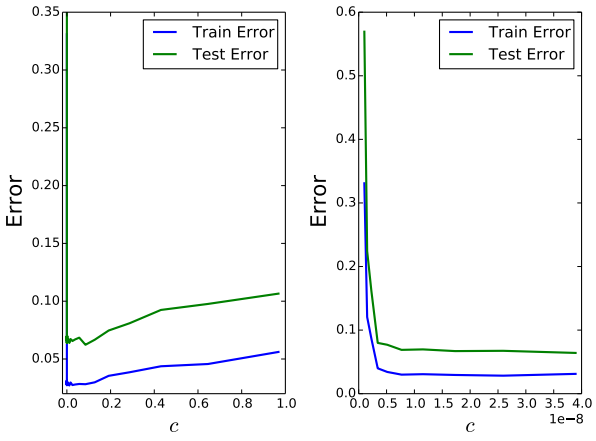


Figure 5.6: Training and prediction error as a function of c, the width of the uniform probability distribution used to generate the random weights. The plot on the right is a zoomed in version of the plot on the left focusing on the trend for small values of c.

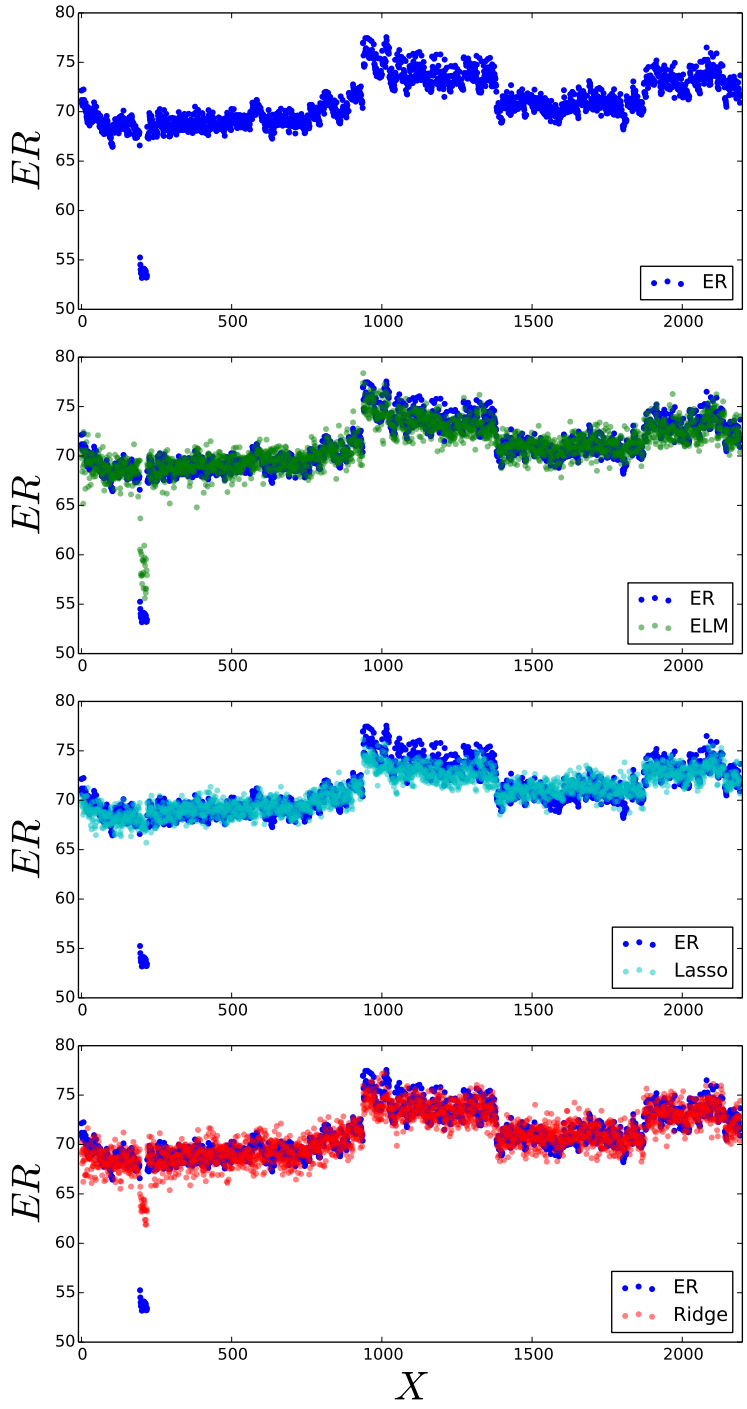


Figure 5.7: A plot of the actual evolution of ER as a function of the number of wafers processed, and the VM estimates obtained using ELM, lasso and ridge regression models.

	$\mathbf{ELM}_{\mathbf{c}=\hat{\mathbf{c}}}$	Lasso	Ridge	$ELM_{c=0.1}$	$ELM_{c=0.5}$	$ELM_{c=1}$
Mean	**0.0617**	0.0697	0.0697	0.0638	0.0809	0.0927
Std	**0.0027**	0.0055	0.0055	0.0039	0.0065	0.0079

Table 5.1: Mean and standard deviation of the prediction error obtained with ELM, the lasso estimator and the ridge estimator.

disjoint sets $(\mathbf{X}^0, \mathbf{y}^0)$, $(\mathbf{X}^1, \mathbf{y}^1)$ respectively composed of N_0 and N_1 samples. The first set $(\mathbf{X}^0, \mathbf{y}^0)$ is used to estimate the model and the second one $(\mathbf{X}^1, \mathbf{y}^1)$ is used as test data to score its prediction performance, defined as:

$$err = \frac{1}{N_1} \sum_{y_i \in \mathbf{y}_1} |y_i - \hat{y}_i|^2 \tag{5.23}$$

This procedure is repeated 100 times and the mean and the standard deviation of the prediction error (err) computed. The results obtained with lasso, ridge and ELM with various values of c are reported in Table 5.1. As can be seen the lasso and the ridge estimators have almost identical performance, while the performance of the ELM model is strongly influenced by the choice of the c parameter. Significantly the best results are achieved if c is optimized in accordance with equation (5.22), in which case the ELM yields the best results overall.

5.3 Novel Nonlinear Unsupervised Features Selection Algorithms

To date nonlinear extensions of the FSCA algorithm have not been investigated. Hence, the focus of this section is to investigate such extensions and whether they can lead to more compact data representations than conventional FSCA, which assumes linear relationships between variables. The nonlinear extensions are achieved by replacing the linear reconstruction model in FSCA with linear-in-the-parameters nonlinear models [174] or neural networks. A classical measure to estimate the goodness of a dimensionality reduction approach is the percentage of explained variance as defined in equation 4.3. The question is how much of the original data can be reconstructed from the selected variables. In linear feature selection this is simply done as in equation 4.10. The equivalent relationship in the nonlinear case between the dataset $\mathbf{X} \in \mathbb{R}^{n \times p}$ and a subset of its columns $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_k) \in \mathbb{R}^{n \times k}$ is defined as

$$\hat{\mathbf{X}} = \Psi_{\mathbf{X}}(\mathbf{Z}) \tag{5.24}$$

where $\Psi_{\mathbf{X}}$ is the nonlinear model that is used to estimate $\mathbf{X}$ from $\mathbf{Z}$.

5.3.0.1 Overfitting

As with standard nonlinear regression, one of the challenges with using nonlinear FSCA is avoiding overfitting, which can occur if too complex a model is used. Even in the case of linear-in-the-parameter models, it can become an issue if the number of regressors approaches, or is greater than, the number of samples. One approach to controlling model complexity is to constrain or penalize the norm of the parameter vector, such that irrelevant parameters are suppressed [175], or forced to zero [63]. An example of node selection trough penalization for ELMs is presented in [22]. This can easily be extended to multi-output problems by considering a multi-level regularization approach such as presented in [176]. This introduces a substantial computational overhead as choosing the appropriate weight for the penalty term, referred to as the regularization gain, requires repeated parameter estimation and model evaluation over a grid of values. Alternatively, the optimal model complexity can be determined by systematically evaluating models of increasing complexity on a dataset that is independent of the data used in model parameter estimation, a process referred to as cross-validation [60]. The optimum model is then selected the one with the minimum cross-validation error.

5.3.0.2 Multi-output Regressor

The function $\Psi_{\mathbf{X}}$ maps $\mathbf{Z} \in \mathbb{R}^{n \times K}$ into $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p) \in \mathbb{R}^{n \times p}$.

$$\Psi_{\mathbf{X}} : \mathbb{R}^K \longrightarrow \mathbb{R}^p \tag{5.25}$$

$\Psi_{\mathbf{X}}$ can in theory be defined as the union of p single output nonlinear regression models, that is:

$$\Psi_{\mathbf{X}} = [\Psi_{\mathbf{X}}^1, \ldots, \Psi_{\mathbf{X}}^p] \tag{5.26}$$

where

$$\Psi_{\mathbf{X}}^i : \mathbb{R}^K \to \mathbb{R} \tag{5.27}$$

$$\hat{\mathbf{x}}_i = \Psi_{\mathbf{X}}^i(\mathbf{Z}) \tag{5.28}$$

and

$$\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_p) \tag{5.29}$$

If general nonlinear models are used this may be too complex from a practical point of view as each function $\Psi^i_{\mathbf{X}}$ would need to be individually tuned according to the target variable $\mathbf{x}_i$. It is thus preferable to consider linear-in-the-parameter nonlinear modelling paradigms, as these naturally generalize to multi-output problems.

5.3.1 Nonlinear FSCA and FSV

An implementation of the FSV algorithm is reported in Pseudocode 4.6.2 in chapter 4. Given nonlinear function $\Psi_{\mathbf{X}}$, the nonlinear version of the FSV algorithm is obtained by replacing line 3 of Pseudocode 4.6.2 with:

$$i_j = \underset{\mathbf{x}_i \in \mathbf{X}}{\operatorname{argmin}} \parallel \Psi_{\mathbf{X}}((\mathbf{Z}_{j-1}, \mathbf{x}_i)) - \mathbf{X} \parallel_F \tag{5.30}$$

In other words, the linear estimation of $\mathbf{X}$ based on the variables $(\mathbf{Z}_{j-1}, \mathbf{x}_i)$ is replaced with the approximation of $\mathbf{X}$ obtained with a nonlinear model $\Psi_{\mathbf{X}}$ trained on the input data $(\mathbf{Z}_{j-1}, \mathbf{x}_i)$ and outputs $\mathbf{X}$.

Similarly to FSCA, the nonlinear version of FSCA is instead obtained with an iterative procedure. At each iteration a variable is chosen in order to minimize the current error.

$$i_j = \underset{\mathbf{e}_i \in \mathbf{E}_j}{\operatorname{argmin}} \parallel \Psi_{\mathbf{E}_j}(\mathbf{e}_i) - \mathbf{E}_j \parallel_F \tag{5.31}$$

The detailed implementation of Nonlinear FSCA is reported in Pseudocode 5.3.1. The considered implementation returns the matrix of components $\mathbf{Q}$, the associated variables $\mathbf{Z}$ and the set of functions required to reconstruct $\mathbf{X}$ from $\mathbf{Z}$.

Observation 5.3.1. *In the linear case FSCA and FSV lead to the same result both in term of explained variance and the variables selected. This is not the case for their nonlinear counterparts. Indeed nonlinear FSCA and nonlinear FSV may lead to totally different results.*

Observation 5.3.2. *In chapter 4 a number of refinement algorithms were introduced in order to improve the performances of FSCA. These algorithms were based on the FSV implementation of FSCA. Since in the nonlinear case FSCA and FSV lead to different results the same algorithms can only be used to improve the performance of nonlinear FSV.*

For nonlinear FSV the reconstruction of the original data is obtained as in equation 5.24, where $\Psi_{\mathbf{X}}(\cdot)$ is estimated in the last step of the FSV algorithm.

Input: Input data $\mathbf{X}$, a function $\Psi : \mathbb{R}^n \to \mathbb{R}^{n \times p}$, the number of components K.

$\mathbf{Q} = ()$
$\mathbf{Z} = ()$
$\mathbf{E}_1 = \mathbf{X}$
for $j = 1, \ldots, K$ **do**
  $i_j = \text{argmin}_{\mathbf{e}_i^j \in \mathbf{E}_j} \parallel \Psi_{\mathbf{E}_j}(\mathbf{e}_i^j) - \mathbf{E}_j \parallel_F$ // (Notation: $\mathbf{E}_j = (\mathbf{e}_1^j, \ldots, \mathbf{e}_p^j)$)
  $\mathbf{Q} = (\mathbf{Q}, \mathbf{e}_{i_j}^j)$
  $\mathbf{Z} = (\mathbf{Z}, \mathbf{x}_{i_j})$
  $\mathbf{E}_{j+1} = \Psi_{\mathbf{E}_j}(\mathbf{e}_{i_j}^j) - \mathbf{E}_j$
end for
return $\mathbf{Q}$, $\mathbf{Z}$ and $(\Psi_{\mathbf{E}_1}, \ldots, \Psi_{\mathbf{E}_k})$

Pseudocode 5.3.1: Nonlinear FSCA

In the nonlinear FSCA algorithm $\Psi_{\mathbf{X}}(\cdot)$ is not explicitly estimated in the algorithm as it does not appear in equation 5.31.

Observation 5.3.3. *In FSV the optimization problem in equation 5.30 is based on all currently selected variables ($\mathbf{Z}_j$). On the other hand the FSCA optimization problem is based on only one component ($\mathbf{e}_j$). The FSV algorithm is then expected to perform better than FSCA, since at each step a multivariate optimization is performed.*

5.3.1.1 Projection and Reconstruction with Nonlinear FSCA

The nonlinear FSCA algorithm returns an ordered set of components that are the columns of the $\mathbf{Q}$ matrix:

$$\mathbf{Q} = \mathbf{Q}_K = (\mathbf{x}_{i_1}, \mathbf{e}_{i_2}, \ldots, \mathbf{e}_{i_K}) \tag{5.32}$$

and a set of functions:

$$\Psi_{\mathbf{E}_j}(\mathbf{v}) : \mathbb{R}^n \to \mathbb{R}^{n \times p} \quad for\ i = 1, \ldots, K \tag{5.33}$$

Given a new matrix $\tilde{\mathbf{X}}$ the components obtained with Nonlinear FSCA are computed as in Pseudocode 5.3.2. The reconstruction of $\tilde{\mathbf{X}}$ (i.e. $\hat{\mathbf{X}}$) is obtained from the nonlinear FSCA components as described in Pseudocode 5.3.3.
In the rest of the chapter several nonlinear models are described. From the theory discussed in this section it follows that for each model it is sufficient to show how the function $\Psi_{\mathbf{X}} : \mathbb{R}^{n \times k} \to \mathbb{R}^{n \times p}$ is defined. The nonlinear

Input: The index of the selected variables $i_1, \ldots, i_K$ as in equation 5.32, the set of functions as in equation 5.33 and a matrix $\tilde{\mathbf{X}}$

1: $\tilde{\mathbf{E}}_1 = \tilde{\mathbf{X}}$
2: **for** $j = 1, \ldots, K$ **do**
3: $\quad \tilde{\mathbf{E}}_{j+1} = \tilde{\mathbf{E}}_j - \Psi_{\mathbf{E}_j}(\tilde{\mathbf{e}}^j_{i_j})$ // (Notation: $\tilde{\mathbf{E}}_j = (\tilde{\mathbf{e}}^j_1, \ldots, \tilde{\mathbf{e}}^j_p)$)
4: **end for**
5: **return** $\tilde{\mathbf{e}}^1_{i_1}, \ldots, \tilde{\mathbf{e}}^K_{i_K}$

Pseudocode 5.3.2: Nonlinear FSCA components

Input: The $\mathbf{Q}$ matrix containing the selected components as in equation 5.32 and the set of functions as in equation 5.33

1: $\hat{\mathbf{X}} = \mathbf{0}$
2: **for** $j = K, \ldots, 1$ **do**
3: $\quad \hat{\mathbf{X}} = \hat{\mathbf{X}} + \Psi_{\mathbf{E}_j}(\mathbf{e}^j_{i_j})$
4: **end for**
5: **return** $\hat{\mathbf{X}}$

Pseudocode 5.3.3: Nonlinear FSCA Reconstruction

FSV algorithm associated with the model will then follows from equation 5.30 while the nonlinear FSCA algorithm will follows from Pseudocode 5.3.1 using the univariate restriction of $\Psi_{\mathbf{X}} : \mathbb{R}^n \to \mathbb{R}^{n \times p}$.

5.3.2 Polynomial Regression

Polynomial regression [174] without interaction terms is the simplest form of nonlinear regression. While it is commonly used for supervised regression it can be easily adapted to the UFS problem. The polynomial version of $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_k)$ of degree d is defined as:

$$\mathbf{Z}_{poly} = (\mathbf{1}, \mathbf{z}_1, \mathbf{z}_1^2, \ldots, \mathbf{z}_1^d, \mathbf{z}_2, \ldots, \mathbf{z}_2^d, \ldots, \mathbf{z}_k^d) \in \mathbb{R}^{n \times (dk+1)} \tag{5.34}$$

where $\mathbf{1}$ is a column vector with all elements equal to 1. Given a target matrix $\mathbf{y} \in \mathbb{R}^{n \times o}$ the relationship between $\mathbf{Z}$ and $\mathbf{y}$ if defined by the linear model between $\mathbf{y}$ and $\mathbf{Z}_{poly}$.

5.3.2.1 Polynomial FSV and FSCA

The polynomial FSV (PFSV) implementation is obtained by replacing $\mathbf{Z}$ with $\mathbf{Z}_{poly}$ at each iteration of the FSV algorithm. The corresponding approximation of $\mathbf{X}$ is given by

$$\hat{\mathbf{X}} = \Psi_{\mathbf{X}}(\mathbf{Z}) = \Phi(\mathbf{Z}_{poly})\mathbf{X} \tag{5.35}$$

PFSV is the simplest nonlinear extension of FSV. While it may lead to better performance than FSV, its capabilities are limited by the absence of nonlinear interactions between variables. This is shown with the aid of some examples in Section 5.3.6.1.

Similarly to PFSV polynomial FSCA (PFSCA) is defined by replacing line 5 of Pseudocode 5.3.1 with

$$i = \underset{\mathbf{e}_i^j \in \mathbf{E}_j}{\operatorname{argmin}} \parallel \Phi(\mathbf{e}_{i\,poly}^j)\mathbf{E}_j - \mathbf{E}_j \parallel_F \tag{5.36}$$

It is interesting to observe that, while PFSV does not include nonlinear interactions between variables, PFSCA does generate interaction terms. This follows from the fact that the matrix $\mathbf{E}_j$ is recursively updated during the PFSCA algorithm.

Computational Complexity The computational complexity of polynomial regression is roughly the same as linear regression. It follows that the computational complexity of the polynomial versions of FSCA and FSV will be computationally similar to the linear implementations.

5.3.3 Extreme Learning Machines FSV

As described in Section 5.2.2 ELMs have been used for both single output and multi-output supervised regression problems. The approach can be easily extended to unsupervised problems. In order to use it in an FSV like algorithm it is sufficient to use $\mathbf{X}$ as the output of the neural network and a subset of its variables ($\mathbf{Z}$) as the input. The SLFNN equation in 5.6 becomes

$$\hat{\vec{\mathbf{x}}}_j = f_L(\vec{\mathbf{z}}_j) = \sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \cdot \vec{\mathbf{z}}_j + b_i) \tag{5.37}$$

where $\vec{\mathbf{z}}_j \in \mathbb{R}^k$ is the j^{th} row of $\mathbf{Z}$. Equation 5.6 can be expressed in matrix form as:

$$\hat{\mathbf{X}} = f_L(\mathbf{Z}) = \mathbf{H}^{\mathbf{Z}}\boldsymbol{\beta} \tag{5.38}$$

where $\mathbf{H}^{\mathbf{Z}}$ is defined as:

$$\mathbf{H}^{\mathbf{Z}} = \mathbf{H}^{\mathbf{Z}}(\mathbf{W},\ \mathbf{b}, \mathbf{Z}) \in \mathbb{R}^{n \times L} :\ \mathbf{H}^{\mathbf{Z}}_{i,j} = g(\mathbf{w}_j \cdot \vec{\mathbf{z}}_i + b_j) \tag{5.39}$$

and $\mathbf{b}$ is defined as $\mathbf{b} = \{b_i\}_{i=1,\ldots,L}$. As described in Section 5.2.2 the weights of the hidden layer $\mathbf{W}$ are randomly initialized, with only the output layer weights optimised as:

$$\hat{\boldsymbol{\beta}}_{\mathbf{X}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{L\times p}}{\operatorname{argmin}} \parallel \mathbf{X} - \mathbf{H}^{\mathbf{Z}}\boldsymbol{\beta} \parallel_2^2 \tag{5.40}$$

$$\hat{\mathbf{X}} = \mathbf{H}^{\mathbf{Z}}\hat{\boldsymbol{\beta}}_{\mathbf{X}} \tag{5.41}$$

The ELM based version of FSV (ELM-FSV) is then obtained by replacing $\Psi_{\mathbf{X}}(\mathbf{Z})$ in equation 5.31 with

$$\hat{\mathbf{X}} = \Psi_{\mathbf{X}}(\mathbf{Z}) = \boldsymbol{\Phi}(\mathbf{H}^{\mathbf{Z}})\mathbf{X} \tag{5.42}$$

The ELM used in this section is a single hidden layer multilayer perception architecture with sigmoid activation functions in the hidden layer neurons. Hence, $g(x)$ is defined as:

$$g(x) = \frac{1}{1+e^{-x}}. \tag{5.43}$$

The ELM-FSCA implementation is obtained in a similar fashion.

5.3.3.1 Random weights initialization in ELM-FSV

In the ELM-FSV algorithm variables are recursively added as input nodes in the neural network. At step k of the ELM-FSV algorithm the ELM has k input variables and L hidden nodes. In the classical version of ELM the input weights $\{w_{i,j}\}_{i=1,\ldots,k;\ j=1\ldots,L}$ are randomly initialized and stored in a matrix $\mathbf{W}^k \in \mathbb{R}^{k\times L}$. Since the dimension of $\mathbf{W}^k$ changes at every step, the ELM needs to be retrained for each new input. If the values of $\mathbf{W}^{k+1}$ are randomly assigned at each step they will have no relationship with the weights at step k. This is not desirable as the first k selected variables are selected based on the values of the weight matrix $\mathbf{W}^k$. If their values change the first k selected variables may no longer be optimal. In order to avoid this problem $\mathbf{W}^{k+1}$ is obtained by adding a randomly generated row to $\mathbf{W}^k$, that is:

$$\mathbf{W}^{k+1} = \begin{pmatrix} \mathbf{W}^k \\ \vec{\mathbf{w}} \end{pmatrix} \in \mathbb{R}^{(k+1)\times L} \tag{5.44}$$

where $\vec{\mathbf{w}} \in \mathbb{R}^L$ is a randomly generated vector. As shown in [22] and discussed in Section 5.2.2.3 the performance of ELMs is strongly influenced by the distribution that is used to generate the weights matrix $\mathbf{W}$. In this section $\mathbf{W} \sim Unif(-\frac{1}{p}, \frac{1}{p})$ where p is the total number of variables. This is reasonable because all the variables are assumed to have zero mean and unit variance.

5.3.3.2 ELM with Direct Linear Feed-through

In FSCA and FSV like algorithms some of the variables appear both as inputs and as outputs. This is because the input is defined as $\mathbf{Z} = (\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k})$ and the output as $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p)$ where $k << p$ (i.e. the input variables are a subset of the output variables). The identity function is the appropriate mapping for these variables, and estimating this mapping using the ELM will be sub-optimal. The same is true if the underlying relationship between inputs and outputs is linear. To overcome this shortcoming of the basic ELM model a modification is introduced whereby the inputs are also directly connected to the output. Mathematically, this corresponds to the selected variables being added as additional columns in the matrix $\mathbf{H}^{\mathbf{Z}}$, that is:

$$\tilde{\mathbf{H}}^{\mathbf{Z}} = (\mathbf{H}^{\mathbf{Z}}, \mathbf{Z}) \in \mathbb{R}^{n \times lk} \tag{5.45}$$

Neural networks where the input neurons are connected to both the first hidden layer and the output layer neurons are called Direct Linear Feed-through Artificial Neural Networks [177]. Hence, we refer to the modified ELMS as an Extreme Learning Machine with Direct Linear Feed-through (ELMDLF). The related ELM-FSCA algorithm is called ELMDLF-FSCA. At the k^{th} step of the ELMDLF-FSCA algorithm the matrix $\mathbf{H}$ is then defined as:

$$\tilde{\mathbf{H}}^{\mathbf{Z}_k} = (\mathbf{H}^{\mathbf{Z}_k}, \mathbf{Z}_k) \tag{5.46}$$

5.3.4 Kernel FSV

In [120] an algorithm equivalent to FSV and a possible kernel generalization is proposed (KFSV). Here the KFSV algorithm is reviewed and integrated in the context of features selection and data reconstruction. The optimization problem defined in equation 4.7 has a unique solution as given by equation 4.10. In order to shrink the size of the coefficients the problem can be reformulated as a Ridge Regression problem:

$$\min_{\mathbf{\Theta}} \parallel \mathbf{X} - \mathbf{Z}\mathbf{\Theta} \parallel_F + \lambda \parallel \mathbf{\Theta} \parallel_F \tag{5.47}$$

The solution is then:

$$\hat{\mathbf{\Theta}} = (\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^T\mathbf{X} \tag{5.48}$$

or equivalently using the dual solution as in [178]

$$\hat{\mathbf{\Theta}} = \mathbf{Z}^T(\mathbf{Z}\mathbf{Z}^T + \lambda\mathbf{I})^{-1}\mathbf{X} \tag{5.49}$$

It follows that:

$$\hat{\mathbf{X}} = \mathbf{Z}\mathbf{Z}^T(\mathbf{Z}\mathbf{Z}^T + \lambda\mathbf{I})^{-1}\mathbf{X} \tag{5.50}$$

or if the reconstruction of new data has to be computed:

$$\hat{\mathbf{X}}_{new} = \mathbf{Z}_{new}\mathbf{Z}^T(\mathbf{Z}\mathbf{Z}^T + \lambda\mathbf{I})^{-1}\mathbf{X} \tag{5.51}$$

In this formulation $\mathbf{Z}$ and $\mathbf{Z}_{new}$ appear only in the form of a matrix product. Nonlinearities can be included through the use of a kernel $\mathcal{K}$ [179]. The previous equations become:

$$\hat{\mathbf{X}} = \mathcal{K}(\mathbf{Z}, \mathbf{Z}^T)(\mathcal{K}(\mathbf{Z}, \mathbf{Z}^T) + \lambda I)^{-1}\mathbf{X} \tag{5.52}$$

$$\hat{\mathbf{X}}_{new} = \mathcal{K}(\mathbf{Z}_{new}, \mathbf{Z}^T)(\mathcal{K}(\mathbf{Z}, \mathbf{Z}^T) + \lambda\mathbf{I})^{-1}\mathbf{X} \tag{5.53}$$

where $\mathcal{K}(\mathbf{Z}, \mathbf{Z}^T) \in \mathbb{R}^{n\times n}$ is the matrix whose (i, j)-th element is given by $\mathcal{K}(\mathbf{z}_i, \mathbf{z}_j)$.

In an FSV context $\hat{\mathbf{X}}_{new}$ represents the reconstruction of $\mathbf{X}$ obtained with the variables in $\mathbf{Z}$ when the kernel $\mathcal{K}$ is used. The selection of variables or components with KFSV is done with the standard FSV or FSCA algorithms using as estimation of $\mathbf{X}$ equations 5.52 or 5.53. Several types of kernel can be used. The most popular are:

- Linear Kernel:
$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j \tag{5.54}$$
In this case KFSV is equivalent to the standard FSV
- Polynomial:
$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T\mathbf{x}_j + c)^d \tag{5.55}$$
- Radial Basis Function:
$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\| \mathbf{x}_i - \mathbf{x}_j \|_2}{2\sigma}} \tag{5.56}$$

Observation 5.3.4. *The KFSV algorithm is defined starting from the ridge regression problem defined in equation 5.47 instead of the least square problem as in equation 4.7. This was required as in equations 5.52 and 5.53 the term* $(\mathcal{K}(\mathbf{Z}, \mathbf{Z}^T) + \lambda\mathbf{I})$ *is not of full rank if* $\lambda = 0$*. This follows from the fact that in the FSV algorithm* $\mathbf{Z} \in \mathbb{R}^{n\times k}$ *with* $k < n$ *and* $\mathbf{Z}\mathbf{Z}^T \in \mathbb{R}^{n\times n}$

Observation 5.3.5. *The term* $(\mathcal{K}(\mathbf{Z}, \mathbf{Z}^T) + \lambda\mathbf{I})^{-1}$ *in equation 5.52 requires the computation of an inverse of an* $n \times n$ *matrix, where* n *is the number of samples. This may be computationally very demanding.*

The penalty value λ can be tuned using bootstrap or cross-validation. In sections 5.3.6.1, 5.3.6.2 where the methods are compared the penalty value is fixed to $\lambda = 0.001$. This is required to reduce the computational complexity of the methodology and to make a fair comparison with the other methods where parameters are decided at priori, rather then tuned with the data.

5.3.5 Deep Learning Based Feature Selection

All the algorithms introduced in this chapter are based on the FSV and FSCA algorithms. They allow nonlinear unsupevised feature selection with roughly the same computational cost as FSV and FSCA. In addition, similarly to their linear counterparts they require minimal user intervention. On the other hand these methods allow only a limited degree of non linearity. PFSV for example is not able to detect nonlinear interaction between variables. ELMFSV is limited by the approximation capabilities of a single hidden layer neural network. Indeed when a function can be compactly represented by a deep architecture, it might need a very large architecture to be represented by an insufficiently deep one [180]. For this reason an unsupervised feature selection algorithm based on a deep neural network architecture is introduced. A fully connected multilayer neural network is defined by the sequence of neurons in each layer

$$Topology = [L_0, L_1, \ldots, L_{l-2}, L_{l-1}] \tag{5.57}$$

and by the activation function of each layer:

$$Activaction = [A_0, A_1, \ldots, A_{l-2}, A_{l-1}] \tag{5.58}$$

where

$$A_i : \mathbb{R} \to \mathbb{R} \tag{5.59}$$

The weights of the neural network, that are in general randomly initialized are then trained in order to solve the minimization problem defined in equation 5.4. In the next algorithms it is shown how a Multilayer neural network can be used to perform unsupervised features selection.

Algorithm 5.3.1. Given a dataset $\mathbf{X} \in \mathbb{R}^{n \times p}$ a neural network with l layers is constructed having $\mathbf{X}$ as both input and output. The layers then have the following sequence of nodes cardinality:

$$Nodes = [p, L_1, \ldots, L_{l-2}, p] \tag{5.60}$$

and the activation functions are

$$Activaction = [Id, A_1, \ldots, A_{l-2}, Id] \tag{5.61}$$

where $Id(x) = x \ \forall x \in \mathbb{R}$ and A_i is a function defined by the user.

Once the structure of the network is defined it is trained with the back-propagation algorithm. Among the input variables the variable whose deletion leads to the smallest training error is removed. The topology is then:

$$Topology = [p-1, L_1, \ldots, L_{l-2}, p] \tag{5.62}$$

The procedure is repeated using only the $p-1$ remaining variables. The algorithm ends when only K input variables remain i.e. when:

$$Topology = [K, L_1, \ldots, L_{l-2}, p] \tag{5.63}$$

Employing ANNs with several hidden layers has lead to state-of-the-art performance with several benchmarking datasets [181]. It follows that they can lead to optimal unsupervised features selection if they can be effectively optimized. A major drawback of ANNs and the method described in Algorithm 5.3.1is that they have numerous hyperparameters. It is indeed necessary to choose the structure of the neural network and the activation function of each neuron. While these parameters may be tuned with a grid search algorithm, it may become computationally impossible to try all combinations of parameters. In the next section, where the performance of all the algorithms is compared, a fixed configuration is used for all datasets considered. The neural network chosen is composed of 5 layers (an input and output layer and 3 hidden layers). The hidden layers each have $2p$ nodes. The rectified activation function:

$$RELU(x) = max(0, x) \tag{5.64}$$

is used for each node in the hidden layers. The rectified function was shown to perform better than sigmoid and radial basis function activation functions when used in deep neural networks [182]. The number of hidden nodes is set to $2p$ in order to have a higher dimensional representation of the data in each hidden layer. The performance of a neural network is strongly influenced by the initialization of the weights [183]. It is therefore important to make sure that the initial weights are close to a good solution when Algorithm 5.3.1 starts. At the beginning of Algorithm 5.3.1 the input and the output are equivalent. The neural network is then trained in order to be the identity function on $\mathbf{X}$ i.e.

$$f_{\mathbf{W}}(\mathbf{X}) = \mathbf{X} \tag{5.65}$$

This is in theory always possible in the specified settings as the hidden layers have an higher dimensionality than the visible ones ($L_1 = \ldots L_{l-2} = 2p$). In the experiments described in Section 5.3.6.2 the training was performed using 30000 iterations of the ADAM optimization algorithm [184].

Observation 5.3.6. *Some of the main drawbacks of Algorithm 5.3.1 are the computational complexity, the sensitivity with respect to the hyperparameters [185] and the general difficulty in training an ANN with a deep architecture [186]. The procedure is particularly complex because it requires training of the ANN to be performed $p - K$ times. In particular, due to the hierarchical procedure used to select variables initial error may significantly influence the final result.*

Observation 5.3.7. *This ANN based method is different from all the others as it is based on a backward elimination procedure. This is in general not effective in datasets where the number of samples is smaller than the number of variables. For this reason and due to the fact that ANNs require a large number of training samples this method will be tested only on datasets with few variables and many samples.*

5.3.6 Performance Evaluation

The performances of the proposed nonlinear FSV and FSCA algorithms are evaluated in this section. For the evaluation procedure several datasets are used. Each one is split into a training and a testing set ($\mathbf{X}_{train}$ and $\mathbf{X}_{test}$). The training set is used to build the model while the test set is only used to evaluate the reconstruction performance, defined as:

$$EV_{test} = 1 - \frac{\| \mathbf{X}_{test} - \hat{\mathbf{X}}_{test} \|_2}{\| \mathbf{X}_{test} \|_2} \tag{5.66}$$

where

$$\hat{\mathbf{X}}_{test} = \Psi_{\mathbf{X}_{train}}(\mathbf{Z}_{test}) \tag{5.67}$$

and $\mathbf{Z}_{test}$ is the matrix composed of the variables of $\mathbf{X}_{test}$ that were selected during training.

As shown in Section 5.2.2 the perfomance of the ELM algorithm is influenced by the choice of the distribution of the random weights. In order to make the experiments simpler, for all the datasets considered the weights were generated according to a uniform distribution whose width is determined by the number of outputs p, that is:

$$\mathbf{W}_{i,j} \sim Unif(-\frac{1}{p}, \frac{1}{p}) \tag{5.68}$$

This is a reasonably good choice as all variables have zero mean and unit variance. Also the test data is then scaled to zero mean and unit variance

as explained in Section 3.5.3.1. While scaling of the data does not affect linear regression models it helps the ELM based methods as it avoids neuron saturation problem as described in equation 5.21. It also avoid potential numerical problems with polynomial regression as the power of a large number can easily become numerical intractable (i.e. $x^d >>> 0$ if $x >> 0$ and $d >> 0$).

5.3.6.1 Simulated Examples

The methods are initially compared on two simple simulated datasets. These are defined in order to highlight the differences between the considered algorithms, in particular with regard to their ability to model nonlinear interaction between variables.

Dataset 5.3.1 (Cubic-Additive)**.** Consider the variables $\mathbf{x}_1, \mathbf{x}_2 \sim N(0,1)$. The variables $\mathbf{x}_3$ and $\mathbf{x}_4$ are obtained from $\mathbf{x}_1$ and $\mathbf{x}_2$ as:

$$\mathbf{x}_3 = \mathbf{x}_1 + \mathbf{x}_2^2 \tag{5.69}$$

$$\mathbf{x}_4 = \mathbf{x}_1^2 + \mathbf{x}_2^3 \tag{5.70}$$

Each variable is then scaled to zero mean and unit variance.

Dataset 5.3.2 (Cubic-Interactions)**.** Consider the variables $\mathbf{x}_1, \mathbf{x}_2 \sim N(0,1)$. The variables $\mathbf{x}_3$ and $\mathbf{x}_4$ are obtained from $\mathbf{x}_1$ and $\mathbf{x}_2$ as:

$$\mathbf{x}_3 = \mathbf{x}_1\mathbf{x}_2^2 + \mathbf{x}_1 \tag{5.71}$$

$$\mathbf{x}_4 = \mathbf{x}_1^2\mathbf{x}_2^3 + \mathbf{x}_2\mathbf{x}_1 \tag{5.72}$$

Each variable is then scaled to zero mean and unit variance.

A noisy version of both datasets is generated by adding a random error to the data matrix:

$$\mathbf{X}_{noise} = \mathbf{X} + \mathbf{E} \tag{5.73}$$

where

$$\mathbf{E} = \{\epsilon_{i,j}\}, \quad \epsilon_{i,j} \sim N(0, 0.1) \; for \; i = 1, \ldots, n \; and \; j = 1, \ldots, p \tag{5.74}$$

For each type of data 2000 samples are generated. The first half is used to train the model and the second half is used to evaluate its performance (test set). The percentage explained variance obtained with each method on the training and test data when 2 variables are selected is used as a metric. The process is repeated 10 times and the mean results obtained on the training and test set are reported in Tables 5.2 and 5.3, respectively.

Results on the simulated dataset The first interesting observation is that all the methods perform relatively well on the training data. Indeed almost all the methods are better than FSCA. On the other hand many methods perform very poorly on the test dataset. These methods overfit the training data. Model selection should then therefore be carefully evaluated in real applications. We observed that FSCA based methods tend to overfit the data more than the FSV ones. For this reason ELMFSCA is used with a smaller number of hidden nodes than ELMFSV. From the results obtained on the test dataset it is possible to make more interesting observations.

- In the first dataset (dataset 5.3.1) all the nonlinear methods perform significantly better than PCA and FSCA. PFSV with $d \geq 3$ generally achieve optimal performance. This was expected as they contain the exact model. The same is true for KFSV with polynomial kernel. Surprisingly slightly better performance is obtained with ELMFSV-DL50. It is interesting to observe that PFSV performs generally better than PFSCA. This was expected as PFSCA is not necessarily able to infer the real model. In particular PFSCA seems to overfit much more than PFSV. This may be observed by comparing PFSCA and PFSV with degrees 7, 8 and 9.

  All the methods have slightly worse performance when noise is introduced. Apart from that the presence of noise does not have much influence on the results.

- In the second dataset (dataset 5.3.2) the performance of PFSV and PFSCA drops. This is expected as the polynomial model is not able to represent nonlinear interactions. It is interesting to observe that PFSCA with degree 2 and 3 is slightly better than PFSV with the same complexity. This may be explained by the fact that PFSCA is able to generate nonlinear interactions. Polynomials with higher degrees tend to overfit resulting in bad performances. In this data the best performance is obtained with ELMFSV, ELMFSV-DLF and KFSCA with RBF kernel. These are all methods that are able to represent nonlinear interactions.

While there is no method that is always better than the others, some general conclusions can be drawn:

- Overly complex models lead to overfitting. Some examples are PFSCA10, PFSV10 and ELMFSV-DLF50.

- For each method the FSCA implementation is in general worse than the FSV one.

- The overall best results are obtained with the ELM methods even if careful tuning is required to choose the optimal number of hidden nodes.
- The introduction of DLF in the ELM based methods leads to improvements especially when a small number of neurons is used.
- Methods with a low degree of nonlinearity, susch as PFSV2, PFSV3 and ELMFSV-DLF with a small number of nodes, provide, in general, better performance than FSCA.

In the next section the various algorithms are compared on real datasets.

Methods	Dataset			
	Cubic Additive	Cubic Interaction	Cubic Additive with noise	Cubic Interaction with noise
PCA	83.17	82.30	82.83	82.14
FSV	72.3	72.87	71.98	72.86
PFSCA2	91.82	75.70	91.02	78.33
PFSCA3	**99.97**	83.01	98.65	83.94
PFSCA4	99.97	84.71	98.65	85.60
PFSCA5	99.96	86.80	98.65	86.90
PFSCA6	99.95	88.60	98.65	87.61
PFSCA7	99.94	89.71	**98.66**	88.09
PFSCA8	99.94	90.50	98.66	88.64
PFSCA9	99.93	90.79	98.66	88.97
PFSCA10	99.92	**91.31**	98.66	**89.70**
PFSV2	91.83	75.25	91.08	77.56
PFSV3	**100.0**	82.33	98.69	83.25
PFSV4	100.0	84.28	98.70	85.01
PFSV5	100.0	86.16	98.70	86.51
PFSV6	100.0	87.96	98.70	87.33
PFSV7	100.0	89.16	98.71	88.04
PFSV8	100.0	90.02	98.72	88.46
PFSV9	100.0	90.63	98.72	89.07
PFSV10	100.0	**91.10**	**98.73**	**89.48**
ELMFSV2	71.58	70.20	71.09	70.08
ELMFSV5	89.02	77.33	88.10	76.62
ELMFSV10	95.97	97.21	94.01	95.92
ELMFSV15	99.87	99.13	98.58	97.88
ELMFSV25	99.92	99.71	98.65	98.40
ELMFSV50	**100.0**	**99.97**	**98.82**	**98.66**
ELMFSV-DLF2	83.25	86.96	82.75	86.45
ELMFSV-DLF5	85.69	93.13	84.65	91.97
ELMFSV-DLF10	99.57	98.52	98.24	97.17
ELMFSV-DLF15	99.61	99.45	98.11	98.20
ELMFSV-DLF25	99.98	99.81	98.74	98.51
ELMFSV-DLF50	**100.0**	**99.98**	**98.83**	**98.66**
ELMFSCA2	87.59	76.34	85.84	78.10
ELMFSCA3	99.11	82.19	96.78	82.43
ELMFSCA4	99.62	83.61	98.47	84.66
ELMFSCA5	99.92	86.21	98.65	86.71
ELMFSCA10	**99.92**	**90.77**	**98.65**	**89.07**
ELMFSCA-DLF2	99.38	82.25	96.62	83.71
ELMFSCA-DLF3	99.91	83.57	98.56	85.54
ELMFSCA-DLF4	99.94	85.92	98.63	86.83
ELMFSCA-DLF5	**99.95**	87.66	98.65	87.57
ELMFSCA-DLF10	99.92	**91.37**	**98.65**	**89.34**
KFSV	99.99	**99.99**	**98.89**	**98.72**
KFSVPoly2	95.67	77.03	93.67	79.07
KFSVPoly3	**100.0**	96.20	98.69	95.68
KFSVPoly4	100.0	97.26	98.71	96.42

Table 5.2: Percentage Variance Explained by various linear and nonlinear unsupervised variable selection technique when applied to the simulated dataset introduced in section 5.3.6.1 The table shows the result for the training dataset when 2 variables are selected by each method.

