UNIVERSITY OF BERGAMO

School of Doctoral Studies

Doctoral Degree in Engineering and Applied Sciences

XXX cycle

SSD: ING/INF-04

Learning meets control

Data analytics for dynamical systems

Advisor: Prof. Fabio Previdi

Co-advisor: Prof. Simone Formentin

Doctoral Thesis

Mirko MAZZOLENI

Student ID: 1007371

Academic Year 2016/2017

Acknowledgements

I'd like to thanks

all the people I came across during this journey. In one way or another, all of them contributed to this work sharing their experiences, talks, thoughts and existence. A special thanks to the collegues of the research group (in no particular order):

- Alberto Cologni, who taught me what it means to be an engineer
- Michele Ermidoro, whose spirit and creativy reflects an uncommon mind
- Fabio Angeloni, whose analysis skills to decomponse complex problem remains unchallenged
- Paolo Sangregorio, who is the archetype of "getting things done"
- Yamuna Maccarana, who beared along with me many burdens
- Stefano Moretti, who reminded me that men live for passions
- Claudio Ghisleni, to whom I envy practical sense and positive attitude
- Dario Nava, a mate whom traits I will not find anywhere else
- Frank Owen, who showed me what a man can do
- Gabriele Maroni, who I trust blindly
- Matteo Scandella, whose rigor and exceptional intuitions will guide him towards the peaks of the world

A heartfelt thanks to Simone Formentin, whose presence is like the green light on the other side of the bay. My sincere gratitude to my advisor, professor Fabio Previdi, whose lead made me a better scientist and whose advices made me a wiser person.

Finally, I'd like to thanks my girlfriend Arianna and my family for having always sustained me.

In God we trust, all others bring data.
—William Edwards Deming (1900-1993)

Abstract

System identification has always been one of the main research focuses of the control community, since the early steps of the automatic control field. The development of a dynamical system's models from experimental data is instrumental for understanding the plant under study and designing its model-based control scheme. In the last decade, a cross-fertilization began between the System Identification and the Statistical Learning communities. This led firstly to the introduction of regularization techniques in system identification, and, more recently, to the application of kernel methods to dynamical system learning. This thesis further investigates the roles that learning methods can have in the control science. In the first part, we lay the theoretical foundations of a new kernel-based regularization method for Nonlinear Finite Impulse Response (NFIR) system identification. The method, called Semi-Supervised Identification (SSI), relies on the *manifold* spanned by the system's inputs. This manifold is built by using not only the measured input/output data, but also inputs data for which there is no corresponding outputs. The effect of this rationale is to impose prior information on the system structure, in the form of local smoothness assumptions. This differs from standard Tikhonov regularization, which imposes a global smoothness behaviour on the learned function. The second part of this work presents practical applications of how statistical learning methods can be used to face control and estimation problems. The case studies span a variety of different applications, from fault detection of electro-mechanical actuators, to clustering methodologies and pure forecasting challenges.

Keywords. System Identification, Statistical Learning, Regularization, Manifold Learning, Fault Detection, Clustering, Forecasting

Contents

Part I

System identification review and new research

CHAPTER 1

Introduction

All types of sciences and theories have been developed with the intent to understand the world around us. The world speaks the language of the data, and if we want to comprehend it, we have to be able to translate these information into something which is interpretable by a human. The main tool we have at our disposal when pursuing this aim is a mathematical *model*. A model is a more or less simplified version of an aspect of reality, that is convenient for some application. Equipped with the right mathematics, we are able to describe in a compact way a huge number of different systems, taking advantage of all the available knowledge. A *system* is an abstract mechanism that transforms *inputs* (causes) into *outputs* (effects). *Statistical learning* is the science of building models of systems from data. In the case of *static systems*, that is, when there is not a temporal relationships between inputs and outputs, the set of learning techniques and concepts is known as *Machine Learning*. The great majority of physical phenomena are however *dynamic systems*, i.e., mechanisms for which the sole knowledge of the input is not sufficient to unequivocally determine the relative output. The missing parts are the system's initial conditions, known as the *system's states*, that embody the concept of *memory*. Control theory relies on dynamic model concepts to impose a particular behaviour to a physical system. In these setting, the inputs and the outputs evolve in time according to differential (or difference) equations. Learning procedures applied to dynamic systems constitute the *System Identification* field. In this setting, the modeling is done from experiments. The system is excited with

some inputs and the outputs are observed; then, these data are used to identify the process that links inputs and outputs. This approach is known as black-box modeling. Learning procedure are not the only way to determine a model of physical systems. A genuine procedure is to decompose the modeling problem into smaller components which already have a model. An example of this is the modeling of electronic circuits. The circuit is decomposed into its basic building blocks, resistors, transistors, capacitors, among others. Each one of these basic building blocks has a well established model. Using Kirchhoff's laws, it is possible to assemble the building blocks to get a model that describes the circuit as a whole. This method is known as white-box modeling. In other applications, the two approaches are often used together (grey-box modeling): the physical approach is used to define a model from first principles, and the identification approach is used to fit its parameters, so that the model agrees with what it is observed [1].

The most widespread approach to the identification of dynamic systems relies on the time domain parametric Prediction Error Methods (PEMs) [2, 3]. The rationale of this approach is to minimize the squared difference between each observed datum and its prediction made by the chosen model. The choice of the parametric model is a twofold problem: in first instance, the engineer has to define a model's structure, then, as a subsequent step, the model's order. This identification step is known as *model selection*. When the model structure is fixed, PEM procedures are shown to converge to the best approximation of the true system contained in the chosen family of models, for large data sets. Model selection is an important and practical topic: it is directly related to the bias-variance tradeoff, and can be handled by various model validation techniques [4]. This is often carried out by resorting to complexity measures, such as the Akaike's criterion (AIC) [5] or cross validation (CV) [4]. Some inefficiencies related to these classical approaches have been recently pointed out [6, 7, 8]. In particular, it has been shown that sample properties of PEM approaches, equipped e.g. with AIC or CV, may be unsatisfactory when tested on experimental data. This makes the PEM estimator a Post Model Selection Estimator (PMSE) [9].

In both static and dynamic cases, learning an (unknown) function from a finite set of data is always an ill-posed problem. Infact, the true function can behave in any way outside the points that we observed. In machine learning, this has generated the theoretical framework of PAC (Probably Approximately Correct) learning [10]. Starting from the seminal works of Tikhonov and

Phillips [11, 12], a number of regularization methods have been proposed in the literature to better condition the static learning problem [13, 14]. Therefore, it is not suprising how regularization has found a flourishing application also in the system identification community. A recent approach that grounds on the concept of regularization was presented in [8]. Here, the authors have adapted to dynamic systems the functional analysis' methods previously employed by the machine learning community to static systems [15, 16]. The new paradigm formulates the identification problem as function estimation possibly in an infinite-dimensional space. In the context of linear system identification, the elements of such space are all possible impulse responses. The adopted method considers a regularization component that admits a Bayesian interpretation [17]. This is in line with standard results in the Ridge and Lasso penalties for machine learning models [4]. In particular, the impulse response is modeled as a zero-mean Gaussian process. In this way, prior information (acting as the regularization term) is introduced in the identification process by assigning a covariance, known as the *kernel* in the machine learning literature [18]. The choice of the kernel is of paramount importance to define the properties of the inferred function. In this context, a recent major advance for linear system identification has been the introduction of new kernel's types, which include information on impulse response exponential stability [8, 6]. This requirement was not investigated in the context of static systems, for obvious reasons. Kernel functions depend on hyperparameters which can be estimated from data using marginal likelihood maximization (also known as Empirical Bayes) approaches [19]. This is a procedure that is conceptually similar to model selection, but it turns out to be much more robust. A comparison of empirical Bayes versus a full Bayes method for hyperparameter tuning is presented in [20]. For a comprehensive survey on kernel methods for linear system identification, see [21], while the adaptation of the aforementioned framework for non-linear systems is faced in [22]. The cited time-domain PEM approach is not the only method to perform system identification. Subspace methods [23, 24] offers the possibility to perform direct identification of models represented in state-space form. Frequency-domain solutions [25] are an alternative tool which is rapidly evolving. In particular, kernel methods have recently caught the attention also of the frequency domain system identification community [26]. This motivates further research on these non-parametric approaches.

The focus of this thesis will be on the black-box system identification approach: given a set of data, the aim is to find a function that best explains

their variability and nature. In the first part, a new regularization approach for non-linear system identification, based on kernel methods, is introduced. The process of learning an unknown function is then applied to various applications of control problems in the second part of this work.

Outline

Chapter 2 gives an overview of the parametric system identification area, describing the classical dynamic system models, the bias-variance tradeoff, model selection and the regularization technique, highlighting the similarities between system identification and statistical learning.

Chapter 3 shows the main tools, borrowed from the functional analysis, used in the context of the nonparametric system identification. The notion of kernel and Reproducing Kernel Hilbert Spaces (RKHS) will be introduced and the state of the art of nonparametric methods in system identification is outlined briefly.

Chapter 4 describes the *first main contribution* of this work, namely a new approach to system identification based on semi-supervided learning. The concept of manifold learning is introduced and the method outlined.

Chapter 5 and following are dedicated to the *second contribution* of this thesis, that is, the application of learning methods to solve practical control and estimation problems. In this chapter, the description of a fault detection project for Electro-Mechanical Actuators (EMA) and its experimental setup are presented.

Chapter 6 presents a model-based fault detection scheme, for the problem in Chapter 5, based on a particle-filter methodology, showing comparisons with the classical extended Kalman filter approach.

Chapter 7 deals with a data-driven fault detection scheme, for the problem in Chapter 5, based on a machine learning pipeline and feature extractions.

Chapter 8 describes a new unsupervised learning algorithm applied to the fault detection of EMA data.

Chapter 9 highlights a forecasting application in the retail food market, where the concepts of closed-loop control are applied as unifying view of the problem.

CHAPTER 2

Parametric system identification

This chapter briefly reviews the concepts of the parametric system identification methods, focusing on Linear Time-Invariant (LTI) Single Input Single Output (SISO), stable and causal systems. Section 2.1 decribes the main parametric models for dynamic systems and the Prediction Error Method (PEM) for system identification. Model order selection is an important component of parametric system identification procedures. Regularization methods are one of the fundamental tools to face this challenge. Section 2.2 introduces the main regularization concepts for static systems, both from the frequentist and bayesian viewpoint. The application of regularization to dynamic systems is finally reviewed in Section 2.3.

2.1 Models of dynamic systems

A model family $\mathcal{M}$ is a parametrized collection of models that describes the relations between the input and output signals of the system. The inputs and outputs are denoted respectively as $u(t)$ and $y(t)$, with t the temporal index. The model family $\mathcal{M}$ defines both the model structural form (the model class) and the model complexity (the number of its parameters). The model's parameters are denoted by $\theta \in \mathbb{R}^{m\times 1}$, thus $\mathcal{M}(\theta)$ defines a particular model. Therefore, $\mathcal{M}$ acts as an *hypothesis space*, and $\mathcal{M}(\theta) \in \mathcal{M}$ is a particular *hypothesis* which belongs to $\mathcal{M}$.

In order to show the main ideas of parametric dynamic systems' modeling,

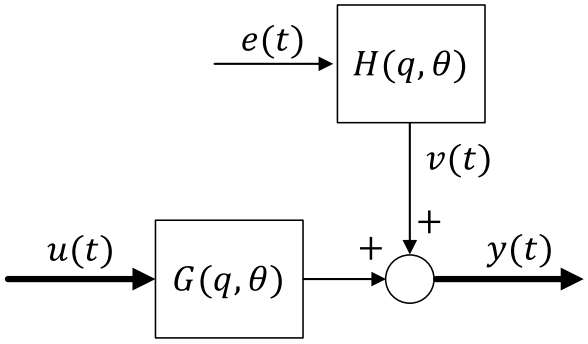


Figure 2.1: General model for linear SISO time-invariant dynamic systems. Bold lines represent measured signals

we restrict our attention to LTI SISO systems. A general modeling of a LTI SISO system is given by the transfer function $G(q,\theta)$ from input to output and the transfer function $H(q,\theta)$ from a zero mean white noise source $e(t)$ to output additive disturbance $v(t)$, see Figure 2.1:

$$\mathcal{M}(\theta):\ y(t) = G(q,\theta)u(t) + H(q,\theta)e(t) \tag{2.1a}$$

$$\mathbb{E}\left[e(t)\right] = 0; \quad \mathbb{E}\left[e(t)^2\right] = \sigma^2; \quad \mathbb{E}\left[e(t)e(k)\right] = 0 \text{ if } k \neq t, \tag{2.1b}$$

In this representation, the transfer function $G(q,\theta)$ represents the deterministic system's component, driven by a known input signal. The model $H(q,\theta)e(t)$ represents both modeling uncertainties and disturbances, complementing the aspects of $y(t)$ that $u(t)$ is not able to explain. The general framework for the identification of the $H(q,\theta)$ lies in interpreting the data sequences as finite-time realizations of stationary stochastic processes. A *stochastic process* $\{v(t,s)\}$(in discrete time) is a sequence of random variables defined from the same random experiment s, indexed via a temporal index t. A stochastic process is said to be *strictly stationary* if its joint probability distribution does not change when shifted in time. In the development of the Kolmogorov-Wiener prediction theory [27], the concept of *weak-sense stationarity* is used, which requires only that the expected value and the autocovariance function do not vary with respect to time and time windows, respectively. The signal $v(t)$ in Figure 2.1 is then interpreted as a stationary stochastic process obtained by feeding a linear dynamic system with a white noise source. This interpretation is made possible by the *spectral factorization theorem* [3], that states the relation between stationary stochastic processes and dynamic systems. The true system

(the data-generating model), denoted as $\mathcal{S}_0$, can be expressed as:

$$\mathcal{S}_0 : \; y(t) = G_0(q)u(t) + H_0(q)\eta(t) \tag{2.2a}$$

$$\mathbb{E}\left[\eta(t)\right] = 0; \quad \mathbb{E}\left[\eta(t)^2\right] = \nu^2; \quad \mathbb{E}\left[\eta(t)\eta(k)\right] = 0 \text{ if } k \neq t, \tag{2.2b}$$

where $G_0(q)$, $H_0(q)$ are the true system's transfer function, and $\eta(t)$ is a white noise. The models (2.1)-(2.2) are in discrete time (assuming sampling interval of one time unit) and q denotes the shift operator such that $qy(t) = y(t+1)$. Since $G(q,\theta)$ and $H(q,\theta)$ are transfer functions, their expansion in the inverse (backward) operator gives the impulse response of, respectively, $g_k(\theta)$ and $h_k(\theta)$:

$$G(q,\theta) = \sum_{k=1}^{\infty} g_k(\theta) q^{-k} \tag{2.3}$$

$$H(q,\theta) = h_0 + \sum_{k=1}^{\infty} h_k(\theta) q^{-k}, \tag{2.4}$$

where $h_0 = 1$ for normalization reasons. The model (2.1) is dynamic: this permits to predict the output at time t, i.e. $y(t)$, based on observations of previous input-output data up to time $t-1$. Under the assumption that $H(q,\theta)$ is inversely stable, the natural one-step ahead predictor for (2.1) is [2]:

$$\hat{y}(t|t-1,\theta) = \frac{H(q,\theta)-1}{H(q,\theta)} y(t) + \frac{G(q,\theta)}{H(q,\theta)} u(t). \tag{2.5}$$

Common black-box models parametrizations are to let G and H be rational in the q operator:

$$G(q,\theta) = \frac{B(q)}{F(q)}; \qquad H(q,\theta) = \frac{C(q)}{D(q)}, \tag{2.6}$$

where $B(q), F(q), C(q)$ and $D(q)$ are polynomials of q^{-1}, and their dependence of θ has been omitted for ease of notation. A common model choice is to impose $F(q) = D(q) = A(q)$ and $C(q) = 1$ which gives the Auto-Regressive with eXogenous input (ARX) model:

$$y(t) = \frac{B(q)}{A(q)} u(t) + \frac{1}{A(q)} e(t), \tag{2.7}$$

where $A(q) = 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a}$, $B = b_1 q^{-1} + \ldots + b_{n_b} q^{-n_b}$, and n_a, n_b are the *orders* of the ARX model. The elements of the parameter vector θ are then the coefficients $a_1, a_2, \ldots, a_{n_a}, b_1, b_2, \ldots, b_{n_b}$, with $n_a + n_b = m$, the dimension of θ. Another useful structure, which will be the focus of the next few chapters,

is the Finite Impulse Response (FIR) model, $F(q) = C(q) = D(q) = 1$:

$$y(t) = B(q)u(t) + e(t). \tag{2.8}$$

Other thoroughly studied models are the Output Error (OE) model, $C(q) = D(q)$, the Auto-Regressive Moving Average with eXogenous input (ARMAX) model, $F(q) = D(q) = A(q)$, and the Box-Jenkins (BJ) model, with all four polynomials different. The chosen model structure needs then to be fitted to measured data. A set of N collected data in time domain, generated from $\mathcal{S}_0$, is indicated as $\mathcal{D} = \{u(1), y(1), \ldots, u(N), y(N)\}$. The PEM method consists in minimizing the mean squared prediction error between actual response and the one predicted by (2.5):

$$J_N(\theta) = \sum_{t=1}^{N} \big(y(t) - \hat{y}(t|t-1, \theta)\big)^2, \tag{2.9}$$

and the corresponding parameters' estimate is:

$$\hat{\theta}_N = \arg\min_{\theta} J_N(\theta). \tag{2.10}$$

The term $J_N(\hat{\theta}) \equiv E_{\text{in}}$ will be referred as *in-sample error* since it measures the performance of the identified model on the training data. The *out-of-sample error*, instead, measures the model's performance on *unseen* data, and can be defined as $E_{\text{out}} = \mathbb{E}\left[\big(y(t) - \hat{y}(t|t-1, \theta)\big)^2\right]$, where the expectation is over the unobserved regressors.

In deriving the PEM method (2.9)-(2.10), no assumptions on the distribution of $e(t)$ has been made: however, it is posssible to show that, if the disturbances are Gaussian, the method coincides with the Maximum Likelihood (ML) approach [28].

This section introduced some common model's structures for linear time invariant dynamic system. As briefly discussed in the Chapter 1, after having chosen a model structure, the next design choice is the model's order. Referring to a fixed model family $\mathcal{M}$ we have that:

- More complex $\mathcal{M}$ leads to better chance of approximating the true system model $\mathcal{S}_0$. Infact, if $\mathcal{M}$ is too simple, we fail to approximate $\mathcal{S}_0$ and we end up with large E_{in}.

- Less complex $\mathcal{M}$ gives better chance of generalizing out of sample. Infact, if $\mathcal{M}$ is too complex, the model will learn also the idiosyncrasies of the particular training dataset, which are not related to the main system's

dynamics. This, in turn, translates to large E_{out}.

It follows that it is not possible to obtain low values for the E_{in} and E_{out} errors at the same time. The aim in any learning problem is to balance the approximation and generalization capability of the chosen model. This fundamental aspect is known as *bias-variance tradeoff*. For a deeper discussion on bias and variance, refer to Appendix A.1.

2.2 Regularization in static systems

The use of PEM methods after having chosen the model order with techniques such such as AIC (FPE) and BIC (see Appendix A.2), can lead to poor sample properties, as noticed in [6]. Furthermore, these procedures perform a "hard" threshold decision on the model's complexity, making the final model's choice not adaptable to all practical situations. *Regularization* permits to effectively contrast overfitting, while at the same time retaining all the ability of complex models to approximate the true unknown function. The following discussion will treat regularization in a generic case of static systems, providing the foundation for the introduction of regularization techniques to dynamic models, treated in Section 2.3. In this section, were static systems are considered, the letter t does not indicate a temporal unit, but merely an indexing variable.

2.2.1 Frequentist interpretation of regularization

Consider a generic static linear model:

$$y(t) = \varphi^T(t)\theta + e(t) \qquad \theta \in \mathbb{R}^{m\times 1}, \tag{2.11}$$

where $y(t)$ (the output) and $\varphi(t)$ (the regression vector) are observed variables, $e(t)$ is a noise disturbance (assumed to be independent of $\varphi(t)$) and θ is the unknown (deterministic) parameters' vector. The least-squares method finds the estimate $\hat{\theta}$ as:

$$\hat{\theta}_{LS} = \arg\min_{\theta} \sum_{t=1}^{N} \left(y(t) - \varphi^T(t)\theta\right)^2, \tag{2.12}$$

which is shown to be unbiased. The same solution is given by the Maximum Likelihood (ML) paradigm, when the errors $e(t)$ are i.i.d. Gaussian variables, $e(t) \sim \mathcal{N}(0, \sigma^2)$.

One of the possible ways to perform regularization consist of penalizing the value of the estimated parameters. Commonly used penalty terms of this type are given by the 2-norm of the parameters' vector, leading to the *Ridge regression*:

$$\hat{\theta}_R = \arg\min_{\theta} \sum_{t=1}^{N} \left(y(t) - \varphi^T(t)\theta\right)^2 + \lambda \sum_{j=1}^{m} \theta_j^2, \tag{2.13}$$

and the parameter vector's 1-norm, giving the *Lasso regression* formula:

$$\hat{\theta}_L = \arg\min_{\theta} \sum_{t=1}^{N} \left(y(t) - \varphi^T(t)\theta\right)^2 + \lambda \sum_{j=1}^{m} |\theta_j|, \tag{2.14}$$

where λ is a *hyperparameter* that controls the regularization strength. There is therefore a tradeoff between minimizing the model fit to the data and minimizing the value of the parameters. If a parameter is driven to zero by the minimization process, we end up with a model that has less parameters. Otherwise, a simpler model is still effectively produced, although the contribution of each regressor is taken into consideration (even if in a small portion). With these regularization schemes, the estimate is no more unbiased, but the variance is in general significatively reduced.