Methods	Dataset			
	Cubic Additive	Cubic Interaction	Cubic Additive with noise	Cubic Interaction with noise
PCA	82.53	82.03	82.23	81.63
FSV	71.31	71.60	70.69	71.19
PFSCA2	90.69	73.17	89.78	72.41
PFSCA3	**99.53**	**77.83**	**98.23**	**78.31**
PFSCA4	99.52	72.64	98.16	54.89
PFSCA5	99.35	43.20	97.44	39.29
PFSCA6	98.00	16.27	87.92	13.91
PFSCA7	89.47	7.83	78.18	20.91
PFSCA8	89.41	15.60	58.86	8.09
PFSCA9	49.67	8.27	47.88	7.95
PFSCA10	49.21	0.00	29.18	4.38
PFSV2	90.76	72.35	88.39	**69.97**
PFSV3	**99.63**	**74.19**	**98.26**	66.41
PFSV4	99.63	72.19	98.24	49.71
PFSV5	99.63	64.21	98.21	45.58
PFSV6	99.63	47.10	98.16	34.02
PFSV7	99.63	29.60	97.96	24.99
PFSV8	99.63	26.43	97.69	21.39
PFSV9	99.63	16.02	95.61	19.99
PFSV10	99.63	10.75	91.33	12.13
ELMFSV2	70.66	68.97	70.26	68.54
ELMFSV5	88.49	73.65	86.94	72.45
ELMFSV10	94.06	95.74	90.94	93.65
ELMFSV15	99.36	**97.52**	**97.93**	**94.73**
ELMFSV25	99.28	96.90	97.60	91.75
ELMFSV50	**99.62**	85.41	95.82	54.47
ELMFSV-DLF2	82.42	84.92	81.29	82.94
ELMFSV-DLF5	84.40	91.55	82.72	89.19
ELMFSV-DLF10	99.04	96.19	97.58	95.30
ELMFSV-DLF15	98.59	**97.79**	96.92	**96.19**
ELMFSV-DLF25	99.52	97.62	**97.99**	95.77
ELMFSV-DLF50	**99.63**	71.32	94.87	45.92
ELMFSCA2	86.09	74.20	84.11	71.78
ELMFSCA3	98.40	78.28	95.88	78.67
ELMFSCA4	99.13	**79.85**	97.88	**79.12**
ELMFSCA5	**99.36**	65.16	**98.01**	68.17
ELMFSCA10	59.19	7.23	58.33	7.20
ELMFSCA-DLF2	98.77	**79.25**	96.02	**78.86**
ELMFSCA-DLF3	99.38	76.81	**98.03**	73.14
ELMFSCA-DLF4	**99.48**	61.80	97.98	54.82
ELMFSCA-DLF5	99.38	48.15	93.33	33.64
ELMFSCA-DLF10	36.09	7.59	35.59	7.21
KFSV	98.79	91.30	96.41	**85.72**
KFSVPoly2	95.36	70.89	92.75	68.69
KFSVPoly3	**99.63**	**93.25**	**98.25**	82.40
KFSVPoly4	99.63	87.60	98.20	75.30

Table 5.3: Percentage Variance Explained by various linear and nonlinear unsupervised variable selection technique when applied to the simulated dataset introduced in section 5.3.6.1 The table shows the result for the test dataset when 2 variables are selected by each method.

5.3.6.2 Real Datasets

The proposed methods are now tested on a set of real world datasets. The datasets considered are:

- J2MRed consisting of $n = 2194$ samples and $p = 200$ variables.
- Nino consisting of $n = 93935$ samples and $p = 11$ variables.
- Parkinson consisting of $n = 5875$ and $p = 22$ variables.
- Community Crime (CC) consisting of $n = 2107$ and $p = 118$ variables.
- Magic consisting of $n = 19020$ samples and $p = 10$ variables.
- Arcene consisting of $n = 99$ samples and $p = 400$ variables.

The J2MRed dataset is defined by the 200 variables with the highest variance from dataset 2.4.2. All the others benchmark datasets are taken from a public online repository [187].

Study methodology A set of $k = 5$ variables is selected with each method using a training dataset. The training dataset is then used to build the model and a testing dataset is used to evaluate the model performance. The experiment is repeated several times. At each iteration the data is randomly split into training and test datasets. The number of samples in the test dataset is randomly selected between 10% and 50% of the total data. Performance is then evaluated according to the percentage of explained variance in the test set as in equation 5.66. The mean and the standard deviation obtained on the training and testing set is reported in Tables 5.4 and 5.5. The best result obtained on the test set for each family of methods is reported in Figure 5.8.

Comments on the results The first obvious observation is that PCA performs well on all the datasets and is only outperformed by nonlinear variable selection methods for two of the datasets: Parkinson 81.58 vs 84.47 and Nino 74.91 vs 82.74. The Arcene data is characterized by more variables than samples. In this scenario all the nonlinear methods tend to overfit while simple linear methods like FSV achieved almost optimal performance. Despite this a small improvement over FSV can be obtained with PFSV with low degree, ELMFSV-DLF with a small number of hidden nodes and KFSV. In other words, adding a small amount of nonlinearity improves the performances of FSV. In all the other datasets ELMFSV-DLF achieves the best solution and is always more effective than its counterparts (ELMFSV,

ELFSCA, ELMFSCA-DLF). Among the polynomial based methods PFSV generally performs better than PFSCA with PFSV often outperforming linear FSCA. While PFSV is normally inferior to ELMFSV-DLF it may be the preferred choice if a lower complexity easy to interpret model is required.

The deep learning based method (NN) is only tested on 3 datasets, due its computational cost, and it performs relatively well in all of them. It obtains the optimal perfomance on the Parkinson dataset and beats PCA on the Nino dataset. However, overall its performance is not good enough to justify the use of such a computationally complex method. The method may perform better on larger datasets and if its parameters are more carefully tuned. Unfortunately due to limited computing power this hypothesis is not investigated here.

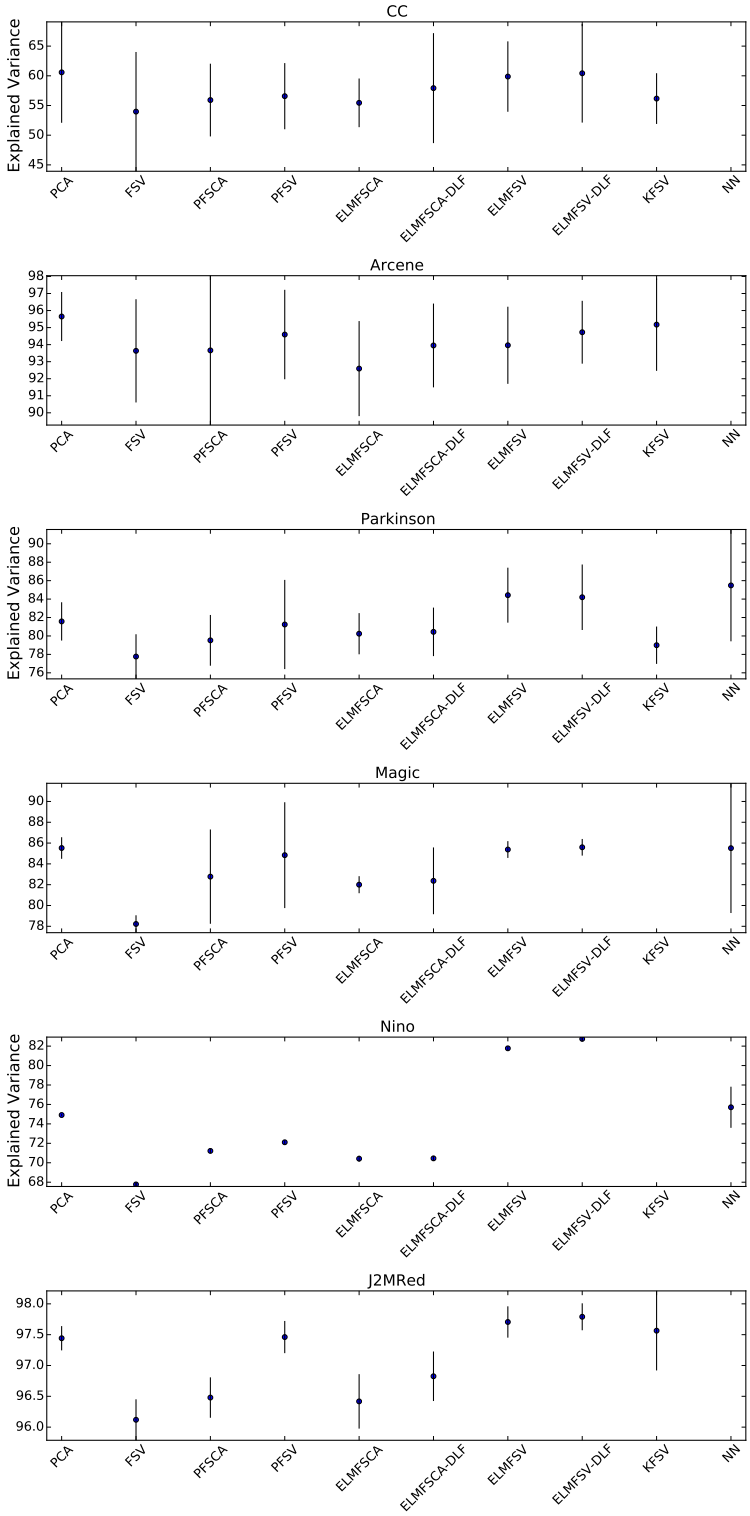


Figure 5.8: The mean and the 95% confidence interval of the explained variance obtained on the testing set on the datasets described in Section 5.3.6.2 with various algorithms when optimal hyper-parameters are chosen and $K = 5$ variables are selected.

	CC	Arcene	Parkinson	Magic	Nino	J2MRed
PCA	60.58 (4.33)	95.65 (0.73)	81.58 (1.06)	85.53 (0.53)	74.91 (0.06)	97.44 (0.1)
FSV	53.97 (5.12)	93.64 (1.55)	77.77 (1.24)	78.22 (0.43)	67.77 (0.11)	96.12 (0.17)
PFSCA2	56.93 (2.41)	95.69 (0.83)	79.52 (1.4)	82.77 (2.31)	69.68 (0.14)	96.48 (0.17)
PFSCA3	58.50 (3.15)	96.06 (0.99)	79.21 (1.51)	81.43 (0.54)	70.34 (0.14)	96.79 (0.11)
PFSCA4	60.03 (3.52)	96.0 (1.46)	79.96 (1.51)	82.18 (0.29)	70.67 (0.12)	96.97 (0.2)
PFSCA5	60.43 (4.04)	96.11 (1.5)	80.63 (1.54)	82.14 (1.22)	71.01 (0.11)	97.12 (0.17)
PFSCA6	61.85 (3.02)	96.59 (1.49)	81.55 (1.12)	82.66 (0.3)	70.97 (0.47)	97.19 (0.32)
PFSCA7	62.13 (3.09)	96.23 (1.58)	82.06 (1.16)	82.81 (0.21)	71.22 (0.12)	97.39 (0.07)
PFSCA8	63.41 (0.39)	96.59 (1.28)	**82.38 (0.93)**	82.89 (0.18)	71.4 (0.09)	97.42 (0.1)
PFSCA9	63.51 (0.54)	**96.69 (1.44)**	82.28 (1.36)	82.95 (0.13)	**71.46 (0.08)**	97.44 (0.07)
PFSCA10	**63.78 (0.6)**	96.68 (1.61)	82.18 (1.43)	**82.97 (0.13)**	71.43 (0.11)	**97.46 (0.08)**
PFSV2	57.40 (2.69)	95.12 (1.46)	80.17 (1.27)	**84.84 (2.6)**	69.73 (0.14)	97.04 (0.13)
PFSV3	59.94 (3.16)	94.68 (1.53)	80.23 (1.64)	82.82 (0.45)	70.44 (0.15)	97.22 (0.14)
PFSV4	59.31 (4.74)	95.09 (1.57)	80.73 (1.68)	83.02 (0.6)	71.51 (0.08)	97.33 (0.12)
PFSV5	62.20 (3.22)	94.59 (1.34)	80.58 (1.57)	82.90 (1.04)	71.74 (0.07)	97.34 (0.13)
PFSV6	62.44 (3.4)	95.24 (2.08)	82.41 (1.2)	82.63 (2.05)	71.91 (0.08)	97.44 (0.1)
PFSV7	62.53 (4.06)	**95.72 (2.23)**	82.67 (1.94)	83.68 (1.31)	71.94 (0.08)	**97.46 (0.13)**
PFSV8	62.66 (4.83)	95.65 (2.27)	83.07 (2.08)	83.86 (1.05)	72.01 (0.1)	97.31 (0.31)
PFSV9	**64.37 (3.0)**	96.40 (2.28)	83.11 (2.17)	83.76 (1.58)	72.08 (0.1)	97.40 (0.26)
PFSV10	64.03 (3.31)	96.84 (2.35)	**83.90 (1.56)**	84.25 (0.43)	**72.11 (0.1)**	97.46 (0.15)
ELMFSV2	32.43 (0.21)	88.08 (0.82)	47.38 (2.16)	49.6 (0.11)	38.83 (0.02)	81.44 (0.41)
ELMFSV5	49.41 (0.39)	93.85 (1.24)	68.73 (0.46)	70.91 (0.13)	63.18 (0.01)	92.17 (0.16)
ELMFSV10	53.70 (2.2)	94.82 (1.67)	74.04 (1.08)	74.89 (0.14)	66.91 (0.01)	95.63 (0.18)
ELMFSV15	57.09 (2.29)	95.92 (1.56)	78.09 (1.26)	78.73 (0.53)	68.62 (0.14)	96.92 (0.11)
ELMFSV25	58.17 (2.32)	97.70 (1.71)	80.49 (1.06)	79.89 (0.42)	70.11 (0.11)	97.31 (0.11)
ELMFSV50	59.81 (2.89)	99.46 (0.05)	83.12 (1.5)	83.98 (0.32)	72.76 (0.09)	97.51 (0.11)
ELMFSV75	58.06 (4.27)	**100.0 (0.0)**	83.47 (1.63)	84.48 (0.3)	73.85 (0.09)	97.7 (0.11)
ELMFSV100	59.42 (5.79)	100.0 (0.0)	84.43 (1.52)	84.56 (0.4)	74.43 (0.1)	97.71 (0.13)
ELMFSV200	**73.49 (0.47)**	100.0 (0.0)	**85.17 (2.57)**	**85.38 (0.41)**	**81.77 (0.11)**	**97.41 (0.12)**
ELMFSV-DLF2	55.17 (5.35)	94.86 (1.32)	78.26 (1.3)	78.62 (0.62)	68.24 (0.12)	96.5 (0.15)
ELMFSV-DLF5	56.04 (5.43)	94.82 (1.04)	79.15 (1.39)	79.05 (0.57)	69.21 (0.12)	97.21 (0.18)
ELMFSV-DLF10	57.27 (4.7)	95.79 (1.32)	79.77 (1.1)	79.88 (0.42)	69.87 (0.09)	97.13 (0.11)
ELMFSV-DLF15	58.61 (4.63)	97.04 (1.24)	81.26 (1.25)	81.75 (0.4)	70.66 (0.12)	97.37 (0.17)
ELMFSV-DLF25	60.43 (4.24)	98.15 (1.46)	82.41 (1.13)	82.22 (0.45)	71.31 (0.28)	97.51 (0.12)
ELMFSV-DLF50	60.32 (3.61)	99.62 (0.03)	83.51 (1.6)	84.38 (0.4)	76.48 (0.09)	97.60 (0.09)
ELMFSV-DLF75	59.54 (5.68)	**100.0 (0.0)**	83.74 (1.65)	85.17 (0.37)	78.50 (0.12)	97.64 (0.11)
ELMFSV-DLF100	**60.68 (8.15)**	100.0 (0.0)	84.20 (1.81)	85.32 (0.32)	79.35 (0.09)	**97.79 (0.11)**
ELMFSV-DLF200	73.54 (0.47)	100.0 (0.0)	**84.79 (3.77)**	**85.60 (0.41)**	**82.74 (0.09)**	97.40 (0.15)
ELMFSCA2	51.99 (1.88)	86.4 (2.65)	77.19 (1.15)	80.23 (1.46)	68.94 (0.52)	94.97 (0.6)
ELMFSCA3	53.63 (2.03)	88.94 (2.57)	78.47 (1.27)	81.34 (1.63)	69.59 (0.67)	95.47 (0.69)
ELMFSCA4	53.59 (2.25)	90.88 (1.73)	78.68 (1.13)	81.68 (1.73)	70.01 (0.15)	96.10 (0.37)
ELMFSCA5	55.17 (2.51)	92.51 (0.67)	79.18 (1.35)	81.66 (1.2)	70.17 (0.19)	96.27 (0.24)
ELMFSCA10	**56.10 (2.28)**	**93.74 (0.97)**	**80.24 (1.14)**	**82.00 (0.42)**	**70.42 (0.16)**	**96.42 (0.23)**
ELMFSCA-DLF2	56.95 (5.44)	94.63 (0.83)	79.2 (1.45)	81.41 (1.41)	69.73 (0.51)	96.54 (0.24)
ELMFSCA-DLF3	56.01 (3.05)	94.46 (0.96)	79.41 (1.6)	81.32 (1.55)	70.09 (0.45)	96.57 (0.17)
ELMFSCA-DLF4	56.30 (5.35)	**94.70 (1.02)**	79.96 (1.49)	82.37 (1.64)	70.16 (0.36)	96.58 (0.19)
ELMFSCA-DLF5	57.02 (5.27)	94.29 (1.24)	79.90 (1.48)	81.75 (0.4)	70.35 (0.17)	96.60 (0.3)
ELMFSCA-DLF10	**57.93 (4.72)**	94.53 (1.02)	**80.45 (1.34)**	**82.14 (0.47)**	**70.45 (0.13)**	**96.83 (0.2)**
KFSV	66.67 (0.97)	95.8 (0.12)	84.72 (0.75)			97.29 (0.12)
KFSVPoly2	64.03 (1.42)	96.46 (0.23)	80.87 (1.79)			97.26 (0.12)
KFSVPoly3	67.92 (1.29)	97.55 (0.18)	84.79 (1.57)			97.59 (0.22)
KFSVPoly4	**71.32 (0.93)**	**98.17 (0.09)**	**88.34 (1.35)**			**98.09 (0.09)**
NN			85.48 (3.09)	**85.51 (3.18)**	75.71 (1.08)	

Table 5.4: Mean an standard deviation of the explained variance obtained on the training dataset over 10 repetitions when $k = 5$ variables are selected with the various algorithms. The experiment is described in section 5.3.6.2.

	CC	Arcene	Parkinson	Magic	Nino	J2MRed
PCA	60.58 (4.33)	95.65 (0.73)	81.58 (1.06)	85.53 (0.53)	74.91 (0.06)	97.44 (0.1)
FSV	53.97 (5.12)	93.64 (1.55)	77.77 (1.24)	78.22 (0.43)	67.77 (0.11)	96.12 (0.17)
PFSCA2	**55.92 (3.12)**	**93.66 (2.24)**	**79.52 (1.4)**	**82.77 (2.31)**	69.68 (0.14)	**96.48 (0.17)**
PFSCA3	<0	92.84 (2.2)	78.91 (1.51)	80.75 (2.16)	70.34 (0.14)	90.41 (13.68)
PFSCA4	<0	<0	<0	<0	70.67 (0.12)	<0
PFSCA5	<0	<0	<0	<0	71.01 (0.11)	<0
PFSCA6	<0	<0	<0	<0	<0	<0
PFSCA7	36.93 (23.44)	<0	<0	<0	**71.22 (0.12)**	<0
PFSCA8	<0	<0	<0	<0	<0	<0
PFSCA9	<0	<0	<0	<0	<0	<0
PFSCA10	<0	<0	<0	<0	<0	<0
PFSV2	**56.57 (2.84)**	94.13 (1.39)	80.17 (1.27)	**84.84 (2.6)**	69.73 (0.14)	97.04 (0.13)
PFSV3	<0	94.52 (1.33)	80.23 (1.64)	82.82 (0.45)	70.44 (0.15)	97.22 (0.14)
PFSV4	<0	94.53 (1.57)	80.73 (1.68)	83.02 (0.6)	71.51 (0.08)	97.33 (0.12)
PFSV5	<0	**94.59 (1.34)**	80.58 (1.57)	82.9 (1.04)	71.74 (0.07)	97.34 (0.13)
PFSV6	<0	93.62 (2.11)	**81.24 (2.47)**	81.63 (3.32)	71.91 (0.08)	97.44 (0.1)
PFSV7	<0	92.90 (3.03)	79.07 (5.44)	70.68 (27.91)	71.94 (0.08)	**97.46 (0.13)**
PFSV8	<0	92.09 (6.16)	67.73 (24.45)	75.94 (9.06)	72.01 (0.1)	97.31 (0.31)
PFSV9	<0	73.54 (60.78)	<0	19.82 (116.69)	72.08 (0.1)	97.23 (0.54)
PFSV10	<0	76.67 (50.49)	<0	1.19 (121.03)	**72.11 (0.1)**	96.40 (2.15)
ELMFSV2	32.66 (4.06)	86.37 (3.25)	46.69 (3.45)	49.67 (0.7)	38.77 (0.08)	81.46 (2.87)
ELMFSV5	47.65 (2.44)	92.96 (1.48)	68.81 (1.39)	70.88 (0.64)	63.14 (0.13)	92.10 (1.0)
ELMFSV10	52.59 (3.85)	**93.96 (1.15)**	73.55 (1.55)	75.02 (0.59)	66.89 (0.1)	95.52 (0.23)
ELMFSV15	54.77 (2.23)	92.62 (4.25)	78.13 (1.47)	78.74 (0.52)	68.62 (0.14)	96.92 (0.11)
ELMFSV25	57.53 (2.28)	90.74 (8.12)	80.49 (1.06)	79.89 (0.42)	70.11 (0.11)	97.31 (0.11)
ELMFSV50	**59.87 (3.02)**	<0	83.12 (1.5)	83.98 (0.32)	72.76 (0.09)	97.51 (0.11)
ELMFSV75	57.70 (3.37)	<0	83.47 (1.63)	84.48 (0.3)	73.85 (0.09)	97.70 (0.11)
ELMFSV100	56.77 (4.56)	<0	**84.43 (1.52)**	84.56 (0.4)	74.43 (0.1)	**97.71 (0.13)**
ELMFSV200	12.70 (26.78)	<0	84.12 (2.5)	**85.38 (0.41)**	**81.77 (0.11)**	97.41 (0.12)
ELMFSV-DLF2	54.77 (5.22)	94.27 (1.27)	78.26 (1.3)	78.43 (0.74)	68.24 (0.12)	96.50 (0.15)
ELMFSV-DLF5	56.04 (5.43)	**94.73 (0.94)**	79.15 (1.39)	79.05 (0.57)	69.21 (0.12)	97.21 (0.18)
ELMFSV-DLF10	57.27 (4.7)	94.39 (1.83)	79.77 (1.1)	79.88 (0.42)	69.87 (0.09)	97.13 (0.11)
ELMFSV-DLF15	58.61 (4.63)	91.53 (7.2)	81.26 (1.25)	81.75 (0.4)	70.66 (0.12)	97.37 (0.17)
ELMFSV-DLF25	**60.43 (4.24)**	76.82 (35.25)	82.41 (1.13)	82.22 (0.45)	71.31 (0.28)	97.51 (0.12)
ELMFSV-DLF50	60.32 (3.61)	<0	83.51 (1.6)	84.38 (0.4)	76.48 (0.09)	97.6 (0.09)
ELMFSV-DLF75	56.56 (3.23)	<0	83.74 (1.65)	85.17 (0.37)	78.50 (0.12)	97.64 (0.11)
ELMFSV-DLF100	53.00 (4.38)	<0	**84.2 (1.81)**	85.32 (0.32)	79.35 (0.09)	**97.79 (0.11)**
ELMFSV-DLF200	<0	<0	82.00 (3.8)	**85.60 (0.41)**	**82.74 (0.09)**	97.40 (0.15)
ELMFSCA2	50.45 (3.66)	85.37 (2.66)	77.14 (1.77)	80.23 (1.46)	68.93 (0.55)	94.84 (0.87)
ELMFSCA3	52.41 (4.15)	87.82 (2.97)	78.59 (1.93)	81.34 (1.63)	69.59 (0.67)	95.34 (1.05)
ELMFSCA4	52.29 (4.17)	89.75 (2.7)	78.56 (1.1)	81.68 (1.73)	70.01 (0.15)	96.06 (0.58)
ELMFSCA5	54.26 (3.47)	91.35 (1.59)	79.24 (1.41)	81.66 (1.2)	70.17 (0.19)	96.24 (0.22)
ELMFSCA10	**55.45 (2.09)**	**92.59 (1.42)**	**80.24 (1.14)**	**82.00 (0.42)**	**70.42 (0.16)**	**96.42 (0.23)**
ELMFSCA-DLF2	54.92 (6.02)	93.72 (1.55)	79.28 (1.52)	81.41 (1.41)	69.73 (0.51)	96.54 (0.25)
ELMFSCA-DLF3	56.67 (5.46)	93.80 (1.62)	79.41 (1.6)	81.32 (1.55)	70.09 (0.45)	96.57 (0.17)
ELMFSCA-DLF4	55.82 (5.39)	**93.95 (1.25)**	79.96 (1.49)	**82.37 (1.64)**	70.16 (0.36)	96.58 (0.19)
ELMFSCA-DLF5	56.84 (5.22)	93.90 (1.21)	79.90 (1.48)	81.75 (0.4)	70.35 (0.17)	96.60 (0.3)
ELMFSCA-DLF10	**57.93 (4.72)**	93.76 (1.34)	**80.45 (1.34)**	82.14 (0.47)	**70.45 (0.13)**	**96.83 (0.2)**
KFSV	53.52 (7.3)	93.85 (1.34)	75.50 (3.89)			96.77 (0.32)
KFSVPoly2	**56.17 (2.18)**	94.84 (1.04)	**79.00 (1.03)**			97.11 (0.23)
KFSVPoly3	<0	**95.17 (1.38)**	76.85 (4.66)			97.34 (0.23)
KFSVPoly4	<0	95.02 (1.32)	<0			**97.56 (0.33)**
NN			85.48 (3.09)	85.51 (3.18)	75.71 (1.08)	

Table 5.5: Mean an standard deviation of the explained variance obtained on the test dataset over 10 repetitions when $k = 5$ variables are selected with the various algorithms. The experiment is described in section 5.3.6.2.

5.4 Conclusion

This chapter has provided a comprehensive presentation of nonlinear unsupervised variables selection algorithms. In the first part of the chapter, an introduction to neural networks with a particular focus on ELM is presented. Methodologies for initializing the weights and choosing the optimal number of nodes in an Extreme Learning Machine regression problem are presented. The proposed methodologies tune the parameter without requiring user intervention. This is a necessary condition for industrial application. In addition, it is shown that the performance of ELMs is often improved with the addition of Direct Linear Feed Through. The chapter then focuses on nonlinear extensions of the FSCA and FSV algorithms originally presented in Chapter 4. A general framework for data decomposition and reconstruction in a nonlinear scenario is presented based on both forward selection of variables and on forward selection of components. These frameworks are then used by several algorithms based on Polynomial Regression, Extreme Learning Machine Neural Networks, Kernel Regression and Multilayer (Deep Learning) Neural Networks. These algorithms are described in detail and compared on real and simulated datasets. Among all the algorithms ELMFSV-DLF achieves the best results in many situations. It is interesting to note that PCA can often be outperformed by forward variables selection methods if nonlinearities are considered.

Chapter 6

Anomaly Detection and the Dimensionality Reduction Problem

6.1 Introduction

In a semiconductor manufacturing context, the ability to detect faults early during the production process reduces the number of incorrectly processed wafers and directly translates into improved overall process yield and throughput [188]. As a result, fault or anomaly detection is an active area of research within the semiconductor manufacturing environment. Some examples are [189], [188] and [190] where clustering is used to separate normal and anomaly samples. Closer to our application area is the work of [191], [192] and [193] where, starting from the measurements of some characteristic of the wafer, a Gaussian Process is used to model the standard geometric profile of a wafer. New wafers are then classified as normal or as anomaly according to how close to the default profile they are. The focus in this chapter is on anomaly detection with OES data. This has previously been considered in [23] and [194] where anomaly detection in OES time series is performed with unsupervised random forest and one class support vector machines (OCSVM).

OES data is generally characterized by high dimension [89], which poses a problem for anomaly detection algorithms. In addition to the exponential growth in computational complexity with dimension, a major challenge is the Curse of Dimensionality [195], [196], [197], [198]. In this chapter the aspect of the "Curse of Dimensionality" we are concetrated with is the fact that distance measures become unreliable in high dimensional spaces. This

is formalized in [199] with the following theorem:

Theorem 6.1.1. *Consider the hypothesis that the ratio of the variance of the length ($\| \cdot \|$) of any point vector ($\vec{\mathbf{x}}_d \in \mathbb{R}^d$) to the length of the mean point vector (denoted by $E[\| \vec{\mathbf{x}}_d \|]$) converges to zero with increasing data dimensionality. (This assumption covers a broad range of data distributions and distance measures (generally: all L_p norms with $p \geq 1$)). Then the proportional difference between the farthest point distance D_{max} and the closest point distance D_{min} (the relative contrast) vanishes. Formally:*

$$If \lim_{d \to \infty} Var\left(\frac{\| \vec{\mathbf{x}}_d \|}{E[\| \vec{\mathbf{x}}_d \|]}\right) = 0 \quad then \quad \frac{D_{max} - D_{min}}{D_{min}} \to 0 \tag{6.1}$$

From the previous theorem it follows that in high dimensions the data is sparse with large distances between samples and that algorithms based on a distance measure between points become unreliable [200]. The work is then oriented toward algorithms that scale well to high dimensional data (in the sense that are not affected by the Curse of Dimensionality) and towards addressing the impact of dimensionality reduction in anomaly detection.

In chapters 4 and 5 the dimensionality reduction problem was considered. The current chapter analyses the effect of high dimension and dimensionality reduction in anomaly detection. The chapter makes several contributions. Firstly a review of unsupervised anomaly detection algorithms is introduced. In this context it is shown that Unsupervised Random Forest [201] and Extremely Randomized Trees [202] are effective algorithms for anomaly detection with OES data. In addition, a new anomaly diagnosis method based on the Isolation Forest [203] algorithm is proposed that allows the variables causing the anomaly to be identified. In certain situations dimensionality reduction cannot be avoided. For this reason a dimensionality reduction algorithm explicitly designed for anomaly detection is developed. It is then shown how the proposed methods can be used for anomaly detection with the PSI data (Dataset 2.4.1). Finally, it is shown how the Similarity Ratio algorithm (SR, [204]), a method particularly designed for anomaly detection with the OES time series, can be improved trough the use of the formalism introduced in chapter 2 and One Class SVM [205].

The remainder of the chapter is divided into 3 sections. The first section provides an introduction to anomaly detection and a review of the most popular algorithms. The second section of the chapter investigates the effect of dimensionality reduction on anomaly detection. In the final section the methodologies proposed in the first 2 sections of the chapter are applied to the PSI data (Dataset 2.4.1).

6.2 Anomaly Detection

This section provides an introduction to anomaly detection. It starts by providing a definition of anomaly. In order to keep the problem as general as possible the whole discussion is based on a generic dataset D and distance measure d. In general for all our applications we may assume that $D = \mathbb{R}^p$ and d is the Euclidean distance.

6.2.1 Definition of Anomalies

The aim of an anomaly detection system is to detect anomalies or outliers. An intuitive definition of outlier was provided by Hawkins [206].

Definition 6.2.1. Hawkins-Outlier: An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism.

A more formal definition based on distance (Distance Based) is instead provided in [207]:

Definition 6.2.2. $DB(\alpha, d_{min})$-Outlier: An object p in a dataset D is a $DB(\alpha, d_{min})$ outlier if at least percentage α of the objects in D lies greater than distance d_{min} from p, i.e., the cardinality of the set

$$\{q \in D : \ d(p, q) \leq d_{min}\} \tag{6.2}$$

is less than or equal to $(100 - \alpha)\%$ of the size of D.

The above definition only captures certain kinds of outliers. Because the definition takes a global view of the dataset, these outliers can be viewed as "global" outliers. However, for many interesting real-world datasets which exhibit a more complex structure, there are other kinds of outliers. These can be objects that are outlying relative to their local neighbourhoods, particularly with respect to the densities of the neighbourhoods. These outliers are regarded as "local" outliers. An example dataset with global and local outliers is depicted in Figure 6.1.

6.2.1.1 Local Outliers

Breunig et al. [208] provide a formal definition of local outliers. The key difference between this notion and the previous notions of outliers is that being outlying is not a binary property. Instead, it assigns to each object an outlier factor, which is the degree to which the object is considered outlying. The proposed local outlier factor (LOF) is mathematically described in equation 6.6 but some preliminary definitions are required.

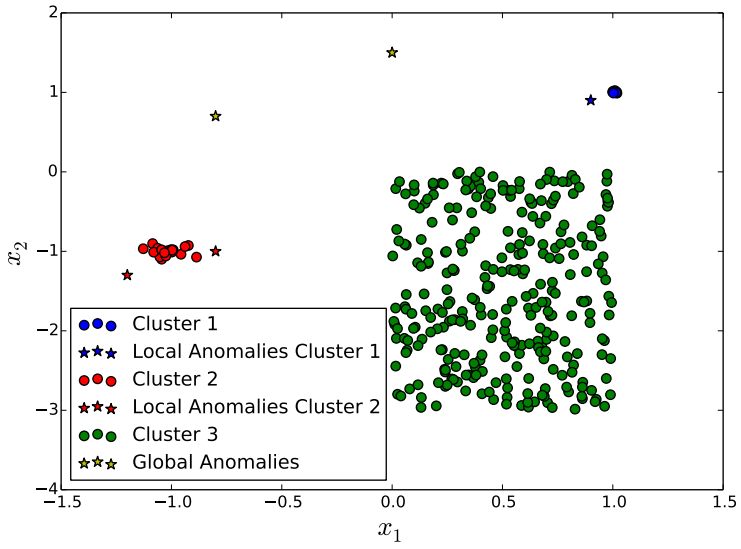


Figure 6.1: A 2 dimensional dataset defined by a main cluster (green) and two smaller clusters (red and blue). Global anomalies are represented with yellows stars while local anomalies associated with the smaller clusters are respectively represented with blue and red stars.

Definition 6.2.3. (k-distance of an object p) For any positive integer k, the k-distance of object p, denoted as $k\text{-}distance(p)$, is defined as the distance $d(p, o)$ between p and an object $o \in D$ such that:

i) for at least k objects $o' \in D/p$ it holds that $d(p, o') \leq d(p, o)$.

ii) for at most $k - 1$ objects $o' \in D/p$ it holds that $d(p, o') < d(p, o)$.

Definition 6.2.4 (k-distance neighbourhood of an object p)**.** Given the k-distance of p, the k-distance neighbourhood of p contains every object whose distance from p is not greater than the k-distance, i.e.

$$N_k(p) = \{q \in D/p : \ d(p, q) \leq k\text{-distance}(p)\} \tag{6.3}$$

The objects q are called the k-nearest neighbours of p.

Definition 6.2.5. (reachability distance of an object p w.r.t. object o) Let k be a natural number. The reachability distance of object p with respect to object o is defined as

$$\text{reach-dist}_k(p, o) = max\{k\text{-distance}(o), d(p, o)\} \tag{6.4}$$

Intuitively, if object p is far away from o, then the reachability distance between the two is simply their actual distance. However, if they are "sufficiently" close, the actual distance is replaced by the k-distance of o. An illustration of this concept (reproduced from the original LOF paper [208]) is provided in Figure 6.2

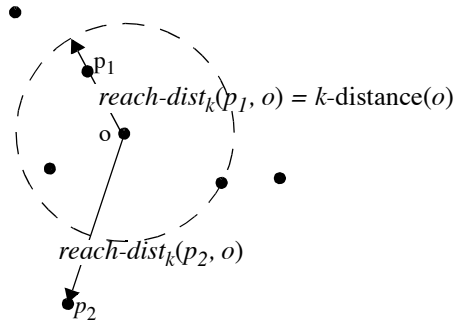


Figure 6.2: reach-dist(p_1, o) and reach-dist(p_2, o), for $k = 4$. Figure from [208].

Definition 6.2.6. The local reachability density of p is defined as the inverse of the reachability distance that is:

$$lrd_v(p) = \left(\frac{\sum_{o \in N_{v(p)}} \text{reach-dist}_v(p, o)}{|N_v(p)|} \right)^{-1} \tag{6.5}$$

Definition 6.2.7. It is finally possible to define the Local Outlier Factor (LOC) for a point p as:

$$LOF = \frac{\sum_{o \in N_v(p)} \frac{lrd_v(o)}{lrd_v(p)}}{N_v(p)} \tag{6.6}$$

$lrd_v(p)$ is a measure of the density of points around p. LOF is the value of the average density of data and the v neighbours of p and p itself. It is clear to see that if the density of the data aound p is much less than that of its v nearest neighbours then LOF for p will be large.

In [208] the authors use the LOF factor to detect anomalies. While the method is important because it provides a formal definition of local outliers and the idea to use an outliers score instead of a binary classifier this will

not be further discussed in this thesis because it has been outperformed by more recent methods both in computing time and accuracy [203].

To complete the classification of the different kind of anomaly the definition of contextual anomaly is now reported:

Definition 6.2.8 (Contextual Anomalies)**.** The term Contextual Anomaly is reported in [209] to describe a particular case of outlier. If a data instance is anomalous in a specific context (but not otherwise), then it is termed as a contextual anomaly (also referred to as conditional anomaly). The notion of a context is induced by the structure in the data set and has to be specified as a part of the problem formulation. For example the time in a time series dataset is an attribute that defines a context.

6.2.2 Introduction to Anomaly Detection Algorithms

Anomaly detection is a classical field of study. Over the years numerous anomaly detection algorithms have been developed. According to [210] anomaly detection algorithms can be classified in three main groups:

- **Supervised Anomaly Detection**. In this case the anomaly detection problem is reduced to a classification task. It assumes the availability of a training data set which has labeled instances for normal as well as anomaly behaviour in which case standard classifiers such as Linear Discriminant Analysis (LDA), Support Vector Machines (SVM) and k-nearest neighbours can be trained to distinguish between normal and anomalous samples [210].

- **Semi-supervised Anomaly Detection**. Assume that the training data has labeled instances for only the normal class. Systems can then be trained to assign an anomaly score to new samples according to how distant they are from the normal behaving ones. Several algorithms have been developed with this aim, including Multivariate Control Charts [211], one-class SVMs [205] and Unsupervised Random Forests [201], [23]. While some methods are explicitly designed for unsupervised anomaly detection, it is possible to transform an unsupervised problem to a supervised one using the technique described in [201]. Since Semi-Supervised anomaly detection methods do not require labels for the anomaly class, they are more widely applicable than supervised techniques.

- **Unsupervised Anomaly Detection**. Techniques that operate in unsupervised mode do not require labelled data and thus are most

widely applicable. The techniques in this category make the implicit assumption that normal instances are far more frequent than anomalies in the test data. This structure is then revealed and potential anomalies identified though the application of unsupervised clustering techniques such as DBSCAN [212] and Max Separation clustering [90] or through isolation based methods such as Isolation Forest [203].

6.2.2.1 Unsupervised Anomaly Detection

In semiconductor manufacturing measurements of normal behaving wafers are in general available while often no information regarding the anomaly data is available. For this reason in this thesis the focus is on semi-supervised and unsupervised techniques as they do not require data labelled as anomaly. In addition, due to the high dimension of the OES datasets, of particular importance are algorithms that can scale to large and high dimensional datasets. Unless otherwise specified, in the reminder of the thesis unsupervised algorithm will be used to refer to both unsupervised and semi-supervised algorithms. Unsupervised algorithms for anomaly detection can be divided in 4 main groups. Good descriptions of these groups are reported in [208] and [213].

- **Distribution-Based**: In distribution-based methods a standard distribution (e.g. Normal, Poisson, etc.) is used to fit the data. Outliers are then defined based on the probability distribution. Over one hundred tests in this category, called discordancy tests, have been developed for different scenarios [214]. A key drawback of this category of tests is that most of the distributions used are univariate. Some examples of multivariate tests are described in [215]. In addition, for the majority of applications, the underlying distribution is unknown. Fitting the data with standard distributions is costly, and may not produce satisfactory results.

- **Depth-Based**. Each data object is represented as a point in a p-dimensional space, and is assigned a depth. With respect to outlier detection, outliers are more likely to be data objects with smaller depths. There are many definitions of depth that have been proposed such as the Mahalanobis depth [216] and the convex hull peeling depth [217]. In theory, depth-based approaches could work for large values of p. However, in practice, while there exist efficient algorithms for $p = 2$ or 3 [218], [219], depth-based approaches become inefficient for large datasets for $p \geq 4$. This is because depth-based approaches rely on the

computation of p-dimensional convex hulls which have a lower bound complexity of $O(np/2)$ for n objects.

- **Cluster-Based**. Most clustering algorithms (e.g. DBSCAN [212], BIRCH [220]), are to some extent capable of handling exceptions. However, since the main objective of a clustering algorithm is to find clusters, they are developed to optimize clustering, and not to optimize outlier detection. The exceptions (called "noise" in the context of clustering) are typically just tolerated or ignored when producing the clustering result. Even if the outliers are not ignored, the notions of outliers are essentially binary, and there is no quantification of how outlying an object is.

- **Density-Based**. As previously mentioned this methodology was originally proposed by Markus et al. [208]. It relies on the local outlier factor (LOF) of each object, which depends on the local density of its neighborhood. Algorithms that are part of this group include: LOF [208], LOCI [213], Isolation Forest [203].

In the next sections a review is presented of selected anomaly detection algorithms. The algorithms have been chosen according to their popularity and for their performance on large and high dimensional datasets.

6.3 Unsupervised Anomaly Detection Algorithms

In this section a number of popular anomaly detection algorithms are introduced and are evaluated with respect to anomaly detection with OES data. The J2M datased (Dataset 2.4.2) is used as illustrative case study.

6.3.1 Univariate and Multivariate Control Chart

Univariate and multivariate control charts are among the most popular fault detection algorithms. This is due to their ease of use, simple implementation and numerical stability. Control charts were introduced by Hotelling in 1930-1940 and they are still the state-of-the-art in many industries.

6.3.1.1 Univariate Control Chart

A univariate control chart is an anomaly detection method designed to monitor a single variable. A probability distribution $\mathcal{D}$ is used to model the normal behaviour of the variable. Given the probability distribution of the

normal behaving data an in control interval is defined as $I_\alpha = [L_\alpha, U_\alpha]$ such that $\mathbb{P}(\vec{\mathbf{d}} \in I_\alpha) = 1 - \alpha$ where $\vec{\mathbf{d}} \sim \mathcal{D}$. A new sample $\vec{\mathbf{x}} \in \mathbb{R}$ is considered in control at level α if $\vec{\mathbf{x}} \in I_\alpha$.

Example 6.3.1 (Normal distribution)**.** In many applications the normal behaving samples are modelled with the normal distribution i.e. $\mathcal{D} = N(\mu, \sigma)$ where μ and σ are parameters estimated with the normal data. Figure 6.3 shows the histogram of the frequencies of 10,000 values sampled from a $N(0, 1)$ distribution. Three pairs of vertical lines are plotted corresponding to the I_α interval for three commonly used values of α.

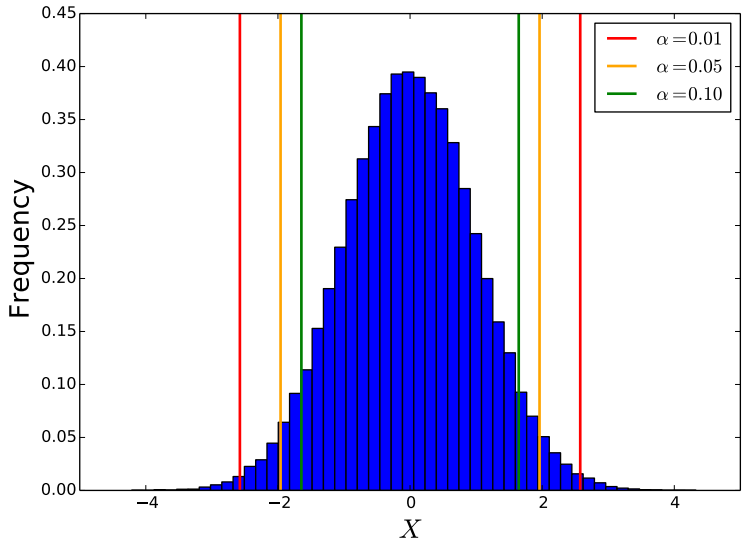


Figure 6.3: The histogram of frequencies for 10,000 samples from a $N(0, 1)$ distribution. Each pair of vertical lines represents the I_α interval for a given value of α.

The univariate control chart is easy to use and interpret, but has one major limitation. It is not able to capture the interaction between variables of a multivariate distribution as shown in the following example.

Example 6.3.2 (Bivariate distribution)**.** Figure 6.4 shows a two dimensional dataset $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2)$. The dataset contains an anomaly (represented by a star in the figure). The limits obtained with two univariate control charts respectively, for variable $\mathbf{x}_1$ and $\mathbf{x}_2$, are also shown. A sample is considered normal if it is inside the black square in the top plot and if it is between the two vertical lines in the second figure. Clearly it is impossible to distinguish the anomaly from the normal data as it is in the region where samples

are labelled as normal. It follows that it is impossible to detect this kind of anomaly with univariate control charts due to their inability to model interactions between variables.

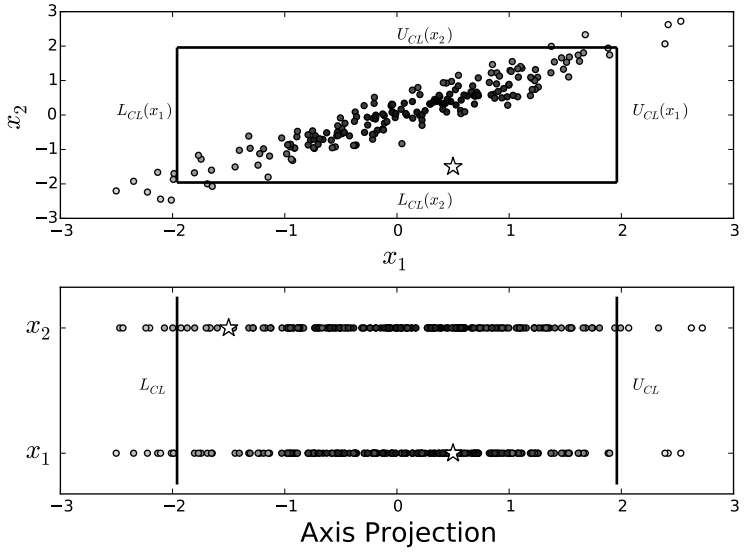


Figure 6.4: A scatter plot of a 2-dimensional dataset and its projection onto its two axes. The white star represents an anomaly. The 95% control limit obtained with the two univariate control charts is represented by black lines as explained in Example 6.3.2.

To address this limitation problem multivariate control charts have been developed as an extension of the univariate control chart.

6.3.1.2 Multivariate Control Chart

Multivariate Control Charts (MCC) are the natural extension of the Univariate Control Chart to multidimensional datasets. Several versions of (MCC) have been proposed in the literature. Among these one of the most popular is the Hotelling T^2 control chart [221]. In the most common form of MCC the normal data is modelled as a multivariate normal distribution $\mathbf{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \mathbf{X} \in \mathbb{R}^{n \times p}$. Given a new sample $\vec{\mathbf{y}} \in \mathbb{R}^p$ the distance from the normal data d^2 is computed as:

$$d^2(\mathbf{X}, \vec{\mathbf{y}}) = (\vec{\mathbf{y}} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\vec{\mathbf{y}} - \boldsymbol{\mu}) \tag{6.7}$$

Figure 6.5 considers the same dataset as used in Example 6.3.2. This time each sample $\vec{\mathbf{y}}$ is coloured as $log(d^2(\mathbf{X}, \vec{\mathbf{y}}))$. As can be seen the MCC al-

gorithm is correctly able to assign an high anomaly score to the sample at coordinates $(0.5, -1.5)$.

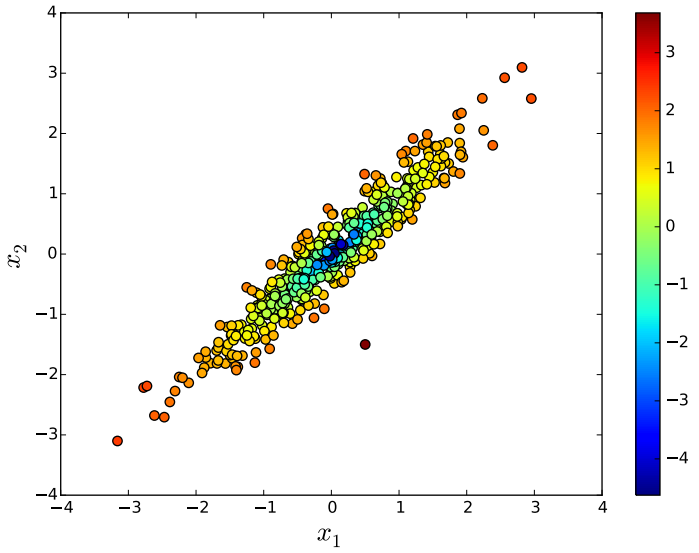


Figure 6.5: The same dataset considered in Example 6.3.2. The color represent the value $log(d^2(\mathbf{X}, \vec{\mathbf{y}}))$ obtained for each sample.

6.3.1.3 MCC and the Curse of Dimensionality

The anomaly score d^2 obtained by MCC is based on the Euclidean distance. Due to the curse of dimensionality [198], the Euclidean distance becomes unreliable as a similarity metric in high dimensional spaces. It follows that MCC is an inefficient method, when the number of variables is large as shown in the following example.

Example 6.3.3 (MCC high dimensional data)**.** Consider the matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{100}) \in \mathbb{R}^{500\times 100}$ where the variables $\mathbf{x}_1, \ldots, \mathbf{x}_{100} \sim N(0, 1)$ $i.i.d.$. An anomaly is introduced by setting the value of the 100^{th} variable of the final sample to 8, i.e. $x_{500,100} = 8$. Figure 6.6 shows the d^2 value assigned by MCC to each sample. It is easy to observe that the last sample represented by a star has a lower d^2 value than many other samples.

OES data is often characterized by high dimension. In order to apply MCC to the OES data a dimensionality reduction step is required. The next example illustrates how the performance of MCC changes when the dimension of the data is first reduced with PCA.

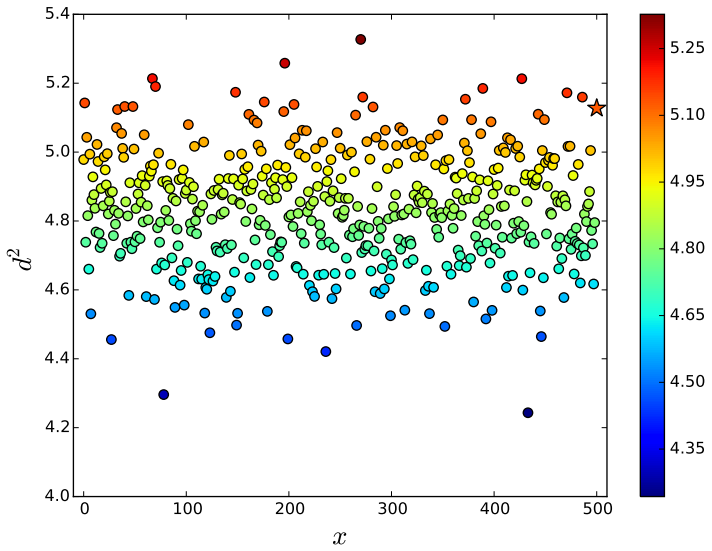


Figure 6.6: The d^2 values obtained with MCC for each sample for the data described in Example 6.3.3. The last sample in the right is the anomaly and it is represented with a star (coordinates $(500, 5.2)$)

Example 6.3.4. Consider $\mathbf{X} \in \mathbb{R}^{2194\times 12277}$ the OES spectrum of the J2M data (dataset 2.4.2). The dimension of the data is reduced with PCA by selecting the first $k = 10$ and then $k = 700$ components. This is done to avoid having more features than samples and to ensure that the $\mathbf{\Sigma}$ matrix is of full rank. 800 wafers are sampled among the ones with normal values of ER and they are used to train a MCC. The d^2 value is then evaluated for all the data. The ER values coloured according to their d^2 value are reported in Figure 6.7. It can be easily observed that when $k = 10$ MCC is able to recognize the anomaly samples and the small shift during production. When $k = 700$ MCC is still able to recognize the anomaly (orange) but it is not able to provide good separation between the normal data and the small shift.

Observation 6.3.1. *In the previous example it is shown that the performances of MCC improves if the dimension of the data is reduced with PCA. Instead of applying MCC to the original matrix* $\mathbf{X} \in \mathbb{R}^{n\times p}$ *it is applied to the score matrix*

$$\mathbf{T}_k = \mathbf{X}\mathbf{P}_k \in \mathbb{R}^{n\times k} \quad k << p \tag{6.8}$$

This is equivalent to the T^2 *statistic described in chapter 4 (Section 4.3.2.4). In the same chapter the* Q *statistic (equation 4.30) was also defined. In an anomaly detection context it is used to monitor how well the PCA model*

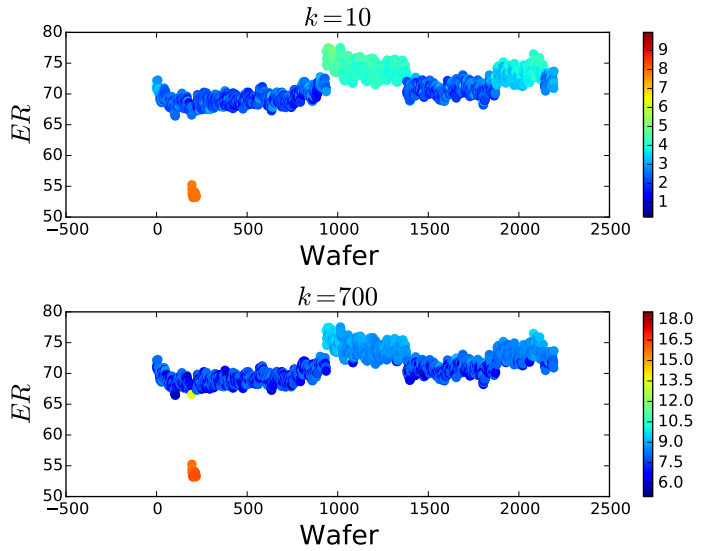


Figure 6.7: The ER for the J2M data. The wafers are coloured according to the d^2 value obtained as described in Example 6.3.4.

monitors the new samples. If Q is too large the anomaly detection system is not effective anymore as too much information is not being modelled.

In the next example the J2M dataset is summarized using only its first two principal components. The resulting two dimensional dataset can be graphically represented and this makes it easier to interpret the model and the process behaviour.

Example 6.3.5 (Elliptic Envelope J2M)**.** A 2 components PCA decomposition is performed on the normal behaving samples of the J2M data (dataset 2.4.2). Figure 6.8 shows the 2 dimensional projection of all the J2M samples coloured according to their ER value. In the Figure the T^2 value is represented by concentric ellipses. It can be observed that samples with an ER which is either too large or too small are far from the center of the ellipse. The samples with small and high ER values are respectively in the upper right and lower left of the plot. This suggests that the process variations are caused by two different mechanisms.

Observation 6.3.2. *While the use of PCA as a preprocessing step with MCC is very common, better performance may be obtained through the use of nonlinear dimensionality reduction. This is for example investigated in [222] where the dimension of the data is reduced with kernel PCA.*

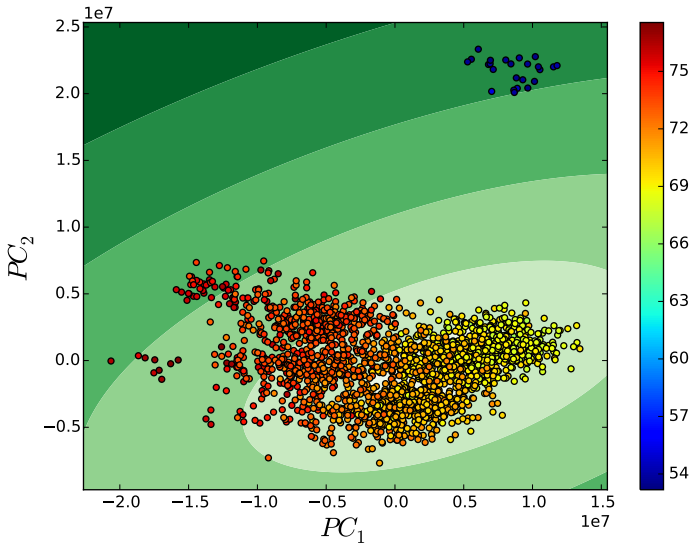


Figure 6.8: The T^2 values for the J2M datasets when represented by its first two principal components.The PC projection was computed using only the samples with normal ER values. All samples are then mapped based on this projection. The samples are coloured according to their ER value.

6.3.2 Clustering-Based Methods

In unsupervised fault detection clustering algorithms are a very popular option. Their aim is to detect the clusters of normal behaving samples and label as anomaly all the samples that are far from these clusters. In this section two of the most popular clustering algorithms are described. The clustering algorithms are by their nature based on a distance measure. This makes then less effective when high dimensional data is used as they are impacted by the curse of dimensionality [223].

6.3.2.1 Self-Organizing Map

The Self Organizing Map (SOM) [224] may be described as a nonlinear, ordered and smooth mapping of high dimensional input data into the elements of a regular low dimensional array (grid). The SOM algorithm assigns to each sample a neuron of the grid. The closer two samples are in the original p dimensional space, the closer their neurons are in the grid. A graphical representation of a self organizing map is shown in Figure 6.9. The grid is composed of M neurons with each one associated with a subset of the data. This is equivalent to obtaining M clusters. Some more complex algorithms

use a two step clustering procedure. In these cases the number of SOM clusters is reduced by merging those that are similar. Some of the techniques used for the second clustering step are described in [225].

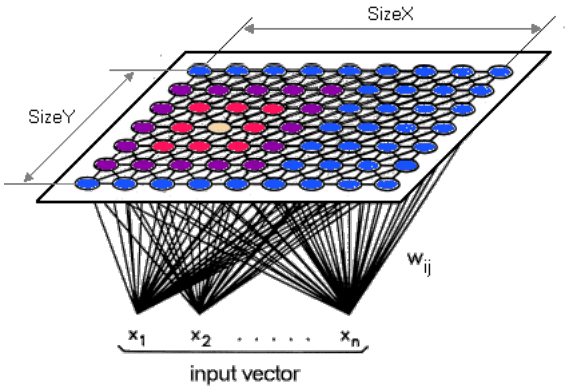


Figure 6.9: A graphical representation of SOM.

Algorithm 6.3.1 (SOM)**.** Consider the input data $\mathbf{X} \in \mathbb{R}^{n \times p}$ and the grid $\mathcal{G}$ defining a discrete 2 dimensional space of cardinality M (for simplicity it is assumed that $M = k^2$).

$$\mathcal{G} = \{\mathbf{r}_i \in \mathbb{Z}_k \times \mathbb{Z}_k : i = 1, \ldots, M\} \tag{6.9}$$

The SOM algorithm is defined by a base step that is recursively repeated T times. It starts with a set of centroids $\{\vec{\mathbf{m}}_i \in \mathbb{R}^p : \ i = 1, \ldots, M\}$ and to each one of them is associated a position on a $k \times k$ grid. A neighbourhood function is then defined as

$$h_{c,i} = \alpha(t) exp\left(-\frac{\| \mathbf{r}_i - \mathbf{r}_c \|^2}{2\sigma^2(t)}\right) \tag{6.10}$$

where $\alpha(t)$ is the learning-rate factor, which decreases monotonically with the regression steps and $\sigma(t)$ corresponds to the width of the neighbourhood function, which also decreases monotonically with the regression steps.
For each sample $\vec{\mathbf{x}} \in \mathbb{R}^p$ the M vectors are recursively updated as

$$c_{\vec{\mathbf{x}}} = \underset{i=1,\ldots,M}{\operatorname{argmin}} \| \vec{\mathbf{x}} - \vec{\mathbf{m}}_i \|^2 \tag{6.11}$$

$$\vec{\mathbf{m}}_i = \vec{\mathbf{m}}_i + h_{c_{\vec{\mathbf{x}}},i}(\vec{\mathbf{x}} - \vec{\mathbf{m}}_i) \tag{6.12}$$

and the process is recursively repeated T times.

The SOM can be used to cluster the data. This is done by assigning to each sample $\vec{\mathbf{x}}$ its closest centroid $\vec{\mathbf{m}}$. SOM also provides a 2-dimensional grid graphical representation of the data with each centroid assigned a position in the grid. Figure 6.10 shows the positions of the centroids at 4 different iterations of the SOM algorithm in a 2 dimensional dataset.

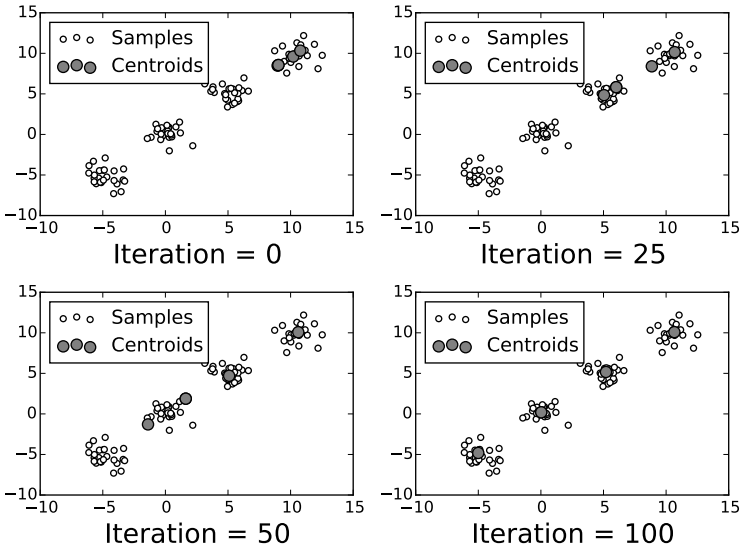


Figure 6.10: The position of the centroids ($\vec{\mathbf{m}}_i,\ i = 1, \ldots, 4$) at 4 different iterations of the SOM algorithm for a sample two dimensional dataset.

6.3.2.2 K-Nearest Neighbours

k-Nearest Neighbour (KNN) is another popular clustering based approach. While KNN is in general considered a supervised algorithm it is used for unsupervised anomaly detection in Verdier et al. [226]. In the paper the authors show that KNN has better performance than a MCC. According to the authors the reason for this is that in general MCC is applied under the hypothesis that the data follows a normal distribution which is often not true in reality.

Algorithm 6.3.2. (Unsupervised KNN) The unsupervised version of KNN is very similar to the supervised one and is obtained with the following steps:

- Consider $\mathbf{X}$ a matrix whose rows are the samples in control and a new

sample $\vec{\mathbf{z}}$. The distance between $\vec{\mathbf{z}}$ and $\mathbf{X}$ is obtained as

$$d_k(\vec{\mathbf{z}}) = \sum_{i=1}^{k} d^2(\vec{\mathbf{x}}_i, \vec{\mathbf{z}}) \tag{6.13}$$

where $\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_k$ are the k closest samples in $\mathbf{X}$ to $\vec{\mathbf{z}}$ according to the distance d.

- $\vec{\mathbf{z}}$ is then labelled an anomaly if $d_k(\vec{\mathbf{z}})$ is larger than a threshold chosen by the user.

Observation 6.3.3. *Observe the similarity between the KNN distance in equation 6.13 and the local reachability density of the LOF algorithm in equation 6.5. In both the cases the obtained value is the distance of a sample from its neighbours based respectively on the d distance (that is in often the euclidean distance) and the k-distance (Definition 6.2.3). The latter one may be considered a more fine grained anomaly distance as it takes into account the locality of the samples.*

6.3.3 Depth Based Anomaly Detection

In depth based anomaly detection a depth is assigned to each sample or region of the space. The outliers are the samples with the lowest depth. Different definitions of depth are described in the literature. Several of them are described in [227]. Here two of the most popular are reported:

- The Mahalanobis depth [216] of $\vec{\mathbf{x}}$ with respect to the data $\mathbf{X}$ is defined as:
$$MD = \frac{1}{1 + (\vec{\mathbf{x}} - \boldsymbol{\mu}_{\mathbf{X}})\boldsymbol{\Sigma}_{\mathbf{X}}^{-1}(\vec{\mathbf{x}} - \boldsymbol{\mu}_{\mathbf{X}})} \tag{6.14}$$
where $\boldsymbol{\Sigma}_{\mathbf{X}}$ and $\boldsymbol{\mu}_{\mathbf{X}}$ are the covariance matrix and the mean vector of $\mathbf{X}$. Observe the similarity to the multivariate control chart distance (equation 6.7)

- The convex hull peeling depth [217] of a sample $\vec{\mathbf{x}}$ in a dataset $\mathbf{X}$ is the level of convex layer that $\vec{\mathbf{x}}$ belongs to. A convex layer is defined as follows. Construct the smallest convex hull which encloses all sample points in $\mathbf{X}$. The sample points on the perimeter have depth 1 and are removed. The convex hull of the remaining points is constructed; the points on the perimeter of this hull are the second convex layer and have depth 2. The process is repeated, and a sequence of nested convex layers is formed. While this approach can be in theory used for an arbitrary dimensional space, efficient algorithms to compute the hull only exist for small dimensional spaces [208].

Depth based anomaly detection methods are implicitly related to an idea of distance. Indeed it is important to understand the geometrical position of a sample respect to all the others. For this reason depth based methods are affected by the curse of dimensionality making it difficult to use them with highly dimensional datasets.

6.4 Algorithms for High-Dimensional Data

All the anomaly detection algorithms described in the previous section are intrinsically linked to the idea of distance. As previously noted most distance measures become unreliable in high dimensional space. It is therefore important to choose algorithms that can scale well to high dimensional problems. In this section the most popular of these algorithms are presented.

6.4.1 One Class SVM

One Class Support Vector Machine (OCSVM) is probably one of the most popular anomalies detection algorithms. It was originally developed to estimate the support of high dimensional distributions [205] and it has been successfully used for anomaly detection in several fields. Some examples are [228], [229], [230] and [231] where it is respectively used to detect anomaly in time series, image classification, fault detection of wearable devices and novelty detection in chiller systems. In the one-class formulation, data are first mapped into a feature space using an appropriate kernel function and then maximally separated from the origin using a hyperplane. The aim of OCSVM is to define a function f that takes the value $+1$ in a small region containing most of the training data and -1 elsewhere.

Algorithm 6.4.1 (OCSVM)**.** Consider a training dataset $\mathbf{X} \in \mathbb{R}^{n \times p}$, $\mathbf{X} \subset \mathcal{X}$ where $\mathcal{X}$ is a compact subset of $\mathbb{R}^p$ and Φ a map into the dot product space:

$$\Phi(\mathbf{X}) : \mathcal{X} \to F \quad and \quad \Phi(\vec{\mathbf{x}}) \cdot \Phi(\vec{\mathbf{y}}) = \mathcal{K}(\vec{\mathbf{x}}, \vec{\mathbf{y}}) \ \forall \vec{\mathbf{x}}, \vec{\mathbf{y}} \in \mathcal{X} \tag{6.15}$$

The OCSVM optimization problem is then defined as:

$$\min_{\mathbf{w} \in F, \boldsymbol{\xi} \in \mathbb{R}^n, \rho \in \mathbb{R}} \frac{1}{2} \parallel \mathbf{w} \parallel_2^2 + \frac{1}{vn} \sum_{i=1}^{n} \xi_i - \rho \tag{6.16}$$

subject to the constraints:

$$(\mathbf{w} \cdot \Phi(\vec{\mathbf{x}}_i)) \geq \rho - \xi_i \quad i = 1, \ldots, n \quad \xi_i \geq 0 \tag{6.17}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n) \in \mathbb{R}^n$ and v is a tunable parameter that determines the trade-off between minimising the size of the support and maximising the number of training data points within the support.

Since the $\boldsymbol{\xi}$ vector is penalized it is expected that for the $\mathbf{w}$ and ρ solution to the problem the decision function

$$f(\vec{\mathbf{x}}) = sign((\mathbf{w} \cdot \Phi(\vec{\mathbf{x}})) - \rho) \tag{6.18}$$

will be positive for most samples $\vec{\mathbf{x}}_i$. At the same time the $\| \mathbf{w} \|_2^2$ is small forcing $f(\vec{\mathbf{x}}) > 0$ only on a small region. The trade-off between the support of $f(\vec{\mathbf{x}})$ and its positiveness on the training data is regulated by v. The minimization problem defined in equation 6.16 can be solved through the use of Lagrange multipliers and quadratic programming as described in [205]. An anomaly score for a given sample $\vec{\mathbf{x}}$ can then be defined as

$$s(\vec{\mathbf{x}}) = -(\mathbf{w} \cdot \Phi(\vec{\mathbf{x}})) + \rho \tag{6.19}$$

This represents how far $\vec{\mathbf{x}}$ is from the support of the data used to train the model.

Example 6.4.1. Figure 6.11 shows the decision function obtained with OCSVM for a simulated two dimensional example. In the Figure three clusters of samples are represented. The OCVSM is trained on this data using the Gaussian kernel $\Phi(\vec{\mathbf{x}}) = e^{-\|\vec{\mathbf{x}}\|_2^2}$. The colour coded shaded regions represent how far each sample is from the estimated support, as measured in equation 6.19.

6.4.1.1 OCSVM Diagnosis

OCSVM is a black box algorithm. While the algorithm provides an efficient anomaly score, it is difficult to extract information and to understand the reasons behind an anomaly. In [232] a fault diagnosis procedure is proposed. The authors use a SVM based Recursive Feature Elimination procedure (SVM-RFE) defined by the following algorithm:

Algorithm 6.4.2 (OCSVM Recursive Features Elimination)**.** The set of k variables causing the anomaly is estimated with the following steps:

- Variables are ranked according to the change in the cost function (equation 6.16) if all the variables but one are used.
- The lowest ranked variable is removed.

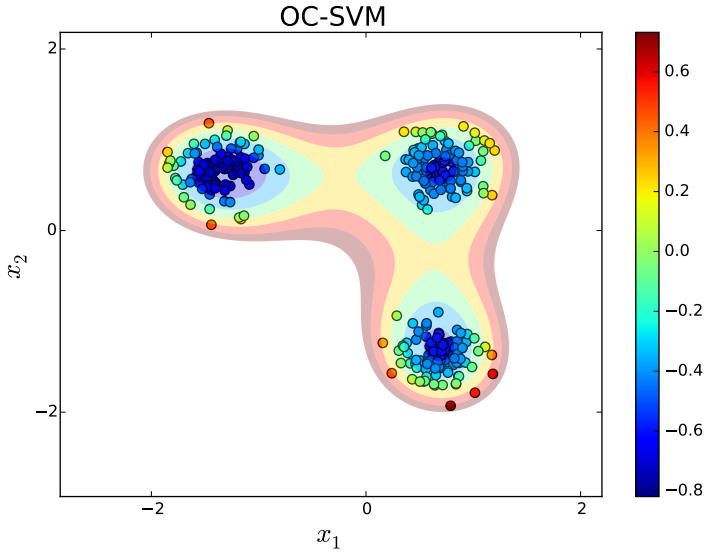


Figure 6.11: OCSVM on a simulated 2 dimensional problem. The samples are coloured according to their anomaly score reported in the lateral bar. The estimated support of the distribution is represented by the shaded area.

- The process is repeated until k variables remain.
- The first three steps can be repeated using different subsamples of the data. Sets of k variables are then obtained and the set with the highest Area Under the Curve (AUC, [233]) score can be used as the set of most relevant variables.

Observation 6.4.1. *It is interesting to observe that the diagnostic system described in Algorithm 6.4.2 has similar characteristics to the unsupervised feature selection algorithms described in chapter 4 (for example FSCA). It is then reasonable to assume that this algorithm can be improved using similar techniques to the ones proposed in Section 4.7.*

6.4.2 Isolation Forest

The Isolation Forest (IF, [203]) is a more recent anomaly detection algorithm based on an isolation procedure. It is based on the idea that anomalies are the minority of the data consisting of only a few instances, and that they have attribute-values that are very different from those of normal instances. Anomalies are therefore more susceptible to a mechanism that the authors call isolation. IF does not rely on any distance or density measure and it

is therefore particularly efficient for high dimensional datasets as it is not influenced by the curse of dimensionality [198].

Algorithm 6.4.3 (Isolation Forest)**.** Isolation forest is an ensemble of isolation trees, similar to the more popular decision trees and random forest which will be introduced in Section 6.5.1. An isolation tree is constructed starting with a matrix $\mathbf{X}$ as described in Pseudocode 6.4.1. An Isolation Forest is then defined by numerous Isolation Trees

$$IF = \{t_1, \ldots, t_T\} \tag{6.20}$$

For each tree t it is possible to compute the number of iterations $h_t(\vec{\mathbf{x}})$ required to isolate a sample $\vec{\mathbf{x}}$. The average number of steps required to isolate a sample $\vec{\mathbf{x}}$ in a forest is then

$$h(\vec{\mathbf{x}}) = \frac{1}{T} \sum_{t \in IF} h_t(\vec{\mathbf{x}}) \tag{6.21}$$

The idea is that only a few steps are required to isolate an anomaly. The number of steps required to isolate an observation $\vec{\mathbf{x}}$ is influenced by the number of samples n in the data. To account for this a normalized anomaly score $s(\vec{\mathbf{x}}, n)$ is defined as:

$$s(\vec{\mathbf{x}}, n) = 2^{-\frac{h(\vec{\mathbf{x}})}{c(n)}} \tag{6.22}$$

where $c(n)$ is:

$$c(n) = \begin{cases} 2H(n-1) - 2(n-1)/n & \text{if } n > 2 \\ 1 & \text{if } n = 2 \\ 0 & otherwise \end{cases} \tag{6.23}$$

and $H(i)$ is the harmonic number estimated as:

$$H(i) \approx ln(i) + 0.5772156649. \tag{6.24}$$

It can be proven that $c(n)$ is the average number of steps required to isolate a sample from the other n samples [203]. In this sense it provides a normalization factor that makes the s value independent of the number of samples (n).

Figure 6.12 shows the $s(\vec{\mathbf{x}})$ and $h(\vec{\mathbf{x}})$ values obtained with the Isolation forest on a simple two dimensional dataset.

Input: Input matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$.

1: $t = \emptyset$ (the empty tree)
2: **if** $nrow(\mathbf{X}) == 1$ **then return** t
3: **end if**
4: Randomly select $\mathbf{x}_i$ a feature of $\mathbf{X}$
5: Randomly select a split point $p \in (\min(\mathbf{x}_i), \max(\mathbf{x}_i))$
6: Add to t the node $N_{\mathbf{x}_i, p}$
7: Define $\mathbf{X}_l$ and $\mathbf{X}_r$ as the matrix composed of the samples of $\mathbf{X}$ where the variable $\mathbf{x}_i$ is respectively larger and smaller than p.
8: Repeat the algorithm with $\mathbf{X} = \mathbf{X}_l$. Link the obtained tree as the left child of t.
9: Repeat the algorithm with $\mathbf{X} = \mathbf{X}_r$. Link the obtained tree as the right child of t.

Pseudocode 6.4.1: Isolation Tree

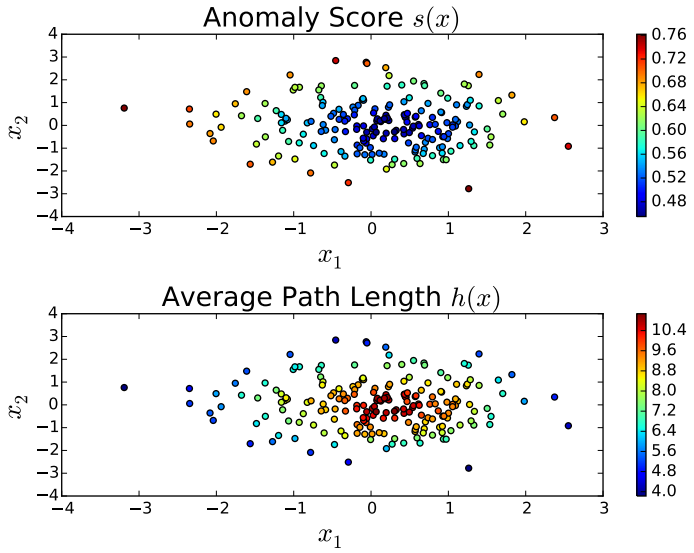


Figure 6.12: Samples in a two dimensional spaces coloured according to the $s(\vec{\mathbf{x}})$ and $h(\vec{\mathbf{x}})$ values obtained with Isolation Forest.

6.4.2.1 Weakness of the Isolation Forest Method

The Isolation Forest algorithm has been shown to be one of the most efficient algorithms both in terms of computational complexity and anomaly detection performance [203]. Despite this the algorithm has several limitations.

Local Anomalies In [234] it is shown that the IF method is not able to recognize local anomalies. This problem is replicated in the following example:

Example 6.4.2. In Figure 6.13 a two dimensional dataset is represented. The dataset is composed of 3 clusters of samples and three global anomalies. All the clusters are composed of 50 samples characterized by different density. The samples in the cluster in the lower right part of the Figure have a lower density. The samples at the bottom of this cluster have a higher anomaly score than some of the global anomalies. The local anomalies in the two smaller clusters have a lower anomaly score than most of the samples in the larger cluster.

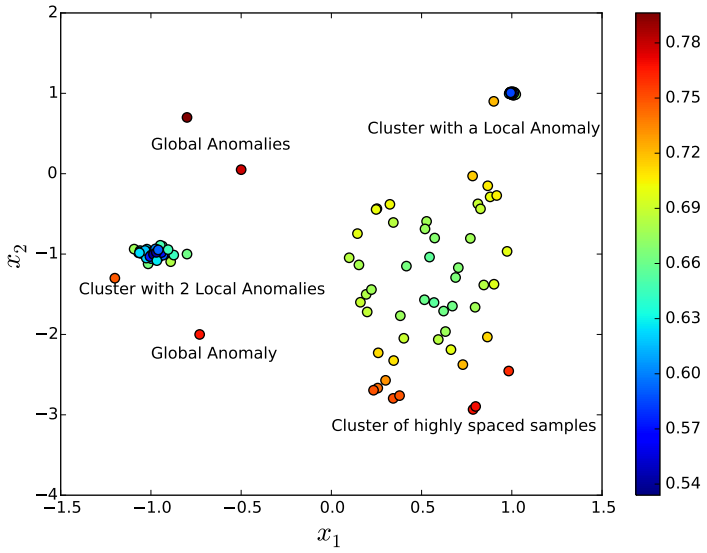


Figure 6.13: A two dimensional dataset containing local and global anomalies. The color represents the anomaly score obtained with the isolation forest method.

The previous example has shown that IF is not able to correctly assign an anomaly score to local anomalies and it is unreliable if the data contains

clusters with different densities. In [234] the authors propose solving this problem through the introduction of a Mass-Based Local Ranking Measure. In general, relative mass of an instance is the ratio of data mass in two regions covering the instance, where one region is a subset of the other. The relative mass measures the degree of anomaly locally by considering the data distribution in the local regions (superset and subset) covering an instance [235]. In each isolation tree T_i the anomaly score of an instance $\vec{\mathbf{x}}$ w.r.t. its local neighbourhood $s_i(\vec{\mathbf{x}})$ is estimated as:

$$s_i(\vec{\mathbf{x}}) = \frac{m(\tilde{T}_i(\vec{\mathbf{x}}))}{m(T_i(\vec{\mathbf{x}})) \times n} \tag{6.25}$$

where $T_i(\vec{\mathbf{x}})$ is the leaf node in T_i in which $\vec{\mathbf{x}}$ falls, $\tilde{T}_i(\vec{\mathbf{x}})$ is the immediate parent of $T_i(\vec{\mathbf{x}})$, $m(\cdot)$ is the data mass of a tree node and n is the size of the training set used. The global anomaly score of an isolation forest composed of t isolation trees is then defined as:

$$s(\vec{\mathbf{x}}) = \frac{1}{t} \sum_{i=1}^{t} s_i(\vec{\mathbf{x}}) \tag{6.26}$$

Clusters of Local Anomalies In [236] it is shown that IF is not able to detect clusters of local anomalies and the SCiFOREST method is proposed as a more powerful alternative. An example of a cluster of local anomalies is shown in Figure 6.14. SCiForest is similar to IF but the trees are built in a slightly different manner. In each node of the tree, given an hyperplane f the data $\mathbf{X}$ is split into $\mathbf{X}^l$ and $\mathbf{X}^r$ based on which side of f a samples falls. Defining the standard deviation gain (Sd_{gain}) as:

$$Sd^f_{gain}(\mathbf{X}) = \frac{\sigma(\pi_f(\mathbf{X})) - \frac{\sigma(\pi_f(\mathbf{X}^l)) + \sigma(\pi_f(\mathbf{X}^r))}{2}}{\sigma(\pi_f(\mathbf{X}))} \tag{6.27}$$

where $\pi_f(\mathbf{X})$ is the projection of $\mathbf{X}$ over f and $\sigma(\pi_f(\mathbf{X}))$ is its standard deviation, f is chosen as the hyperplane with the best split point p that yields the highest Sd^f_{gain} among τ hyperplanes of q randomly selected attributes. The parameters q (number of attributes used in a hyperplane) and τ (number of hyperplanes considered in a node) are chosen by the user.

Locality of the Isolation Forest Method Consider an isolation forest trained on a dataset $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p)$. In [203] the authors states that efficient anomaly detection is possible without the need of retraining. It is

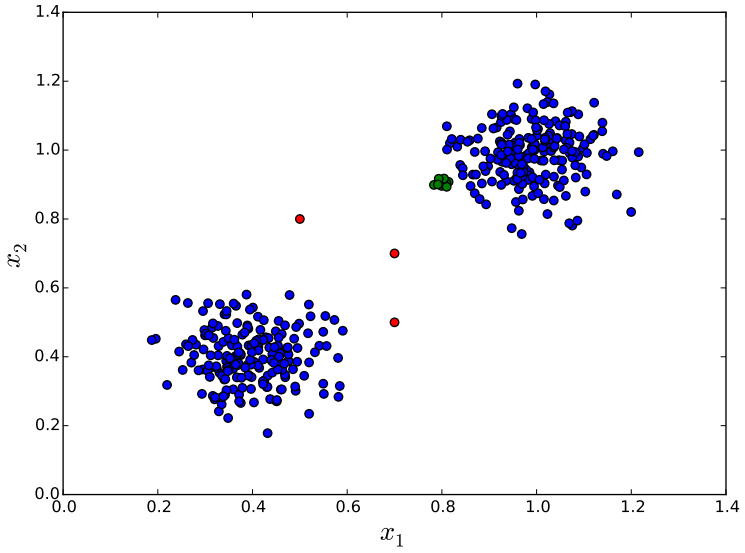


Figure 6.14: An two dimensional dataset defined by normal data (blue), global anomalies (red) and a cluster of local anomalies (green)

clear that the method is able to efficiently assign an anomaly score to a new sample that is inside the hypercube defined by:

$$D = [min(\mathbf{x}_1), max(\mathbf{x}_1)] \times \cdots \times [min(\mathbf{x}_p), max(\mathbf{x}_p)] \tag{6.28}$$

On the other hand the anomaly score assigned to new samples outside of the hypercube D is not reliable.

Example 6.4.3. A two-dimensional dataset is represented in Figure 6.15. In the two dimensional problem hypercube D is represented by a black square. The samples used to train the isolation forest are all contained inside D and they are represented by triangles. New samples represented by circles are then introduced. It can be observed that while the new points inside the black square have a reasonable anomaly score it is not the same for the points outside of the square. In particular the points on the higher side of the figure have a low anomaly score and the points in the lower part of the figure have an high one. According to the distribution of the data those points should all be classified with the same anomaly score.