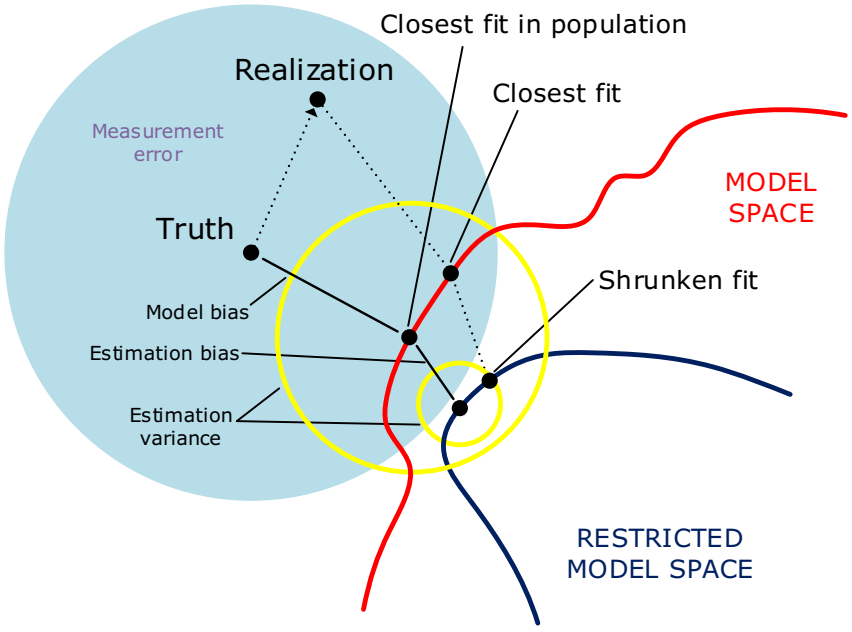


Figure 2.2: Regularization and bias-variance decomposition for linear models (adapted from [4])

The intuition behind regularization is depicted in Figure 2.2. In the case of linear models, the model space is the set of all linear predictions from m inputs

and the black dot labeled "closest fit" is best linear model fit. The blue-shaded region indicates the error's variance with which we see the truth in the training sample, as stated in the model (2.11). The "closest fit in population" represents the fit obtained when no stochastic disturbance is present. In this case, the figure depicts with the yellow circle the variance of the estimate. The model bias is indicated as the distance from the truth, when a linear model is not sufficient to correctly fit all the data. When we further restrict the model space by regularization, the estimation bias becomes greater than zero, since we have no more the best linear model, but the variance is drastically reduced. For linear models fit by ordinary least squares, the estimation bias is zero. As long as this gain in variance is greater than the increment in bias, the procedure is worthwhile.

The Ridge and Lasso penalties can be alternatively seen as constrained minimization problems:

$$\hat{\theta}_R = \arg\min_{\theta} \sum_{t=1}^{N} \left(y(t) - \varphi^T(t)\theta\right)^2 \tag{2.15}$$

$$\text{subject to} \qquad \sum_{j=1}^{m} \theta_j^2 \leq C, \tag{2.16}$$

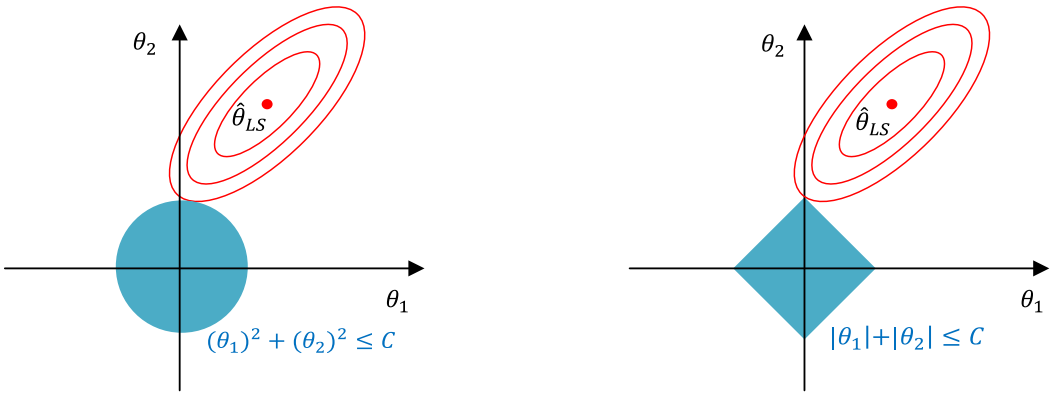
and respectively:

$$\hat{\theta}_L = \arg\min_{\theta} \sum_{t=1}^{N} \left(y(t) - \varphi^T(t)\theta\right)^2 \tag{2.17}$$

$$\text{subject to} \qquad \sum_{j=1}^{m} |\theta_j| \leq C. \tag{2.18}$$

There is an inverse proportionality between λ and C: if C is lower, then the constraint on the parameters' value is tighter, meaning that λ is higher. A geometrical interpretation of the Ridge and Lasso is depicted in Figure 2.3, for a 2-dimensional cost function. The regularized solution is the intersection (in the paramters space) of the cost function's contours with the region dictated by the constraint. It is possible to observe that the Ridge penalty shrinks the parameters to a lower value which is different from zero, while the Lasso solution tends to exactly set one (or more) parameters to be identically null.

Computing the Lasso solution is a quadratic programming problem, and therefore it is computed as a byproduct of very computationally efficient algorithms such as the Least Angle Regression (LAR) [29]. On the contrary,



(a) Ridge regression contour surfaces: the method tends to give soutions with a low parameters' value

(b) Lasso regression contour surfaces: the method tends to give a sparse parameter's vector

Figure 2.3: Comparison between Ridge and Lasso regularizations from a geometrical viewpoint

the Ridge regression has a closed form solution [4]. It is useful, for this purpose, to express model (2.11) in matrix notation:

$$Y = \Phi\theta + E, \tag{2.19}$$

where $Y \in \mathbb{R}^{N\times 1}$ contains the $y(t)$ stacked in row, $\Phi \in \mathbb{R}^{N\times m}$ is the regressors' matrix obtained by stacking the single regressors $\varphi^T(t) \in \mathbb{R}^{1\times m}$, $\theta \in \mathbb{R}^{m\times 1}$ is the parameters' vector and $E \in \mathbb{R}^{N\times 1}$ contains the stacked error terms. The Least Square formulation (2.12) becomes:

$$\hat{\theta}_{LS} = \arg\min_{\theta} \|Y - \Phi\theta\|^2 \tag{2.20}$$

$$= \left(\Phi^T\Phi\right)^{-1}\Phi^T Y, \tag{2.21}$$

where $\|\cdot\|$ indicates the Euclidean norm. Using the matrix notations, the regularized Ridge regression problem (2.13) becomes:

$$\hat{\theta}_R = \arg\min_{\theta} \|Y - \Phi\theta\|^2 + \lambda\|\theta\|^2 \tag{2.22}$$

$$= \left(\Phi^T\Phi + \lambda I_m\right)^{-1}\Phi^T Y, \tag{2.23}$$

where $I_m \in \mathbb{R}^{m\times m}$ is the identity matrix.

The solution (2.22) to the Ridge regression problem (2.23) is nearly identical to the standard least squares formulation (2.21). The only difference lies in the λI_m term, which acts as a tool to better condition the estimate.

2.2.2 Bayesian interpretation of regularization

The result in (2.23) admits an interesting Bayesian interpretation. Let $x \in \mathbb{R}^{d\times 1}$ be a Gaussian random vector with mean $\mu \in \mathbb{R}^{d\times 1}$ and covariance $\Sigma \in \mathbb{R}^{d\times d}$, such that $x \sim \mathcal{N}(\mu, \Sigma)$. We can partition x into two disjoint subsets $x_a \in \mathbb{R}^{m\times 1}$ and $x_b \in \mathbb{R}^{N\times 1}$ with $d = m + N$, being m the dimension of x_a and N the dimension of x_b, so that:

$$x = \begin{bmatrix} x_a \\ x_b \end{bmatrix}. \tag{2.24}$$

The joint distribution of x_a and x_b can be expressed as:

$$\begin{bmatrix} x_a \\ x_b \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix}, \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix} \right), \tag{2.25}$$

where $\Sigma_{aa} \in \mathbb{R}^{m\times m}$ and $\Sigma_{bb} \in \mathbb{R}^{N\times N}$are symmetric, $\Sigma_{ab} \in \mathbb{R}^{m\times N}$, $\Sigma_{ba} \in \mathbb{R}^{N\times m}$ and $\Sigma_{ab}^T = \Sigma_{ba}$. Then, the conditional distribution $x_a|x_b$ is still Gaussian [30]:

$$x_a|x_b \sim \mathcal{N}(\mu_{a|b}, \Sigma_{a|b}) \tag{2.26}$$

$$\mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b) \tag{2.27}$$

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}. \tag{2.28}$$

It is often useful to express the Gaussian distribution as a function of the *precision matrix* instead of the covariance one:

$$\Lambda \equiv \Sigma^{-1}. \tag{2.29}$$

We refer to equations (2.27) - 2.28 as the *covariance representation* of the Gaussian conditional distribution: expressing them in terms of the precision matrix leads to the *precision form*:

$$\mu_{a|b} = \Sigma_{a|b} \cdot \{\Lambda_{aa}\mu_a - \Lambda_{ab}(x_b - \mu_b)\} \tag{2.30}$$

$$= \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b) \tag{2.31}$$

$$\Sigma_{a|b} = \Lambda_{aa}^{-1}, \tag{2.32}$$

where:

$$\Lambda_{aa} = \left(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}\right)^{-1} \tag{2.33}$$

$$\Lambda_{ab} = -\left(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}\right) \cdot \Sigma_{ab}\Sigma_{bb}^{-1}. \tag{2.34}$$

The formulas outlined until now refer to the general case of two jointly Gaussian vectors x_a and x_b. When a specific parametric structure is imposed on these variables, it is possible to exploit such formulation to restate the more general formulas, deriving a practical solution for the problem at hand. Consider now the linear model (2.19), where $E \sim \mathcal{N}(0, L^{-1})$ and $L \in \mathbb{R}^{N \times N}$ is the errors' precision matrix. It is possible to define a *prior distribution* over the parameters' vector θ such that:

$$p(\theta) = \mathcal{N}(\mu, \Lambda^{-1}), \tag{2.35}$$

where $\Lambda \in \mathbb{R}^{m \times m}$ is the parameters' precision matrix. The vector θ is now considered a random variable, as opposite to the frequentist approach where it was a deterministic vector. The conditional distribution of the data can be expressed as:

$$p(Y|\theta) = \mathcal{N}(\Phi\theta, L^{-1}). \tag{2.36}$$

Having imposed $p(\theta)$ in (2.35) and computed $p(Y|\theta)$ (2.36), the aim is to derive $p(\theta|Y)$. Since $p(\theta)$ and $p(Y|\theta)$ are Gaussian, their joint pdf is still Gaussian, with $z = [\theta, Y]^T \in \mathbb{R}^{d \times 1}$, $d = m + N$, and $p(z) = \mathcal{N}(\mu_z, \Sigma_z)$:

$$z = \begin{bmatrix} \theta \\ Y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \Phi\mu \end{bmatrix}, \begin{bmatrix} \Lambda^{-1} & \Lambda^{-1}\Phi^T \\ \Phi\Lambda^{-1} & L^{-1} + \Phi\Lambda^{-1}\Phi^T \end{bmatrix} \right). \tag{2.37}$$

The precision matrix R of $p(z)$ can be found to be [30]:

$$R \equiv \Sigma_z^{-1} = \begin{bmatrix} \Lambda + \Phi^T L \Phi & -\Phi^T L \\ -L\Phi & L \end{bmatrix} \tag{2.38}$$