Observation 6.4.2. *In the J2M case study (dataset 2.4.2) the OES data is characterized by a drift. This can be observed both in Figure 6.16 and Figure 6.17. The first figure shows the projection of the J2M data on the first two principal components. It can be observed that the training data*

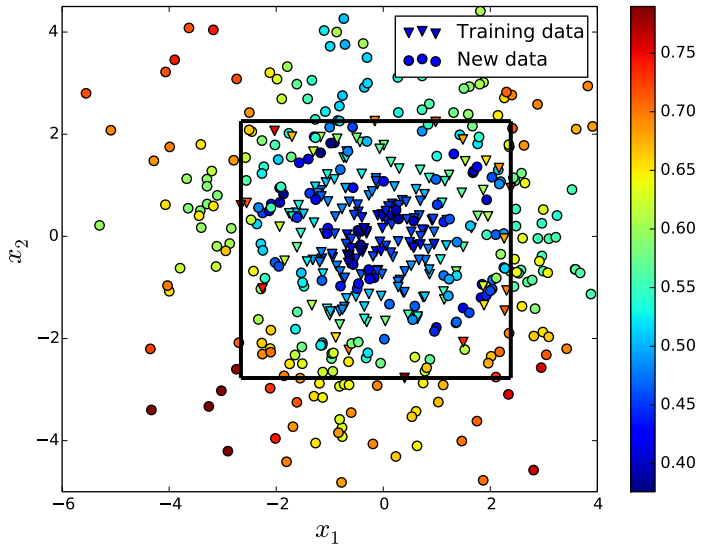


Figure 6.15: The data described in Example 6.4.3. The training samples are represented by triangles while the new samples are points. The black square is the D hypercube.

(represented by triangles) is all in the left part of the figure while most of the new data is on the right side (i.e. it is outside of the hypercube D). In the second Figure the first variable selected by FSCA (FSV_1) is plotted. It can be observed that this variable shows a decreasing pattern. In both cases most of the new samples are outside of the hypercube defined by the training data. As previously observed their anomaly score may be biased.

Isolation Forest can still be used with OES data. In order to avoid the locality problem the forest can be retrained each time that a new sample is included. This may negatively influence the computational performance of the method. In addition, this makes it difficult to use the method to measure the distance between two datasets. With OCSVM or MCC, for example, a dataset composed of normal behaving samples is used to train the model which is then used as reference to measure how distant the new samples are from the normal behaving ones. This would not be possible with IF if it is retrained every time a new sample is introduced.

Bias Toward Correlated Variables IF tends to assign an higher anomaly score to the samples which have the anomaly in a group of correlated variables. This is shown in the next example.

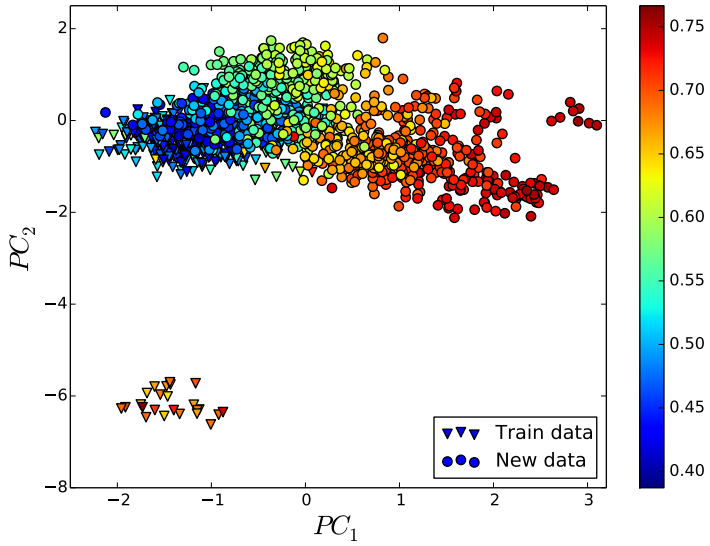


Figure 6.16: The projection of the J2M dataset on the first 2 principal components. The samples are coloured according to their anomaly score obtained with isolation forest.

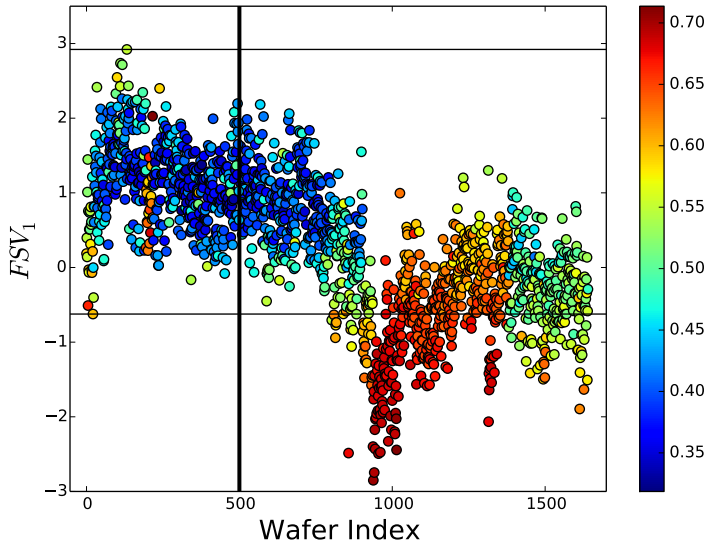


Figure 6.17: The first forward selection variable (FSV_1) of the J2M dataset. The vertical line represents the end of the training set while the horizontal ones indicate the maximum and the minimum of FSV_1 in the training set. The color represents the anomaly score obtained with the isolation forest method.

Example 6.4.4. Two base variables are generated as $\mathbf{x}_1, \mathbf{x}_2 \sim N(0,1)$ and two different samples in $\mathbf{x}_1$ and $\mathbf{x}_2$ are replaced with the value 6. These samples are anomalies and are denoted as n_1 and n_2, respectively. A set of variables correlated with $\mathbf{x}_2$ is generated as: $\mathbf{x}_i = \mathbf{x}_2 + \boldsymbol{\epsilon}_i$ for $i = 3, \ldots, 10$ where $\boldsymbol{\epsilon}_i \sim N(0, 0.01)$. It is clear that sample n_2 is also an anomaly with respect to the variables $\{\mathbf{x}_i\}_{i=3,\ldots,10}$. The full data $\mathbf{X}$ is then defined as $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{10})$. The anomaly score obtained for each sample with IF is represented in Figure 6.18. It can be observed that the anomaly in the vertical direction has an higher anomaly score than the one in the horizontal direction. The reason for such bias can easily be found by considering the operations of the isolation forest algorithm. Consider the anomaly samples n_1 and n_2. When the data is split in an isolation tree n_2 has 9 times the probability of being isolated from the rest of the samples as n_1. This is because n_1 can be isolated only by the variable $\mathbf{x}_1$ while n_2 can be isolated by all the others variables.

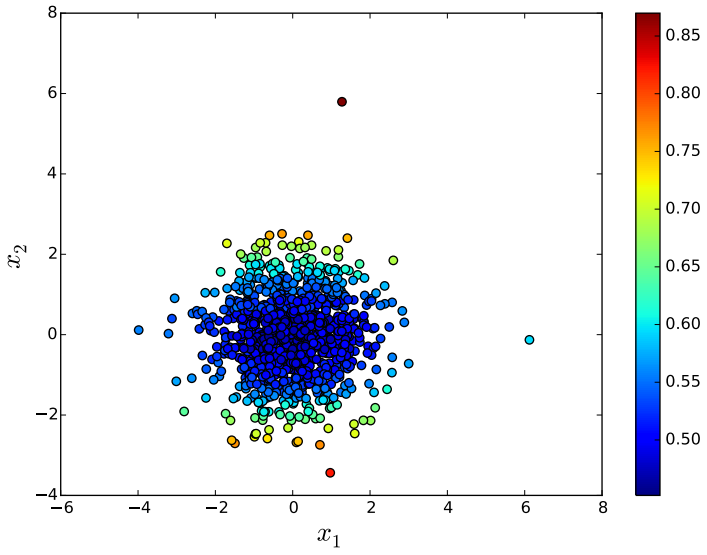


Figure 6.18: The data described in Example 6.4.4 projected on the variables $\mathbf{x}_1$ and $\mathbf{x}_2$. The samples are coloured according to their anomaly score assigned with IF.

The performance of IF can therefore be improved by preprocessing the data and removing redundant variables. Some variables selection algorithms explicitly designed for anomaly detection are introduced in Section 6.6.

6.4.2.2 Fault Diagnosis with Isolation Forest

Consider an anomaly $\vec{\mathbf{a}} \in \mathbb{R}^p$ and $h(\vec{\mathbf{a}})$ the average number of splits that are required to isolate $\vec{\mathbf{a}}$. Consider the two subsets of variables $\mathbf{X}_i = \{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}\}$ and $\mathbf{X}_j = \left\{\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_{(p-k)}}\right\}$ such that $\mathbf{X} = (\mathbf{X}_i, \mathbf{X}_j)$. $\mathbf{X}_i$ is the set of variables from which subsets of variables can be extracted which can isolate $\vec{\mathbf{a}}$ with a few splits. $\mathbf{X}_j$ instead contains the set of variables from which it is more difficult to extract a subset than can isolate $\vec{\mathbf{a}}$ in a few splits. Since

$$h(\vec{\mathbf{a}}) = \frac{1}{T} \sum_{t \in IF} h_t(\vec{\mathbf{a}}) \tag{6.29}$$

it is clear that trees which have as an initial split variables that are mainly from set $\mathbf{X}_j$ require more steps to isolate $\vec{\mathbf{a}}$ than ones that have most of their initial split variables from the set $\mathbf{X}_i$. Using this observation it is possible to detect which variables are causing the anomaly. Consider the following example:

Example 6.4.5. Let $\mathbf{X} \sim N(0, I) \in \mathbb{R}^{500 \times 100}$ and compute $h(\vec{\mathbf{x}})$ for each $\vec{\mathbf{x}}$ sample of $\mathbf{X}$. In this example the value $\vec{\mathbf{a}}$ defined as:

$$\vec{\mathbf{a}} = \underset{\vec{\mathbf{x}} \in \mathbf{X}}{\operatorname{argmax}}\, h(\vec{\mathbf{x}}) \tag{6.30}$$

is used as an anomaly. Figure 6.19 shows the distribution of the value $\{h_t(\vec{\mathbf{a}})\}_{t \in IF}$. It is possible to observe that there are trees with $h_t(\vec{\mathbf{a}}) < 5$ and trees with value $h_t(\vec{\mathbf{a}}) > 20$. The first set of trees are probably the ones where initial split variables are from $\mathbf{X}_i$ while the second group has probably most of the initial split variables from $\mathbf{X}_j$. Table 6.4.5 reports the split variables that are used to isolate $\vec{\mathbf{a}}$ in the trees with $h_t(\vec{\mathbf{a}}) < 3$. It is easy to observe that the variables $\mathbf{x}_{a_1} = 0$ and $\mathbf{x}_{a_2} = 46$ are the most frequently occurring. The projection of the data on these two variables is represented in Figure 6.19 and from the figure it is easy to understand that $\vec{\mathbf{a}}$ is an anomaly for $\mathbf{x}_{a_1}, \mathbf{x}_{a_2}$.

t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
0	40	0	49	0	46	17	43	92	55
	82	46	0	99	60	46	65	14	95
					32	12	98	50	97

Table 6.1: The split variables used to isolate $\vec{\mathbf{a}}$ for all the trees such that $h_t(\vec{\mathbf{a}}) \leq 3$

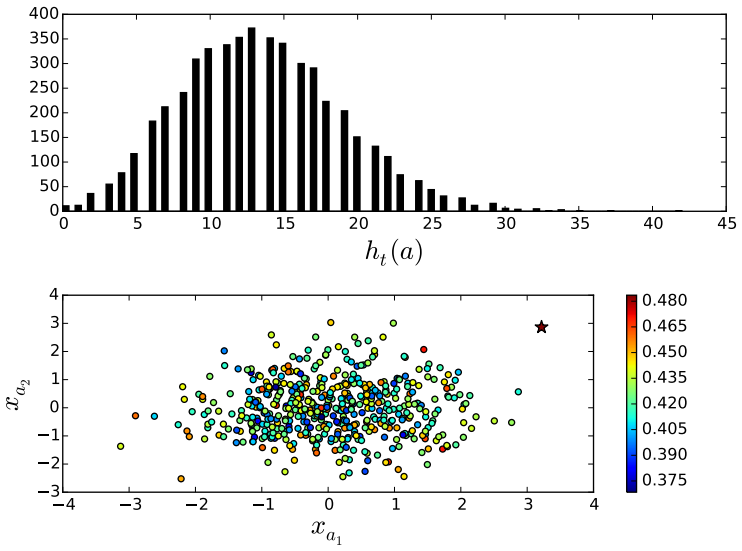


Figure 6.19: The distribution of the $h_t(\vec{\mathbf{a}})$ across the forest and the projection of the data on the variables $\mathbf{x}_{a_1}$ and $\mathbf{x}_{a_2}$ for the problem described in Example 6.4.5.

In the next example the IF diagnosis system is used to diagnose the anomaly in the J2M data.

Example 6.4.6. An IF with 200 trees is build on the J2M data (Dataset 2.4.2). Figure 6.20 shows the J2M ER coloured according to the anomaly score assigned to each sample by IF. It is easy to observe that the faulty wafers have a higher anomaly score than the rest of the samples. The shift is not recognized as when all the data is used to train the model the samples in the shift are as numerous as the normal samples. Therefore they are not anomalies. Some of the trees in the isolation forest have height smaller than 3. The values of some couples of variables for these small trees are represented in Figure 6.21. It is easy to see that these variables are sufficient to isolate the anomaly.

6.5 Unsupervised Training of a Supervised Algorithm

An unsupervised anomaly detection algorithm can be derived from a supervised classifier with a simple trick. In this section this methodology is explained using Random Forest (RF, [237]) as an example classifier yielding

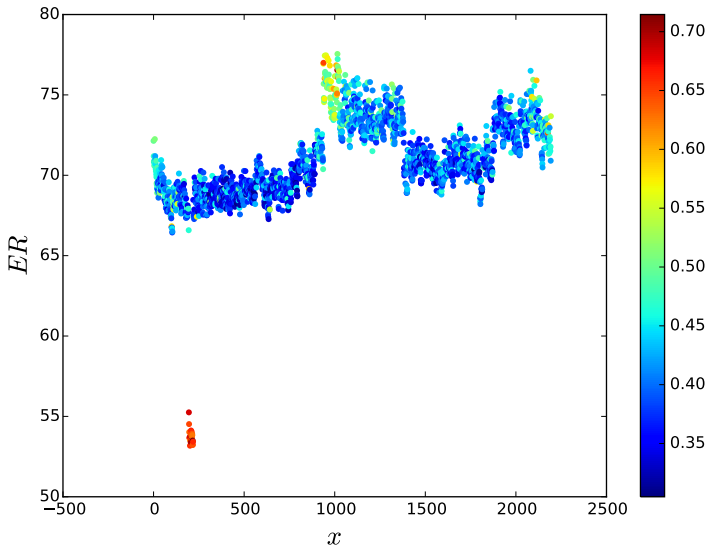


Figure 6.20: The ER of the J2M data coloured according to the anomaly score obtained with IF.

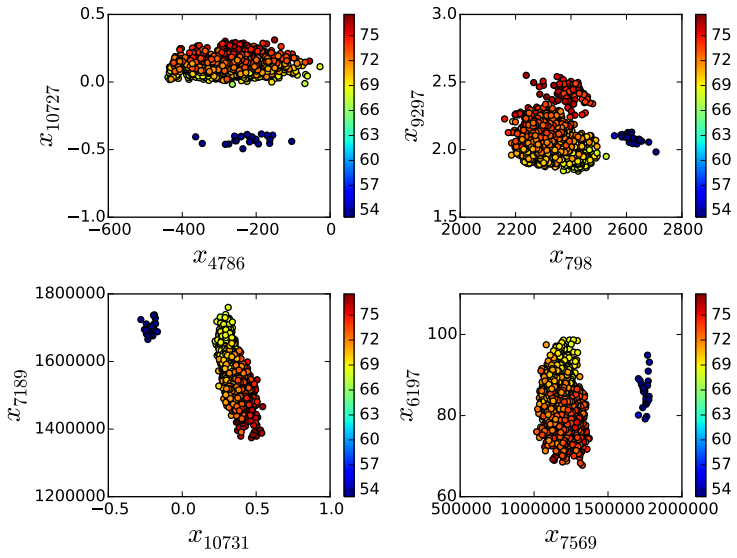


Figure 6.21: Projection of the J2M data on some of the variables detected with the diagnosis algorithm based on IF.

Unsupervised Random Forest (URF, [201]). It will be shown that URF is particularly efficient for highly correlated datasets such as those generated by OES. We begin with an introduction to decision trees and random forest.

6.5.1 Decision Trees and Random Forest

Decision trees and Random Forest are popular techniques used in regression and supervised classification [238], [239] and [237]. A decision tree is a tree-like graph or hierarchical decision structure in which leaves represent class labels and branches represent divisions of features that lead to those class labels. In literature numerous algorithms have been proposed to build decision trees such as the CART algorithm [240], extremely randomized trees [202] and the unbiased recursive partitions [241] where each split is based on a statistical test.

Algorithm 6.5.1 (CART)**.** The CART is probably the most popular algorithm for training decision trees. In CART the data is recursively split trying to minimize an impurity measure. Defining s as a set of samples and f_i the fraction of samples in the set s labelled with value i, the Gini impurity for a set is defined as:

$$I(s) = \sum_i f_i(1 - f_i) \tag{6.31}$$

This is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset. Total purity is obtained when all the samples in the set s have the same label, i.e. $I(s) = 0$.
Given a variable $\mathbf{x}_k$ the set s can be split by $\mathbf{x}_k$ into two subsets:

$$s_1^k = \{i : x_{i,k} < c\} \quad and \quad s_2^k = \{i : x_{i,k} \geq c\} \tag{6.32}$$

The value of c is chosen in order to maximise the decrease in impurity DI, where

$$DI(s,k) = I(s) - \frac{|s_1^k|}{|s|} I(s_1^k) - \frac{|s_2^k|}{|s|} I(s_2^k) \tag{6.33}$$

Given a set of variables $(\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_{p^*}})$ the data is split using the variable $\mathbf{x}_{i_j}$ where $i_j = \text{argmax}_k \ DI(s,k)$.

In the CART algorithm each split is done using the variable that maximizes the DI. In general we may assume that there are variables that are stronger i.e. they are more likely to be selected as they lead to larger decreases in the Gini Impurity. The resulting tree will be so characterized by a higher

presence of these strong variables. While this may be a desiderable property in many situations, in anomaly detection it is also important to keep track also of the weakest variables. For this reason extremely randomized trees are also considered in this thesis. Observe that a similar argument will be used to justify the introduction of alternative dimensionality reduction techniques to FSCA and PCA in Section 6.6.

Algorithm 6.5.2 (Randomized Trees)**.** Randomized Tree [202] is an alternative algorithm to train decision trees. The data is recursively split into subsets as described in equation 6.32 but, the variable $\mathbf{x}_k$ and the split value c are chosen almost randomly. In a tree constructed with Extremely Randomized Trees all the variables and all the split points appear roughly the same number of times. This ensure the presence in the tree of the weak variables i.e. the ones that will not be included by the CART algorithm.

6.5.1.1 Random Forest

Random Forest is an ensemble learning method, where the output from several trees are combined to produce a single decision. We refer to the original random forest, the ones where trees are trained with the CART algorithm as RF and to the forest composed of Randomized Trees as ERT. In RF each tree is trained using a subset of equally sized random samples (with replacement), from the original dataset. The ERT forest instead is composed of all the randomly generated trees trained on the original data. In RF and ERT each tree returns a classification result, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest). An estimation of the prediction error by RF can be extracted using the out of bag error (OOB, [242]). In the CART algorithm the number of variables used to find the optimal split point is p^*. In [237] it is shown that a large value of p^* increases the correlation of the trees in the forest and the strength of the single threes. These are two values that influence the forest error rate [237]. It is important to find the right balance between the correlation of the trees and their strength as individual predictors. It is indeed important to choose the right value p^* in order to optimize the performance of the algorithm. A random forest similarity distance [243] can be used in an unsupervised context to detect anomalies as described in the following algorithm.

Algorithm 6.5.3 (Unsupervised Random Forest)**.** Given a dataset for normal operation $\mathbf{X}^0 \in \mathbb{R}^{n\times p}$, we label all the samples in $\mathbf{X}^0$ as class 0. We create a synthetic second class $\mathbf{X}^1$ of the same size that will be labelled as class 1. The synthetic second class is created by sampling at random from

the univariate distributions of the original data. In order to create a single member of class two, the first coordinate is sampled from the n values $\left\{X_{i,1}^0 : \ i = 1, \ldots, n\right\}$. The second coordinate is sampled independently from the n values $\left\{X_{i,2}^0 : \ i = 1, \ldots, n\right\}$, and so forth. Thus, class two has the distribution of independent random variables, each one having the same univariate distribution as the corresponding variable in the original data. Class 2 therefore destroys the dependency structure in the original data. As a result, there are now two classes and this artificial two-class problem can be run through a random forest.

Once the random forest has been trained using the two classes, we can compute the proximity between samples. This is done as follows:

- After a tree is grown, put all data trough the tree.
- In a tree each sample ends up in a leaf. The proximity function $\mathbf{Pr}(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ computes the number of times that samples $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ end up in the same leaf of the tree.
- Normalise the estimated proximity by dividing by the total number of trees and then replacing each value with its square.

The average proximity of a sample $\vec{\mathbf{z}}$ to a class k is defined as

$$\overline{Pr}_k(\vec{\mathbf{z}}) = \frac{1}{|k|} \sum_{\vec{\mathbf{j}}: \ class(\vec{\mathbf{j}})=k} Pr(\vec{\mathbf{j}}, \vec{\mathbf{z}}) \ . \tag{6.34}$$

The distance of a new sample $\vec{\mathbf{s}}$ from the class k is then defined as

$$\tilde{D}_k(\vec{\mathbf{s}}) = \frac{|k|}{\overline{Pr}_k(\vec{\mathbf{s}})} \tag{6.35}$$

and reported in its scaled version:

$$D_k(\vec{\mathbf{s}}) = \frac{\tilde{D}_k(\vec{\mathbf{s}}) - \bar{D}_k}{std(D_k)} \ . \tag{6.36}$$

where

$$\bar{D}_k = \frac{1}{|k|} \sum_{\vec{\mathbf{z}}: class(\vec{\mathbf{z}})=k} \tilde{D}_k(\vec{\mathbf{z}}) \tag{6.37}$$

and

$$std(\tilde{D}_k) = \left(\frac{1}{|k| - 1} \sum_{\vec{\mathbf{z}}: class(\vec{\mathbf{z}})=k} \left(\tilde{D}_k(\vec{\mathbf{z}}) - \bar{D}_k \right)^2 \right)^{1/2} \ . \tag{6.38}$$

To perform anomaly detection, a single class k may be constructed from normal behaving samples. The observed distance a new sample is from this class can be a measure of how dissimilar the new sample is from previously observed normal behaving data. For this method to be effective, the training samples must reflect the structure of the normally observed samples. To quantify the D_k range that corresponds to normal samples, the mean (μ) and standard deviation (δ) from normal behaving samples is calculated. Using the limit $\mu + 3\delta$ as a bound on typically observed D_k, outliers are identified as samples where $D_k \geq \mu + 3\delta$. OOB error can be used to test if the training set size was sufficiently large in this regard. In a classification problem, the distance of a sample from the previous considered classes can be used to discover class membership or the identification of a new class.

An intuitive explanation of how the method works is provided in the next example.

Example 6.5.1. The dataset $\mathbf{X}^0$ is composed of 100 samples generated according to a two dimensional normal multivariate distribution $N(0, \mathbf{\Sigma}^2)$, where the covariance matrix is defined as:

$$\mathbf{\Sigma}^2 = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}. \tag{6.39}$$

In order to introduce an artificial anomaly, the value of the second variable $\mathbf{x}_2$ for sample $n = 100$ was set equal to 3. The dataset $\mathbf{X}^1$ is then computed from $\mathbf{X}^0$, as detailed above. The datasets $\mathbf{X}^0$ and $\mathbf{X}^1$ are depicted in Figure 6.22. Here, $\mathbf{X}^0$ and $\mathbf{X}^1$ samples are indicated by "$\bigcirc$" and "$\bigtriangledown$", respectively. The anomaly introduced at $n = 100$ is indicated by a '$*$'and is located close to coordinate $(-1, 3)$. This figure illustrates the lack of correlation between $\mathbf{x}_1$ and $\mathbf{x}_2$ within dataset $\mathbf{X}^1$, and the high correlation between $\mathbf{x}_1$ and $\mathbf{x}_2$ within dataset $\mathbf{X}^0$.

The similarity measurement, D_k, was applied to $\mathbf{X}^0$ and the results are depicted in Figure 6.23. From examining Figure 6.23, it is clear that the sample $n = 100$ has been identified as the outlier from samples within $\mathbf{X}^0$.

The next examples illustrate how Unsupervised RF works well when variables are very correlated and performs poorly if they are not. OES data are often characterized by highly correlated variables, hence unsupervised RF is expected to perform well on this data.

Example 6.5.2 (Unsupervised RF and uncorrelated variables)**.** The unsupervised RF algorithm is applied to the dataset described in Example 6.3.3

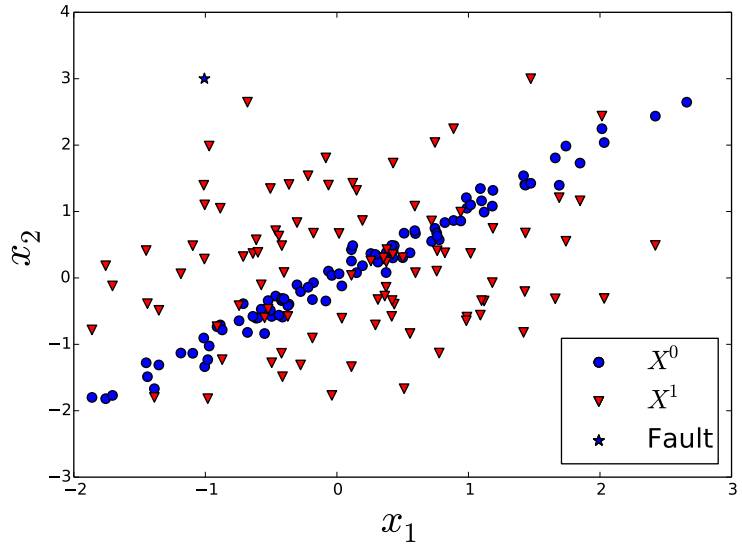


Figure 6.22: Graphical representation of the dataset described in Example 6.5.1. The original data $\mathbf{X}^0$ is represented with blue points; $\mathbf{X}^1$, the dataset obtained by randomly sampling from the columns of $\mathbf{X}^0$, is represented by red triangles; the anomaly is indicated by a "star".

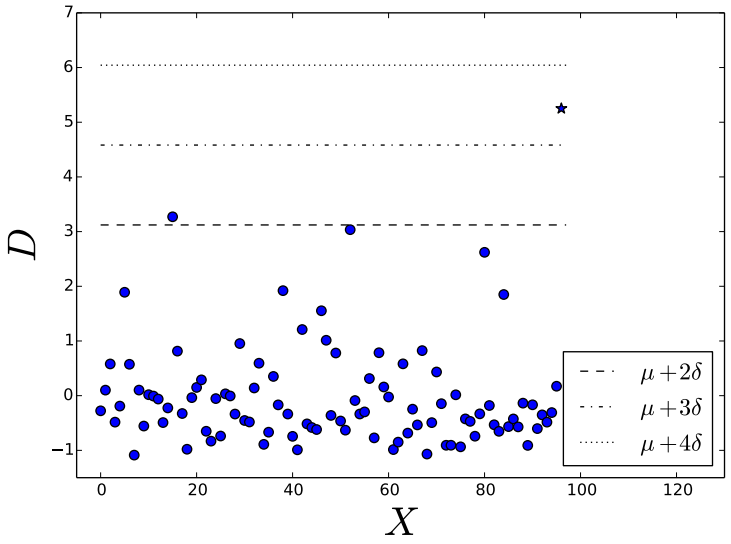


Figure 6.23: The distance measure obtained when URF is applied to the data in Example 6.5.1

and the results are represented in Figure 6.24. It is clear that the RF algorithm is not able to recognize the anomaly. This is because the method works only if the data shows a certain level of correlation. This easily follows from how the synthetic anomalies are created (i.e. class 1).

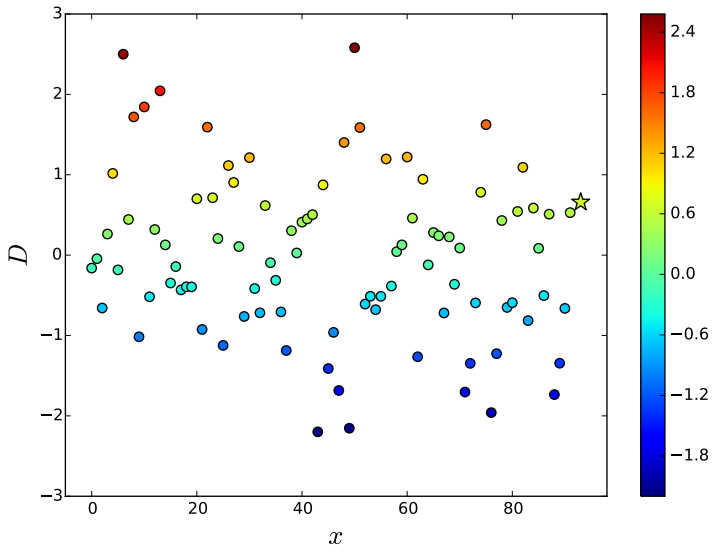


Figure 6.24: The distance obtained with random forest on the dataset describe in Example 6.3.3. The anomaly is denoted with a star.

Example 6.5.3 (Highly correlated data)**.** A dataset $\mathbf{X} \in \mathbb{R}^{100\times 30}$ is generated as in Example 4.7.2 i.e.:

- $\mathbf{X}^0 \in \mathbb{R}^{100\times 3}$: $\mathbf{X}^0_{i,j} \sim N(0,1)$
- $\phi \in \mathbb{R}^{3\times 27}$: $\phi_{i,j} \sim N(0,1)$
- $\epsilon \in \mathbb{R}^{100\times 27}$: $\epsilon_{i,j} \sim N(0,0.1)$
- $\mathbf{X}^1 = \mathbf{X}^0 \cdot \phi + \epsilon$
- $\mathbf{X} = (\mathbf{X}^0, \mathbf{X}^1)$

An anomaly is introduced by replacing the first 15 elements in the last row of the matrix with values sampled from an $N(0,1)$ distribution. The value of D obtained with RF and ERT for each sample is represented in Figure 6.25. Both RF and ERT are correctly able to recognize the anomaly but ERT distinguishes it more clearly.

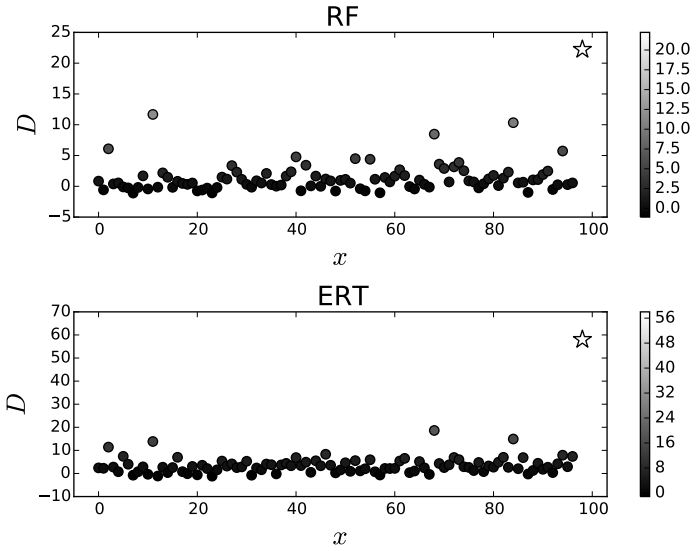


Figure 6.25: The D value obtained with unsupervised RF and ERT on the data described in Example 6.5.3

Unsupervised Random Forest is particularly effective on highly correlated datasets and scales well to high dimensional datasets. It is indeed a suitable algorithm for anomaly detection with OES data, which is commonly characterized by highly correlated variables as shown in the next example.

Example 6.5.4 (Unsupervised Random Forest on J2M)**.** The Unsupervised RF technique is applied to the J2M case study (Dataset 2.4.2). Taking 400 randomly sampled normal wafers from $\mathbf{Z}$ (i.e. wafers whose ER is in the normal operating range) as a training dataset, the random forest similarity distance for all other wafers was calculated using the methodology detailed in Algorithm 6.5.3. The computed random forest similarly distance is visualised in Figure 6.26. Here, the number of variables selected in each split to train the forest is $p^* = 50$ and the number of trees used within the forest was set to 100. From visual inspection, it is clear that both the process shift and faulty wafers can be identified using the applied random forest similarity distance.

Typically in the context of fault detection systems applied to high dimensional datasets, if the system is trained only on normal behaving data and a dimensionality reduction step is required, there is the potential to exclude a variable which captures the information that would describe a previously unseen fault or anomaly. However, due to the unsupervised random variable

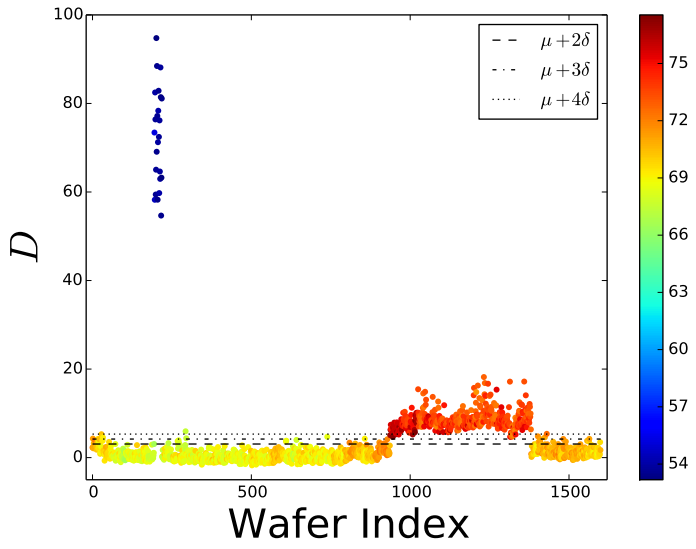


Figure 6.26: Random forest similarity distance with respect to normal behaving process wafers. Sample points are coloured according to their ER value. Normal operation ER is defined to be in the range $[66, 72]$.

selection process used within random forest, dimensionality reduction is not applied. Therefore, there exists, given a sufficient forest size and $p*$ value, the opportunity to include the variable which captures this information.

Forest size, N, and $p*$ value are important parameters in the calculation of the random forest similarly distance. To evaluate the impact of each parameter, a study was carried out in which, for a given training and validation dataset, one value of either N or $p*$ was fixed, while the other was varied. The results of a fixed $p*$ and varying N are presented in Figures 6.27, 6.28 and 6.29. Here $p* = 50$, while $N = \{50, 100, 200, 500, 1000, 10000\}$. The results of varying $p*$ are depicted in Figures 6.30 , 6.31 and 6.32. In this instance, $N = 100$, while $p* = \{20, 50, 100, 200, 300, 400, 500, 750\}$.

Due to the nature of OES recordings, there is typically a high correlation between observed variables. As a result, there is an increased probability that each tree within the forest would be a strong predictor and the majority of trees within the forest will be highly correlated. From inspecting Figures 6.27, 6.28, 6.29, 6.30, 6.31 and 6.32, it is clear that forest size, N, has a greater impact on calculated similarity distance compared to the value of $p*$. Indeed increasing N improved the distinguishing power in relation to

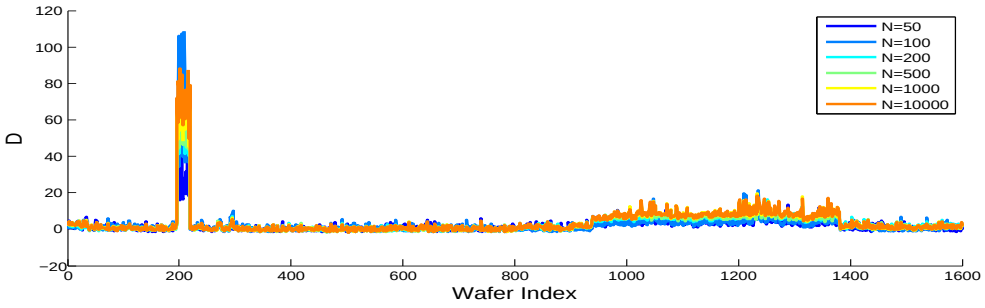


Figure 6.27: Random forest similarity distance for different values of N, where $p* = 50$: wafers 1 to 1600.

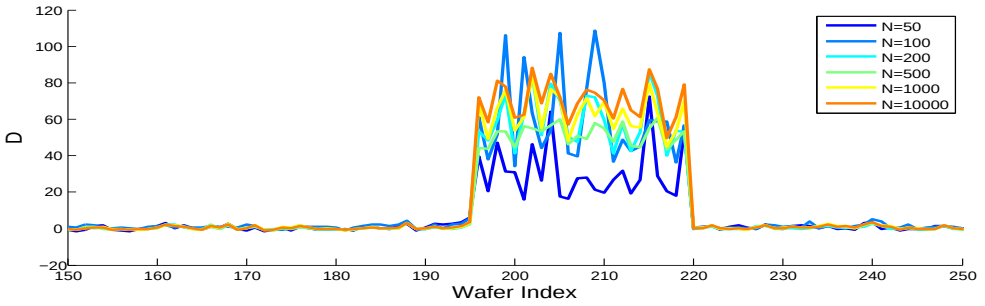


Figure 6.28: Random forest similarity distance for different values of N, where $p* = 50$: wafers 150 to 250.

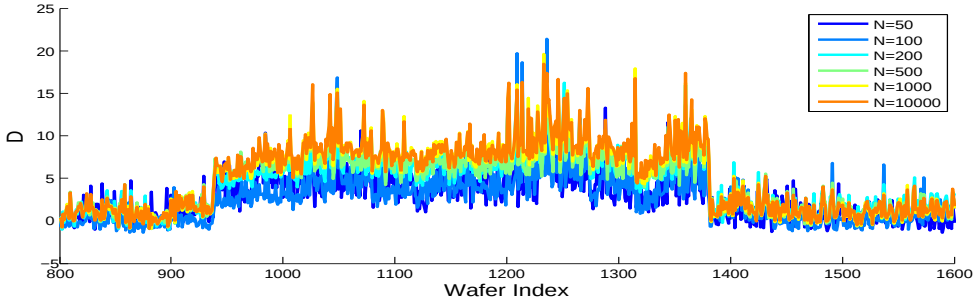


Figure 6.29: Random forest similarity distance for different values of N, where $p* = 50$: wafers 800 to 1600.

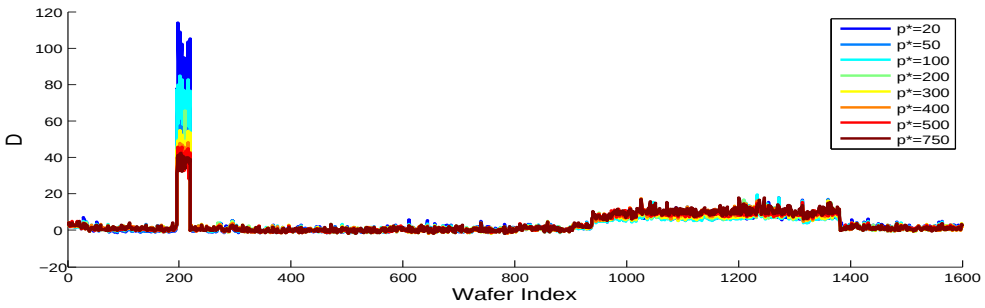


Figure 6.30: Random forest similarity distance for different values of $p*$, where $N = 100$: wafers 1 to 1600.

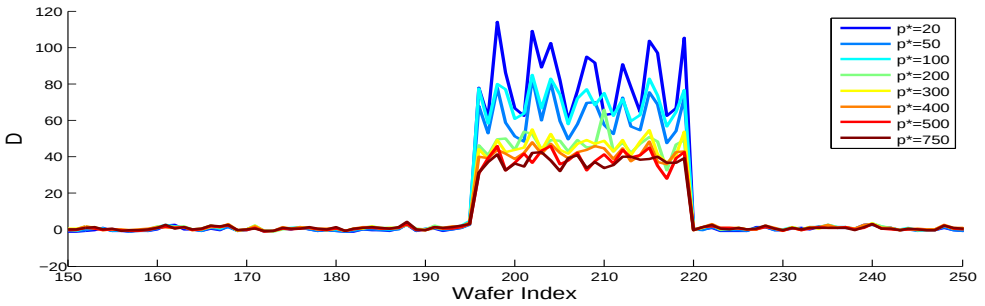


Figure 6.31: Random forest similarity distance for different values of $p*$, where $N = 100$: wafers 150 to 250.

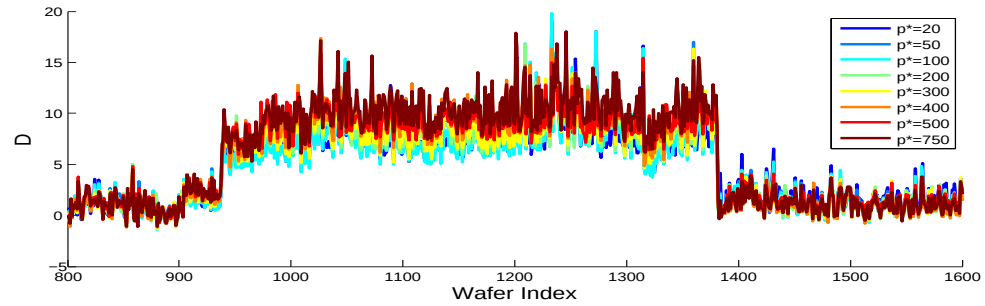


Figure 6.32: Random forest similarity distance for different values of $p*$, where $N = 100$: wafers 800 to 1600.

the process shift, as illustrated in Figure 6.32 while if N is large enough the fault and the drift are well distinguished from the normal samples for all the values of p^* as shown in Figure 6.30.

Observation 6.5.1 (RF Importance)**.** *RF is classically used as a supervised algorithm for classification and regression problems. In such situations it is common to extract from RF a measure defining how important each variable is [244], [237]. Practitioners may be tempted to use such a measure with unsupervised RF in order to detect the variables causing a fault. This will result in wrong conclusions.Taking into account how class 1 and class 2 are defined in Section 6.5.1 it is clear that the most important variables from a classification perspective are the variables that are more correlated with each other. The importance assigned to each variable will then not have any relation with the anomaly detection task.*

6.6 Dimensionality Reduction for Anomaly Detection

OES spectra are generally characterized by high dimension and anomaly detection can then be affected by the curse of dimensionality. In order to avoid the curse of dimensionality data is often preprocessed with dimensionality reduction techniques [245]. In supervised anomaly detection, where a classifier is used to separate normal samples from anomalies, dimensionality reduction is based on the same principles that are used to avoid overfitting in supervised classification and regression algorithms. Some examples in this sense are [246] where forward selection and backward elimination like methods are used for cloud classification, [247] where features are selected through random mutation hill climbing, [248] where PCA is used to pre-process the data and a single hidden layer neural network is then used to classify each sample as normal or anomaly and [249] where nonlinear PCA is used as a preprocessing step. In unsupervised anomaly detection the dimension of the data must be reduced with unsupervised dimensionality reduction algorithms like PCA or unsupervised features selection algorithms such as FSCA and similar approaches, as presented in chapters 4 and 5. In other circumstances techniques that take the structure of the data into account can be used. Some examples are [250] where a discrete wavelength transform is used to reduce the dimension of time dependent process data or as shown in Chapter 2 where the dimension of the data is reduced through wavelength selection or with summary statistics. The situation in an black box unsupervised framework is more complicated as most of the dimensionality reduction techniques are

not specifically designed for anomaly detection. This may lead to some side effects as described in the next section.

6.6.1 Side Effect of Dimensionality Reduction

Dimensionality reduction techniques such as PCA and FSCA seek to obtain lower dimensional approximations of datasets from which it is possible to reconstruct the majority of the information in the original high dimensional datasets, usually defined in terms the percentage of explained variance. While they are generally very useful for generating compact representations of highly correlated datasets, the reduced representations are not guaranteed to retain sufficient information to detect isolated anomalies. In particular, in datasets with several large clusters of correlated variables, the contributions of isolated uncorrelated variables to explained variance may be insignificant, with the result that such variables may not be included in the reduced data representation. It is then not possible to detect an anomaly if it is only reflected in such isolated variables. Some side effects of dimensionality reduction are shown in the following examples.

Example 6.6.1. Figure 6.33 shows a two dimensional dataset $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2)$. All the data lies on a diagonal line apart from the anomaly. It is very easy to spot the anomaly from the scatter plot of $\mathbf{X}$. A very good approximation of the data which reconstructs more than 99.99% of the variance can be obtained with a single PCA or FSCA component. Unfortunately in both cases it is impossible to distinguish the anomaly from the normal data when it is projected onto the resulting lower dimensional subspace. This can be observed in the second and third plots of the figure. It follows that the percentage of explained variance is not a reliable measure for the quality of dimensionality reduction in anomaly detection.

Example 6.6.2. Consider a two dimensional time series $\mathbf{X}(t) = (\mathbf{x}_1(t), \mathbf{x}_2(t))$ where the variables $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ are as represented in Figure 6.34. It is common in anomaly detection to build a model of the normal data and then to measure how different the new data is from the normal one. In this example the first 50 samples of the data are considered normal. A one component PCA is computed on the normal data and the projection of all the data on this component is given in Figure 6.34. It is clear from the first plot that there is a change in the signal after $t = 50$ as the variable $\mathbf{x}_2$ changes its behaviour. The same change of behaviour cannot be spotted in the second plot where the PCA approximation of the data is represented. This can be explained by the fact that the PCA trained on the normal data assigns a very small loading to $\mathbf{x}_2$. This is because $\mathbf{x}_2$ has a very small variation in the

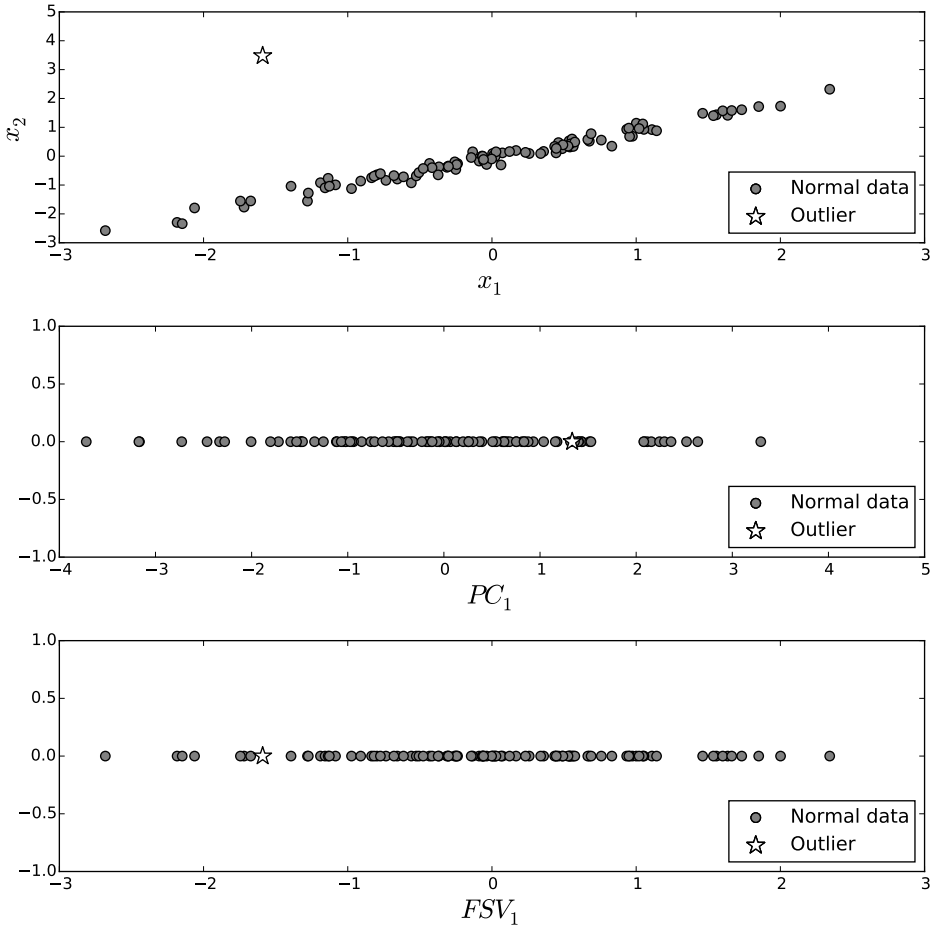


Figure 6.33: The original two dimensional data and the one dimensional approximation obtained with PCA and FSCA

normal data and it provides only a small contribution to the total variance. The influence of $\mathbf{x}_2$ on the data is lost after the dimensionality reduction while it was fundamental to spotting the anomaly.

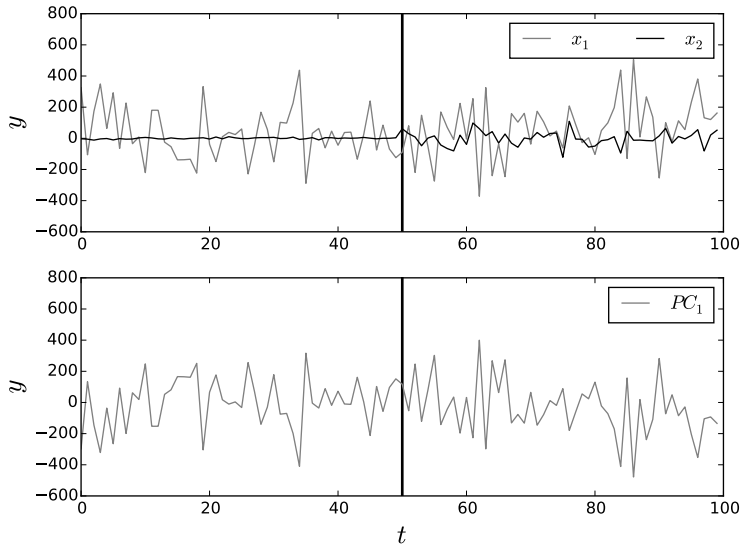


Figure 6.34: The variables $\mathbf{x}_1$ and $\mathbf{x}_2$ of the data described in Example 6.6.2 and the projection of the data onto first PC

6.6.2 Dimensionality Reduction that Keeps the Small Patterns

Most dimensionality reduction algorithms in the literature focus on the minimization of the reconstruction error. As previously observed those algorithms may not be the most suitable for anomaly detection. In the following sections some algorithms that perform dimensionality reduction while seeking to keep all the variability in the data are described.

6.6.2.1 Most Distinctive Variables

In [95] an algorithm to select the most dissimilar features is proposed which will be referred to as MDV in this thesis. The MDV algorithm is based on a similarity measure $s(\mathbf{x}, \mathbf{y})$ that measures the similarity between the variables $\mathbf{x}$ and $\mathbf{y}$. The authors propose three different similarity measures:

- Correlation Coefficient: $s(\mathbf{x}, \mathbf{y}) = 1 - |cor(\mathbf{x}, \mathbf{y})|$

- Linear Regression: $s(\mathbf{x}, \mathbf{y}) = min_{a,b} \parallel \mathbf{y} - a\mathbf{x} - b \parallel_2$
- Maximal information compression: $s(\mathbf{x}, \mathbf{y})$ is the smallest eigenvalue of the covariance between $\mathbf{x}$ and $\mathbf{y}$.

In all the cases $s(\mathbf{x}, \mathbf{y}) = 0$ for maximal similarity and the bigger $s(\mathbf{x}, \mathbf{y})$ is the less similar $\mathbf{x}$ and $\mathbf{y}$ are.
Having chosen $s(\mathbf{x}, \mathbf{y})$ variables are then recursively clustered and the ones that are more similar to their centroid are discarded. A detail description of the algorithm is presented in Pseudocode 6.6.1.

Input: Input matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, k the number of neighbours considered and $s(\mathbf{x}, \mathbf{y})$ is a function that measures the similarity between two variables
1: Start with F as the set of all the variables
2: $\forall F_i \in F$ compute the similarity with its k^{th} neighbour r_k^i
3: Choose F_j : $j = \text{argmin}_i\, r_k^i$ and set $\epsilon = -\infty$ (Most similar Neighbour)
4: In F remove the first k neighbours of F_j but keep F_j
5: **while** $r_k^j > \epsilon$ **do**
6: **if** $k = 1$ **then**
7: **return** F
8: **end if**
9: $\epsilon = r_k^j$
10: $k = k - 1$ (reduce k until the k^{th} nearest neighbour of at least one of the variables is less than ϵ-dissimilar with the variable)
11: $r_k^j = inf_{F_i \in F} \;\; r_i^k$
12: **end while**
13: **Go to step** 2
14: **return** F

Pseudocode 6.6.1: Unsupervised Feature Selection Using Feature Similarity (MDV)

6.6.2.2 Max separation clustering

The Max Separation Clustering (MSC) algorithm [90] is based on similar ideas of MDV. In this algorithm two centroids are initially selected as the most distant variable in the dataset and their cluster are defined by the variables whose correlation with the centroid is more than a threshold ξ. In subsequent iterations a new centroid is selected as the sample that is furthest from the existing clusters. The procedure is iteratively repeated until each variable is in a cluster. A computationally efficient variant of the algorithm is reported in Pseudocode 6.6.2. In the code the newly selected variable $\mathbf{m}$ is

the most distant from the centroids of existing clusters while in the original version of the algorithm $\mathbf{m}$ was chosen as the most distant sample from all smaples in the clusters.

Input: Input matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, $d(\mathbf{x}, \mathbf{y})$ a distance measure between the variables $\mathbf{x}$ and $\mathbf{y}$ and ξ a threshold value
1: Define the set of available variables $F = \{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$
2: Define the set of maxoids $M = \emptyset$
3: Select $\mathbf{x}_i, \mathbf{x}_j$ such that $(i, j) = \underset{(u,v):\mathbf{x}_u, \mathbf{x}_v \in F}{argmax}\ d(\mathbf{x}_u, \mathbf{x}_v)$
4: Choose one between $\mathbf{x}_i$ and $\mathbf{x}_j$. Usually the one with higher variance. Denote this variable $\mathbf{m}$
5: $M = M \cup \{\mathbf{m}\}$
6: $F = F - \{\mathbf{f} \in F : corr(\mathbf{m}, \mathbf{f}) > \xi\}$
7: **if** F is empty **then**
8: **return** M
9: **end if**
10: $\mathbf{m} = \underset{\mathbf{x}_u \in F}{argmax}\ d(\mathbf{x}_u, \mathbf{m})$ (The new $\mathbf{m}$ is the most distant variable from the previous $\mathbf{m}$.)
11: **go to** step 5

Pseudocode 6.6.2: Max Separation Clustering

6.7 Novel Dimensionality Reduction Algorithm for Anomaly Detection

In this section a novel dimensionality reduction algorithm specifically designed for anomaly detection is proposed. This is superior to the ones previously described as it select variables based on a multivariate procedure rather than through the use of a bivariate similarity measure.

6.7.1 Forward Selection Independent Variable Analysis

Both [95] and [90] select features based on a function $s(x, y)$ that measures the similarity between two variables. In general, instead of discarding variables that are similar to those already selected, it is more interesting to know which variables are not adequately represented by the selected ones. In addition, as shown in the next example using a univariate measure such as the correlation is a suboptimal choice. With this in mind Forward Selection

Independent Variable (FSIV) analysis is proposed in Algorithm 6.7.1 as a multivariate extension of MDV and MSC and as a tool for efficient unsupervised features selection for anomaly detection.

It is known that uncorrelated variables are linearly independent. On the other hand weakly correlated variables can be linearly dependent as shown in the following example.

Example 6.7.1 (Weakly correlated but linearly dependent variables)**.** Let $\{\mathbf{x}_i\}_{i=1,\ldots,p}$ be a set of uncorrelated variables with unit variance. Define

$$\mathbf{y}_i = \mathbf{x}_i - \frac{1}{p}\sum_{j=1}^{p}\mathbf{x}_j \tag{6.40}$$

The variables $\{\mathbf{y}_i\}_{i=1}^{p}$ are linearly dependent because

$$\sum_{i=1}^{p}\mathbf{y}_i = 0 \tag{6.41}$$

$\mathbf{y}_i$ can be rewritten as:

$$\mathbf{y}_i = \frac{p-1}{p}\mathbf{x}_i - \frac{1}{p}\sum_{j\neq i}\mathbf{x}_j \tag{6.42}$$

it follows:

$$Var(\mathbf{y}_i) = \frac{p-1}{p} \tag{6.43}$$

and

$$Cov(\mathbf{y}_i, \mathbf{y}_j) = -\frac{1}{p} \tag{6.44}$$

The correlation is then

$$corr(\mathbf{y}_i, \mathbf{y}_j) = -\frac{1}{p-1} \tag{6.45}$$

The previous example shows that by increasing p the correlation can be made arbitrary small while having a linearly dependant set of variables. It follows that feature selection based on pairwise similarity check is suboptimal and that better results may be obtained selecting the variables based on a multivariate model as described in the following algorithm.

Algorithm 6.7.1 (Forward Selection Independent Variables (FSIV))**.** The FSIV algorithm begins by selecting its first k variables $(\mathbf{z}_1, \ldots, \mathbf{z}_k)$ using the FSCA algorithm. This step is required to ensure the presence of the variables that represent the largest variation in the data. Then, additional variables are added in order to model significant isolated variations that are not captured by the first k variables. The process ends when K variables are selected or when the error $\epsilon_{\hat{j}}$ defined according to equations 6.48 and 6.49 is smaller than a given threshold. The FSIV algorithm is thus defined as follows:

1) Start with the full data $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p) \in \mathbb{R}^{n \times p}$ and set k and a stop criterion.

2) Scale the data such that each variable has zero mean.

3) Select k variables $\mathbf{z}_1, \ldots, \mathbf{z}_k$ using the FSCA algorithm.

4) Define the matrix $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_k)$.

5) Compute the linear approximation of $\mathbf{X}$

$$\hat{\mathbf{X}} = \mathbf{Z}(\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{X} \in \mathbb{R}^{n \times k} \tag{6.46}$$

where

$$\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_p) \tag{6.47}$$

6) For each variable $\mathbf{x}_i$ in $\mathbf{X}$ compute its approximation error

$$\epsilon_i = \| \mathbf{x}_i - \hat{\mathbf{x}}_i \|_2^2 \tag{6.48}$$

where $\hat{\mathbf{x}}_i$ is the ith column of $\hat{\mathbf{X}}$.

7) Select $\mathbf{x}_{\hat{j}}$ the variable with the highest approximation error where:

$$\hat{j} = \underset{i}{\operatorname{argmax}}\, \epsilon_i \tag{6.49}$$

8) Add $\mathbf{x}_{\hat{j}}$ to the $\mathbf{Z}$ matrix.

9) Stop if the termination criterion is reached, otherwise set $k = k + 1$ and repeat from step 5

The distinguishing feature of FSIV is that a variable is added to the model if it cannot be adequately reconstructed by a linear combination of those already selected. In FSIV variables are selected through the use of a multivariate model while in MDV and MSC variables are selected through the use of univariate operators such as the euclidean distance or the correlation. In the previous example it was shown that weakly correlated variables can be linearly dependent. In the next example the benefit of using FSIV in such circumstances is illustrated.

Example 6.7.2. Define a matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{20})$ where $\mathbf{x}_1, \ldots, \mathbf{x}_{19} \sim N(0,1)$ and $\mathbf{x}_{20} = \sum_{i=2}^{19} \mathbf{x}_i + \epsilon$ where $\epsilon \sim N(0, 0.001)$ and consider it in its scaled form so that each variable has mean 0 and unit variance. The problem is to detect the two variables that can best describe all the variation in the data. It is clear that the two variables are $\mathbf{x}_1$ and $\mathbf{x}_{20}$. $\mathbf{x}_1$ has to be considered as it is independent of all the rest while $\mathbf{x}_{20}$ is a linear combination of $\mathbf{x}_2, \ldots, \mathbf{x}_{19}$. FSCA, FSIV, MDV and MSC are applied to this dataset and the results are reported in Table 6.2. Only MDV and FSIV are able to correctly select the variables $\mathbf{x}_1$ and $\mathbf{x}_{20}$. There is not a value of the threshold ξ that results in only two selected variables with MSC. A minimum of 4 variables is selected by MSC, as reported. Significantly neither $\mathbf{x}_1$ and $\mathbf{x}_{20}$ are among them.

MDV	**FSIV**	MSC	FSCA
1	20	17	20
20	1	12	7
		7	
		8	

Table 6.2: Variables selected by MDV, FSIV, MSC and FSCA in the problem defined in Example 6.7.2

Observation 6.7.1. *Similarly to FSCA,* $\hat{\mathbf{X}}$ *is the linear reconstruction of* $\mathbf{X}$ *obtained from* $\mathbf{Z}$*. It follows that the algorithm can easily be extended to nonlinear cases as presented for FSCA in chapter 5.*

The following example illustrates the difference in performance between PCA, FSCA and FSIV and why it is important to keep track of all the variables in a fault detection context.

Example 6.7.3. Consider the simulated data $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_7) \in \mathbb{R}^{n \times 7}$ defined by three groups of variables $\mathbf{X}_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$, $\mathbf{X}_2 = \{\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6\}$ and

$\mathbf{X}_3 = \{\mathbf{x}_7\}$. Each variable has correlation 0.9 with the others in the same group and between the variables in $\mathbf{X}_1$ and $\mathbf{X}_2$ there is a correlation of 0.4. The variable in $\mathbf{X}_3$ is instead isolated and has only correlation 0.1 with all other variables. Specifically, $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_7) \sim N(0, \mathbf{\Sigma})$ where $\mathbf{\Sigma} = \{\Sigma_{i,j}\}$ is defined as:

$$\Sigma_{i,j} = \begin{cases} 1 & if\ i = j \\ 0.9 & if\ i,j \in \{1,2,3\}\ or\ i,j \in \{4,5,6\} \\ 0.4 & if\ i \in \{1,2,3\}\ and\ j \in \{4,5,6\} \\ 0.4 & if\ j \in \{1,2,3\}\ and\ i \in \{4,5,6\} \\ 0.1 & if\ i = 7\ or\ j = 7 \end{cases}$$

An anomaly is then introduced by replacing one of the samples in $\mathbf{x}_7$ with the value 10. Dimensionality reduction is performed with PCA, FSCA and FSIV. In each case only two variables are selected. In FSIV parameter k is chosen as $k = 1$. The two dimensional representations of the data obtained with the various methods is reported in Figure 6.35. From the figure it can be observed that only FSIV is able to isolate the anomaly. In particular, FSCA tends to select one variable from $\mathbf{X}_1$ and one from $\mathbf{X}_2$ while FSIV selects a variable from $\mathbf{X}_1$ and $\mathbf{x}_7$. The PCA components instead are obtained as a weighted linear combination of all the variables. However, the weighting associated with $\mathbf{x}_7$ is insufficient to materially affect the behaviour of the components, with the result that the anomaly is not distinguishable from the normal samples.

In general the challenge with FSIV is knowing what value to choose for k and K. It is reasonable to choose k in order to have the percentage of explained variance higher than a predefined threshold (as for example 99.99%) or to stop when the percentage of explained variance grows too slowly (similar criteria are used to chose the number of components in PCA and FSCA). The value of K may instead be chosen as the total number of desired variables or to be large enough in order to have the error $\epsilon_{\hat{j}}$ from equations 6.48 and 6.49 small enough.

6.8 Anomaly Detection with OES Time Series

In this last section the techniques developed are applied to the problem of anomaly detection with OES time series data. The PSI dataset (Dataset 2.4.1), described in Chapter 2, is used as a case study. In Algorithm 6.8.1 the Similarity Ratio (SR) methodology [204] is described. SR was originally

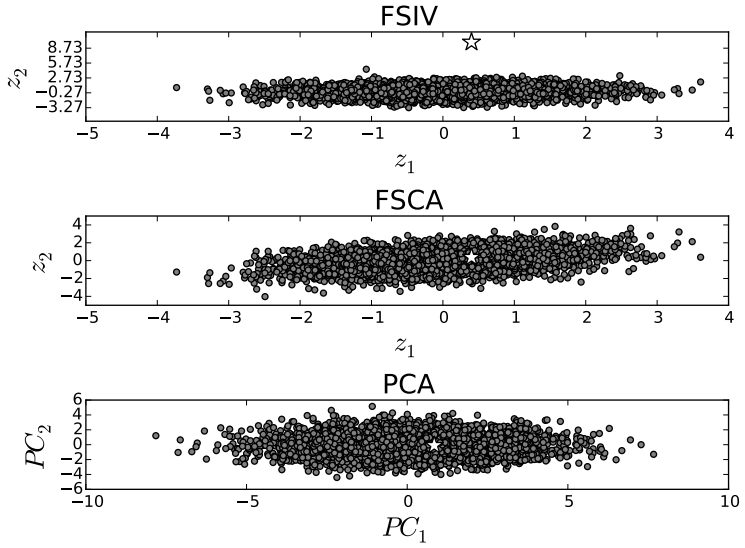


Figure 6.35: The two dimensional approximation of the data obtained with PCA, FSCA and FIV. The anomaly is represented by a white star

designed for anomaly detection with OES time series, it is now presented and extended taking into account the framework introduced in Chapter 2. Throughout the section the different ways in which the data can be aggregated plays an important role. The following matrices are particularly important: $\mathbf{\Lambda}$ (equation 2.11), $\mathbf{W}$ (equation 2.15) and $\mathbf{W}_{t_i}^{t_j}$ (equation 2.20).

Algorithm 6.8.1. Similarity Ratio (SR, [204]) is a recent method for early stage fault detection for plasma etching processes. For each wavelength a pair of fitting functions describe the wavelength intensity upper boundary ($UBF(t)$)and lower boundary ($LBF(t)$). Given a dataset $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p)$ composed of p wavelengths, for each wafer w each wavelength $\mathbf{x}_i$ can be considered as a function

$$x_i^w(t) \quad t \in [0, \tau] \tag{6.50}$$

where $t = 0$ when w starts to be processed and $t = \tau$ when the production of w is completed. Given a set of normal behaving wafers $\mathcal{W}$ for each value of t the mean and the variance can be computed for each wavelength as:

$$\mu_i(t) = \frac{1}{|\mathcal{W}|} \sum_{w \in \mathcal{W}} x_i^w(t) \tag{6.51}$$

$$\sigma_i(t)^2 = \frac{1}{|\mathcal{W}| - 1} \sum_{w \in \mathcal{W}} \| \, x_i^w(t) - \mu(t) \, \|^2 \tag{6.52}$$

a matrix, which the authors denote as, the SRA -Library is then obtained as:

$$SRA - Library(t) = \begin{pmatrix} UBF_1(t) & LBF_1(t) \\ UBF_2(t) & LBF_2(t) \\ \cdots & \cdots \\ UBF_p(t) & LBF_p(t) \end{pmatrix} \tag{6.53}$$

where $UBF_i(t) = \mu_i(t) + c\sigma_i(t)$, $LBF_i(t) = \mu_i(t) - c\sigma_i(t)$ and c is a constant defining how large the confidence interval is. The level of anomaly of a new wafer $\hat{w}$ is measured according to how many wavelengths are outside the control limits:

$$x_i^{\hat{w}}(t) > UBF_i(t) \quad or \quad x_i^{\hat{w}}(t) < LBF_i(t) \tag{6.54}$$

The similarity ratio SR_t is then defined as the percentage of wavelengths that are inside of the control limit at time t. A graphical representation of the $SRA - Library$ for a given wavelength is shown in Figure 6.36.

Taking into account the framework introduced in Chapter 2 the SR algorithm reduces to p univariate control charts as explained in the following observation.

Observation 6.8.1 (Similarity Ratio and Univariate Control Chart)**.** *Consider the representation of the data in the* $\mathbf{W}$ *matrix format defined in equation 2.16. In this case it is easy to observe that SR is equivalent to p univariate control charts. Indeed in a real world scenario the time measurements take discrete values* $t_1 < t_2 < \cdots < t_\tau$ *and the* $x_i^{w_j}(t)$ *function is a discrete vector*

$$\vec{\mathbf{x}}_i^{w_j}(t) = (x_i^{w_j}(t_1), x_i^{w_j}(t_2), \cdots, x_i^{w_j}(t_\tau)) \tag{6.55}$$

In this case the wavelengths measured for each wafer can be aggregated in a matrix. Suppose for example that K wafers and p wavelengths are available, then the data can then be rewritten as:

$$\mathbf{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_p) \tag{6.56}$$

where

$$\mathbf{W}_i = \begin{pmatrix} x_i^{w_1}(t_1) & x_i^{w_1}(t_2) & \cdots & x_i^{w_1}(t_\tau) \\ x_i^{w_2}(t_1) & x_i^{w_2}(t_2) & \cdots & x_i^{w_2}(t_\tau) \\ & & \cdots & \\ x_i^{w_K}(t_1) & x_i^{w_K}(t_2) & \cdots & x_i^{w_K}(t_\tau) \end{pmatrix} \quad i = 1, \ldots, p \tag{6.57}$$

Observe that the obtained $\mathbf{W}$ *matrix is equivalent to the matrix defined in equation 2.16. It follows that the SRA-Library (equation 6.53) is equivalent to applying a univariate control chart to each column of* $\mathbf{W}$*. Indeed* $\mu_i(t_j)$ *and* $\sigma_i(t_j)$ *are the mean and the standard deviation of the* j^{th} *column of* $\mathbf{W}_i$*.*

As explained in Section 6.3.1.1 the main weakness of Univariate Control Charts (UCC) is their inability to detect interactions between variables. In this case a UCC is not able to model the interaction between wavelengths and the evolution of a wavelength during different time points.

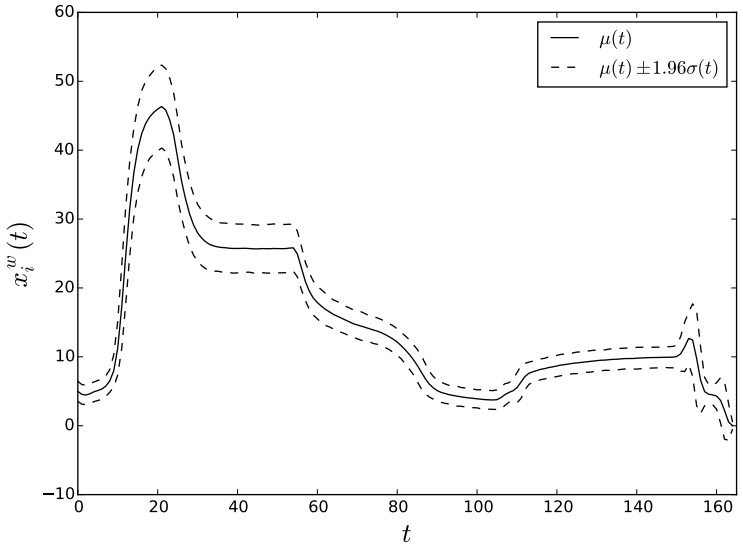


Figure 6.36: The functions $\mu(t)$, $UBF(t)$ and $LBF(t)$ when $c = 1.96$ for a given wavelength.

6.8.1 OCSVM as a Multivariate Extension of SR

Once it is clear that SR is nothing more than an application of UCC to the $\mathbf{W}$ matrix it is easy to extend and generalize the method in order to take advantage of the multivariate structure of the data. In order to obtain a multivariate alternative to SR it is enough to apply a multivariate fault detection algorithm to the matrix $\mathbf{W}$. Unfortunately the $\mathbf{W}$ matrix has a huge number of columns ($\tau \times p = 288255$) that makes the application of an algorithm on the full matrix impractical from a computational point of view. It is then necessary to reduce the dimension of $\mathbf{W}$. This can be done in several ways. It is for the moment assumed that it is possible to reduce the dimension of $\mathbf{W}$. This can be done using some of the techniques described in chapters 5

and 4 or the ones particularly designed for anomaly detection introduced in Section 6.6 of this chapter.

Once a lower dimensional representation of the data is obtained the anomaly detection performance of the method is determined as reported in Pseudocode 6.8.1. As shown in Pseudocode 6.8.1 in semi-supervised anomaly detection it is necessary to know which samples are normal. In addition, we need to have labelled normal and abnormal samples in the test set in order to evaluate the performance of the methodology. Domain knowledge is used to label the normal and abnormal samples in the OES spectrum. Note that the algorithm only needs to be trained on normal behaving wafers; the samples labelled as anomalies are only used for evaluation purposes. In plasma etching wafers are normally processed is batches of 25, called lots, with the chamber undergoing a cleaning cycle between each lot. As a consequence of the cleaning step there is a seasoning effect during the processing of the first few wafers in each lot as chemicals absorb into the chamber walls, with the result that the processing of wafers $1, 2, 3$ and 4 differ slightly from the remaining wafers 5 to 25. For the purposes of evaluating the performance of the anomaly detection methodology, we will consider wafers $1, 2, 3$ and 4 in each lot as abnormal wafers.

The next examples show how the use of feature selection algorithms and the two formulations of the PSI data can be used for anomaly detection. The first example is specifically designed to show the differences between FSCA and FSIV in an anomaly detection context.

Input: The data $\mathbf{X}$, an anomaly detection algorithm Ψ and a dimensionality reduction algorithm $\mathcal{D}$

1: Define $\mathbf{X}_{train}$ a subset of $\mathbf{X}$ containing only normal samples.
2: Define $\mathbf{X}_{test}$ as the samples that are in $\mathbf{X}$ but not in $\mathbf{X}_{train}$.
3: Compute $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ the mean of each variable in $\mathbf{X}_{train}$.
4: $\mathbf{X}_{train} = (\mathbf{X}_{train} - \boldsymbol{\mu})/\boldsymbol{\sigma}$.
5: $\mathbf{X}_{test} = (\mathbf{X}_{test} - \boldsymbol{\mu})/\boldsymbol{\sigma}$.
6: Train the dimensionality reduction algorithm on $\mathbf{X}_{train}$
7: $\mathbf{X}_{train}^{reduced} = \mathcal{D}(\mathbf{X}_{train})$
8: $\mathbf{X}_{test}^{reduced} = \mathcal{D}(\mathbf{X}_{test})$
9: Train Ψ on $\mathbf{X}_{train}^{reduced}$.
10: Obtain an anomaly score for $\mathbf{X}_{test}^{reduced}$ from Ψ.
11: Use the AUC error to evaluate the performance of Ψ.

Pseudocode 6.8.1: Anomaly detection evaluation procedure algorithm.

Example 6.8.1. In this example a smaller version of the PSI dataset (Dataset 2.4.1) is considered. The data is composed of $K = 500$ wafers each one containing the light intensity measurements of $p = 1747$ wavelengths at $\tau = 165$ time points. Initially the data is stored in the $\mathbf{\Lambda} \in \mathbf{R}^{K\tau \times p}$ format (Equation 2.11). The number of wavelengths is reduced to 200 by selecting the ones with the highest standard deviation in recorded light intensity. In order to better show the difference between FSCA and FSIV an artificial wavelength $x_l^{w_k}(t)$ is added to the OES spectrum. $x_l^{w_k}(t)$ is defined for each wafer as

$$x_l^{w_k}(t) = 285(sin(t) + \epsilon) \quad t \in [-\pi, \pi] \tag{6.58}$$

where $\epsilon \sim N(0, 0.05)$ and the amplitude 285 is selected to give a signal power that is similar to the other wavelengths. A fault is then introduced in the final wafer in the dataset by clamping the l^{th} wavelength to lie between -100 and 100, that is:

$$x_l^{w_K}(t) = \begin{cases} x_l^{w_K}(t) & if \ \ |x_l^{w_K}(t)| < 100 \\ 100 + \epsilon & if \ x_l^{w_K}(t) > 100 \\ -100 + \epsilon & if \ x_l^{w_K}(t) < -100 \end{cases} \tag{6.59}$$

where $\epsilon \sim N(0, 10)$. Figure 6.37 shows the time evolution of the light intensity for the artificial wavelength $x_l^{w_k}(t)$ and a normal wavelength for a group of five wafers. It follows that the faulty wafer can be classified as anomaly only if the wavelength l is among the selected ones. In order to perform anomaly detection the data must be aggregated in the $\mathbf{W}$ format. This data format is particularly useful for comparing wafers and performing anomaly detection as each row corresponds to all the data observed for a given wafer. Even in this reduced version of the data, the dimension of the matrix is very large as it has $p\tau = 33000$ columns. The dimension of the data can be drastically reduced by selecting only a subset of the wavelengths from the $\mathbf{\Lambda}$ matrix. If 14 wavelengths are selected from $\mathbf{\Lambda}$ the number of columns in the $\mathbf{W}$ matrix is reduced to $14\tau = 2310$ columns. The W matrix is still high dimensional but now it is small enough for high dimensional anomaly detection algorithms such as Unsupervised Random Forest and OCSVM to be efficiently applied.

Wavelengths are selected from the $\mathbf{\Lambda}$ format of the data using the FSCA and FSIV algorithms. FSCA and FSIV select variables using different criteria. Figure 6.38 shows the maximal error $\epsilon_{\hat{j}}$ defined according to equations 6.48 and 6.49, as a function of the number components selected by FSCA and FSIV, while Figure 6.39 shows the corresponding explained variance (EV,

Equation 4.3). In total 14 components are selected by both the FSCA and FSIV algorithms. The first 7 components of the FSIV algorithm are selected with FSCA, hence it follows that the performances of the both methods in terms of both EV and $\epsilon_{\hat{j}}$ are identical for these components. In contrast for the remaining 7 components we can observe that as expected the variables selected with FSIV lead to a lower error $\epsilon_{\hat{j}}$, while those selected by FSCA yield a larger percentage of EV. Notably, the l^{th} wavelength is not selected by FSCA but is selected by FSIV as the 8^{th} component. It can be observed in Figure 6.38 that the 8^{th} component is where the performance of FSCA and FSIV begin to deviate in terms of the error $\epsilon_{\hat{j}}$.

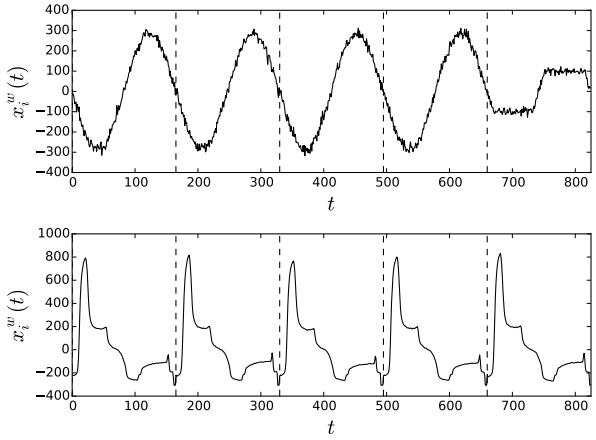


Figure 6.37: Light Intensity variation of the artificial wavelength and a normal wavelength over 800 time points and 5 wafers.

In order to train and test the anomaly detector the wafers in W are split into a training set of 300 wafers containing measurements of only normal behaving wafers and a test set of 200 wafers containing normal and abnormally behaving wafers.

Observation 6.8.2. *Observe that the defined scenario is a semi-supervised anomaly detection problem where the training data is known to be normal behaving. The presence of normal and abnormal behaving wafers in the test set is required only to asses the performance of the model.*

The OCSVM algorithm is used to assign an anomaly score to each wafer. The anomaly score assigned by OCSVM to each wafer in the test dataset when using each of the dimensionality reduction techniques is given in Figure 6.40. For completeness, the results obtained without dimensionality reduction are also reported. The results show that in general a larger anomaly score is

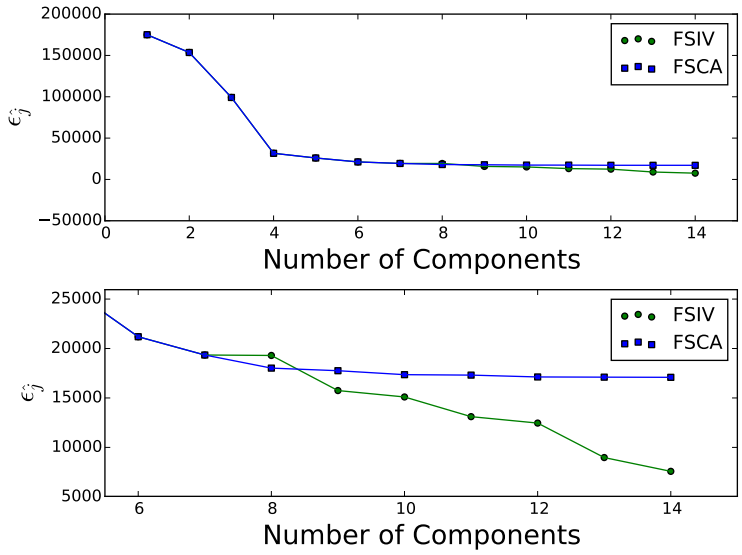


Figure 6.38: The error $\epsilon_{\hat{j}}$ as a function of the number of components selected with FSCA or FSIV.

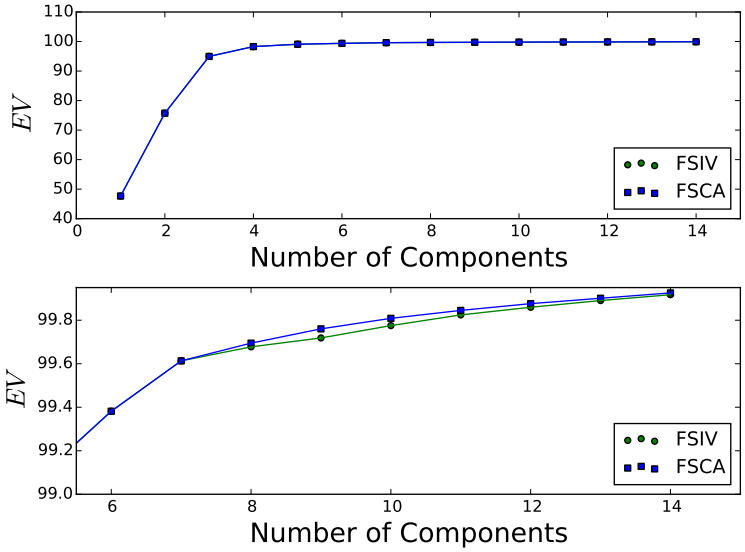


Figure 6.39: The error percentage of explained variance as a function of the number of components selected with FSCA or FSIV.

	A.W.	FSCA	FSIV	PCA
AUC	0.9564	0.9507	**0.9650**	0.9527

Table 6.3: The AUC score obtained using OCSVM when all the wavelengths are used (A.W.), when a subset of 14 wavelengths is selected with FSCA and FSIV, and when 14 PCA components are employed as inputs.

assigned to the abnormal wafers allowing them to be identified. The one exception is the artificially created abnormal wafer, denoted by the star, which is only correctly identified as an anomaly when FSIV is used. Even when all the wavelengths are used the artificial anomaly has a low anomaly score. This may be due to over fitting caused by the excessive number of variables. The performance of each method is also summarized in terms of the Area Under the Curve (AUC) classifier performance metric in Table 6.3 and again this underscores the superiority of FSIV.

The previous example was mainly used to show the differences between FSCA and FSIV in an industrial context. The same analysis is now shown for the full original dataset without the introduction of the artificial wavelength and considering all 1747 wavelengths.