Substituting (2.37) into equations (2.26)-(2.28) yields the conditional distribution $p(\theta|Y) \sim \mathcal{N}(\mu_{\theta|Y}, \Sigma_{\theta|Y})$ in covariance form:

$$\mu_{\theta|Y} = \mu + \Lambda^{-1}\Phi^T \left(L^{-1} + \Phi\Lambda^{-1}\Phi^T\right)^{-1} (Y - \Phi\mu) \tag{2.39}$$

$$\Sigma_{\theta|Y} = \Lambda^{-1} - \Lambda^{-1}\Phi^T \left(L^{-1} + \Phi\Lambda^{-1}\Phi^T\right)^{-1} \Phi\Lambda^{-1}, \tag{2.40}$$

while using (2.38) in (2.30)-(2.32) gives $p(\theta|Y)$ in precision form:

$$\mu_{\theta|Y} = \left(\Lambda + \Phi^T L \Phi\right)^{-1} \cdot \left\{ \left(\Lambda + \Phi^T L \Phi\right) \mu + \Phi^T L \left(Y - \Phi\mu\right) \right\} \tag{2.41}$$

$$= \left(\Lambda + \Phi^T L \Phi\right)^{-1} \cdot \left\{ \Phi^T L Y + \Lambda\mu \right\} \tag{2.42}$$

$$\Sigma_{\theta|Y} = \left(\Lambda + \Phi^T L \Phi\right)^{-1} \tag{2.43}$$

Once the posterior distribution is computed, if one wants a point value estimate for the unknown parameters, the most common choice is to take the Maximum A Posteriori (MAP) estimate. This is the value of the parameters which maximizes the posterior probability density. Since $P(\theta|Y)$ is Gaussian, $\hat{\theta}_{MAP}$ corresponds to $\mathbb{E}[p(\theta|Y)] = \mu_{\theta|Y}$. If the prior distributions on errors and parameters are taken such that:

$$p(\theta) = \mathcal{N}(\mu = 0,\ \Lambda^{-1} = \gamma^2 I_m) \tag{2.44}$$

$$p(E) = \mathcal{N}(0,\ L^{-1} = \sigma^2 I_N), \tag{2.45}$$

where $I_m \in \mathbb{R}^{m\times m}$ and $I_N \in \mathbb{R}^{N\times N}$ are identity matrices, then the conditioned probability of the data becomes:

$$p(Y|\theta) = \mathcal{N}(\Phi\theta,\ \sigma^2 I_N). \tag{2.46}$$

The posterior mean, by substituting (2.44)-(2.45) in (2.41) is therefore:

$$\mu_{\theta|Y} = \left[\left(\gamma^2 I_m\right)^{-1} + \Phi^T\left(\sigma^2 I_N\right)^{-1}\Phi\right]^{-1} \cdot \left\{\Phi^T\left(\sigma^2 I_N\right)^{-1} Y\right\} \tag{2.47}$$

$$= \left[\frac{1}{\gamma^2} I_m + \frac{1}{\sigma^2}\Phi^T\Phi\right]^{-1} \cdot \frac{1}{\sigma^2}\Phi^T I_N Y \tag{2.48}$$

$$= \left[\frac{1}{\sigma^2} \cdot \left(\frac{\sigma^2}{\gamma^2} I_m + \Phi^T\Phi\right)\right]^{-1} \cdot \frac{1}{\sigma^2}\Phi^T Y \tag{2.49}$$

$$= \left[\Phi^T\Phi + \frac{\sigma^2}{\gamma^2} I_m\right]^{-1} \cdot \Phi Y = \hat{\theta}_{MAP}. \tag{2.50}$$

The posterior covariance, by elaborating on (2.43) is:

$$\Sigma_{\theta|Y} = \left(\Lambda + \Phi^T L \Phi\right)^{-1} = \left[\frac{1}{\gamma^2} I_m + \frac{1}{\sigma^2}\Phi^T\Phi\right]^{-1}. \tag{2.51}$$

By setting $\lambda = \frac{\sigma^2}{\gamma^2}$ in (2.50), the MAP estimate can be restated as:

$$\hat{\theta}_{MAP} = \left(\Phi^T\Phi + \lambda I_m\right)^{-1}\Phi Y, \tag{2.52}$$

which is equivalent to the frequentist formulation (2.23). *A Gaussian prior on the unknown parameters is a way to induce regularization in the learning model.* The interpretation of the term $\frac{\sigma^2}{\gamma^2}$ is as follows. If σ^2 is near zero, then there is little noise in the data, and the expression reduces to the simple least square estimate, since an unbiased solution is preferred and there is no need to

regularize to avoid the risk of overfitting caused by the noise. The same result appears if the parameters' prior is very vague: in this case, no preference is given to particular parameters' values, and so the regularization effect vanishes. If the parameters' prior is very concentrated, then the regularization term is higher, since we are more confident that the parameters should not deviate much from that we have imposed with the prior information.

As a concluding remark for this section, it possible to show that the Lasso penalty formulation (2.13) can be recovered in a Bayesian framework, when a Laplace distribution is taken as prior information for the parameters [4].

2.3 Regularization in dynamic systems

The natural application of the regularization's concepts introduced in Section 2.2 concerns those dynamic models that can be casted as linear regression ones. Here, the indexing letter t denotes a temporal unit. As an example, consider the generic Finite Impulse Response (FIR) model:

$$y(t) = G(q,\theta)u(t) + e(t) = \sum_{k=1}^{m} g_k u(t-k) + e(t) \tag{2.53}$$

$$= \varphi_u^T(t)\theta_g + e(t), \tag{2.54}$$

where in $\varphi_u^T(t)$ and θ_g are collected the m elements of $u(t-k)$ and the m impulse response coefficients g_k, respectively. This is again a linear regression problem. All that was said above about linear regressions, regularization and estimation of hyper-parameters can thus be applied to the estimation of FIR models. Consider a more general version of the regularization problem (2.22):

$$\hat{\theta}_T = \arg\min_{\theta} \|Y - \Phi\theta\|^2 + \lambda\theta^T P^{-1}\theta \tag{2.55}$$

$$= \left(P\Phi^T\Phi + \lambda I_m\right)^{-1} P\Phi^T Y, \tag{2.56}$$

where $P \in \mathbb{R}^{m\times m}$ is a proper regularization matrix. This is a general formulation of the Tikhonov regularization. Note that problem (2.55) is equivalent to problem (2.22) if $P = I_m$. Suitable choices of P should reflect what is reasonable to assume about an impulse response. If the system is exponentially stable, the impulse response coefficients g_k should decay exponentially, and if the impulse response is smooth, neighbouring values should have a positive correlation. A

suitable regularization matrix P^g for θ_g is a matrix whose k, j element is:

$$\text{DC} \quad P^g_{kj}(\eta) = \delta\alpha^{(k+j)/2}\rho^{|j-k|}; \tag{2.57}$$

$$\delta \geq 0, \;\; 0 \leq \alpha < 1, \;\; |\rho| \leq 1; \;\; \eta = [\delta, \alpha, \rho]. \tag{2.58}$$

Here α accounts for the exponential decay along the diagonal, while ρ describes the correlation across the diagonal (the correlation between neighbouring impulse response coefficients).
We call this matrix, or *kernel*, DC for Diagonal-Correlated [6].

A special case derived from (2.57) is obtained by posing $\rho = \sqrt{\alpha}$, leading to the Tuned/Correlated kernel [31, 32]:

$$\text{TC} \quad P^g_{kj}(\eta) = \delta\alpha^{\max(k,j)}; \tag{2.59}$$

$$\delta \geq 0, \;\; 0 \leq \alpha < 1, \;\; \eta = [\delta, \alpha]. \tag{2.60}$$

A third useful kernel is the Stable Spline (SS) one [8]:

$$\text{SS} \quad P^g_{kj}(\eta) = \delta\left(\frac{\alpha^{k+j+\max(k,j)}}{2} - \frac{\alpha^{3\cdot\max(k,j)}}{6}\right); \tag{2.61}$$

$$\delta \geq 0, \;\; 0 \leq \alpha < 1, \;\; \eta = [\delta, \alpha]. \tag{2.62}$$

The hyperparameter η can be tuned by marginal likelihood optimization via (A.11). Efficient numerical implementation of this minimization problem is discussed in [33, 34]. The impulse response can then be computed from (2.55) with $\lambda = \sigma^2$. This method of estimating the impulse response, possibly followed by a model reduction of the high order FIR model, has been extensively tested in Monte Carlo simulations in [6]. The method is shown to be a viable alternative to the classical PEM/ML estimation, mainly because the question of model order determination is avoided.

Example: 2.3 (i) *Diagonal-Correlated DC kernel*

Consider the problem (2.55) with kernel matrix as defined in (2.57). Suppose $m = 2, \delta = 1$ for ease of explanation. Then, the kernel matrix $P^g \in \mathbb{R}^{2\times 2}$ is:

$$P^g = \begin{bmatrix} \alpha & \alpha^{3/2}\rho \\ \alpha^{3/2}\rho & \alpha^2 \end{bmatrix} \tag{2.63}$$

The regularization term $\theta^T P^{-1}\theta$ in (2.55) is therefore:

$$\begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix} \begin{bmatrix} \alpha & \alpha^{3/2}\rho \\ \alpha^{3/2}\rho & \alpha^2 \end{bmatrix}^{-1} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \tag{2.64}$$

$$\frac{1}{\det P^g} \cdot \begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix} \begin{bmatrix} \alpha^2 & -\alpha^{3/2}\rho \\ -\alpha^{3/2}\rho & \alpha \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \tag{2.65}$$

$$\frac{1}{\det P^g} \cdot \left[\theta_1^2\alpha^2 \; + \theta_2^2\alpha \; - \; 2\theta_1\theta_2\alpha^{3/2}\rho\right] \tag{2.66}$$

Equation (2.66) encodes the prior information on the FIR model impulse response. The term θ_1^2 is weighted less than the term θ_2^2, meaning that g_2 is less important with respect to g_1 in predicting the next impulse response value. If θ_1 and θ_2 have equal sign, then the last term avoids bringing to zero the interaction between the impulse responses coefficients, with a strength proportional to the ρ parameter. If θ_1 and θ_2 have different sign, the minimization of the cost function tries to bring both the parameters to lower values, imposing the required smoothness.

CHAPTER 3

Nonparametric system identification

This chapter introduces the time-domain nonparametric approach to system identification, based on the framework of Reproducing Kernel Hilbert Spaces (RKHS), described in Section 3.1. The aim is to learn an unknown function from a set of measured data. This function is searched in a functions set defined by an RKHS. By interpreting the system identification problem as a function estimation one, the methods and concepts that belong to the latter framework can be employed (with the due modifications) in the former one. In Section 3.2 the framework of the RKHS is presented for the static function's learning case. Then, Section 3.3 introduces the application of nonparametric learning procedure to LTI SISO dynamic systems.

A brief review of the fundamental concepts of functional analysis are reported in Appendix B. For a detailed exposition of these concepts, see [35].

3.1 Reproducing Kernel Hilbert Spaces

Definition 3.1.1 *A Reproducing Kernel Hilbert Space (RKHS) is a Hilbert space V such that:*

a) Its elements are functions $u : \Omega \to \mathbb{R}$, where Ω is a generic set

b) $\forall x \in \Omega$, $L_x : \underset{u \mapsto u(x)}{V \to \mathbb{R}}$ is a continous linear functional, that is, $\exists M \in \mathbb{R}$ such that $|u(x)| \leq M \cdot \|u\| \ \ \forall u \in V$

For the Riesz's representation theorem (see Appendix B), there exists a function $r_x \in V$ (called the representer of x) such that $L_x = \langle \cdot, r_x \rangle$, that is, $L_x(u) = u(x) = \langle u, r_x \rangle$. By setting $u = r_z$, we have that $r_z(x) = \langle r_z, r_x \rangle$. We define the reproducing kernel as:

$$K(x, z) = \langle r_x, r_z \rangle \,, \tag{3.1}$$

$$K : \Omega \times \Omega \to \mathbb{R}. \tag{3.2}$$

The reproducing kernel K is:

a) symmetric: $K(x, z) = K(z, x)$

b) semidefinite positive: $\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0 \quad \forall n, c_i \in \mathbb{R}, \forall x_i \in \Omega$

The theory of Reproducing Kernel Hilbert Spaces (RKHS) was first introduced in [36]. Their adoption as hypothesis space provides a general solutions to functions' estimation and approximation problems, see [37, 38].