Example 6.8.2. From the PSI data in the $\mathbf{\Lambda}$ format (Equation 2.11) two sets of wavelengths are selected. In the first case all the wavelength are selected with FSCA while in the second case the first 20 wavelengths are selected with FSCA and 20 additional wavelengths are selected with FSIV. In both the cases a total of 40 wavelengths are selected. The remaining data is then stored in the $\mathbf{W}$ matrix which now has dimensions $K \times 40\tau$, where the number of columns, 40τ is only 6600. The data is divided into a training and test datasets. The OCSVM algorithm is then used to assign an anomaly score to the samples in the test dataset. The process is repeated several times using a bootstrap algorithm. The mean and the standard deviation of the obtained AUC scores are reported in Table 6.4. In the table the results obtained using the first 40 principal components computed from the original $\mathbf{W}$ are also reported. All the methods lead to very good AUC scores. PCA is the method with the best performance followed by FSIV and FSCA. This confirms FSIV to be better than FSCA for variable selection in an anomaly detection context. Figures 6.41 and 6.42 show the anomaly score assigned to wafers in the test dataset. It can be observed that in general normal and anomaly wafers are well separated but a certain number of false positives is present. It is interesting to observe in Figure 6.42 that the distribution of anomaly scores as a function of slot number is slightly U shaped with

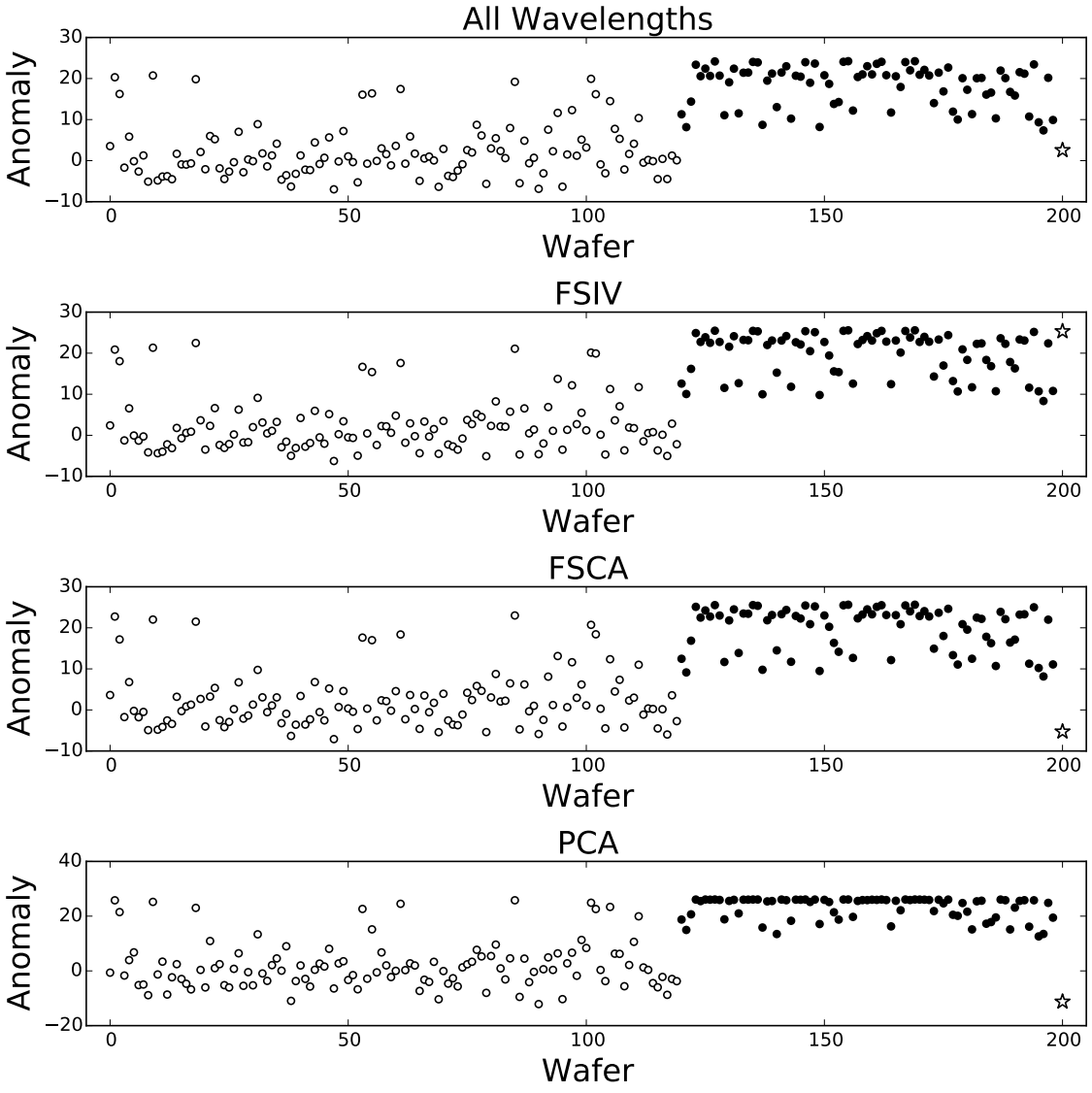


Figure 6.40: Anomaly score assigned to each wafer in the test set: Black circles denote the anomaly wafers ($1-4$ in each lot), white circles denote the normal wafers, and the artificial anomaly wafer is represented by a star.

lower anomaly scores for wafers in slots 10 to 20. This seems to reflect the seasoning effect of the process.

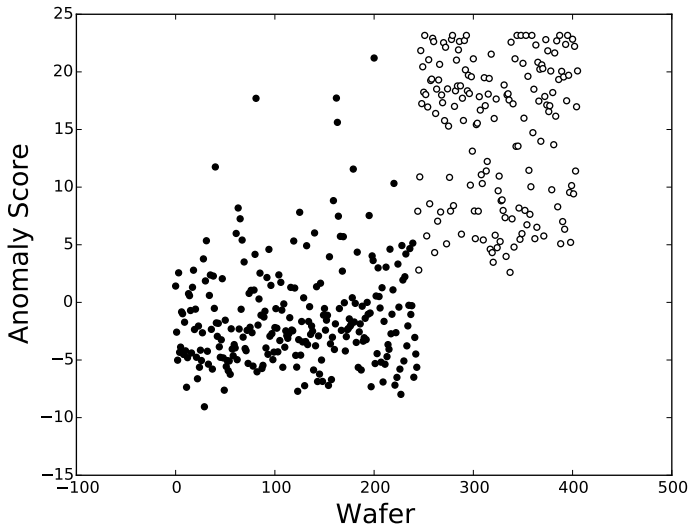


Figure 6.41: Anomaly score assigned to the wafers in the test set when the dimension of **W** is reduced with PCA. The white circles are the anomalies while the black ones are the normal wafers.

	AUC	
	Mean	Std
FSIV	0.96713	0.00749
FSCA	0.96657	0.00843
PCA	0.97998	0.00820

Table 6.4: AUC score obtained with OC-SVM when a subset of wavelengths is selected with FSCA or FSIV or when the dimension of **W** is reduced with PCA.

In conclusion FSIV provides an effective way to reduce the dimensionality of **W** by performing wavelengths selection on the $\mathbf{\Lambda}$ matrix. In both the examples considered better performance is achieved if wavelengths are chosen with FSIV rather than with FSCA. As illustrated in the first example, if the anomaly appears in a wavelength that is weakly correlated with the rest of the data FSIV is even preferable to PCA. In conclusion, FSIV assists with

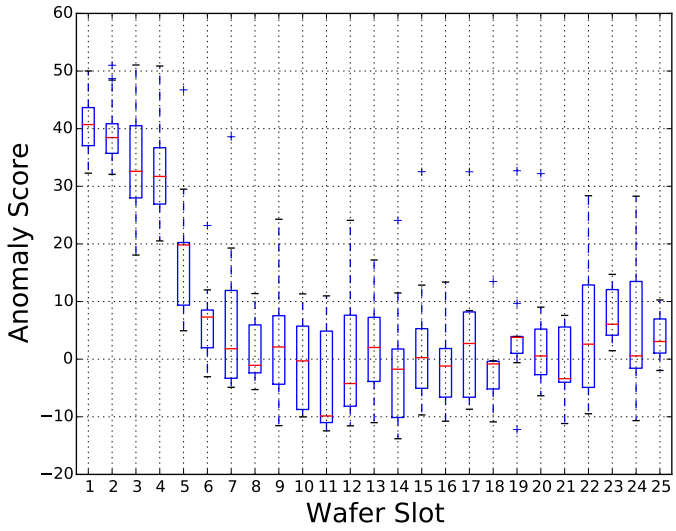


Figure 6.42: Anomaly score obtained with OCSVM + FSIV as a function of wafer slot number.

building an effective and easily interpretable anomaly detection system with the OES data. Its performance is similar to PCA with the added advantage that the model keeps a physical meaning. This is an important property as it is often not enough to detect a fault; it is also necessary to understand the physical reason behind the fault.

6.8.1.1 False Positive Analysis

In the two previous examples some false positives were found. These where wafers with slot number larger than 4 to which a large anomaly score was assigned. This can, for example, be observed in Figures 6.41 and 6.42. A more detailed analysis shows that the reason behind the large number of false positives is an imperfect alignment of the wafers. Consider for example the False Positive with the largest anomaly score. This was in slot 20. The values of a single wavelength over all the production time for this and all the wafers in the same slot are reported in Figure 6.43. From the figure it can be observed that the wafer is aligned with all the others if $t < 45$ while it is not aligned anymore if $t > 85$. A similar problem is shown in Figure 6.44 for a false positive in slot 15.

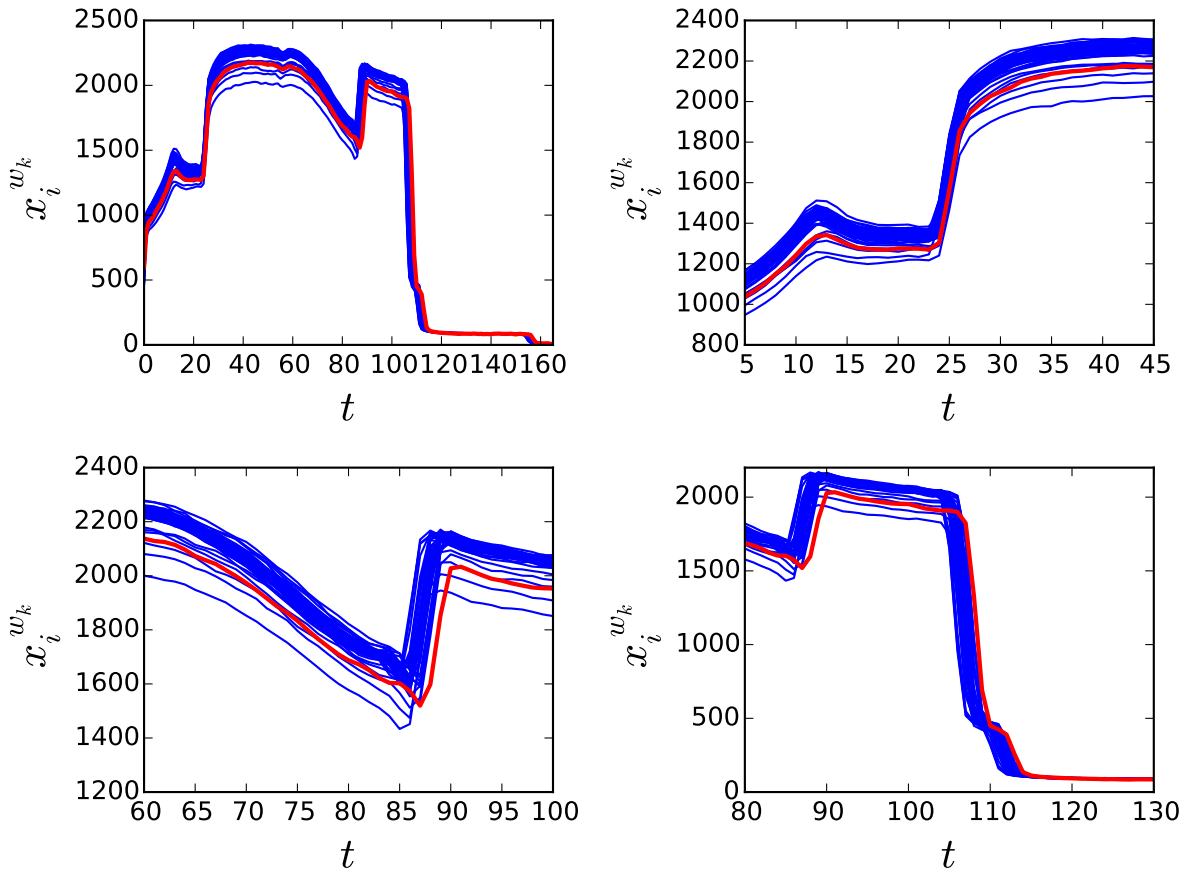


Figure 6.43: The profile of a wavelength for all wafers in slot 20. The red line is the wafer that has a large anomaly score.

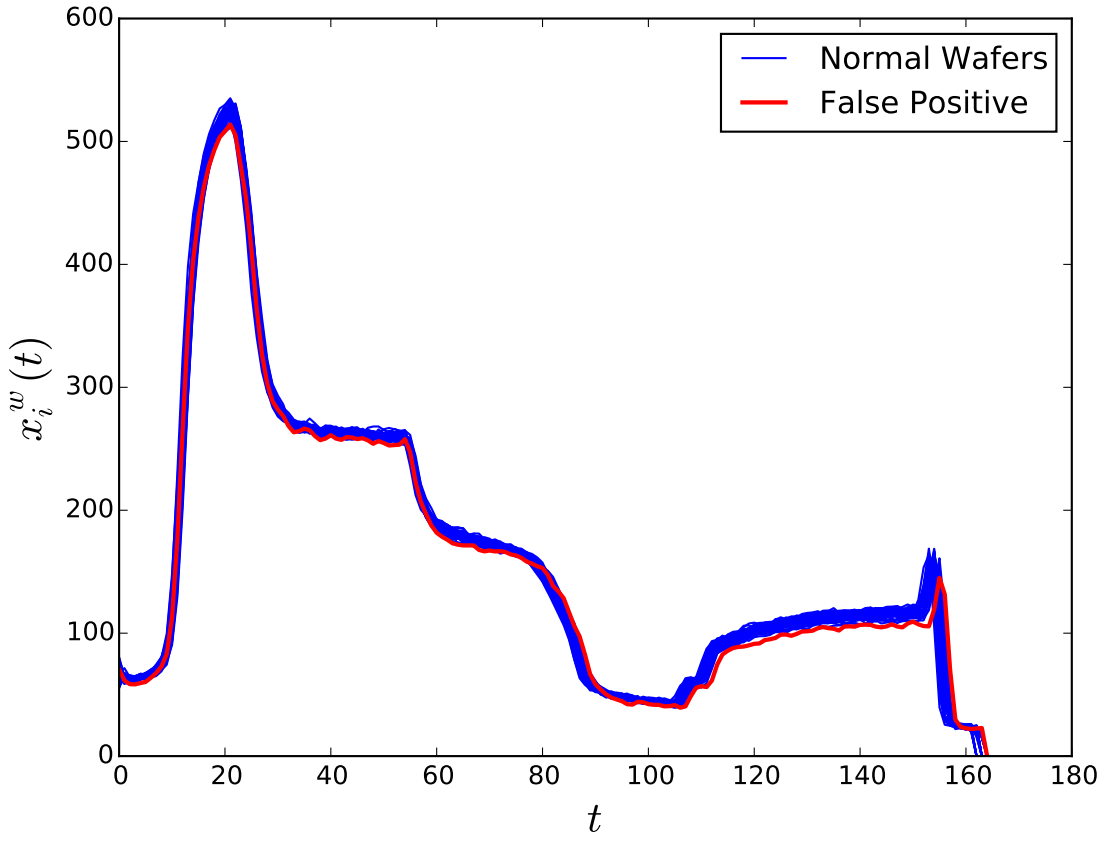


Figure 6.44: The value of a wavelength for all the wafers in slot 15. One of them is a false positive and it is represented by a red line.

6.8.2 Time Window

As stated in Section 6.6, dimensionality reduction in an anomaly detection context comes with several side effects. In this section a way to perform anomaly detection avoiding the dimensionality reduction of the $\mathbf{W}$ matrix is proposed. Taking into account that to each column of $\mathbf{W}$ is associated a time point t, it is possible to study the behaviour of the process in a time interval. Given a starting time point t_i and a final one t_j only the matrix $\mathbf{W}_{t_i}^{t_j}$ defined as in equation 2.20 is considered. The following example shows the kind of analysis that can be performed using the matrix $\mathbf{W}_{t_i}^{t_j}$.

Example 6.8.3. In this example anomaly detection on the PSI data (dataset 2.4.1 is performed using the matrix $\mathbf{W}_{t_i}^{t_j}$. Different values of t_i and t_j are tried with the restriction of not considering more than 14 time points i.e. $j = i, i+1, \ldots, i+13$. This is necessary for computational reasons as the number of variables used each time is

$$(j - i + 1) * 1747 \tag{6.60}$$

The minimum number of considered variables is then 1747 obtained when $j = i$ and the maximum number is 24458 obtained with $j = i+13$. The AUC score obtained for different values of the starting time t_i and number of time points considered $(j-i)$ are represented in Figure 6.45. The figure shows that the AUC score drops if $t_i \geq 110$. This may suggest that anomaly wafers are not very different from normal wafers during the final part of their production process. This impression seems to be confirmed in Figure 6.46 where some sample wavelengths for normal and anomaly wafers are represented. Indeed it can be observed that the anomaly wafers show a different behaviour with respect to the normal ones only in the first half of the process. This is also reflected in Figure 6.47 where the boxplot of the anomaly score obtained with OCSVM when $t_i > 110$ for normal and anomaly wafers is reported. In the figure normal an anomaly wafers have roughly the same anomaly score.

From the previous example it follows that the time window study, based on the matrix $\mathbf{W}_{t_i}^{t_j}$ can be used to understand when during the production process the anomaly occurred. In addition it suggests that an anomaly can be detected in real time during wafer production rather than having to wait the full wafer to be processed. Instead of a single anomaly score for each wafer the approach provides a set of values indicating how anomalous a wafer is at a given point in time during production.

Observation 6.8.3. *Since anomalous wafers are different from normal wafers only for certain values of t it follows that the $\mathbf{\Lambda}$ data format is not suitable*

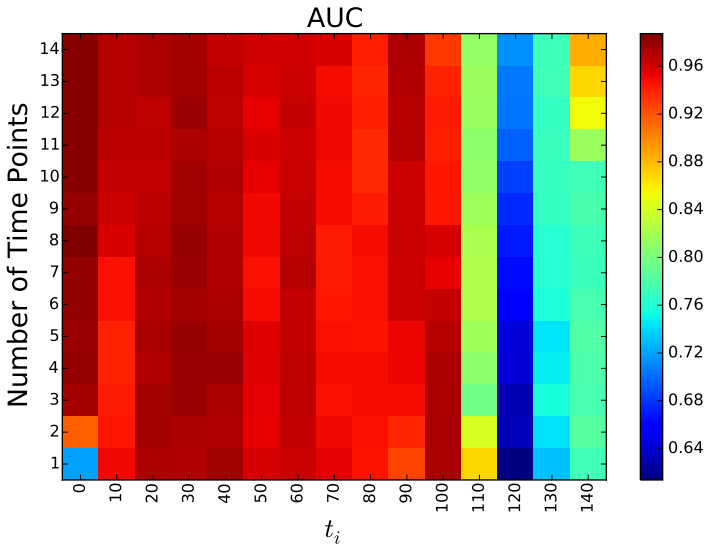


Figure 6.45: The AUC error as a function of the number of time points and of the starting point t_i as described in Example 6.8.3.

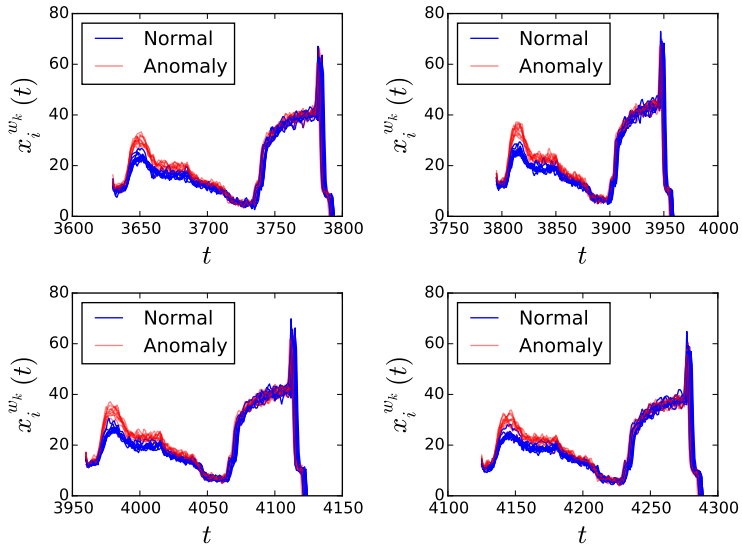


Figure 6.46: A random set of normally and anomalously behaving wafers for 4 wavelengths as described in Example 6.8.3.

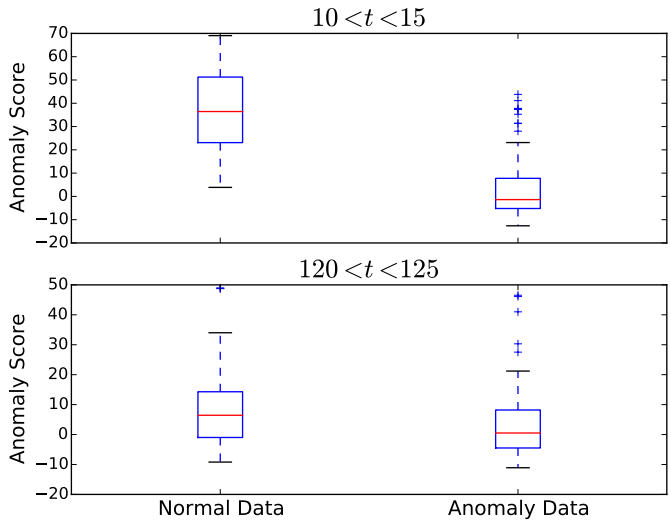


Figure 6.47: A box plot showing the score assigned to the normal and anomalous data using the OC-SVM algorithm for starting point $t = 10$ and $t = 120$, as described in Example 6.8.3.

for anomaly detection. Consider for example the row of $\mathbf{\Lambda}$ *containing the measurement of an anomaly wafer at a time* t_a *where it is not distinguishable from a normal wafer. That sample will always be considered normal even if the wafer is faulty. The* $\mathbf{W}$ *matrix format is therefore more appropriate for anomaly detection as each sample contains all the history of a wafer production.*

6.9 Conclusion

This chapter has provided a comprehensive presentation of unsupervised anomaly detection in semiconductor manufacturing. In the first part of the chapter an introduction to anomaly detection and a review of the most popular algorithms is reported. It is shown that the Unsupervised Random Forest algorithm can successfully be applied to the OES data and better results are obtained if trees are constructed with the Extremely Randomized Trees algorithm rather than the more popular CART algorithm. In addition a fault diagnosis algorithm based on Isolation forest is proposed. In the second part of the chapter the effect of dimensionality reduction on anomaly detection is investigated. Forward Selection of Independent Variables (FSIV) is proposed as a new unsupervised variables selection algorithm. This leads to

better anomaly detection performance than PCA and FSCA. The last part of the chapter describes how the OES time series can be used for anomaly detection in addition using FSIV to obtain interpretable and effective models. In addition the Similarity Ratio algorithm [251], that was originally designed for anomaly detection with OES time series, is generalized and improved.

Chapter 7

Concluding Summary and Future Work

7.1 Concluding Summary

The work presented in this thesis was dedicated to the development of advanced analytical tools for data analysis with a particular focus on Optical Emission Spectroscopy (OES) data collected during plasma etching in semiconductor manufacturing processes.

The thesis starts with a description of the process behind the generation of OES data and how it can be represented in matrix formats convenient for analysis. Temporal and wavelength alignment issues are highlighted and addressed. In particular, the problem of multi sensor spectral alignment is described and a novel retrospective alignment methodology developed. In the reminder of the thesis OES measurements are assumed to be aligned.

OES data is very high dimensional, posing major challenges to its use in practical applications such as process monitoring and virtual metrology. Consequently the rest of the thesis focuses on the variable selection problem as a means of reducing the dimensionality. Several variable selection techniques are developed to address different scenarios. These can be classified as either supervised or unsupervised algorithms.

Among the supervised variable selection algorithms, the lasso estimator is one of the most popular and has previously been employed in semiconductor manufacturing problems. Lasso has good prediction performance and is computationally efficient. As the OES data is characterized by highly corre-

lated variables, the set of variables selected by lasso may change significantly with small variations in the training set. In order to improve the robustness of lasso, several lasso based model selection procedures are reviewed and a number of new algorithms are proposed. This results in sparser or more stable models with equivalent or better performance than lasso.

In the thesis supervised variable selection is considered in the context of Virtual Metrology (VM), the specific instance being the estimation of etch rate using OES signals as model inputs. VM is a particularly attractive proposition as ER values are often only available several hours after production is completed and then only for a few selected wafers.

Unsupervised dimensionality reduction is an important area or research with application in many domains. Principal Component Analysis (PCA) is one of the most popular dimensionality reduction techniques in semiconductor manufacturing. The main drawback of PCA is that since the PCs are linear combination of all underlying variables, it is difficult to identify the key variables. In the case of OES this means that PCA cannot be used to determine the most important wavelengths, which is an important requirement for plasma etch process monitoring, as the variation in wavelength intensities define the underlying variations in plasma chemistry, and hence the drivers of process changes. For this reason the thesis focuses on unsupervised variables selection instead of unsupervised dimensionality reduction. As an alternative to PCA the recently proposed Forward Selection Component Analysis algorithm is investigated in detail and several enhancement proposed. Its computational efficiency is improved with the introduction of optimized algorithms and its variables selection performance is increased with the introduction of refinement steps. The new FSCA implementation highlights the similarity between FSCA and multi-output regression models. This led to a natural extension of FSCA and its alternative implementations to the nonlinear case. The use of nonlinear multi-output models substantially improves the FSCA performance in terms of explained variance. Particularly good performance is obtained when Extreme Learning Machine with Direct Linear Feed-Through are used. The associated algorithm ELMFSV-DLF beats linear FSCA in all datasets investigated and PCA in some of the considered cases. Good performance is also obtained with polynomial models. (PFSV). PFSV has in general worse performance than ELMFSV-DLF but it performs generally better than FSCA and keeps the model easy to interpret.

Anomaly detection is one of the most important applications of OES data. As a consequence, the last chapter of the thesis is dedicated to anomaly

detection. The high dimension of the data poses several challenges in this context. This is, in particular, expressed by the, Curse of Dimensionality, (COD), according to which relative distances between samples are not reliable in high dimensions. COD implies that anomalies, that are by definition samples that are distant from the normal behaving ones, are difficult to detect when high dimensional datasets are considered. After a review of the most popular anomaly detection algorithms and the discussion of their efficiency with high dimensional datasets, the effect of dimensionality reduction on anomaly detection is investigated. It is clear that classical dimensionality reduction algorithms that try to maximize the percentage of explained variance, like PCA and FSCA, are not always suitable for anomaly detection. This is because the information contained in variables that do not contribute much in terms of explained variance may be absent or weakly represented in the obtained lower dimensional representation of the data. This led to the development of a new unsupervised variables selection technique called Forward Selection Independent Variables (FSIV) with the aim of providing a lower dimensional representation of the data that keeps track of all the isolated variations. The proposed algorithm is then used to perform wavelength selection in an anomaly detection case study based on OES data and it performs better than PCA and FSCA when the dataset contains uncorrelated wavelengths.

In conclusion it is shown that OES data can be successfully used for anomaly detection given that effective dimensionality reduction is performed. It is in addition shown that dimensionality reduction can be avoided if the wafers are analysed in time intervals resulting in a measure of anomaly at each time instant during production. This may be useful when seeking to identify when during production the anomaly occurred.

As a summary the proposed variables selection algorithms are here reported.

- Supervised Variables Selection (Lasso based methods)
  - HM, HF, MCHM, MCHF are used to identify a stable set of variables.
  - HF(max) allows the number of selected variables with lasso to be reduced as hyperparameters chosen with cross-validation may be too generous in terms of number of selected variables.
- Unsupervised Variables Selection

- FSCA and FSV perform linear unsupervised variables selection and their performance is improved with the introduction of refinement steps denoted as: SPBR, MPBR, R-SPBR and R-MPBR.
- Several linear in the parameter algorithms are developed as nonlinear extension of FSCA and FSV. These were obtained using nonlinear multi-output models. The considered models and the associated nonlinear version of FSCA and FSV were: Polynomial (PFSCA and PFSV), Extreme Learning Machine Neural Networks (ELMFSCA and ELMFSV), and Extreme Learning Machine Direct with Linear Feed-Through Neural Networks (ELMFSCA-DLF and ELMFSV-DLF).
- Variables selection based on multilayer neural networks.
- FSIV for unsupervised variables selection for anomaly detection.

7.1.1 Guidelines for Practical Applications.

All the work in this thesis is oriented toward the development of new algorithms that are of practical use in semiconductor manufacturing. It is then important to guide practitioners in the use of the algorithms. While there isn't an algorithm that stands out from the rest, as the method of choice under all conditions, as each method has its advantages and disadvantages, some of them work well in general and, as a consequence, are more valuable for practitioners that do not want to go into the details of each method.

- For supervised problems HF(max) is the most effective method as it is easy to use and, at the same time, produces robust and accurate predictive models.
- For generic unsupervised problems FSCA provides an interpretable alternative to PCA. If the percentage of explained variance obtained with FSCA is not sufficient its performance can be improved through the introduction of refinement steps or nonlinearities. Among these ELMFSV-DLF is the method that is most likely to achieve the best performance and therefore the suggested one.
- For anomaly detection, and in particular for semi-supervised anomaly detection, the most common scenario in semiconductor manufacturing, variables that behave differently from the majority of the data need to be included in the final model, even if they do not contribute much in terms of explained variance. For this reason, the FSIV algorithm is recommended for anomaly detection.

7.2 Future Work

The algorithms proposed in this thesis open up new possibilities for future work. Some suggestions are discussed below, which are certainly not an exhaustive list.

Time context

Most of the developed algorithms do not take into account normal variation in the process over time. The currently developed algorithms need to be periodically retrained in order to keep track of these variations. Wavelengths that are important at the beginning of the wafer etching may not be relevant any more later in the cycle. An automatic update of the model can be obtained through the introduction of time series analysis techniques or the development of automatic retraining procedures. This may reduce the need for human intervention during production leading to a more automated and effective production control.

PCA for anomaly detection

FSIV shows the limit of classical dimensionality reduction techniques like PCA in an anomaly detection context. FSIV is defined as a constrained minimization procedure where the cost function is defined in terms of the L^∞ norm rather the L^2 norm as used in PCA and FSCA. It is then reasonable to wonder if a modified version of PCA defined in order to minimize a convex combination of the L^2 and L^∞ norms may lead to better performance than FSIV due to its lack of constraints.

Microarray Data Analysis

Most of the algorithms developed and discussed in the thesis are not limited to OES data and can be applied in various domains. They are specifically designed for high dimensional datasets with a significant level of redundancy among variables.

DNA microarray data has characteristics similar to OES data. It contains the measurements of many thousands of genes and is characterized by high dimension, high volume and highly levels of redundancy. The SPCA technique was, in fact, originally developed for microarray data analysis and many of the algorithms that can handle high dimensional data, for example lasso and random forest, are commonly used for microarray data analysis. Microarrays

can be used diagnostically to determine the disease that an individual is suffering from and to predict the effectiveness of a course of therapy. The use of microarray technology is rapidly spreading in medicine and pharmaceutical industries with several applications among which personalised medicine has major growth potential. It will be interesting to establish the utility of the algorithms developed in the thesis for analysis of this type of data.

Appendices

Appendix A

Stable Lasso

This appendix tabulates the numerical results obtained in the experiments described in Section 3.5.3 of Chapter 3.

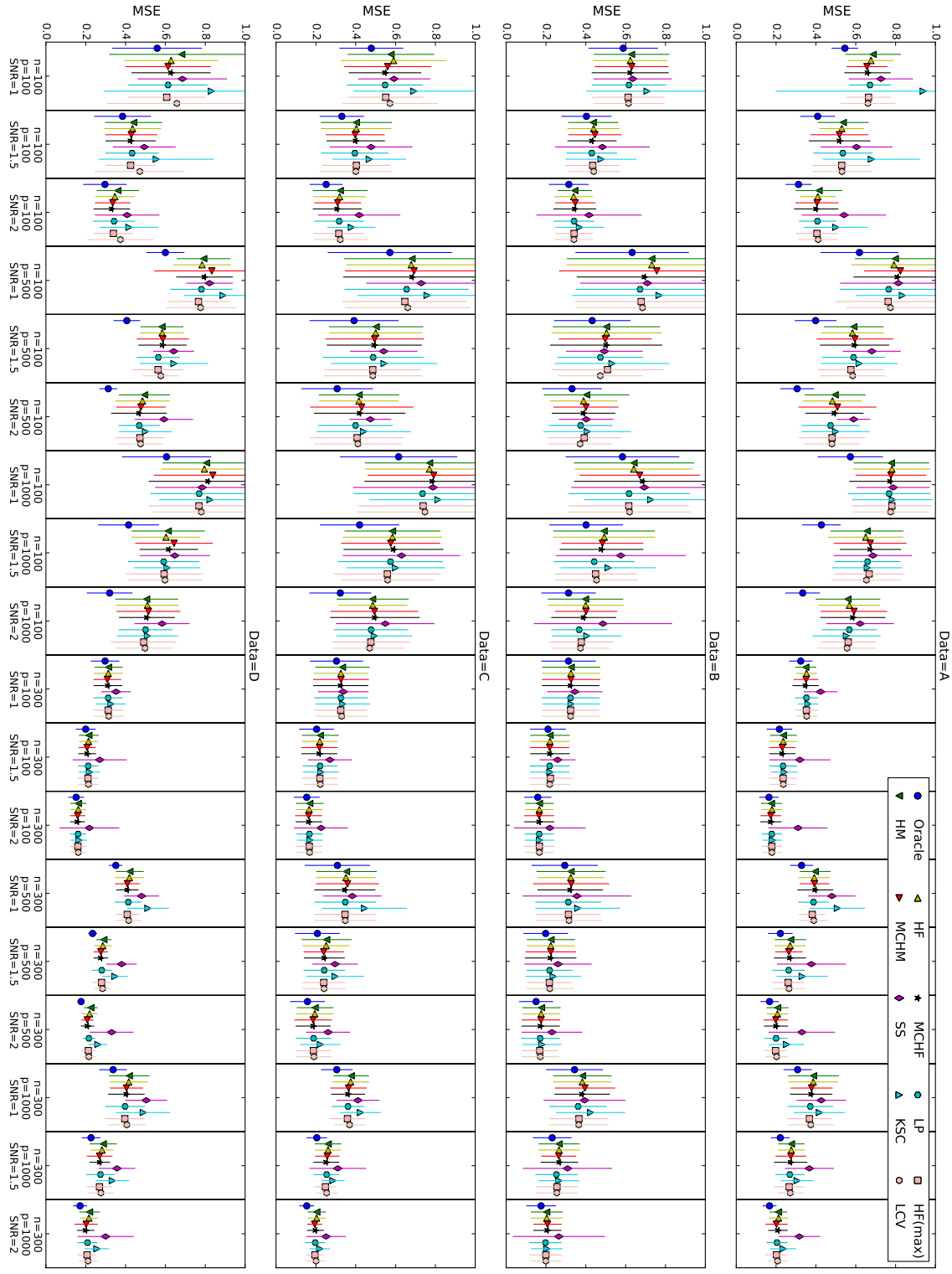


Figure A.1: Mean Square Error with 95% confidence interval on the testing data generated as described in Section 3.5.3

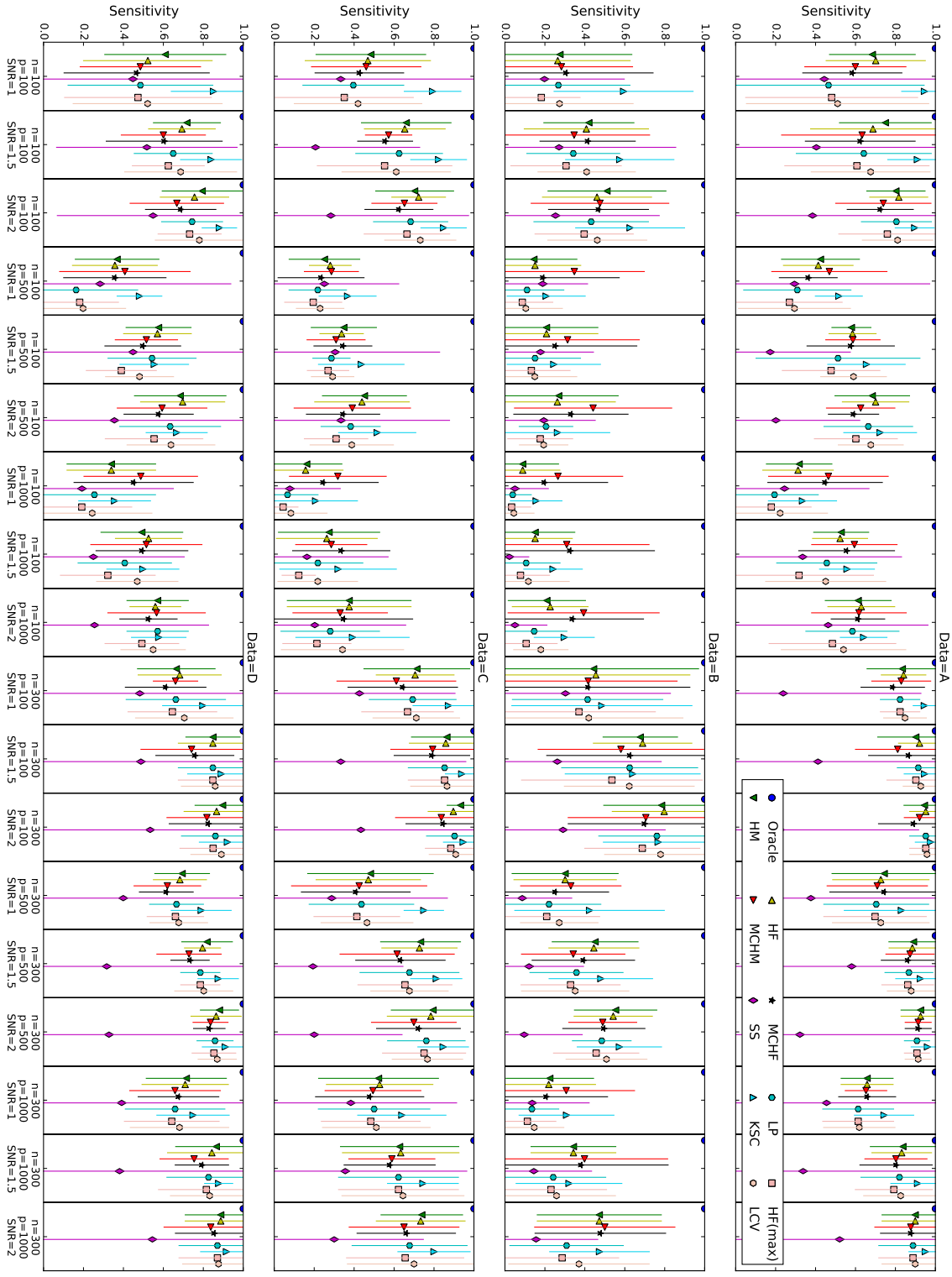


Figure A.2: Sensitivity with 95% confidence interval for the experiment described in Section 3.5.3

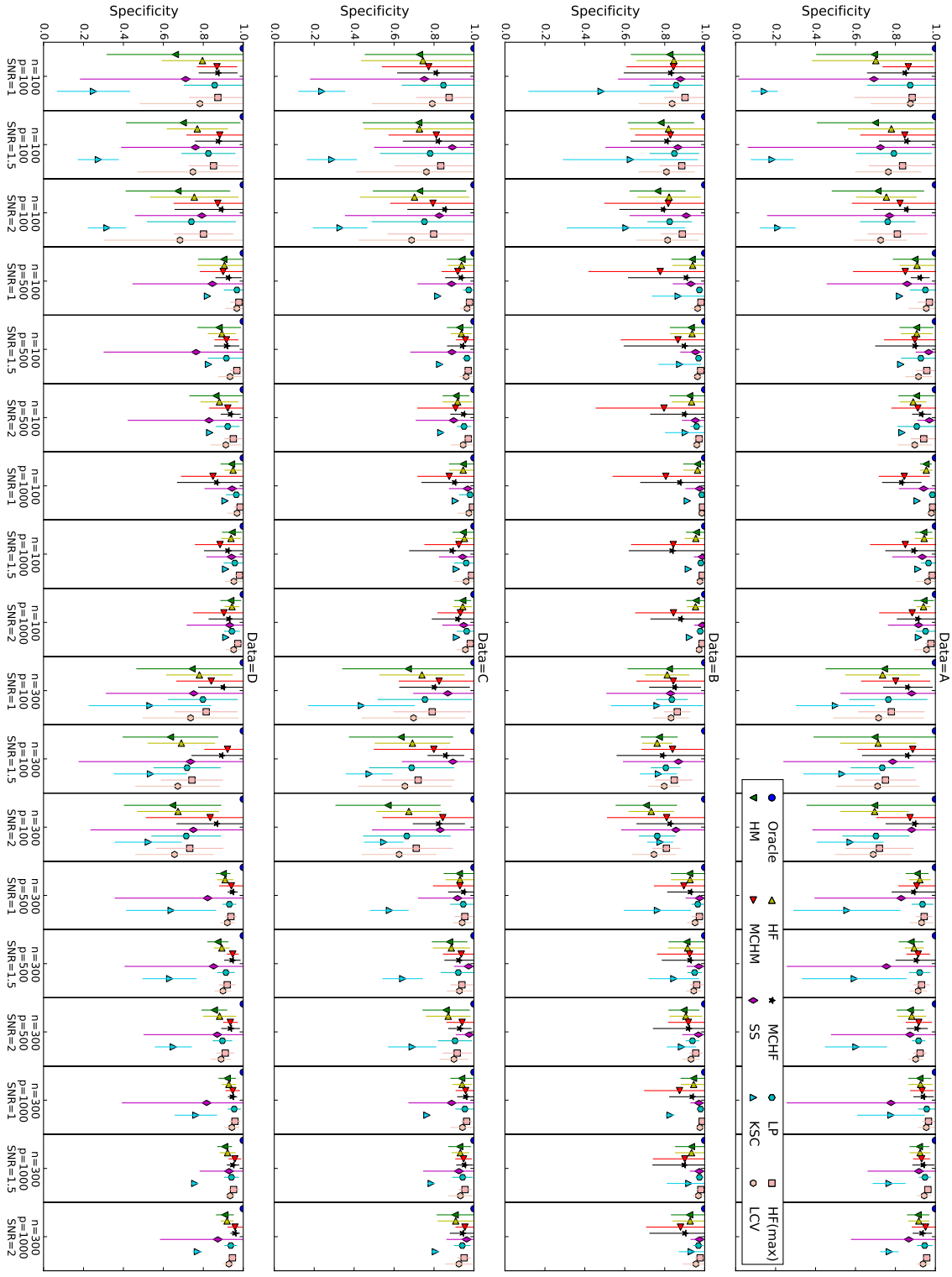


Figure A.3: Specificity with 95% confidence interval for the experiment described in Section 3.5.3