Example: 3.1 (i) *Reproducing Kernel Hilbert Space*

Let $V = \{u \in L^2(\mathbb{R}) \mid \text{supp}(\hat{u}) \subset [-a, a]\}$, where $a > 0$ is fixed and $\mathcal{F}[u](\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{-jx\omega} u(x)\, dx$ is the Fourier transform of u, with j the imaginary unit. Then, V is the space of functions with limited bandwidth. It can be shown that V is a Hilbert space with respect to the inner product $\langle u, v \rangle = \int_{-\infty}^{+\infty} u(x)v(x)\, dx$. It can also be shown that:

- L_x is continous $\forall x \in \mathbb{R}$
- V is a RKHS
- The reproducing kernel is:

$$K(x, z) = \frac{\sin\big(a\,(z - x)\big)}{\pi\,(z - x)}$$

An important result is the following theorem [37]:

Theorem 3.1.1 ***Moore-Aronszajn theorem***

A RKHS defines a corresponding reproducing kernel. Conversely, a reproducing kernel defines a unique RKHS.

So, if we are able to define a reproducing kernel, we know that there exists an associated RKHS $\mathcal{H}$. The kernel defines the properties of the functions that belong to the respective RKHS.

Example: 3.1 (ii) *Reproducing kernels*
Examples of reproducing kernels are the following:

- **Linear kernel**: $K(x,z) = x \cdot z$
- **Gaussian kernel**: $K(x,z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}}$
- **Polynomial kernel**: $K(x,z) = (x \cdot z + 1)^d$

The idea is then, in the case of a learning problem, to search the unknown function in $\mathcal{H}$. The advantage is that, by selecting an appropriate kernel, we are able to impose the desired characteristics to the functions in $\mathcal{H}$ and, as a conseguence, to the learned one. An example of this is as follows.

Example: 3.1 (iii) *Kernels and induced function space*

Suppose that the (non-normalized) Gaussian kernel $K(x,z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}}$ is employed. Let $C_0^0(\mathbb{R}^m)$ denote the set of continous function with compact support. The set

$$\mathcal{H}_\sigma \equiv \left\{ u \in C_0^0(\mathbb{R}^m) \,:\, u \in L_1(\mathbb{R}^m) \text{ and } \int_{\mathbb{R}^m} |\mathcal{F}[u](\omega)|^2 e^{\frac{\sigma^2\|\omega\|^2}{2}} \, d\omega < +\infty \right\}$$

is the RKHS associated with the Gaussian kernel $K(x,z)$ [39]. It can be shown that the functions $u \in \mathcal{H}_\sigma$ have induced norm:

$$\|u\|^2 \propto \int_{\mathbb{R}^m} |\mathcal{F}[u](\omega)|^2 e^{\frac{\sigma^2\|\omega\|^2}{2}} \, d\omega.$$

Notice that the induced norm is greater for functions that have higher frequency components. Then, as a conseguence of the characterization of the Gaussian RKHS $\mathcal{H}_\sigma$, the following statements hold:

1. For any $0 < \sigma < \tau$, we have that:

$$\mathcal{H}_\tau \subset \mathcal{H}_\sigma \subset L_2(\mathbb{R}^m)$$

2. For any $u \in \mathcal{H}_\tau$, we have that:

$$\|u\|_{\mathcal{H}_\tau} \geq \|u\|_{\mathcal{H}_\sigma} \geq \|u\|_{\mathcal{H}_{L_2}}$$

The interpretation of the previous statements is that, as σ gets higher, less functions satisfy the definition of $\mathcal{H}_\sigma$, since the functions' norm increases.

3.2 Regularization in RKHS

A typical formulation of a static nonparametric learning problem is to find the function $\hat{g} : \Omega \to \mathbb{R}$, in a RKHS $\mathcal{H}$, such that:

$$\hat{g} = \arg\min_{g \in \mathcal{H}} \sum_{i=1}^{N} \big(y_i - g(x_i)\big)^2 + \lambda \cdot \|g\|_{\mathcal{H}}^2, \tag{3.3}$$

for a finite set N of points $(x_i, y_i) \in \Omega \times \mathbb{R}$. Here, we used the shorthand notation $x_i = x(t_i)$ and $y_i = y(t_i)$. This means that y_i is the t_i-th observation of the outcome variable y, and x_i is the t_i-th observation of the regressor variable x. The variational problem (3.3) is similar to the formulations (2.13) and (2.14). The first term in (3.3) is the cost which measures the function's fit to data. A condition for the existence of a unique minimizer of (3.3) is that this cost has to be strictly convex (no flat regions) and coercive (meaning that it grows rapidly at extrema). The second term represents a regularization component, penalizing the norm of each hypothesis g in the hypothesis space $\mathcal{H}$, where $\lambda > 0$ is an hyperparameter. When a RKHS is adopted as hypothesis space, problem (3.3) is well-posed: the solution is unique and is *stable*, meaning that small perturbations in the data do not change the final solution [12].

The rationale of the method is clearly different from the parametric case: instead of constraining the unknown function to a specific parametric structure, g is searched over a possibly infinite-dimensional functional space $\mathcal{H}$ (a space is infinite-dimensional if it does not have a finite basis). One question naturally arises: how it is possible to practically manage an infinite number of values? The answer is given by the following result [40]:

Theorem 3.2.1 ***Representer theorem***

The minimizer of (3.3)*, where $\mathcal{H}$ is a RKHS, is given by:*

$$\hat{g}(x) = \sum_{i=1}^{N} c_i K(x, x_i) = \sum_{i=1}^{N} c_i K(x_i, x) \tag{3.4}$$

$$= \sum_{i=1}^{N} c_i r_{x_i}(x), \tag{3.5}$$

for some N-tuple $c = [c_1, c_2, \ldots, c_N]^T \in \mathbb{R}^{N \times 1}$.

Hence, minimizing over the (possibly infinite-dimensional) Reproducing Kernel Hilbert Space $\mathcal{H}$ boils down to minimizing over $\mathbb{R}^N$, that is, to estimate the coefficients' vector $c \in \mathbb{R}^N$. The expression (3.5) has an intuitive interpretation:

the estimated function's value at a point x, i.e. $\hat{g}(x)$, is obtained as a weighted sum of the representer functions r_{x_i} of every point x_i in the dataset, evaluated in the new point x. The solution appears as just a sum of basis functions (the representers): however, unlike the standard parametric approach, in this solution the number of basis functions is not fixed, but it depends on the number of data points. The estimator (3.5) is known in the literature as *regularization network* [41] or *least squares support vector machine* [42].

Example: 3.2 (i) *Impact of kernel choice on the learned function*

Consider the RKHS $\mathcal{H}_\sigma$ induced by the (non-normalized) Gaussian kernel $K(x,z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}}$. Then, the function $u \in \mathcal{H}_\sigma$ has induced norm [39]:

$$\|u\|^2 \propto \int_{\mathbb{R}^m} |\mathcal{F}[u](\omega)|^2 e^{\frac{\sigma^2\|\omega\|^2}{2}} \, d\omega.$$

The formulation (3.3) then penalizes more heavily functions with high frequency components, striving for smoother solutions. As σ gets higher, the norm increases and the regularization term becomes more important, restraining the function space to a set of even smoother functions.

It turns out that the expression for the vector of coefficients c admits a closed-form solution. Using the representer theorem (3.5), an alternative expression for the squared norm $\|g\|_{\mathcal{H}}^2$ is:

$$\|g\|_{\mathcal{H}}^2 = \langle g, g \rangle_{\mathcal{H}} = \left\langle \sum_{i=1}^{N} c_i r_{x_i}, \sum_{j=1}^{N} c_j r_{x_j} \right\rangle \tag{3.6}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j \left\langle r_{x_i}, r_{x_j} \right\rangle \tag{3.7}$$

$$= c^T \mathcal{K} c, \tag{3.8}$$

where $\mathcal{K} \in \mathbb{R}^{N \times N}$ is a semidefinite positive matrix (also called Gram matrix or kernel matrix) such that $\mathcal{K}_{ij} = K(x_i, x_j)$. Using (3.5) and (3.8) it is possible to rewrite the minimization problem (3.3) as function of only the vector of coefficients c:

$$\hat{c} = \underset{c \in \mathbb{R}^N}{\arg\min} \, \|Y - \mathcal{K}c\|_2^2 + \lambda \cdot c^T \mathcal{K} c, \tag{3.9}$$

where $Y \in \mathbb{R}^{N \times 1}$ contains the y_i stacked in row, and $\|\cdot\|_2$ denotes the 2-norm operator. Solving problem (3.9) (by setting the partial derivatives with respect

to c to zero) yields:

$$(\mathcal{K} + \lambda I_N) \cdot \hat{c} = Y. \tag{3.10}$$

The estimate $\hat{c}$ of the coefficients' vector c can then be obtained by solving the linear system (3.10).

In the case of the linear kernel, i.e. $K(x, z) = x^T \cdot z$, the kernel matrix $\mathcal{K}$ can be written as $\mathcal{K} = \Phi\Phi^T$, where $\Phi \in \mathbb{R}^{N \times m}$ contains the N observed regressors $x \in \mathbb{R}^{m \times 1}$ stacked in row. The problem (3.9) can then be reformulated as:

$$\hat{c} = \underset{c \in \mathbb{R}^N}{\arg\min} \|Y - \Phi\Phi^T c\|_2^2 + \lambda \cdot c^T \Phi\Phi^T c \tag{3.11}$$

By defining $\theta \equiv \Phi^T \cdot c \in \mathbb{R}^{m \times 1}$ we have that (3.11) becomes:

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^m}{\arg\min} \|Y - \Phi\theta\|_2^2 + \lambda \cdot \theta^T \theta \tag{3.12}$$

$$= \underset{\theta \in \mathbb{R}^m}{\arg\min} \|Y - \Phi\theta\|_2^2 + \lambda \cdot \|\theta\|_2^2 \tag{3.13}$$

$$= \left(\Phi^T\Phi + \lambda I_m\right)^{-1} \Phi^T Y \tag{3.14}$$

The solution to the problem (3.3), when the function space $\mathcal{H}$ generated by the linear kernel $K(x, z) = x^T \cdot z$ is adopted as hypothesis space, *is therefore the same* as the Ridge regression solution (2.23) and the MAP estimate of the bayesian linear model problem with a Gaussian prior on the parameters (2.52).