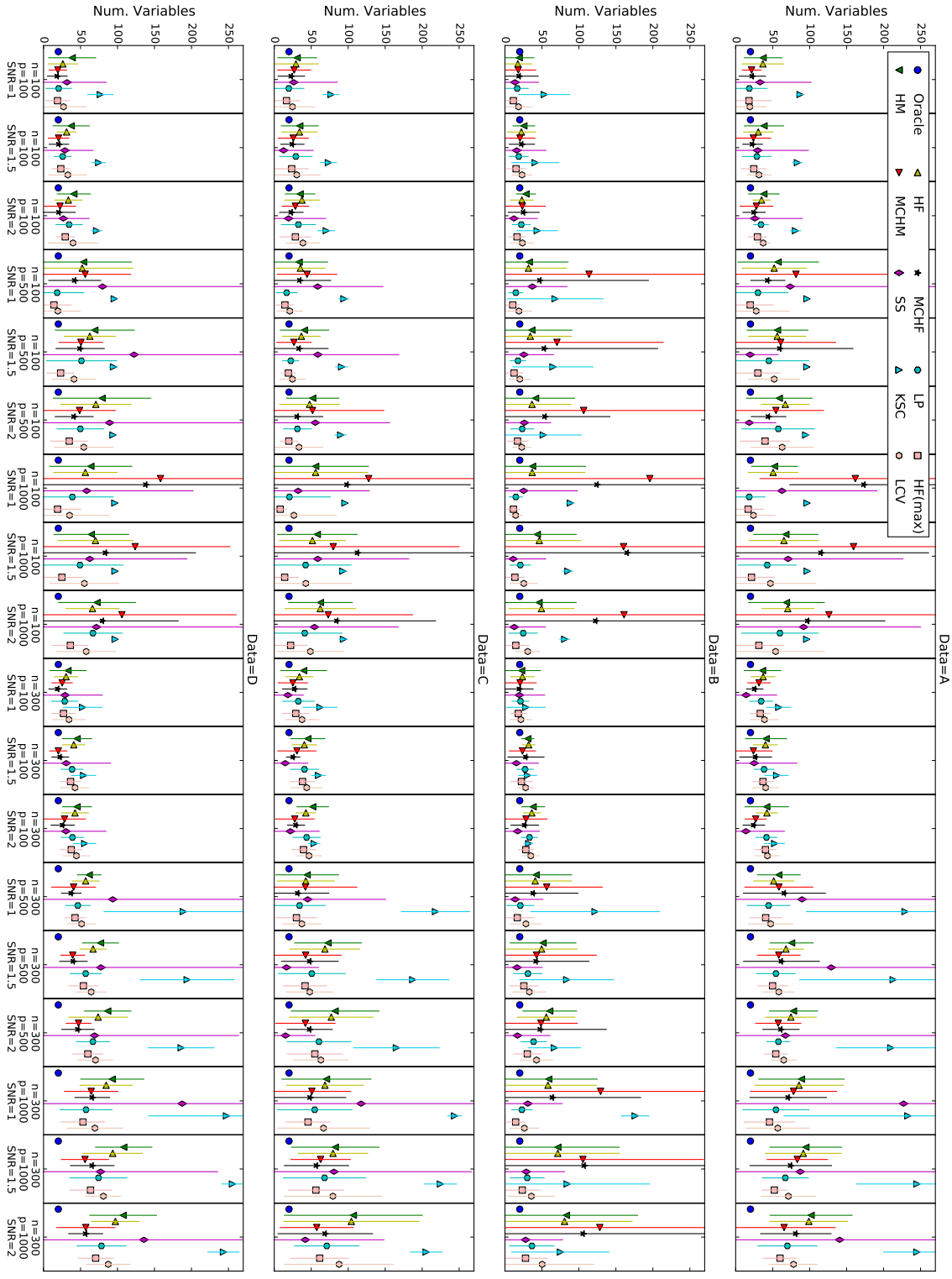


Figure A.4: Number of selected variables with 95% confidence interval for the experiment described in Section 3.5.3

	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec
	n=100 p=100 SNR=1 Data=A				n=100 p=100 SNR=1.5 Data=A				n=100 p=100 SNR=2 Data=A			
Real	0.034	0.000	0.000	0.000	0.045	0.000	0.000	0.000	0.033	0.000	0.000	0.000
HM	0.071	13.299	0.111	0.149	0.065	13.865	0.118	0.150	0.060	10.879	0.076	0.118
HF	0.058	14.601	0.128	0.163	0.057	10.477	0.160	0.111	0.046	6.245	0.075	0.077
MCHM	0.052	6.729	0.130	0.066	0.074	12.166	0.207	0.114	0.055	11.597	0.123	0.123
MCHF	0.059	9.396	0.128	0.098	0.079	7.132	0.141	0.072	0.057	8.000	0.086	0.085
HF (max)	0.057	15.540	0.221	0.148	0.074	8.890	0.186	0.086	0.054	6.784	0.125	0.077
SS	0.082	35.518	0.426	0.347	0.092	35.302	0.442	0.339	0.108	32.964	0.440	0.313
LP	0.054	12.561	0.245	0.110	0.076	10.161	0.174	0.097	0.048	5.426	0.091	0.071
KSC	0.378	3.060	0.059	0.034	0.124	3.644	0.077	0.054	0.082	3.887	0.051	0.046
LCV	0.071	11.624	0.234	0.102	0.076	8.546	0.153	0.085	0.051	5.357	0.091	0.068
	n=100 p=500 SNR=1 Data=A				n=100 p=500 SNR=1.5 Data=A				n=100 p=500 SNR=2 Data=A			
Real	0.100	0.000	0.000	0.000	0.053	0.000	0.000	0.000	0.043	0.000	0.000	0.000
HM	0.103	28.161	0.101	0.056	0.076	21.232	0.051	0.044	0.078	23.016	0.096	0.046
HF	0.109	22.391	0.09	0.045	0.080	20.017	0.061	0.040	0.075	16.744	0.086	0.034
MCHM	0.093	66.365	0.148	0.134	0.098	38.251	0.071	0.079	0.100	33.087	0.089	0.068
MCHF	0.113	12.074	0.075	0.024	0.089	50.702	0.113	0.102	0.075	12.238	0.065	0.025
HF (max)	0.133	16.233	0.136	0.030	0.085	15.588	0.126	0.028	0.085	17.420	0.107	0.033
SS	0.149	105.133	0.347	0.206	0.074	19.724	0.206	0.033	0.043	18.748	0.216	0.031
LP	0.125	21.056	0.138	0.040	0.081	27.859	0.211	0.051	0.074	25.353	0.114	0.049
KSC	0.120	1.581	0.061	0.004	0.099	1.202	0.100	0.004	0.087	2.784	0.094	0.006
LCV	0.113	23.118	0.144	0.045	0.084	17.940	0.085	0.034	0.073	21.688	0.084	0.043
	n=100 p=1000 SNR=1 Data=A				n=100 p=1000 SNR=1.5 Data=A				n=100 p=1000 SNR=2 Data=A			
Real	0.084	0.000	0.000	0.000	0.049	0.000	0.000	0.000	0.045	0.000	0.000	0.000
HM	0.097	16.203	0.085	0.015	0.094	22.452	0.071	0.022	0.083	26.467	0.085	0.026
HF	0.099	17.328	0.091	0.016	0.096	24.165	0.071	0.024	0.079	18.242	0.087	0.018
MCHM	0.091	66.056	0.154	0.065	0.093	89.997	0.110	0.090	0.085	84.954	0.122	0.084
MCHF	0.105	51.525	0.148	0.050	0.079	74.487	0.124	0.074	0.084	53.577	0.070	0.054
HF (max)	0.103	11.016	0.103	0.010	0.089	16.303	0.191	0.013	0.070	17.299	0.163	0.015
SS	0.094	65.913	0.217	0.063	0.100	79.433	0.254	0.077	0.087	80.944	0.256	0.078
LP	0.106	11.147	0.113	0.010	0.084	20.365	0.129	0.019	0.071	26.649	0.120	0.025
KSC	0.103	1.333	0.088	0.002	0.090	1.732	0.071	0.003	0.087	1.803	0.061	0.003
LCV	0.098	15.186	0.122	0.014	0.089	31.595	0.155	0.030	0.076	33.890	0.160	0.032
	n=300 p=100 SNR=1 Data=A				n=300 p=100 SNR=1.5 Data=A				n=300 p=100 SNR=2 Data=A			
Real	0.030	0.000	0.000	0.000	0.033	0.000	0.000	0.000	0.025	0.000	0.000	0.000
HM	0.027	12.874	0.091	0.150	0.034	14.469	0.099	0.157	0.027	15.297	0.053	0.174
HF	0.027	8.448	0.058	0.095	0.035	8.526	0.055	0.097	0.026	7.579	0.042	0.088
MCHM	0.032	8.033	0.077	0.088	0.034	13.210	0.109	0.141	0.027	7.574	0.04	0.087
MCHF	0.034	5.932	0.082	0.063	0.033	11.303	0.103	0.117	0.027	7.715	0.090	0.075
HF (max)	0.029	7.259	0.052	0.085	0.036	7.253	0.075	0.078	0.026	7.645	0.042	0.087
SS	0.043	21.439	0.353	0.182	0.078	29.745	0.412	0.279	0.076	26.841	0.356	0.253
LP	0.029	8.187	0.052	0.100	0.036	7.126	0.055	0.081	0.026	7.500	0.042	0.086
KSC	0.026	8.555	0.03	0.101	0.034	8.338	0.053	0.098	0.026	7.159	0.04	0.085
LCV	0.027	9.905	0.056	0.116	0.034	9.167	0.046	0.107	0.025	8.358	0.045	0.098
	n=300 p=500 SNR=1 Data=A				n=300 p=500 SNR=1.5 Data=A				n=300 p=500 SNR=2 Data=A			
Real	0.029	0.000	0.000	0.000	0.031	0.000	0.000	0.000	0.023	0.000	0.000	0.000
HM	0.040	15.050	0.135	0.029	0.040	15.199	0.064	0.032	0.028	16.865	0.051	0.035
HF	0.040	14.187	0.124	0.027	0.040	12.258	0.061	0.025	0.028	17.769	0.051	0.037
MCHM	0.038	23.744	0.129	0.047	0.035	15.040	0.063	0.030	0.032	15.820	0.034	0.033
MCHF	0.046	28.603	0.140	0.057	0.041	26.46	0.069	0.054	0.031	12.824	0.034	0.027
HF (max)	0.034	11.392	0.132	0.020	0.040	10.717	0.067	0.021	0.029	8.090	0.033	0.017
SS	0.061	113.069	0.367	0.221	0.088	129.215	0.423	0.255	0.085	105.021	0.458	0.202
LP	0.039	14.993	0.135	0.028	0.042	13.648	0.062	0.027	0.030	8.182	0.033	0.017
KSC	0.071	68.414	0.147	0.137	0.066	64.801	0.064	0.134	0.045	37.888	0.043	0.079
LCV	0.037	15.875	0.125	0.030	0.041	11.091	0.062	0.022	0.029	8.860	0.034	0.019
	n=300 p=1000 SNR=1 Data=A				n=300 p=1000 SNR=1.5 Data=A				n=300 p=1000 SNR=2 Data=A			
Real	0.036	0.000	0.000	0.000	0.024	0.000	0.000	0.000	0.018	0.000	0.000	0.000
HM	0.065	29.808	0.068	0.03	0.034	25.067	0.084	0.025	0.023	28.614	0.086	0.029
HF	0.064	30.781	0.068	0.031	0.034	26.392	0.077	0.027	0.024	27.034	0.086	0.027
MCHM	0.054	29.949	0.054	0.031	0.039	21.148	0.081	0.022	0.029	35.599	0.093	0.035
MCHF	0.054	26.676	0.074	0.026	0.043	28.470	0.094	0.028	0.025	24.637	0.079	0.025
HF (max)	0.057	17.769	0.092	0.016	0.039	9.025	0.100	0.008	0.029	11.726	0.089	0.011
SS	0.063	268.399	0.390	0.267	0.063	135.275	0.373	0.131	0.054	152.551	0.426	0.148
LP	0.058	23.104	0.092	0.022	0.037	16.167	0.099	0.015	0.027	20.757	0.089	0.020
KSC	0.066	85.668	0.077	0.086	0.042	42.070	0.070	0.042	0.033	23.170	0.043	0.023
LCV	0.059	22.245	0.092	0.021	0.033	18.999	0.096	0.018	0.026	16.776	0.086	0.017

Table A.1: The standard deviation of MSE, the number of selected variables, the sensitivity and the specificity of each method when applied to dataset A. The reported values are the standard deviation over 10 Monte Carlo repetitions.

	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec
	n=100 p=100 SNR=1 Data=B				n=100 p=100 SNR=1.5 Data=B				n=100 p=100 SNR=2 Data=B			
Real	0.089	0.000	0.000	0.000	0.064	0.000	0.000	0.000	0.051	0.000	0.000	0.000
HM	0.097	10.38	0.185	0.100	0.064	7.748	0.116	0.085	0.044	6.839	0.152	0.072
HF	0.095	9.951	0.186	0.096	0.067	10.060	0.160	0.099	0.050	8.177	0.141	0.081
MCHM	0.094	12.369	0.183	0.120	0.068	10.571	0.194	0.090	0.052	16.156	0.177	0.163
MCHF	0.099	13.371	0.224	0.120	0.063	9.020	0.123	0.094	0.055	10.977	0.129	0.113
HF (max)	0.093	5.099	0.100	0.053	0.070	6.910	0.142	0.057	0.047	5.979	0.127	0.055
SS	0.100	16.516	0.205	0.160	0.121	20.374	0.299	0.185	0.135	16.496	0.266	0.145
LP	0.095	7.939	0.184	0.069	0.065	6.837	0.120	0.063	0.052	6.314	0.147	0.057
KSC	0.156	17.915	0.179	0.186	0.091	16.412	0.140	0.172	0.063	14.293	0.140	0.151
LCV	0.091	9.695	0.189	0.086	0.067	7.114	0.126	0.073	0.050	8.079	0.128	0.080
	n=100 p=500 SNR=1 Data=B				n=100 p=500 SNR=1.5 Data=B				n=100 p=500 SNR=2 Data=B			
Real	0.145	0.000	0.000	0.000	0.098	0.000	0.000	0.000	0.077	0.000	0.000	0.000
HM	0.218	27.016	0.111	0.053	0.138	28.045	0.132	0.055	0.110	27.180	0.153	0.051
HF	0.218	26.352	0.118	0.052	0.141	28.778	0.132	0.056	0.087	26.968	0.149	0.051
MCHM	0.250	91.571	0.180	0.184	0.119	73.573	0.185	0.146	0.083	87.788	0.202	0.175
MCHF	0.173	75.127	0.196	0.149	0.144	78.363	0.210	0.155	0.081	44.964	0.147	0.088
HF (max)	0.170	4.927	0.079	0.009	0.143	6.205	0.100	0.010	0.093	7.550	0.084	0.014
SS	0.172	24.254	0.116	0.046	0.098	20.761	0.137	0.039	0.070	18.496	0.133	0.034
LP	0.165	5.123	0.095	0.008	0.000	5.570	0.117	0.009	0.082	8.246	0.070	0.016
KSC	0.222	33.29	0.101	0.067	0.147	27.957	0.120	0.055	0.112	26.584	0.135	0.052
LCV	0.171	9.089	0.095	0.016	0.109	7.322	0.108	0.013	0.083	6.690	0.063	0.014
	n=100 p=1000 SNR=1 Data=B				n=100 p=1000 SNR=1.5 Data=B				n=100 p=1000 SNR=2 Data=B			
Real	0.146	0.000	0.000	0.000	0.095	0.000	0.000	0.000	0.070	0.000	0.000	0.000
HM	0.154	36.871	0.093	0.036	0.130	27.133	0.102	0.026	0.097	25.877	0.100	0.025
HF	0.151	36.728	0.096	0.036	0.131	29.005	0.096	0.028	0.097	22.804	0.098	0.022
MCHM	0.154	136.567	0.167	0.136	0.105	109.714	0.211	0.108	0.079	99.28	0.194	0.098
MCHF	0.176	102.785	0.163	0.102	0.107	113.282	0.217	0.111	0.083	79.857	0.183	0.078
HF (max)	0.154	4.773	0.050	0.005	0.104	6.692	0.075	0.006	0.083	9.055	0.080	0.008
SS	0.187	37.366	0.087	0.037	0.166	22.746	0.051	0.022	0.177	21.807	0.083	0.021
LP	0.155	4.848	0.049	0.005	0.103	7.328	0.088	0.007	0.072	10.124	0.085	0.010
KSC	0.171	3.000	0.067	0.004	0.121	3.790	0.076	0.004	0.089	3.701	0.078	0.004
LCV	0.157	4.693	0.053	0.005	0.106	9.580	0.106	0.009	0.075	8.423	0.071	0.009
	n=300 p=100 SNR=1 Data=B				n=300 p=100 SNR=1.5 Data=B				n=300 p=100 SNR=2 Data=B			
Real	0.070	0.000	0.000	0.000	0.046	0.000	0.000	0.000	0.035	0.000	0.000	0.000
HM	0.074	13.249	0.269	0.107	0.050	4.400	0.096	0.047	0.036	8.253	0.149	0.078
HF	0.074	8.411	0.242	0.057	0.050	4.664	0.127	0.039	0.037	6.000	0.134	0.058
MCHM	0.075	11.469	0.228	0.094	0.050	9.341	0.213	0.079	0.038	14.833	0.200	0.152
MCHF	0.073	9.734	0.261	0.066	0.051	12.778	0.213	0.118	0.038	9.821	0.196	0.086
HF (max)	0.073	5.788	0.196	0.034	0.054	7.855	0.232	0.047	0.039	5.495	0.148	0.036
SS	0.071	17.728	0.269	0.164	0.046	15.517	0.267	0.142	0.092	15.252	0.262	0.140
LP	0.074	6.144	0.193	0.041	0.051	5.869	0.175	0.038	0.037	5.652	0.150	0.047
KSC	0.074	13.638	0.231	0.118	0.051	6.540	0.174	0.047	0.038	3.180	0.141	0.033
LCV	0.073	7.715	0.242	0.046	0.051	6.030	0.167	0.041	0.037	6.234	0.144	0.055
	n=300 p=500 SNR=1 Data=B				n=300 p=500 SNR=1.5 Data=B				n=300 p=500 SNR=2 Data=B			
Real	0.085	0.000	0.000	0.000	0.057	0.000	0.000	0.000	0.044	0.000	0.000	0.000
HM	0.089	24.867	0.137	0.048	0.062	22.990	0.111	0.047	0.049	18.869	0.106	0.040
HF	0.086	25.502	0.132	0.049	0.062	24.764	0.116	0.050	0.048	20.196	0.101	0.042
MCHM	0.097	38.685	0.129	0.077	0.065	41.551	0.133	0.083	0.049	25.387	0.088	0.052
MCHF	0.084	31.013	0.138	0.060	0.066	36.637	0.132	0.073	0.049	45.778	0.106	0.091
HF (max)	0.082	11.662	0.137	0.020	0.057	10.138	0.128	0.018	0.046	9.808	0.110	0.018
SS	0.140	19.499	0.128	0.036	0.086	17.642	0.140	0.031	0.078	22.751	0.150	0.041
LP	0.085	9.615	0.134	0.016	0.059	10.050	0.120	0.019	0.050	9.055	0.076	0.017
KSC	0.109	44.648	0.192	0.086	0.071	32.738	0.133	0.064	0.050	18.361	0.109	0.037
LCV	0.084	10.989	0.100	0.020	0.060	11.511	0.138	0.020	0.049	11.630	0.105	0.023
	n=300 p=1000 SNR=1 Data=B				n=300 p=1000 SNR=1.5 Data=B				n=300 p=1000 SNR=2 Data=B			
Real	0.073	0.000	0.000	0.000	0.050	0.000	0.000	0.000	0.038	0.000	0.000	0.000
HM	0.075	33.707	0.113	0.033	0.052	42.632	0.109	0.042	0.041	49.456	0.160	0.048
HF	0.074	33.599	0.121	0.033	0.052	42.779	0.109	0.042	0.041	46.782	0.160	0.045
MCHM	0.077	91.778	0.176	0.091	0.045	83.400	0.214	0.081	0.036	89.424	0.180	0.088
MCHF	0.071	61.145	0.158	0.059	0.048	83.895	0.224	0.082	0.036	91.919	0.168	0.091
HF (max)	0.075	6.784	0.074	0.006	0.053	12.208	0.143	0.010	0.042	15.246	0.146	0.013
SS	0.105	23.929	0.147	0.022	0.114	26.804	0.149	0.025	0.118	25.780	0.159	0.023
LP	0.074	7.288	0.070	0.007	0.054	12.015	0.135	0.010	0.041	15.661	0.146	0.013
KSC	0.089	9.862	0.122	0.012	0.052	57.240	0.136	0.056	0.041	34.000	0.128	0.032
LCV	0.076	10.208	0.077	0.009	0.052	16.508	0.152	0.014	0.040	36.000	0.181	0.034

Table A.2: The standard deviation of MSE, the number of selected variables, the sensitivity and the specificity of each method when applied to dataset B. The reported values are the standard deviation over 10 Monte Carlo repetitions.

	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec
	n=100 p=100 SNR=1 Data=C				n=100 p=100 SNR=1.5 Data=C				n=100 p=100 SNR=2 Data=C			
Real	0.082	0.000	0.000	0.000	0.057	0.000	0.000	0.000	0.042	0.000	0.000	0.000
HM	0.111	13.684	0.141	0.140	0.091	13.030	0.116	0.143	0.070	10.476	0.101	0.119
HF	0.136	15.628	0.161	0.158	0.090	12.524	0.104	0.142	0.067	12.32	0.070	0.140
MCHM	0.111	11.677	0.141	0.120	0.076	10.651	0.060	0.122	0.060	9.422	0.084	0.109
MCHF	0.093	9.493	0.114	0.100	0.075	8.338	0.071	0.091	0.063	8.418	0.088	0.096
HF (max)	0.097	9.554	0.178	0.084	0.087	11.759	0.173	0.117	0.065	10.933	0.110	0.118
SS	0.092	30.360	0.370	0.292	0.106	20.809	0.267	0.200	0.105	25.981	0.353	0.241
LP	0.097	10.724	0.130	0.107	0.087	11.465	0.112	0.128	0.064	12.083	0.096	0.135
KSC	0.154	5.674	0.074	0.060	0.094	5.932	0.072	0.064	0.062	6.144	0.059	0.070
LCV	0.123	15.540	0.164	0.155	0.092	16.417	0.140	0.179	0.069	11.660	0.092	0.135
	n=100 p=500 SNR=1 Data=C				n=100 p=500 SNR=1.5 Data=C				n=100 p=500 SNR=2 Data=C			
Real	0.159	0.000	0.000	0.000	0.114	0.000	0.000	0.000	0.092	0.000	0.000	0.000
HM	0.174	19.962	0.091	0.039	0.121	16.966	0.084	0.033	0.103	18.317	0.108	0.034
HF	0.166	17.349	0.054	0.035	0.120	13.295	0.057	0.027	0.102	20.843	0.122	0.039
MCHM	0.177	20.839	0.070	0.041	0.126	12.166	0.075	0.024	0.132	49.411	0.150	0.098
MCHF	0.177	21.499	0.111	0.042	0.122	20.185	0.075	0.040	0.117	17.892	0.095	0.034
HF (max)	0.160	6.227	0.074	0.012	0.124	5.183	0.054	0.011	0.119	6.164	0.082	0.011
SS	0.141	45.406	0.192	0.087	0.086	56.208	0.268	0.106	0.053	51.563	0.279	0.097
LP	0.159	7.480	0.075	0.014	0.129	5.809	0.049	0.012	0.095	9.899	0.077	0.018
KSC	0.178	3.833	0.074	0.006	0.136	4.387	0.110	0.007	0.121	4.295	0.100	0.008
LCV	0.161	9.028	0.062	0.018	0.125	8.946	0.055	0.018	0.111	16.898	0.108	0.031
	n=100 p=1000 SNR=1 Data=C				n=100 p=1000 SNR=1.5 Data=C				n=100 p=1000 SNR=2 Data=C			
Real	0.150	0.000	0.000	0.000	0.102	0.000	0.000	0.000	0.079	0.000	0.000	0.000
HM	0.166	36.657	0.091	0.036	0.123	27.797	0.131	0.028	0.092	22.317	0.159	0.022
HF	0.166	36.347	0.097	0.036	0.127	22.951	0.130	0.022	0.089	24.621	0.159	0.024
MCHM	0.170	81.265	0.125	0.081	0.127	87.224	0.092	0.088	0.112	58.498	0.122	0.059
MCHF	0.174	85.289	0.142	0.085	0.128	109.385	0.125	0.110	0.115	68.262	0.178	0.067
HF (max)	0.167	4.873	0.039	0.005	0.137	9.584	0.044	0.010	0.100	11.624	0.088	0.011
SS	0.205	49.398	0.130	0.048	0.150	63.085	0.209	0.060	0.126	58.000	0.234	0.055
LP	0.176	28.557	0.079	0.028	0.135	31.663	0.116	0.031	0.096	25.956	0.127	0.025
KSC	0.175	1.667	0.108	0.003	0.126	3.073	0.150	0.004	0.097	3.245	0.146	0.004
LCV	0.174	29.462	0.094	0.028	0.119	31.639	0.103	0.031	0.092	23.267	0.157	0.023
	n=300 p=100 SNR=1 Data=C				n=300 p=100 SNR=1.5 Data=C				n=300 p=100 SNR=2 Data=C			
Real	0.068	0.000	0.000	0.000	0.045	0.000	0.000	0.000	0.033	0.000	0.000	0.000
HM	0.070	16.169	0.136	0.169	0.047	12.095	0.092	0.133	0.035	11.158	0.035	0.134
HF	0.070	9.785	0.101	0.109	0.045	8.775	0.094	0.096	0.034	7.207	0.066	0.083
MCHM	0.072	10.402	0.153	0.103	0.047	13.491	0.108	0.153	0.035	13.664	0.118	0.154
MCHF	0.071	8.775	0.141	0.090	0.046	4.899	0.098	0.047	0.034	6.187	0.096	0.067
HF (max)	0.068	9.427	0.118	0.100	0.045	8.482	0.094	0.093	0.034	8.383	0.066	0.094
SS	0.065	10.967	0.245	0.088	0.056	15.951	0.321	0.130	0.069	20.216	0.362	0.174
LP	0.069	11.159	0.112	0.121	0.045	9.968	0.094	0.109	0.033	9.757	0.073	0.112
KSC	0.070	12.036	0.092	0.137	0.046	5.061	0.043	0.060	0.034	4.243	0.052	0.051
LCV	0.066	12.012	0.112	0.132	0.044	10.868	0.094	0.120	0.033	8.602	0.069	0.096
	n=300 p=500 SNR=1 Data=C				n=300 p=500 SNR=1.5 Data=C				n=300 p=500 SNR=2 Data=C			
Real	0.084	0.000	0.000	0.000	0.057	0.000	0.000	0.000	0.045	0.000	0.000	0.000
HM	0.078	21.932	0.162	0.040	0.064	23.355	0.103	0.045	0047	30.594	0.108	0.060
HF	0.076	20.000	0.135	0.037	0.060	24.409	0.097	0.048	0.048	29.187	0.112	0.057
MCHM	0.081	35.904	0.174	0.069	0.052	24.83	0.147	0.047	0.048	20.799	0.109	0.040
MCHF	0.078	21.616	0.140	0.04	0.052	19.871	0.115	0.038	0.045	15.788	0.108	0.030
HF (max)	0.078	14.080	0.111	0.026	0.055	15.313	0.121	0.029	0.045	19.348	0.107	0.037
SS	0.075	54.032	0.297	0.101	0.058	22.333	0.231	0.037	0.056	20.523	0.226	0.034
LP	0.078	18.054	0.135	0.034	0.053	23.296	0.128	0.045	0.045	22.317	0.100	0.044
KSC	0.108	23.822	0.052	0.049	0.074	25.208	0.067	0.052	0.052	29.793	0.066	0.062
LCV	0.079	13.379	0.119	0.024	0.055	16.432	0.100	0.032	0.045	19.026	0.091	0.037
	n=300 p=1000 SNR=1 Data=C				n=300 p=1000 SNR=1.5 Data=C				n=300 p=1000 SNR=2 Data=C			
Real	0.041	0.000	0.000	0.000	0.026	0.000	0.000	0.000	0.019	0.000	0.000	0.000
HM	0.045	31.008	0.154	0.029	0.034	30.546	0.152	0.029	0.023	47.877	0.105	0.047
HF	0.047	26.773	0.137	0.025	0.033	24.198	0.150	0.023	0.024	46.882	0.115	0.046
MCHM	0.046	26.909	0.124	0.025	0.031	20.833	0.111	0.021	0.019	25.505	0.141	0.024
MCHF	0.043	24.845	0.139	0.023	0.033	22.464	0.118	0.022	0.022	32.928	0.127	0.033
HF (max)	0.045	15.500	0.127	0.013	0.027	19.365	0.154	0.017	0.022	20.175	0.151	0.018
SS	0.055	113.100	0.272	0.110	0.073	95.265	0.312	0.092	0.051	54.567	0.229	0.052
LP	0.042	25.952	0.144	0.024	0.033	28.644	0.154	0.026	0.025	22.517	0.148	0.021
KSC	0.052	4.944	0.114	0.004	0.030	11.402	0.092	0.010	0.025	11.281	0.094	0.010
LCV	0.039	31.86	0.139	0.031	0.027	33.838	0.157	0.032	0.026	37.868	0.171	0.036

Table A.3: The standard deviation value of MSE, the number of selected variables, the sensitivity and the specificity of each method when applied to dataset C. The reported values are the standard deviation over 10 Monte Carlo repetitions.

	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec	MSE	N. Var	Sens	Spec
	n=100 p=100 SNR=1 Data=D				n=100 p=100 SNR=1.5 Data=D				n=100 p=100 SNR=2 Data=D			
Real	0.115	0.000	0.000	0.000	0.073	0.000	0.000	0.000	0.056	0.000	0.000	0.000
HM	0.186	16.529	0.156	0.176	0.073	12.788	0.087	0.146	0.054	11.758	0.103	0.134
HF	0.120	10.529	0.165	0.104	0.073	6.833	0.086	0.078	0.051	9.697	0.089	0.113
MCHM	0.109	6.227	0.155	0.052	0.066	7.348	0.109	0.085	0.045	11.113	0.121	0.113
MCHF	0.101	7.106	0.187	0.05	0.065	6.972	0.149	0.060	0.047	11.396	0.091	0.120
HF (max)	0.099	8.551	0.189	0.074	0.065	6.265	0.093	0.063	0.050	6.652	0.082	0.076
SS	0.114	27.432	0.308	0.270	0.080	19.554	0.232	0.190	0.083	18.061	0.246	0.171
LP	0.102	9.043	0.186	0.078	0.069	6.234	0.101	0.069	0.054	9.501	0.079	0.114
KSC	0.276	8.997	0.109	0.094	0.147	4.876	0.079	0.053	0.075	3.962	0.045	0.049
LCV	0.179	15.664	0.191	0.154	0.115	13.402	0.144	0.142	0.084	17.324	0.114	0.194
	n=100 p=500 SNR=1 Data=D				n=100 p=500 SNR=1.5 Data=D				n=100 p=500 SNR=2 Data=D			
Real	0.049	0.000	0.000	0.000	0.034	0.000	0.000	0.000	0.023	0.000	0.000	0.000
HM	0.069	33.292	0.108	0.066	0.055	27.559	0.084	0.056	0.066	33.918	0.118	0.068
HF	0.074	35.189	0.110	0.07	0.057	17.98	0.088	0.036	0.070	24.529	0.109	0.049
MCHM	0.147	31.540	0.168	0.059	0.067	15.363	0.081	0.030	0.063	24.823	0.116	0.048
MCHF	0.072	18.254	0.132	0.034	0.062	16.935	0.098	0.032	0.071	13.427	0.090	0.026
HF (max)	0.080	12.221	0.100	0.022	0.066	9.701	0.091	0.018	0.063	13.036	0.126	0.023
SS	0.060	104.567	0.335	0.204	0.052	120.634	0.386	0.236	0.074	106.421	0.352	0.208
LP	0.080	18.709	0.159	0.033	0.055	24.504	0.114	0.047	0.053	16.400	0.130	0.030
KSC	0.099	2.128	0.058	0.004	0.087	3.087	0.089	0.006	0.068	1.764	0.079	0.003
LCV	0.090	15.748	0.110	0.029	0.047	15.023	0.088	0.03	0.057	20.501	0.113	0.039
	n=100 p=1000 SNR=1 Data=D				n=100 p=1000 SNR=1.5 Data=D				n=100 p=1000 SNR=2 Data=D			
Real	0.114	0.000	0.000	0.000	0.078	0.000	0.000	0.000	0.059	0.000	0.000	0.000
HM	0.112	28.548	0.114	0.028	0.094	26.116	0.106	0.026	0.081	26.851	0.080	0.027
HF	0.111	22.176	0.114	0.022	0.088	26.295	0.086	0.026	0.080	18.72	0.066	0.019
MCHM	0.151	82.142	0.146	0.082	0.099	65.674	0.143	0.064	0.082	79.090	0.126	0.078
MCHF	0.152	102.261	0.153	0.102	0.075	62.315	0.118	0.062	0.072	52.37	0.074	0.052
HF (max)	0.128	16.492	0.129	0.014	0.098	13.944	0.122	0.012	0.084	12.578	0.095	0.012
SS	0.120	73.622	0.235	0.071	0.090	67.330	0.233	0.064	0.071	113.160	0.293	0.110
LP	0.125	28.513	0.157	0.027	0.091	30.004	0.120	0.028	0.069	20.382	0.080	0.020
KSC	0.131	1.500	0.093	0.002	0.085	1.590	0.093	0.002	0.079	1.716	0.071	0.002
LCV	0.123	27.435	0.154	0.025	0.096	24.020	0.105	0.023	0.074	20.773	0.084	0.020
	n=300 p=100 SNR=1 Data=D				n=300 p=100 SNR=1.5 Data=D				n=300 p=100 SNR=2 Data=D			
Real	0.036	0.000	0.000	0.000	0.025	0.000	0.000	0.000	0.020	0.000	0.000	0.000
HM	0.036	12.722	0.100	0.143	0.025	10.452	0.070	0.122	0.021	10.235	0.072	0.124
HF	0.037	8.293	0.107	0.085	0.024	7.897	0.089	0.086	0.020	9.528	0.084	0.105
MCHM	0.035	7.311	0.058	0.089	0.022	6.305	0.131	0.059	0.020	14.664	0.103	0.164
MCHF	0.037	6.500	0.104	0.065	0.023	6.103	0.101	0.078	0.019	8.337	0.101	0.103
HF (max)	0.038	8.276	0.115	0.080	0.026	7.213	0.089	0.080	0.020	7.650	0.086	0.086
SS	0.038	25.964	0.434	0.224	0.07	30.948	0.435	0.286	0.077	27.924	0.385	0.263
LP	0.037	9.341	0.128	0.089	0.026	7.764	0.089	0.086	0.021	8.038	0.088	0.089
KSC	0.039	13.728	0.104	0.157	0.027	8.614	0.087	0.094	0.021	7.890	0.073	0.086
LCV	0.039	11.436	0.126	0.123	0.025	10.064	0.088	0.108	0.020	9.235	0.079	0.100
	n=300 p=500 SNR=1 Data=D				n=300 p=500 SNR=1.5 Data=D				n=300 p=500 SNR=2 Data=D			
Real	0.017	0.000	0.000	0.000	0.012	0.000	0.000	0.000	0.009	0.000	0.000	0.000
HM	0.035	8.423	0.071	0.019	0.020	12.713	0.066	0.027	0.017	16.354	0.050	0.033
HF	0.038	9.500	0.069	0.021	0.022	9.407	0.047	0.020	0.020	20.651	0.065	0.042
MCHM	0.032	15.476	0.086	0.031	0.018	8.531	0.083	0.015	0.016	8.803	0.046	0.019
MCHF	0.029	7.091	0.070	0.013	0.019	9.812	0.05	0.021	0.017	11.425	0.041	0.024
HF (max)	0.030	6.984	0.073	0.014	0.025	10.252	0.051	0.021	0.018	10.671	0.057	0.022
SS	0.045	122.379	0.467	0.237	0.039	117.862	0.463	0.227	0.056	99.594	0.468	0.188
LP	0.035	8.803	0.070	0.018	0.025	11.000	0.051	0.023	0.016	11.819	0.047	0.025
KSC	0.054	55.039	0.078	0.114	0.033	32.743	0.053	0.069	0.025	23.022	0.060	0.047
LCV	0.033	9.645	0.075	0.019	0.021	10.944	0.075	0.021	0.017	12.460	0.050	0.025
	n=300 p=1000 SNR=1 Data=D				n=300 p=1000 SNR=1.5 Data=D				n=300 p=1000 SNR=2 Data=D			
Real	0.036	0.000	0.000	0.000	0.024	0.000	0.000	0.000	0.018	0.000	0.000	0.000
HM	0.052	22.014	0.104	0.022	0.035	19.743	0.105	0.019	0.026	23.324	0.092	0.023
HF	0.049	18.296	0.112	0.018	0.029	20.749	0.114	0.020	0.023	16.714	0.092	0.016
MCHM	0.043	18.716	0.117	0.018	0.033	16.654	0.088	0.016	0.030	20.429	0.121	0.019
MCHF	0.047	12.278	0.104	0.012	0.026	15.341	0.069	0.016	0.022	12.021	0.100	0.012
HF (max)	0.051	15.492	0.122	0.014	0.033	14.867	0.123	0.014	0.024	12.320	0.099	0.011
SS	0.054	219.308	0.375	0.216	0.047	81.008	0.38	0.075	0.072	151.151	0.404	0.148
LP	0.051	18.256	0.128	0.017	0.037	19.931	0.108	0.019	0.026	17.255	0.099	0.016
KSC	0.069	54.088	0.094	0.054	0.043	7.399	0.037	0.008	0.031	11.077	0.067	0.012
LCV	0.048	19.462	0.127	0.019	0.032	12.584	0.101	0.013	0.024	15.388	0.093	0.015

Table A.4: The standard deviation of MSE, the number of selected variables, the sensitivity and the specificity of each method when applied to dataset D. The reported values are the standard deviation over 10 Monte Carlo repetitions.

Appendix B

Linear Unsupervised Variables Selection

The proofs of two results discussed in Chapter 4 are reported here.

Theorem B.0.1. *Given projection matrix* $\Phi(\mathbf{S})$ *as defined in eqt. (4.9) and* $V_{\mathbf{X}}(.)$ *as defined in eqt. (4.3), if* $\hat{\mathbf{X}} = \Phi(\mathbf{S})\mathbf{X}$ *then* $V_{\mathbf{X}}(\hat{\mathbf{X}}) \geq 0 \ \forall \mathbf{S} \in \mathbb{R}^{n \times k}$.

Proof. Choosing $\hat{\mathbf{X}} = \Phi(\mathbf{S})\mathbf{X}$ is equivalent to choosing $\hat{\mathbf{X}} = \mathbf{S}\boldsymbol{\Theta}$, where $\boldsymbol{\Theta}$ is the solution to the convex minimization problem in eqt. (4.7). By definition, the $\boldsymbol{\Theta}$ that minimizes eqt. (4.7) maximizes $V_{\mathbf{X}}(\hat{\mathbf{X}})$, that is:

$$\boldsymbol{\Theta} = \underset{\tilde{\boldsymbol{\Theta}}}{argmax} V_{\mathbf{X}}(\mathbf{S}\tilde{\boldsymbol{\Theta}}) \tag{B.1}$$

It then follows that $\mathbf{V}_{\mathbf{X}}(\mathbf{S}\boldsymbol{\Theta}) \geq \mathbf{V}_{\mathbf{X}}(\mathbf{S}\tilde{\boldsymbol{\Theta}}) \ \forall \tilde{\boldsymbol{\Theta}}$. Choosing $\tilde{\boldsymbol{\Theta}}$ as the zero matrix $\mathbf{0}$ gives $\hat{\mathbf{X}} = \mathbf{0}$ and hence $\mathbf{V}_{\mathbf{X}}(\mathbf{0}) = 0$. Therefore $\mathbf{V}_{\mathbf{X}}(\mathbf{S}\boldsymbol{\Theta}) \geq \mathbf{V}_{\mathbf{X}}(\mathbf{0}) = 0$.

□

Theorem B.0.2. *If* $V_{\mathbf{X}}(.)$ *is defined as in eqt. (4.3) and* $\Phi(.)$ *defined as in eqt. (4.9) then*

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \ V_{\mathbf{X}}(\Phi(\mathbf{x}_i)\mathbf{X}) = \underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \frac{\mathbf{x}_i \mathbf{X}\mathbf{X}^T \mathbf{x}_i}{\mathbf{x}_i^T \mathbf{x}_i}. \tag{B.2}$$

Proof. It immediately follows from eqt.(4.3) that

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmax} \ V_{\mathbf{X}}(\hat{\mathbf{X}}) = \underset{\mathbf{x}_i \in \mathbf{X}}{argmin} \parallel \mathbf{X} - \hat{\mathbf{X}} \parallel_F^2, \tag{B.3}$$

where $\hat{\mathbf{X}} = \Phi(\mathbf{x}_i)\mathbf{X}$. Expressing the F-norm in terms of the trace operator gives

$$\begin{aligned} \| \mathbf{X} - \hat{\mathbf{X}} \|_F^2 &= tr((\mathbf{X} - \hat{\mathbf{X}})(\mathbf{X} - \hat{\mathbf{X}})^T) \\ &= tr(\mathbf{X}\mathbf{X}^T) + tr(\hat{\mathbf{X}}\hat{\mathbf{X}}^T) - 2tr(\mathbf{X}\hat{\mathbf{X}}) \\ &= tr(\mathbf{X}\mathbf{X}^T) + tr(\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T\Phi(\mathbf{x}_i)) - 2tr(\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T), \end{aligned} \tag{B.4}$$

where the last equivalence is obtained by replacing $\hat{\mathbf{X}}$ with $\Phi(\mathbf{x}_i)\mathbf{X}$ and noting that $\Phi(\mathbf{x}_i) = \Phi(\mathbf{x}_i)^T$. By application of the cyclic property of the trace operator and observing that $\Phi(\mathbf{x}_i)^2 = \Phi(\mathbf{x}_i)$ we can write $tr(\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T\Phi(\mathbf{x}_i)) = tr(\Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T) = tr(\mathbf{X}\mathbf{X}^T)$, and therefore

$$\| \mathbf{X} - \hat{\mathbf{X}} \|_F^2 = 2tr(\mathbf{X}\mathbf{X}^T) - 2tr(\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T).$$

It immediately follows that

$$\underset{\mathbf{x}_i \in \mathbf{X}}{argmin} \| \mathbf{X} - \hat{\mathbf{X}} \|_F = \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ tr(\Phi(\mathbf{x}_i)\mathbf{X}\mathbf{X}^T).$$

Finally, by application of the definition of $\Phi(.)$ in eqt. (4.9) and the properties of trace, the r.h.s. can be rewritten as

$$\begin{aligned} \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ tr(\frac{\mathbf{x}_i\mathbf{x}_i^T}{\mathbf{x}_i^T\mathbf{x}_i}\mathbf{X}\mathbf{X}^T) &= \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ tr(\frac{\mathbf{x}_i^T}{\mathbf{x}_i\mathbf{x}_i^T}\mathbf{X}\mathbf{X}^T\mathbf{x}_i) \\ &= \underset{\mathbf{x}_i \in \mathbf{X}}{argmax}\ \frac{\mathbf{x}_i^T\mathbf{X}\mathbf{X}^T\mathbf{x}_i}{\mathbf{x}_i^T\mathbf{x}_i}. \end{aligned} \tag{B.5}$$

□

Bibliography

[1] H. H. Henkle, S. Lubetzky, and F. Rider. *The Scholar and the Future of the Research Library.* 1945.