3.3 System identification as function estimation

One successful approach, that incorporated the kernel methods previously presented into the system identification field, has been proposed in [8]. Here the authors faced the problem of identifying a continous stable SISO system's impulse response. The impulse response is seen as a continous function of time, and lends itself naturally to the applications of the methods described in Section 3.2, that dealt with continous static functions g. Assuming an output error model, one has that the response of a dynamic system is:

$$y(t_i) = \int_0^{+\infty} u(t_i - s)g(s)\, ds + e(t_i) \;\; i = 1, \ldots, N \tag{3.15}$$

where $u(t_i), y(t_i)$ are the system's input and output (at time t_i) respectively, $e(t_i)$ is a white noise and g is now the impulse response. The dataset is now composed by the couples $\{u(t_i), y(t_i)\}$. The system identification problem

presents some peculiarities with respect to the static case:

1. the domain of the function g is one-dimensional and given by the positive real axis (the domain is now the time), i.e. $\Omega = \mathbb{R}^+$ and $g : \mathbb{R}^+ \to \mathbb{R}$

2. compared to the static case, the nature of the data is more complex since, in place of $g(x_i)$, the measurement model involves the functional L_i given by:
$$L_i[g] = \int_0^{+\infty} u(t_i - s)g(s)\, ds \tag{3.16}$$

The optimization problem can now be formulated as:

$$\hat{g} = \arg\min_{g \in \mathcal{H}} \sum_{i=1}^{N} \left(y_i - L_i[g]\right)^2 + \lambda \cdot \|g\|_{\mathcal{H}}^2, \tag{3.17}$$

which coincides with (3.3) except that $L_i[g]$ replaces $g(x_i)$. The authors in [8] established that the representer theorem is still valid for problem (3.17). This is the case because the functional L_i is linear and continous, so the Riesz's theorem holds (see Appendix B). The functional L_i can therefore be expressed as $L_i[\cdot] = \langle \cdot, r_{t_i} \rangle$, and so we are in a situation like in the case of static systems. The solution to (3.17), as function of time t, can be expressed using the representer theorem 3.2.1 as:

$$\hat{g}(t) = \sum_{i=1}^{N} c_i L_i[r_t], \tag{3.18}$$

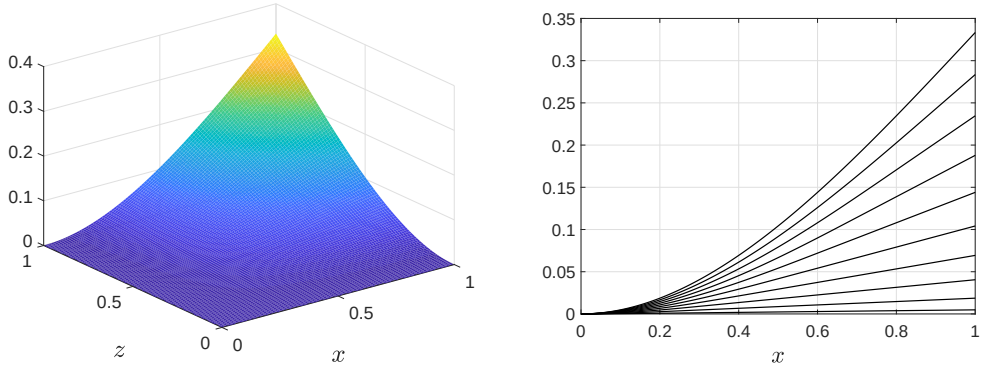
where $t \in \mathbb{R}^+$ and the coefficients' vector c admits a closed-form solution. The difference with respect to the static case is that the basis functions are no more the representers, but their convolution with the input u, i.e. $L_i[r_t] = \int_0^{+\infty} u(t_i - s)K(t, s)ds$.

A key issue is the choice of the kernel function. As stated by the representer theorem (3.5), the representer functions $r_{x_i}(x)$ (also known as *kernel's sections*) constitute the elements that, linearly combined, compose the final solution. For stable dynamic systems, the impulse response decreases exponentially to zero. Therefore, a suited kernel is one that generates kernel's sections which decrease to zero, imposing a sort of "prior knowledge" about the unknown function. Broadly speaking, this represents the fact that the uncertainty about the impulse response behaviour decreases as time increases, since we are sure that it will eventually go to zero. Let's now introduce the *cubic spline kernel*

as [21]:

$$K(x,z) = \frac{x \cdot z \cdot \min\{x,z\}}{2} - \frac{\min\{x,z\}^3}{6} \tag{3.19}$$

where $K : \Omega \times \Omega \to \mathbb{R}$, $\Omega = [0,1]$, and which kernel's sections are cubic smoothing splines, consisting of piecewise third-order polynomials, see Figure 3.1.



(a) Cubic spline kernel

(b) Cubic spline representers

Figure 3.1: Cubic spline kernel and kernel's sections $r_z(x)$ for $z = 0.1, 0.2, \ldots, 1$

The use of spline functions, originally employed in the interpolation scenario, permits to leverage notable numerical properties. In particular, piecewise polynomials avoid Runge's phenomenon [43] (presence of large oscillations in the reconstructed function). As it is possible to notice in Figure 3.1, cubic splines representers grow as the independent variable (the unkown function's input) x gets higher. In our formulation for dynamic systems, the input domain is the positive real time axis. Let's rename the independent input variable as t: then, $t \in \mathbb{R}^+$. We therefore need representers that decrease to zero as time t gets higher, in order to reflect the behaviour of the impulse response (the unknown function that needs to be found). A solution could be to employ the following change of coordinates using the cubic spline kernel $K(x,z)$ defined in (3.19):

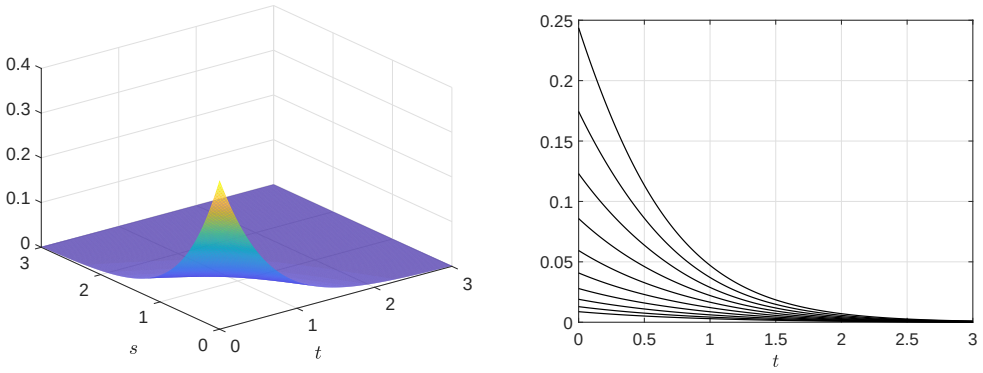
$$S(t,s) = K(\underbrace{e^{-\beta t}}_{x}, \underbrace{e^{-\beta s}}_{z}), \tag{3.20}$$

where $t, s \in \mathbb{R}^+$ and β is a hyperparameter that can be tuned by empirical Bayes (see Appendix (A.3)). This procedure maps time values t and s (belonging to $\mathbb{R}^+$) to suitable input, which belongs to $(0,1]$, for the cubic spline kernel. Therefore, when $t = 0$, we have that $e^{-\beta t} = 1$ and the cubic spline kernel assumes the maximum value. As time increases, the value of $e^{-\beta t}$ gets lower, which translates into a lower value of the cubic spline kernel. By substituting

expression (3.19) in (3.20), we obtain the *stable spline kernel*, see (3.2):

$$S(t,s) = \frac{e^{-\beta(t+s+\max(t,s))}}{2} - \frac{e^{-3\beta\max(t,s)}}{6}, \tag{3.21}$$

which is a valid kernel as shown in [21] and exibits the desidered properties.



(a) Stable spline kernel

(b) Stable spline representers

Figure 3.2: Second-order stable spline kernel with $\beta = 1$ and kernel sections $r_s(t)$ for $s = 0.2, 0.4, \ldots, 2$

The stable spline kernel (3.20), defined for continous time systems, can be straightforwardly extented to the discrete time setting by setting $\alpha = e^{-\beta}$ and obtainining the sampled version (2.61).

Just as Ridge regression (which we obtained as special case of kernel regression equipped with a linear kernel) admitted a Bayesian interpretation, the generic formulation of kernel regression (3.3) can be given a probabilistic interpretation in a Bayesian framework [17, 38, 44, 45]. The rationale is to interpret the unknown function as as continous-time Gaussian stochastic process on $\mathbb{R}^+$. The observations y_i are modeled as in (3.15) and (3.18), i.e., $y_i = L_i[g] + e_i$, where:

- the e_i are zero-mean Gaussian of variance σ^2, mutually independent and independent of g;
- the prior for the system's impulse response g is a zero-mean Gaussian stochastic process, on $\mathbb{R}^+$, with autocovariance $\gamma^2 \cdot K$ and independent of e_i.

Then, it is possible to show that the MAP estimation of the system's impulse respose g given the data is obtained by minimizing (3.3) with $\lambda = \frac{\sigma^2}{\gamma^2}$. The reader can find a connection with the Bayesian derivation of Ridge regression (2.52).

The application of kernel methods to system identification has been extented to the non-linear case in [22]. In both cases, the main advantage of these methods is the elimination of the model order selection step. This is performed by the regularization term and the correspective (robust) tuning of the kernel's hyperparameters, which permit to explore a richer range of solutions with respect to the "hard thresholding" of model order selection methods.

Other types of regularization employed for system identification relied on the Lasso formulation [46, 47] (presented in Section 2.2.1). Regularized approaches for nonlinear and state-space models identification can be found in [48, 49, 50].

The use of kernel methods in system identification is an actual and promising line of research. After that much work has alredy been done in time-domain, kernel methods are currently finding more and more applications also in the frequency-domain system identification methodologies [26]. In addition to the research of new kernel's types, that embody particular prior information on the system, another possible research direction is the introduction of new types of regularization terms, in place of the squared norm $\|g\|^2_{\mathcal{H}}$ in the cost function 3.3. This is exacly the focus that we will carry on in the next section, constituting the *first contribution* of this thesis.

CHAPTER 4

Semi-supervised system identification

This chapter lays the foundation for the introduction of a new regularization term, employed in the cost function (3.3), where the system identification problem is cast into the framework of Reproducing Kernel Hilbert Spaces (RKHS). The focus will be on Nonlinear Finite Impulse Response (NFIR) systems. Therefore, the problem can be considered as a special case of learning a static function, and the formulation (3.3) remains valid. The remainder of the chapter is organized as follows. In Section 4.1, the proposed method is motivated via intuitions and examples, highlighting the first main contributions of this work. Section 4.2 defines the problem boundaries. Section 4.3 introduces the theoretical foundations and assumptions in order to apply the semi-supervised learning scheme. Section 4.5 suggests a method for generating additional input points. Section 4.4 presents the proposed semi-supervised system identification methodology. Section 4.6 shows the main results and comparison of the described method with respect to employing standard Tikhonov regularization in place of the proposed regularization term. Lastly, Section 4.7 is devoted to concluding remarks and future challenges.

4.1 Motivation

An humble attempt of this thesis is to collect, organize and show the main interlacements that system identification had (and still has) with machine learning. The author's opinion is that there is still much that the two

communities can give and share to each other, in both directions. The aim of this work is to foster and encourage, even if in a small portion, the dialog between the two research areas. This is the position that we took in Chapter 1. The present contribution born just beneath this line of reasoning. A main method that system identification borrowed from machine learning is the technique of regularization. In Chapter 2, we introduced this concept and the empirical Bayes method, highlighting the effectiveness of their combined use (the former technique to deal with the bias/variance tradeoff, the latter for hyperparameters' tuning). In Chapter 3, the nonparametric function learning approach, based on the RKHS framework, has been introduced. Originally developed for the static function learning case (Section 3.2), the methodology has been extended, and incorporated with relevant results, to the identification of dynamic systems (Section 3.3). Both contributions to the system identification worlds came from machine learning. The question that now arises naturally is: how much can still give the machine learning world to system identification? The author's belief is, as previously explained, a lot.

In order to introduce the methodology that is the main actor of this chapter, it is useful to recall the main machine learning fields of application. Broadly speaking, machine learning deals with four types of problems:

- **Supervised learning**[30, 4]: this is the problem formulation that we dealt with until now. The inputs and outputs to the unknown true function are given, and the aim is to estimate this function from the available dataset. We say that we have *labeled or supervised data*, since the outputs (the labels) are known. In the following, we will use the two definitions interchangeably. Supervised learning problems can be further decomposed into *classification* and *regression* ones. In the former case, the aim is to predict a category or a class, represented as categorical discrete variables; in the latter, the purpose is to predict continous real-valued data

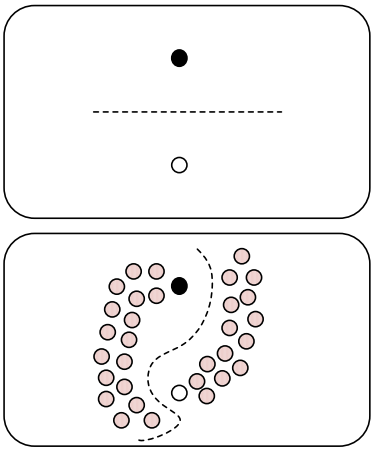
- **Unsupervised learning**[30, 4]: in this setting, only the inputs are given. The data are therefore *unlabeled or unsupervised*. In the following, we will use the two definitions interchangeably. Thus, it is not possible to induce a function. The purpose is to find patterns, clusters and relations between the input data

- **Semi-supervised learning**[51]: this type of problems is characterized by having both supervised and unsupervised data. A similar scenario

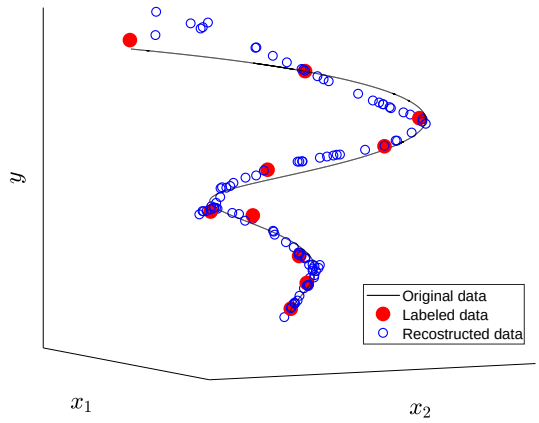
may occur when performing a measurement is costly or it is a destructive experiment (see [52] for a selection of practical examples on the topic). A variety of techniques have been developed to make use of the additional (not labeled) data, in order to represent a function which maps inputs to outputs

- **Reinforcement learning**[53]: this machine learning branch assumes that, as well as inputs and outputs, a *reward* (for having taken a specific action) is also available. The aim of the algorithm is then to learn a *policy*, that is, what actions to take in correspondence of certain inputs, in order to maximise the reward. This field shares many similarities with the classic theory of optimal control [54].

Supervised learning concepts, such as regularization and kernel methods, have found application in system identification. With the same spirit, by looking at what machine learning still has to offer, we can legitimately wonder *if* and *how* the presence of *inputs with no corresponding outputs* (i.e. unlabeled data) can be useful to the system identification purpose. This would translate the system identification problem from *supervised* to *semi-supervised*, opening for the inclusion of related techniques. The addition of more training points (although not supervised) can be beneficial in a small-data regime.



(a) **Classification**: Badly conditioned classification task using only two labeled points (above) and well conditioned problem exploiting additional unlabeled samples (below)



(b) **Regression**: learning a static function in presence of few labeled points and additional unlabeled regressors. The additional inputs lie in the $x_1 - x_2$ plane. The information about their disposition aids the reconstruction of the output y (blue), together with the points for which the y value is known (red)

Figure 4.1: Examples of semi-supervised learning of static systems

In the case of static systems, semi-supervised methods have already been employed for both classification [52] and regression [55] cases. This is sketched in Figure 4.1. In the classification example, the aim is to find a function (called *classifier*) which correctly separates the black dot from the white one. Seeing only this two points - top panel of Figure 4.1 (left) - a possible classifier would be the horizontal dotted line. Suppose now that additional *unlabeled* points are added (represented in grey in the bottom panel of Figure 4.1 (left). We do not know if they are black or white, but we do know how they are arranged in space. By exploiting the additional information provided by their placement, it is possible to induce a different (and more reasonable) classification bound. In the regression example, the aim is to learn the function that generated the labeled data. These are data that are at our disposal, and are such that both inputs (the vectors $[x_1, x_2]^T \in \mathbb{R}^{2\times 1}$) and outputs $y \in \mathbb{R}$ are known. When, in addition to the supervised data, other regressors (which lie in the $x_1 - x_2$ plane) are available (but without the corresponding output y), their disposition in the $x_1 - x_2$ plane gives additional information about how the unknown y could be, given that, for certain (supervised) points, the output y is known in advance. The algorithm for producing Figure 4.1(right) has been presented in [55, 56]. Here, however, the described method (which makes use of additional input data) is a *transductive learning* one. This means that its purpose is to correcly predict *only specific* test data. This is in contrast with the *inductive learning* reasoning, that tries to generalize from specific training examples to general rules (which are then applied to specific test cases). System identification belongs to this latter group, since the aim is to find a model that is able to describe the true system in a generic set of applications. There is therefore the need for a mathematical function, describing the relation between generic inputs and outputs variables. As an example of how the use of additional input data can be beneficial in the system identification case, consider the example of Figure 4.2. Here, we supposed that an input signal $u(t)$ has been passed through a stable LTI SISO system, generating the corresponding output signal $y(t)$. The filtering effect of the dynamic system is that of attenuating and delaying the input. Suppose now that the input is measured with a sampling frequency that is greater with respect to the output one. We would end up with more input data than output measurements, leading to a possible application of the semi-supervised concepts.

The method presented in this chapter, named *Semi-supervised System Identification* (SSI), gives a solution for the inductive semi-supervised learning

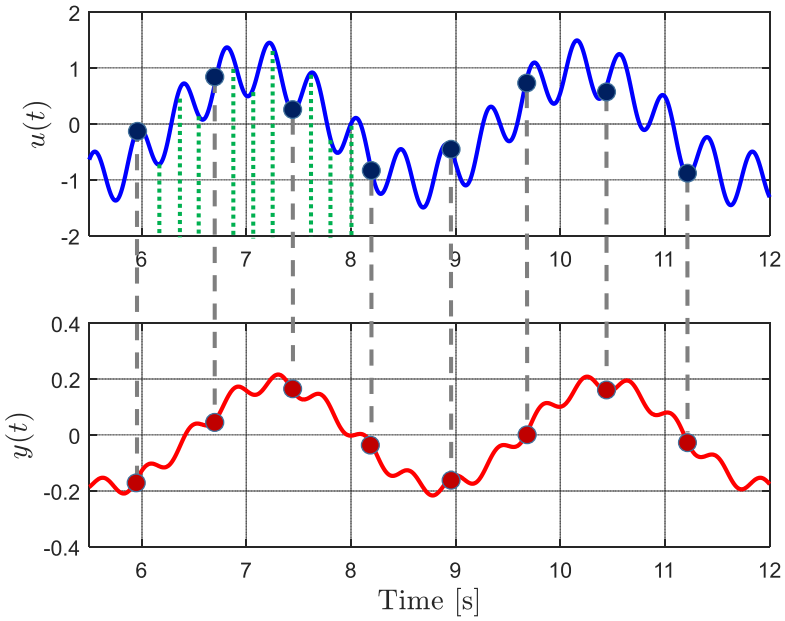


Figure 4.2: Example of obtaining a semi-supervised dataset in a dynamic scenario. The input signal (top) is sampled more frequently than the output one (bottom). This leads to having a set of supervised data (represented with dots and connected by the dashed grey line), and a set of unsupervised data (represented by the orange dashed vertical lines on the input signal)

problem. The aim is to identify the unknown true system $\mathcal{S}_0$ given a set of input-output data (supervised dataset), leveraging also on the information contained in a set of input-only measurements (unsupervised dataset). The prior information embedded in the distribution of the unsupervised dataset are employed in the form of an additional regularization term, called *manifold regularization*. This is in line with the standard Tikhonov regularization term, that can be interpreted, via a Bayesian treatment, as a prior information on the parameters' value (see Section 2.2). As a conseguence, Tikhonov regularization impacts the *global smoothness* of the learned function. We will show that the manifold regularization term implies, instead, the concept of *local smoothness*.

Since, in dynamic systems, the regressors may also contain the past output samples (not available in correspondence of the unlabeled input points), we will restrict our analysis to the case of Nonlinear Finite Impulse Response (NFIR) systems, leaving this extension to future research. The point here is that *this work is a first (and unique) step into carrying the semi-supervised framework into the system identification community*. The principal virtue of this contribution relies on the *implications* that this shipment could bring to the system identification community when, arrived at destination through a cargo truck full of goods, will be opened.

In this setting, it is appropriate to acknowledge that the application of a manifold regularization term in (inductive) static fuction learning has been faced in [57, 58, 59]. We built on these results, first by applying the methodology to the system identification case, and, afterwards, by introducing a new method for generating the unsupervised input data (without the need to perform an additional experiment).

4.2 Problem statement

Let the NFIR Single-Input Single-Output (SISO) model be defined as:

$$\mathcal{S}_0 : \; y\left(t+1\right) = g\big(\varphi\left(t\right)\big) + e\left(t\right), \tag{4.1}$$

where $y\left(t\right) \in \mathbb{R}$ denotes the system output, g is a nonlinear function, $\varphi\left(t\right) \in \mathbb{R}^{m\times 1}$ is the regression vector made of past samples of the input $u(t)$ such that $\varphi\left(t\right) = \left[u\left(t\right), \cdots, u\left(t-m+1\right)\right]^T$, and $e\left(t\right) \in \mathbb{R}$ is an additive white noise. From now on, m will be referred to as the *model order*. The objective of this work is to identify a system of type (4.1), assuming that m is known[1]. We suppose furthermore that two different datasets are available: a *supervised* data set $\mathcal{D}_S$ and an *unsupervised* one $\mathcal{D}_U$. The supervised dataset is such that:

$$\mathcal{D}_S = \left\{ \big(u_S\left(t\right), y\left(t\right)\big) \;\; | \;\; 1 \leq t \leq N_S \right\}, \tag{4.2}$$

where $u_S(t)$ is the input at time t, $y(t)$ is the output associated with the input $u_S(t)$, and N_S is the number of supervised data. The unsupervised dataset $\mathcal{D}_U$ is defined as:

$$\mathcal{D}_U = \left\{ \big(u_U\left(t\right)\big) \;\; | \;\; 1 \leq t \leq N_U \right\}, \tag{4.3}$$

where $u_U(t)$ is an input for which the associated output has not been measured, and N_U is the number of these unsupervised input measurements. Notice that the dataset $\mathcal{D}_S$ contains both input and output samples, while the dataset $\mathcal{D}_U$ consists of only input measurements. In order to obtain a more compact representation, it is useful to represent the observations and the regressors in matrix form. By using the supervised dataset $\mathcal{D}_S$, we obtain the output vector $Y \in \mathbb{R}^{N\times 1}$:

$$Y = \left[\begin{array}{ccc} y\left(m+1\right) & \cdots & y\left(N_S\right) \end{array} \right]^T, \tag{4.4}$$

[1] The knowledge of m is generally not available and an estimate is usually derived from the data. Since the issue is not trivial, this is postponed to future research.

which contains the observations $y(t)$ stacked in row, and $N = N_S - m$ is the number of outputs that it is possible to employ for the identification stage, given the model order m. In the same way, it is possible to construct the N *supervised* model's regressors as:

$$\varphi_S(t) = \begin{bmatrix} u_S(t) & \cdots & u_S(t-m+1) \end{bmatrix}^T \quad m \leq t \leq N_S - 1, \tag{4.5}$$

where $\varphi_S(t) \in \mathbb{R}^{m \times 1}$. The regressors' matrix $\Phi \in \mathbb{R}^{N \times m}$ can be defined as in (2.19), by stacking all supervised regressors $\varphi_S^T(t) \in \mathbb{R}^{1 \times m}$, leading to:

$$\Phi = \begin{bmatrix} \varphi_S^T(m) \\ \vdots \\ \varphi_S^T(N_S - 1) \end{bmatrix}. \tag{4.6}$$

Remember now that, in addition to the supervised dataset $\mathcal{D}_S$, we have at our disposal also the unsupervised dataset $\mathcal{D}_U$, containing only input samples. It is therefore possible to construct the model's regressors as in (4.5), by leveraging on $\mathcal{D}_U$. There are therefore $N_{rU} = N_U - m + 1$ available *unsupervised* model's regressors, each one of them defined as:

$$\varphi_U(t) = \begin{bmatrix} u_U(t) & \cdots & u_U(t-m+1) \end{bmatrix}^T \quad m \leq t \leq N_U, \tag{4.7}$$

where $\varphi_U(t) \in \mathbb{R}^{m \times 1}$. It is then possible to group all of these unsupervised regressors into an (unsupervised) regressors' matrix:

$$\Phi_U = \begin{bmatrix} \varphi_U^T(m) \\ \vdots \\ \varphi_U^T(N_U) \end{bmatrix}, \tag{4.8}$$

with $\Phi_U \in \mathbb{R}^{N_{rU} \times m}$.