[2] R. J. Morris and B. J. Truskowski. "The evolution of storage systems". In: *IBM systems Journal* 42.2 (2003), pp. 205–217.

[3] A. McAfee et al. "Big data". In: *The management revolution. Harvard Bus Rev* 90.10 (2012), pp. 61–67.

[4] F. Yinug. *U.S. Semiconductor Industry Employment.* 2015.

[5] S. I. Association. *Global Semiconductor Sales Top 335 Billion Dollars in 2015.* 2016.

[6] R. R. Schaller. "Moore's law: past, present and future". In: *IEEE spectrum* 34.6 (1997), pp. 52–59.

[7] *International Roadmap Committee. Executive summary. Technical report, International Technology Roadmap for Semiconductors, 2007.*

[8] T. F. Edgar et al. "Automatic control in microelectronics manufacturing: Practices, challenges, and possibilities". In: *Automatica* 36.11 (2000), pp. 1567–1603.

[9] M. Quirk and J. Serda. *Semiconductor manufacturing technology.* Vol. 1. Prentice Hall Upper Saddle River, NJ, 2001.

[10] M. A. Lieberman and A. J. Lichtenberg. *Principles of plasma discharges and materials processing.* John Wiley & Sons, 2005.

[11] J. Coburn and M. Chen. "Optical emission spectroscopy of reactive plasmas: A method for correlating emission intensities to reactive particle density". In: *Journal of applied physics* 51.6 (1980), pp. 3134–3136.

[12] H. H. Yue et al. "Fault detection of plasma etchers using optical emission spectra". In: *Semiconductor Manufacturing, IEEE Transactions on* 13.3 (2000), pp. 374–385.

[13] R. Winge et al. "ICP emission spectrometry: on the selection of analytical lines, line coincidence tables, and wavelength tables". In: *Applied Spectroscopy* 36.3 (1982), pp. 210–221.

[14] R. Gribonval, V. Cevher, and M. E. Davies. "Compressible distributions for high-dimensional statistics". In: *IEEE Transactions on Information Theory* 58.8 (2012), pp. 5016–5034.

[15] C. Ding and H. Peng. "Minimum redundancy feature selection from microarray gene expression data". In: *Journal of bioinformatics and computational biology* 3.02 (2005), pp. 185–205.

[16] J. F. Edward A. Feigenbaum et al. *Computers and thought*. New York, 1963.

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

[18] L. Puggini, J. Doyle, and S. McLoone. "Towards multi-sensor spectral alignment through post measurement calibration correction". In: *Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014). 25th IET*. IET. 2013, pp. 402–407.

[19] L. Puggini and S. McLoone. "Forward Selection Component Analysis: Algorithms and Applications". In: *Transaction on Pattern Analysis and Machine Intelligence* (2017).

[20] L. Puggini and S. McLoone. "Nonlinear Forward Selection Component Analysis for Optical Emission Spectroscopy Wavelength Selection." In: *Signals and Systems Conference (ISSC), 2016 27th Irish. IEEE, 2015*. 2016.

[21] L. Puggini and S. McLoone. "Feature Selection for Anomaly Detection Using Optical Emission Spectroscopy." In: *4th IFAC International Conference on Intelligent Control and Automation Sciences (ICONS 2016)*. 2016.

[22] L. Puggini and S. McLoone. "Extreme learning machines for virtual metrology and etch rate prediction". In: *Signals and Systems Conference (ISSC), 2015 26th Irish*. IEEE. 2015, pp. 1–6.

[23] L. Puggini, J. Doyle, and S. McLoone. "Fault Detection using Random Forest Similarity Distance". In: *IFAC-PapersOnLine* 48.21 (2015), pp. 583–588.

[24] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Neural Networks, 1995. Proceedings., IEEE International Conference on.* Vol. 4. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[25] B. N. Chapman. *Glow discharge processes.* Wiley, 1980.

[26] H. Abe, M. Yoneda, and N. Fujiwara. "Developments of plasma etching technology for fabricating semiconductor devices". In: *Japanese Journal of Applied Physics* 47.3R (2008), p. 1435.

[27] T. Ono et al. "Reactive ion stream etching utilizing electron cyclotron resonance plasma". In: *Journal of Vacuum Science & Technology B* 4.3 (1986), pp. 696–700.

[28] S. Grauby et al. "High resolution photothermal imaging of high frequency phenomena using a visible charge coupled device camera associated with a multichannel lock-in scheme". In: *Review of Scientific Instruments* 70.9 (1999), pp. 3603–3608.

[29] G. Bacelli and J. Ringwood. *Tracking plasma etch process variations using Principal Component Analysis of OES data.* Tech. rep. Maynooth University, 2007.

[30] S. J. Qin et al. "Semiconductor manufacturing process control and monitoring: A fab-wide framework". In: *Journal of Process Control* 16.3 (2006), pp. 179–191.

[31] D. A. Ross et al. "Incremental learning for robust visual tracking". In: *International Journal of Computer Vision* 77.1-3 (2008), pp. 125–141.

[32] J. Lin et al. "A symbolic representation of time series, with implications for streaming algorithms". In: *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery.* ACM. 2003, pp. 2–11.

[33] Z. Li et al. "Dimensionality reduction for anomaly detection in electrocardiography: A manifold approach". In: *Wearable and Implantable Body Sensor Networks (BSN), 2012 Ninth International Conference on.* IEEE. 2012, pp. 161–165.

[34] P. Monchamp et al. "Signal Processing Methods for Mass Spectrometry". In: *Systems Bioinformatics: An Engineering Case-Based Approach, Artech House Publishers* (2007).

[35] N. Jeffries. "Algorithms for alignment of mass spectrometry proteomic data". In: *Bioinformatics* 21.14 (2005), pp. 3066–3073.

[36] G. C. Lee and D. L. Woodruff. "Beam search for peak alignment of NMR signals". In: *Analytica Chimica Acta* 513.2 (2004), pp. 413 –416. ISSN: 0003-2670. DOI: http://dx.doi.org/10.1016/j.aca.2004.02.068.

[37] J. Forshed et al. "A comparison of methods for alignment of {NMR} peaks in the context of cluster analysis". In: *Journal of Pharmaceutical and Biomedical Analysis* 38.5 (2005). Quantitative {NMR} Spectroscopy principles and applications, pp. 824 –832. ISSN: 0731-7085. DOI: http://dx.doi.org/10.1016/j.jpba.2005.01.042.

[38] R. J. Torgrip et al. "Peak alignment using reduced set mapping". In: *Journal of Chemometrics* 17.11 (2003), pp. 573–582.

[39] Q. P. He et al. "Self-calibrated warping for mass spectra alignment". In: *Cancer informatics* 10 (2011), p. 65.

[40] N. P. V. Nielsen, J. M. Carstensen, and J. Smedsgaard. "Aligning of single and multiple wavelength chromatographic profiles for chemometric data analysis using correlation optimised warping". In: *Journal of Chromatography A* 805.1 (1998), pp. 17–35.

[41] G. Tomasi, F. van den Berg, and C. Andersson. "Correlation optimized warping and dynamic time warping as preprocessing methods for chromatographic data". In: *Journal of Chemometrics* 18.5 (2004), pp. 231–241.

[42] A. van Nederkassel et al. "A comparison of three algorithms for chromatograms alignment". In: *Journal of Chromatography A* 1118.2 (2006), pp. 199 –210. ISSN: 0021-9673. DOI: http://dx.doi.org/10.1016/j.chroma.2006.03.114.

[43] V. Pravdova, B. Walczak, and D. L. Massart. "A comparison of two algorithms for warping of analytical signals". In: *Analytica Chimica Acta* 456.1 (2002), pp. 77–92.

[44] P. H. C. Eilers. "Parametric Time Warping". In: *Analytical Chemistry* 76.2 (2004), pp. 404–411. DOI: 10.1021/ac034800e. eprint: http://pubs.acs.org/doi/pdf/10.1021/ac034800e.

[45] J. W. Wong, C. Durante, and H. M. Cartwright. "Application of fast Fourier transform cross-correlation for the alignment of large chromatographic and spectral datasets". In: *Analytical Chemistry* 77.17 (2005), pp. 5655–5661.

[46] J. W. Wong, G. Cagney, and H. M. Cartwright. "SpecAlign—processing and Alignment of Mass Spectra Datasets". In: *Bioinformatics* 21.9 (May 2005), pp. 2088–2090. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bti300.

[47] C. De Boor et al. *A practical guide to splines*. Vol. 27. Springer-Verlag New York, 1978.

[48] C. A. C. Coello. "A comprehensive survey of evolutionary-based multiobjective optimization techniques". In: *Knowledge and Information systems* 1.3 (1999), pp. 269–308.

[49] R. T. Marler and J. S. Arora. "Survey of multi-objective optimization methods for engineering". In: *Structural and multidisciplinary optimization* 26.6 (2004), pp. 369–395.

[50] K. E. Parsopoulos and M. N. Vrahatis. "Particle swarm optimization method in multiobjective problems". In: *Proceedings of the 2002 ACM symposium on Applied computing*. ACM. 2002, pp. 603–607.

[51] J. Kennedy. "Social interaction is a powerful optimiser: The particle swarm". In: *Bio-Inspired Computing: Theories and Applications, 2008. BICTA 2008. 3rd International Conference on*. IEEE. 2008, pp. 9–10.

[52] M. Clerc and J. Kennedy. "The particle swarm-explosion, stability, and convergence in a multidimensional complex space". In: *Evolutionary Computation, IEEE Transactions on* 6.1 (2002), pp. 58–73.

[53] F.-A. Fortin et al. "DEAP: Evolutionary algorithms made easy". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 2171–2175.

[54] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart. "The mahalanobis distance". In: *Chemometrics and intelligent laboratory systems* 50.1 (2000), pp. 1–18.

[55] G. A. Susto and A. Beghi. "A virtual metrology system based on least angle regression and statistical clustering". In: *Applied Stochastic Models in Business and Industry* 29.4 (2013), pp. 362–376.

[56] M. Zanon et al. "Regularised Model Identification Improves Accuracy of Multisensor Systems for Noninvasive Continuous Glucose Monitoring in Diabetes Management". In: *Journal of Applied Mathematics* (2013).

[57] T. T. Wu et al. "Genome-wide association analysis by lasso penalized logistic regression". In: *Bioinformatics* 25.6 (2009), pp. 714–721.

[58] M. Zanon, G. A. Susto, and S. McLoone. "Root Cause Analysis by a Combined Sparse ClassiÃđcation and Monte Carlo Approach". In: *Proc. IFAC World Congress*. 2014.

[59] P. Bühlmann and S. Van De Geer. *Statistics for high-dimensional data: methods, theory and applications*. Springer, 2011.

[60] T. Hastie et al. *The elements of statistical learning*. Vol. 2. Springer, 2009.

[61] P. Pudil, J. Novovičová, and J. Kittler. "Floating search methods in feature selection". In: *Pattern recognition letters* 15.11 (1994), pp. 1119–1125.

[62] K. Li, J. X. Peng, and E. W. Bai. "A two-stage algorithm for identification of nonlinear dynamic systems". In: *Automatica* 42.7 (2006), pp. 1189–1197.

[63] R. Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[64] S. S. Chen, D. L. Donoho, and M. A. Saunders. "Atomic decomposition by basis pursuit". In: *SIAM journal on scientific computing* 20.1 (1998), pp. 33–61.

[65] H. Zou and T. Hastie. "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320.

[66] L. Meier, S. Van De Geer, and P. Bühlmann. "The group lasso for logistic regression". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70.1 (2008), pp. 53–71.

[67] M. Yuan and Y. Lin. "Model selection and estimation in regression with grouped variables". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.1 (2006), pp. 49–67.

[68] E. Candes and T. Tao. "The Dantzig selector: Statistical estimation when p is much larger than n". In: *The Annals of Statistics* (2007), pp. 2313–2351.

[69] M. Yuan and Y. Lin. "On the non-negative garrotte estimator". In: *ournal of the Royal Statistical Society: Series B (Statistical Methodology)* 69.2 (2007), pp. 143–161.

[70] H. Xu, C. Caramanis, and S. Mannor. "Sparse algorithms are not stable: A no-free-lunch theorem". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.1 (2012), pp. 187–193.

[71] N. Meinshausen and P. Bühlmann. "Stability selection". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.4 (2010), pp. 417–473.

[72] W. Sun, J. Wang, and Y. Fang. "Consistent selection of tuning parameters via variable selection stability". In: *The Journal of Machine Learning Research* 14.1 (2013), pp. 3419–3440.

[73] S Roberts and G Nowak. "Stabilizing the lasso against cross-validation variability". In: *Computational Statistics & Data Analysis* 70 (2014), pp. 198–211.

[74] E. T. Jaynes. "Information theory and statistical mechanics". In: *Physical review* 106.4 (1957), p. 620.

[75] B. Efron et al. "Least angle regression". In: *The Annals of statistics* 32.2 (2004), pp. 407–499.

[76] T. T. Wu and K. Lange. "Coordinate descent algorithms for lasso penalized regression". In: *The Annals of Applied Statistics* (2008), pp. 224–244.

[77] Q. S. Xu and Y. Z. Liang. "Monte Carlo cross validation". In: *Chemometrics and Intelligent Laboratory Systems* 56.1 (2001), pp. 1–11.

[78] J. Shao. "Linear model selection by cross-validation". In: *Journal of the American statistical Association* 88.422 (1993), pp. 486–494.

[79] Y. Kim, H. Choi, and H.-S. Oh. "Smoothly clipped absolute deviation on high dimensions". In: *Journal of the American Statistical Association* 103.484 (2008), pp. 1665–1673.

[80] J. Huang, J. L. Horowitz, and S. Ma. "Asymptotic properties of bridge estimators in sparse high-dimensional regression models". In: *The Annals of Statistics* (2008), pp. 587–613.

[81] S. A. Van De Geer, P. Bühlmann, et al. "On the conditions used to prove oracle results for the Lasso". In: *Electronic Journal of Statistics* 3 (2009), pp. 1360–1392.

[82] N. Meinshausen and P. Bühlmann. "High-dimensional graphs and variable selection with the lasso". In: *The Annals of Statistics* (2006), pp. 1436–1462.

[83] P. Zhao and B. Yu. "On model selection consistency of Lasso". In: *The Journal of Machine Learning Research* 7 (2006), pp. 2541–2563.

[84] H. Zou. "The adaptive lasso and its oracle properties". In: *Journal of the American statistical association* 101.476 (2006), pp. 1418–1429.

[85] J. Huang, S. Ma, and C. H. Zhang. "Adaptive Lasso for sparse high-dimensional regression models". In: *Statistica Sinica* 18.4 (2008), p. 1603.

[86] J. Cohen. "A coefficient of agreement for nominal scales." In: *Educational and Psychological Measurement* (1960).

[87] F. R. Bach. "Bolasso: model consistent lasso estimation through the bootstrap". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 33–40.

[88] J. H. Kim. "Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap". In: *Computational Statistics & Data Analysis* 53.11 (2009), pp. 3735–3745.

[89] P. Prakash et al. "Optimal wafer site selection using forward selection component analysis". In: *Advanced Semiconductor Manufacturing Conference (ASMC), 2012 23rd Annual SEMI*. IEEE. 2012, pp. 91–96.

[90] B. Flynn and S. McLoone. "Max Separation Clustering for Feature Extraction From Optical Emission Spectroscopy Data". In: *Semiconductor Manufacturing, IEEE Transactions on* 24.4 (2011), pp. 480–488.

[91] Y. Fang, J. Wang, and W. Sun. "A note on selection stability: combining stability and prediction". In: *arXiv preprint arXiv:1301.7118* (2013).

[92] A. Miller. *Subset selection in regression*. CRC Press, 2012.

[93] R. Díaz-Uriarte and S. A. De Andres. "Gene selection and classification of microarray data using random forest". In: *BMC bioinformatics* 7.1 (2006), p. 3.

[94] X. Geng, D. C. Zhan, and Z. H. Zhou. "Supervised nonlinear dimensionality reduction for visualization and classification". In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 35.6 (2005), pp. 1098–1107.

[95] P. Mitra, C. Murthy, and S. K. Pal. "Unsupervised feature selection using feature similarity". In: *IEEE transactions on pattern analysis and machine intelligence* 24.3 (2002), pp. 301–312.

[96] J. G. Dy and C. E. Brodley. "Feature selection for unsupervised learning". In: *The Journal of Machine Learning Research* 5 (2004), pp. 845–889.

[97] Z. Zhao and H. Liu. "Spectral feature selection for supervised and unsupervised learning". In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 1151–1157.

[98] Y. Saeys, I. Inza, and P. Larrañaga. "A review of feature selection techniques in bioinformatics". In: *bioinformatics* 23.19 (2007), pp. 2507–2517.

[99] S. T. Roweis and L. K. Saul. "Nonlinear dimensionality reduction by locally linear embedding". In: *Science* 290.5500 (2000), pp. 2323–2326.

[100] M. Belkin and P. Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural computation* 15.6 (2003), pp. 1373–1396.

[101] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[102] L. Mariey et al. "Discrimination, classification, identification of microorganisms using FTIR spectroscopy and chemometrics". In: *Vibrational Spectroscopy* 26.2 (2001), pp. 151–159.

[103] J. Trygg, E. Holmes, and T. Lundstedt. "Chemometrics in metabonomics". In: *Journal of proteome research* 6.2 (2007), pp. 469–479.

[104] G. P. McCabe. "Principal variables". In: *Technometrics* 26.2 (1984), pp. 137–144.

[105] J. Cadima and I. T. Jolliffe. "Loading and correlations in the interpretation of principle compenents". In: *Journal of Applied Statistics* 22.2 (1995), pp. 203–214.

[106] I. T. Jolliffe, N. T. Trendafilov, and M. Uddin. "A modified principal component technique based on the LASSO". In: *Journal of Computational and Graphical Statistics* 12.3 (2003), pp. 531–547.

[107] A. d'Aspremont et al. "A direct formulation for sparse PCA using semidefinite programming". In: *SIAM review* 49.3 (2007), pp. 434–448.

[108] H. Zou, T. Hastie, and R. Tibshirani. "Sparse principal component analysis". In: *Journal of computational and graphical statistics* 15.2 (2006), pp. 265–286.

[109] R. Jenatton, G. Obozinski, and F. Bach. "Structured sparse principal component analysis". In: *arXiv preprint arXiv:0909.1440* (2009).

[110] H. Shen and J. Z. Huang. "Sparse principal component analysis via regularized low rank matrix approximation". In: *Journal of multivariate analysis* 99.6 (2008), pp. 1015–1034.

[111] D. M. Witten, R. Tibshirani, and T. Hastie. "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis". In: *Biostatistics* (2009), kxp008.

[112] E. Ragnoli et al. "Identifying key process characteristics and predicting etch rate from high-dimension datasets". In: *2009 IEEE/SEMI Advanced Semiconductor Manufacturing Conference*. IEEE. 2009, pp. 106–111.

[113] I. T. Jolliffe. "Discarding variables in a principal component analysis. I: Artificial data". In: *Applied statistics* (1972), pp. 160–173.

[114] W. Krzanowski. "Selection of variables to preserve multivariate data structure, using principal components". In: *Applied Statistics* (1987), pp. 22–33.

[115] K. Mao. "Identifying critical variables of principal components for unsupervised feature selection". In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 35.2 (2005), pp. 339–344.

[116] Y. Lu et al. "Feature selection using principal feature analysis". In: *Proceedings of the 15th international conference on Multimedia*. ACM. 2007, pp. 301–304.

[117] M. Masaeli et al. "Convex Principal Feature Selection." In: *SDM*. SIAM. 2010, pp. 619–628.

[118] D. C. Whitley, M. G. Ford, and D. J. Livingstone. "Unsupervised forward selection: a method for eliminating redundant variables". In: *Journal of chemical information and computer sciences* 40.5 (2000), pp. 1160–1168.

[119] H.-L. Wei and S. A. Billings. "Feature subset selection and ranking for data dimensionality reduction". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29.1 (2007), pp. 162–166.

[120] R. Liu, R. Rallo, and Y. Cohen. "Unsupervised feature selection using incremental least squares". In: *International Journal of Information Technology and Decision Making* 10.06 (2011), pp. 967–987.

[121] Z. Zhao et al. "Massively parallel feature selection: an approach based on variance preservation". In: *Machine Learning* 92.1 (2013), pp. 195–220. ISSN: 0885–6125.

[122] K. Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572.

[123] H. Hotelling. "Analysis of a complex of statistical variables into principal components." In: *Journal of educational psychology* 24.6 (1933), p. 417.

[124] G. H. Golub and C. Reinsch. "Singular value decomposition and least squares solutions". In: *Numerische mathematik* 14.5 (1970), pp. 403–420.

[125] H. O. A. Wold. *Nonlinear estimation by iterative least square procedures*. 1968.

[126] J. J. Gerbrands. "On the relationships between SVD, KLT and PCA". In: *Pattern recognition* 14.1 (1981), pp. 375–381.

[127] N. Kettaneh, A. Berglund, and S. Wold. "PCA and PLS with very large data sets". In: *Computational Statistics & Data Analysis* 48.1 (2005), pp. 69–85.

[128] I. W. Evett and J. S. Ernest. "Rule Induction in Forensic Science. Central Research Establishment. Home Office Forensic Science Service. Aldermaston". In: *Reading, Berkshire RG7 4PN* (1987).

[129] J. M. Sutter and J. H. Kalivas. "Comparison of forward selection, backward elimination, and generalized simulated annealing for variable selection". In: *Microchemical journal* 47.1 (1993), pp. 60–66.

[130] F. G. Blanchet, P. Legendre, and D. Borcard. "Forward selection of explanatory variables". In: *Ecology* 89.9 (2008), pp. 2623–2632.

[131] P. C. Austin and J. V. Tu. "Automated variable selection methods for logistic regression produced unstable models for predicting acute myocardial infarction mortality". In: *Journal of clinical epidemiology* 57.11 (2004), pp. 1138–1146.

[132] V. Cevher and A. Krause. "Greedy dictionary selection for sparse representation". In: *Selected Topics in Signal Processing, IEEE Journal of* 5.5 (2011), pp. 979–988.

[133] M. Jaggi. "Revisiting Frank-Wolfe: Projection-free sparse convex optimization". In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013, pp. 427–435.

[134] Y. Cui and J. G. Dy. *Orthogonal principal feature selection*. Tech. rep. Northeastern University, 2008.

[135] T. Zhang. "Adaptive forward-backward greedy algorithm for learning sparse representations". In: *Information Theory, IEEE Transactions on* 57.7 (2011), pp. 4689–4708.

[136] K. Li, J. X. Peng, and E. W. Bai. "Two-stage mixed discrete–continuous identification of radial basis function (RBF) neural models for nonlinear systems". In: *Circuits and Systems I: Regular Papers, IEEE Transactions on* 56.3 (2009), pp. 630–643.

[137] G. H. Golub and C. F. Van Loan. *Matrix computations*. Vol. 3. JHU Press, 2012.

[138] M. S. Bartlett. "An Inverse Matrix Adjustment Arising in Discriminant Analysis". In: *Ann. Math. Statist* 22.1 (Mar. 1951), pp. 107–111.

[139] P. Jain, A. Tewari, and I. S. Dhillon. "Orthogonal matching pursuit with replacement". In: *Advances in Neural Information Processing Systems*. 2011, pp. 1215–1223.

[140] N. Rao, P. Shah, and S. Wright. "Forward–Backward Greedy Algorithms for Atomic Norm Regularization". In: *Signal Processing, IEEE Transactions on* 63.21 (2015), pp. 5798–5811.

[141] J. Jeffers. "Two case studies in the application of principal component analysis". In: *Applied Statistics* (1967), pp. 225–236.

[142] D. Zeng and C. J. Spanos. "Virtual metrology modeling for plasma etch operations". In: *Semiconductor Manufacturing, IEEE Transactions on* 22.4 (2009), pp. 419–431.

[143] M. B. Kursa, W. R. Rudnicki, et al. "Feature selection with the Boruta package". In: *Journal of Statistical Software* 36.11 (2010).

[144] Y. LeCun et al. "Optimal brain damage." In: *NIPs*. Vol. 89. 1989.

[145] G. E. Hinton and R. R. Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786 (2006), pp. 504–507.

[146] B. Schölkopf, A. Smola, and K. R. Müller. "Kernel principal component analysis". In: *Artificial Neural Networks-ICANN'97*. Springer, 1997, pp. 583–588.

[147] F. Questier et al. "The use of CART and multivariate regression trees for supervised and unsupervised feature selection". In: *Chemometrics and Intelligent Laboratory Systems* 76.1 (2005), pp. 45–54.

[148] Y. Kim, W. N. Street, and F. Menczer. "Feature selection in unsupervised learning via evolutionary search". In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2000, pp. 365–369.

[149] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[150] G. B. Huang, Q. Y. Zhu, and C. K. Siew. "Extreme learning machine: theory and applications". In: *Neurocomputing* 70.1 (2006), pp. 489–501.

[151] Y. Chen, A. Abraham, and B. Yang. "Feature selection and classification using flexible neural tree". In: *Neurocomputing* 70.1 (2006), pp. 305–313.

[152] D. W. Ruck, S. K. Rogers, and M. Kabrisky. "Feature selection using a multilayer perceptron". In: *Journal of Neural Network Computing* 2.2 (1990), pp. 40–48.

[153] R. Setiono and H. Liu. "Neural-network feature selector". In: *Neural Networks, IEEE Transactions on* 8.3 (1997), pp. 654–662.

[154] M. M. Kabir, M. M. Islam, and K. Murase. "A new wrapper feature selection approach using neural network". In: *Neurocomputing* 73.16 (2010), pp. 3273–3283.

[155] R. K. Sivagaminathan and S. Ramakrishnan. "A hybrid approach for feature subset selection using neural networks and ant colony optimization". In: *Expert systems with applications* 33.1 (2007), pp. 49–60.

[156] R. Collobert and J. Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 160–167.

[157] G. B. Orr and K.-R. Müller. *Neural networks: tricks of the trade*. Springer, 2003.

[158] G. B. Huang, D. H. Wang, and Y. Lan. "Extreme learning machines: a survey". In: *International Journal of Machine Learning and Cybernetics* 2.2 (2011), pp. 107–122.

[159] R. Hecht-Nielsen. "Theory of the backpropagation neural network". In: *Neural Networks, 1989. IJCNN., International Joint Conference on*. IEEE. 1989, pp. 593–605.

[160] M. T. Hagan and M. B. Menhaj. "Training feedforward networks with the Marquardt algorithm". In: *Neural Networks, IEEE Transactions on* 5.6 (1994), pp. 989–993.

[161] S. McLoone et al. "A hybrid linear/nonlinear training algorithm for feedforward neural networks". In: *Neural Networks, IEEE Transactions on* 9.4 (1998), pp. 669–684.

[162] S. Chen, C. F. Cowan, and P. M. Grant. "Orthogonal least squares learning algorithm for radial basis function networks". In: *Neural Networks, IEEE Transactions on* 2.2 (1991), pp. 302–309.

[163] C. R. Rao and S. K. Mitra. *Generalized inverse of matrices and its applications*. Vol. 7. Wiley New York, 1971.

[164] Z. Bai et al. "Sparse extreme learning machine for classification". In: *Cybernetics, IEEE Transactions on* 44.10 (2014), pp. 1858–1870.

[165] G. B. Huang. "An insight into extreme learning machines: random neurons, random features and kernels". In: *Cognitive Computation* 6.3 (2014), pp. 376–390.

[166] G. B. Huang et al. "Extreme learning machine for regression and multiclass classification". In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 42.2 (2012), pp. 513–529.

[167] G. B. Huang et al. "Local receptive fields based extreme learning machine". In: *Computational Intelligence Magazine, IEEE* 10.2 (2015), pp. 18–29.

[168] G. B. Huang and L. Chen. "Convex incremental extreme learning machine". In: *Neurocomputing* 70.16 (2007), pp. 3056–3062.

[169] G. B. Huang and L. Chen. "Enhanced random search based incremental extreme learning machine". In: *Neurocomputing* 71.16 (2008), pp. 3460–3468.

[170] S. Haykin and N. Network. "A comprehensive foundation". In: *Neural Networks* 2.2004 (2004).

[171] W. Deng, Q. Zheng, and L. Chen. "Regularized extreme learning machine". In: *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*. IEEE. 2009, pp. 389–395.

[172] P. L. Bartlett. "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network". In: *Information Theory, IEEE Transactions on* 44.2 (1998), pp. 525–536.

[173] J. Park and I. W. Sandberg. "Universal approximation using radial-basis-function networks". In: *Neural computation* 3.2 (1991), pp. 246–257.

[174] J. R. Edwards and M. E. Parry. "On the use of polynomial regression equations as an alternative to difference scores in organizational research". In: *Academy of Management Journal* 36.6 (1993), pp. 1577–1613.

[175] A. E. Hoerl. "Ridge analysis". In: *Chemical engineering progress symposium series*. Vol. 60. 1964, pp. 67–78.

[176] G. Swirszcz and A. C. Lozano. "Multi-level lasso for sparse multi-task regression". In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. 2012, pp. 361–368.

[177] A Goel, S. C. Saxena, and S. Bhanot. "Modified functional link artificial neural network". In: *International Journal of Electrical and Computer Engineering* 1.1 (2006), pp. 22–30.

[178] G. A. Susto et al. "Supervised aggregative feature extraction for big data time series regression". In: *IEEE Transactions on Industrial Informatics* 12.3 (2016), pp. 1243–1252.

[179] N. Kwak. "Nonlinear projection trick in kernel methods: An alternative to the kernel trick". In: *Neural Networks and Learning Systems, IEEE Transactions on* 24.12 (2013), pp. 2113–2119.

[180] Y. Bengio. "Learning deep architectures for AI". In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[181] J. Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (2015), pp. 85–117.

[182] X. Glorot, A. Bordes, and Y. Bengio. "Deep sparse rectifier neural networks". In: *International Conference on Artificial Intelligence and Statistics*. 2011, pp. 315–323.

[183] Y. Bengio et al. "Greedy layer-wise training of deep networks". In: *Advances in neural information processing systems* 19 (2007), p. 153.

[184] D. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[185] J. Bergstra and Y. Bengio. "Random search for hyper-parameter optimization". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 281–305.

[186] D. Erhan et al. "The difficulty of training deep architectures and the effect of unsupervised pre-training". In: *International Conference on artificial intelligence and statistics*. 2009, pp. 153–160.

[187] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: `http://archive.ics.uci.edu/ml`.

[188] Q. P. He and J. Wang. "Fault Detection Using the k-Nearest Neighbor Rule for Semiconductor Manufacturing Processes". In: *IEEE Transactions on Semiconductor Manufacturing* 20.4 (Nov. 2007), pp. 345–354. ISSN: 0894-6507. DOI: 10.1109/TSM.2007.907607.

[189] L. Ren and W. Lv. "Fault Detection via Sparse Representation for Semiconductor Manufacturing Processes". In: *IEEE Transactions on Semiconductor Manufacturing* 27.2 (May 2014), pp. 252–259. ISSN: 0894-6507. DOI: 10.1109/TSM.2014.2302011.

[190] G. Verdier and A. Ferreira. "Adaptive Mahalanobis Distance and k-Nearest Neighbor Rule for Fault Detection in Semiconductor Manufacturing". In: *IEEE Transactions on Semiconductor Manufacturing* 24.1 (Feb. 2011), pp. 59–68. ISSN: 0894-6507. DOI: 10.1109/TSM.2010.2065531.

[191] R. Jin, C. J. Chang, and J. Shi. "Sequential measurement strategy for wafer geometric profile estimation". In: *IIE Transactions* 44.1 (2012), pp. 1–12.

[192] H. Zhao et al. "PDE-constrained Gaussian process model on material removal rate of wire saw slicing process". In: *Journal of Manufacturing Science and Engineering* 133.2 (2011), p. 021012.

[193] L. Zhang, K. Wang, and N. Chen. "Monitoring wafers' geometric quality using an additive Gaussian process model". In: *IIE Transactions* (2015).

[194] S. Mahadevan and S. L. Shah. "Fault detection and diagnosis in process data using one-class support vector machines". In: *Journal of Process Control* 19.10 (Dec. 2009), pp. 1627–1639. ISSN: 09591524. DOI: 10.1016/j.jprocont.2009.07.011.

[195] H. P. Kriegel, A. Zimek, et al. "Angle-based outlier detection in high-dimensional data". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2008, pp. 444–452.

[196] G. P. Hughes. "On the mean accuracy of statistical pattern recognizers". In: *Information Theory, IEEE Transactions on* 14.1 (1968), pp. 55–63.

[197] R. Marimont and M. Shapiro. "Nearest neighbour searches and the curse of dimensionality". In: *IMA Journal of Applied Mathematics* 24.1 (1979), pp. 59–70.

[198] M. Köppen. "The curse of dimensionality". In: *5th Online World Conference on Soft Computing in Industrial Applications (WSC5)*. 2000, pp. 4–8.

[199] A. Zimek, E. Schubert, and H. P. Kriegel. "A survey on unsupervised outlier detection in high-dimensional numerical data". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 5.5 (2012), pp. 363–387.

[200] K. Beyer et al. "When is "nearest neighbor" meaningful?" In: *Database theory-ICDT'99*. Springer, 1999, pp. 217–235.

[201] T. Shi and S. Horvath. "Unsupervised learning with random forest predictors". In: *Journal of Computational and Graphical Statistics* 15.1 (2006).

[202] P. Geurts, D. Ernst, and L. Wehenkel. "Extremely randomized trees". In: *Machine learning* 63.1 (2006), pp. 3–42.

[203] F. T. Liu, K. M. Ting, and Z. H. Zhou. "Isolation forest". In: *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE. 2008, pp. 413–422.

[204] J. Yang, C. McArdle, and S. Daniels. "Similarity Ratio Analysis for Early Stage Fault Detection with Optical Emission Spectrometer in Plasma Etching Process". In: *PloS one* 9.4 (2014), e95679.

[205] B. Schölkopf et al. "Estimating the support of a high-dimensional distribution". In: *Neural computation* 13.7 (2001), pp. 1443–1471.

[206] D. M. Hawkins. *Identification of outliers*. Vol. 11. Springer, 1980.

[207] E. M. Knox and R. T. Ng. "Algorithms for mining distancebased outliers in large datasets". In: *Proceedings of the International Conference on Very Large Data Bases*. Citeseer. 1998, pp. 392–403.

[208] M. M. Breunig et al. "LOF: identifying density-based local outliers". In: *ACM sigmod record*. Vol. 29. ACM. 2000, pp. 93–104.

[209] X. Song et al. "Conditional anomaly detection". In: *Knowledge and Data Engineering, IEEE Transactions on* 19.5 (2007), pp. 631–645.

[210] V. Chandola, A. Banerjee, and V. Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

[211] C. A. Lowry and D. C. Montgomery. "A review of multivariate control charts". In: *IIE transactions* 27.6 (1995), pp. 800–810.

[212] M. Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: 96.34 (1996), pp. 226–231.

[213] S. Papadimitriou et al. "Loci: Fast outlier detection using the local correlation integral". In: *Data Engineering, 2003. Proceedings. 19th International Conference on*. IEEE. 2003, pp. 315–326.

[214] V. Barnett and T. Lewis. *Outliers in statistical data*. Vol. 3. Wiley New York, 1994.

[215] S. J. Schwager and B. H. Margolin. "Detection of multivariate normal outliers". In: *The annals of statistics* (1982), pp. 943–954.

[216] P. C. Mahalanobis. "On the generalized distance in statistics". In: *Proceedings of the National Institute of Sciences (Calcutta)* 2 (1936), pp. 49–55.

[217] V. Barnett. "The ordering of multivariate data". In: *Journal of the Royal Statistical Society. Series A (General)* (1976), pp. 318–355.

[218] I. Ruts and P. J. Rousseeuw. "Computing depth contours of bivariate point clouds". In: *Computational Statistics & Data Analysis* 23.1 (1996), pp. 153–168.

[219] F. P. Preparata, M. I. Shamos, and F. P. Preparata. *Computational geometry: an introduction*. Vol. 5. Springer-Verlag New York, 1985.

[220] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: an efficient data clustering method for very large databases". In: *ACM SIGMOD Record*. Vol. 25. ACM. 1996, pp. 103–114.

[221] A. Faraz and A. Parsian. "Hotelling's T2 control chart with double warning lines". In: *Statistical Papers* 47.4 (2006), pp. 569–593.

[222] H. Hoffmann. "Kernel PCA for novelty detection". In: *Pattern Recognition* 40.3 (2007), pp. 863–874.

[223] C. Domeniconi et al. "Locally adaptive metrics for clustering high dimensional data". In: *Data Mining and Knowledge Discovery* 14.1 (2007), pp. 63–97.

[224] T. Kohonen. "The self-organizing map". In: *Neurocomputing* 21.1 (1998), pp. 1–6.

[225] J. Vesanto and E. Alhoniemi. "Clustering of the self-organizing map". In: *Neural Networks, IEEE Transactions on* 11.3 (2000), pp. 586–600.

[226] G. Verdier and A. Ferreira. "Adaptive Mahalanobis Distance and Nearest Neighbor Rule for Fault Detection in Semiconductor Manufacturing". In: *Semiconductor Manufacturing, IEEE Transactions on* 24.1 (2011), pp. 59–68.

[227] R. Y. Liu, J. M. Parelius, K. Singh, et al. "Multivariate analysis by data depth: descriptive statistics, graphics and inference,(with discussion and a rejoinder by Liu and Singh)". In: *The annals of statistics* 27.3 (1999), pp. 783–858.

[228] J. Ma and S. Perkins. "Time-series novelty detection using one-class support vector machines". In: *Neural Networks, 2003. Proceedings of the International Joint Conference on*. Vol. 3. IEEE. 2003, pp. 1741–1745.

[229] L. M. Manevitz and M. Yousef. "One-class SVMs for document classification". In: *the Journal of machine Learning research* 2 (2002), pp. 139–154.

[230] T. Zhang et al. "Fall detection by wearable sensor and one-class SVM algorithm". In: *Intelligent Computing in Signal Processing and Pattern Recognition*. Springer, 2006, pp. 858–863.

[231] A. Beghi et al. "A one-class svm based tool for machine learning novelty detection in hvac chiller systems". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 1953–1958.

[232] S. Mahadevan and S. L. Shah. "Fault detection and diagnosis in process data using one-class support vector machines". In: *Journal of Process Control* 19.10 (2009), pp. 1627–1639.

[233] J. A. Hanley and B. J. McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve." In: *Radiology* 143.1 (1982), pp. 29–36.

[234] S. Aryal et al. "Improving iForest with Relative Mass". In: *Advances in Knowledge Discovery and Data Mining*. Springer, 2014, pp. 510–521.

[235] K. M. Ting et al. "Mass estimation". In: *Machine learning* 90.1 (2013), pp. 127–160.

[236] F. T. Liu, K. M. Ting, and Z. H. Zhou. "On detecting clustered anomalies using SCiForest". In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 274–290.

[237] L. Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[238] S. R. Safavian and D. Landgrebe. "A survey of decision tree classifier methodology". In: *IEEE transactions on systems, man, and cybernetics* 21.3 (1991), pp. 660–674.

[239] S. K. Murthy. "Automatic construction of decision trees from data: A multi-disciplinary survey". In: *Data mining and knowledge discovery* 2.4 (1998), pp. 345–389.

[240] R. J. Lewis. "An introduction to classification and regression tree (CART) analysis". In: *Annual Meeting of the Society for Academic Emergency Medicine in San Francisco, California*. 2000, pp. 1–14.

[241] T. Hothorn, K. Hornik, and A. Zeileis. "Unbiased recursive partitioning: A conditional inference framework". In: *Journal of Computational and Graphical statistics* 15.3 (2006), pp. 651–674.

[242] T. Bylander. "Estimating Generalization Error on Two-Class Datasets Using Out-of-Bag Estimates". en. In: *Machine Learning* 48.1-3 (July 2002), pp. 287–297. ISSN: 1573-0565. DOI: 10.1023/A:1013964023376.

[243] J. Zhang and M. Zulkernine. "Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection". In: *2006 IEEE International Conference on Communications*. Vol. 5. IEEE, 2006, pp. 2388–2393. DOI: 10.1109/ICC.2006.255127.

[244] C. Strobl et al. "Bias in random forest variable importance measures: Illustrations, sources and a solution". In: *BMC bioinformatics* 8.1 (2007), p. 25.

[245] V. J. Hodge and J. Austin. "A survey of outlier detection methodologies". In: *Artificial Intelligence Review* 22.2 (2004), pp. 85–126.

[246] D. W. Aha and R. L. Bankert. "Feature selection for case-based classification of cloud types: An empirical comparison". In: *AAAI-94 Workshop on Case-Based Reasoning*. Seattle, WA. 1994, pp. 106–112.

[247] D. B. Skalak. "Prototype and feature selection by sampling and random mutation hill climbing algorithms". In: *Proceedings of the eleventh international conference on machine learning*. 1994, pp. 293–301.

[248] A. Malhi and R. X. Gao. "PCA-based feature selection scheme for machine defect classification". In: *Instrumentation and Measurement, IEEE Transactions on* 53.6 (2004), pp. 1517–1525.

[249] G. K. Kuchimanchi et al. "Dimension reduction using feature extraction methods for Real-time misuse detection systems". In: *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*. IEEE. 2004, pp. 195–202.

[250] E. K. Lada, J. C. Lu, and J. R. Wilson. "A wavelet-based procedure for process fault detection". In: *Semiconductor Manufacturing, IEEE Transactions on* 15.1 (2002), pp. 79–90.

[251] J. Yang, C. McArdle, and S. Daniels. "Similarity ratio analysis for early stage fault detection with optical emission spectrometer in plasma etching process." In: *PloS one* 9.4 (Jan. 2014). Ed. by M. Yousfi, e95679. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0095679.