Combining the input datasets, we can define the joint matrix, containing both supervised (4.5) and unsupervised (4.7) regressors, as:

$$\widetilde{\Phi} = \begin{bmatrix} \Phi \\ \Phi_U \end{bmatrix}, \tag{4.9}$$

where $\widetilde{\Phi} \in \mathbb{R}^{N_r \times m}$ and $N_r = N + N_{rU}$ is the total number of regressors, both supervised and unsupervised. From now on, for simplicity, the i-th row of $\widetilde{\Phi}$ and Y will be denoted as $\varphi(i)$ and $y(i)$, respectively.

The aim now is to *identify the system* $\mathcal{S}_0$ *by employing the information contained in* $\mathcal{D}_S$ *and* $\mathcal{D}_U$.

4.3 Manifold regularization

When can $\mathcal{D}_U$ be of some use into discovering the relation between inputs and outputs? This is the case if the marginal probability density $p(\varphi)$ which, we suppose, generates the inputs, happens to be informative about the conditional distribution $p(y|\varphi)$, describing the possible outputs values in correspondence of the input regressor φ [58]. We can make this possible by stating a specific assumption about the connection between the marginal and the conditional distributions [52]:

Assumption 4.3.1 ***Semi-supervised smoothness assumption***
If two regressors $\varphi(i)$ *and* $\varphi(j)$ *in a high-density region are close, then so should be their corresponding outputs* $y(i)$ *and* $y(j)$.

Assumption 4.3.1, therefore, constrains the solution to be smooth with respect to the *manifold* onto which the regressors lie. This can be enforced by a *proper regularization term*, that should reflect the intrinsic structure of $p(\varphi)$. This term has a different taste with respect to the standard Tikhonov one, that instead, enforces a *global smooth behaviour* to the unknown function.

One of the first attempts to formalize Assumption 4.3.1 has been taken in [60]. Here, we take the opposite path: our aim is to define a regularization term which, in turn, enforces Assumption 4.3.1, without a specific set of choices and ad-hoc definitions. In this view, a possible choice for the manifold regularization term has been advocated in [58] as:

$$S_g = \int_{\mathcal{G}} \|\nabla g(\varphi)\|^2 \cdot p(\varphi)\, d\varphi, \tag{4.10}$$

where $\mathcal{G} \subseteq \mathbb{R}^{m\times 1}$ is the regressor space and $p(\varphi)$ denotes the probability density function of the regressors defined over $\mathcal{G}$. The main idea behind the manifold regularization rationale considered here is that, if Assumption 4.3.1 holds, the gradient of g, and so S_g, must be small. Then, minimizing S_g with respect to model parameters on missing output labels is a way to enforce Assumption 4.3.1.

In the standard supervised learning approach, the information about the input distribution $p(\varphi)$ is rarely used. This is the case because, most of the times,

$p(\varphi)$ is unknown and the smoothness index S_g cannot be computed exactly. It turns out that S_g can be approximated using the *regressor graph* [57, 58]. This is a weighted complete graph with the (supervised and unsupervised) regressors as its vertexes, and the weight of each edge defined as:

$$w_{i,j} = e^{-\frac{\|\varphi(i)-\varphi(j)\|^2}{2\sigma_e{}^2}}, \tag{4.11}$$

where $\sigma_e \in \mathbb{R}$ is a tuning parameter. Now, let's consider the Laplacian matrix $\widetilde{L} = D - W$, where $D \in \mathbb{R}^{N_r \times N_r}$ is the diagonal matrix with elements $D_{ii} = \sum_{j=1}^{N_r} w_{i,j}$, and $W \in \mathbb{R}^{N_r \times N_r}$ is the matrix composed by the weights $w_{i,j}$. A higher value of (4.11) indicates that two regressors are similar. This rationale derives from the Laplacian Eigenmaps algorithm [61]. It can be shown that [58]:

$$S_g \simeq \widetilde{Y}^T \cdot \widetilde{L} \cdot \widetilde{Y} \tag{4.12}$$

where $\widetilde{Y} = \left[\widetilde{y}(1), \cdots, \widetilde{y}(N_r)\right]^T = \left[g\big(\varphi(1)\big), \cdots, g\big(\varphi(N_r)\big)\right]^T \in \mathbb{R}^{N_r \times 1}$ contains the *noiseless* outputs, corresponding to *both supervised and unsupervised input regressors*. In order to obtain the approximation of S_g (4.12), *only the input regressors are needed*. Thus, both supervised and unsupervised regressors can be employed for this purpose. Notice that Y differs from $\widetilde{Y}$, since the former is a *noisy* vector of N observations, while the latter is a *noiseless* vector of N_r observations.

It is interesting to observe that the same problem structure (4.12) is shared by other manifold learning methods, although they do not use $\widetilde{L}$, but a different symmetric matrix [62]. The reason of this fact is that such manifold learning algorithms are still based on Assumption 4.3.1, but they interpret it from different perspectives.

4.4 The semi-supervised approach

In this work, we will consider the realistic case where g is unknown and therefore no prior parameterization is available. A powerful tool for dealing with such challenges is the framework of the Reproducing Kernel Hilbert Spaces (RKHS), presented in Section 3. Therefore, a kernel-based nonparametric approach, based on RKHS, is proposed. The method embodies the notion of manifold regularization, in order to take advantage of the presence of unsupervised data points.

Suppose now that g belongs to a RKHS $\mathcal{H}$ defined using the kernel K.

The tipical variational formulation, involving the Tikhonov regularization term $\|g\|_{\mathcal{H}}^2$, has been presented in (3.3). In order to include information about the local smoothness of the function (leveraging on the unsupervised data points), it is meaningful to add the manifold regularization term (4.12) to (3.3), leading to the following modified variational problem [58]:

$$\hat{g} = \underset{g \in \mathcal{H}}{\arg\min} \sum_{t=1}^{N} \Big(y\left(t\right) - g\big(\varphi\left(t\right)\big) \Big)^2 + \lambda \cdot \|g\|_{\mathcal{H}}^2 + \lambda_M \cdot \widetilde{Y}^T \cdot \widetilde{L} \cdot \widetilde{Y}, \tag{4.13}$$

where $\lambda_M \in \mathbb{R}$ is an hyperparameter which controls the weight given to the manifold regularization term.

It is possible to show that the representer theorem 3.2.1 still holds for the cost function (4.13), and the solution is given by considering all N_r regressors, both the N supervised and the N_{rU} unsupervised ones [58]:

$$\hat{g}\big(\varphi(t)\big) = \sum_{s=1}^{N_r} \tilde{c}_s K\big(\varphi\left(t\right), \varphi\left(s\right)\big) = \sum_{s=1}^{N_r} \tilde{c}_s K\big(\varphi\left(s\right), \varphi\left(t\right)\big) \tag{4.14}$$

$$= \sum_{s=1}^{N_r} \tilde{c}_s r_{\varphi(s)}\big(\varphi\left(t\right)\big), \tag{4.15}$$

for some N_r-tuple $\tilde{c} = \left[\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_{N_r}\right]^T \in \mathbb{R}^{N_r \times 1}$.

The formulation (4.15) can be expressed in a more compact way by introducing the kernel matrix $\widetilde{\mathcal{K}} \in \mathbb{R}^{N_r \times N_r}$ that spans all available N_r regressors:

$$\widetilde{\mathcal{K}} = \begin{bmatrix} K\big(\varphi\left(1\right), \varphi\left(1\right)\big) & \cdots & K\big(\varphi\left(1\right), \varphi\left(N_r\right)\big) \\ \vdots & \ddots & \vdots \\ K\big(\varphi\left(N_r\right), \varphi\left(1\right)\big) & \cdots & K\big(\varphi\left(N_r\right), \varphi\left(N_r\right)\big) \end{bmatrix}, \tag{4.16}$$

Notice that (4.16) differs from the kernel matrix $\mathcal{K} \in \mathbb{R}^{N \times N}$ introduced in Section 3.2, which considers only the N supervised points. The vector $\widetilde{Y}$ introduced in (4.12) can the be rewritten as:

$$\widetilde{Y} = \widetilde{\mathcal{K}}\tilde{c}. \tag{4.17}$$

The solution to the general problem, that depends only on the unknown vector $\tilde{c} \in \mathbb{R}^{N_r \times 1}$, can be found by substituting (4.17) in (4.13) as in problem (3.9):

$$\widehat{c} = \underset{\tilde{c} \in \mathbb{R}^{N_r}}{\arg\min} \left\| \begin{bmatrix} Y \\ 0_{N_{rU}} \end{bmatrix} - P \cdot \widetilde{\mathcal{K}}\tilde{c} \right\|_2^2 + \lambda_T \cdot \tilde{c}^T \widetilde{\mathcal{K}} \tilde{c} + \lambda_M \cdot \tilde{c}^T \cdot \widetilde{\mathcal{K}} L \widetilde{\mathcal{K}} \cdot \tilde{c}. \tag{4.18}$$

In order to proper evaluate the effect of the new introduced regularization term (4.12), we will suppose from now on that the Tikhonov term in the cost function (4.13) is set to zero, leading to the following purely semi-supervised formulation:

$$\hat{g} = \underset{g \in \mathcal{H}}{\arg\min} \sum_{t=1}^{N} \Big(y\,(t) - g\big(\varphi\,(t)\,\big) \Big)^2 + \lambda_M \cdot \widetilde{Y}^T \cdot \widetilde{L} \cdot \widetilde{Y}. \tag{4.19}$$

The solution of (4.19) can be therefore simply expressed as:

$$\widehat{c} = \underset{\tilde{c} \in \mathbb{R}^{N_r}}{\arg\min} \left\| \begin{bmatrix} Y \\ 0_{N_{rU}} \end{bmatrix} - P \cdot \widetilde{\mathcal{K}} \tilde{c} \right\|_2^2 + \lambda_M \cdot \tilde{c}^T \cdot \widetilde{\mathcal{K}} \widetilde{L} \widetilde{\mathcal{K}} \cdot \tilde{c}, \tag{4.20}$$

where $0_{N_{rU}} \in \mathbb{R}^{N_{rU} \times 1}$ is column vector of all zeros. The matrix $P \in \mathbb{R}^{N_r \times N_r}$ permits to select only the elements of $\widetilde{\mathcal{K}}$ that contribute to explain the N supervised data points, such that:

$$P = \begin{bmatrix} I_N & 0 \\ 0 & 0 \end{bmatrix}. \tag{4.21}$$

Since (4.20) is now quadratic in $\tilde{c}$, its minimization can be carried out analytically, just as in (3.9). The minimizer of (4.20), therefore, can be found by solving the linear system:

$$\left[P \cdot \widetilde{\mathcal{K}} + \lambda_M \cdot \widetilde{L} \cdot \widetilde{\mathcal{K}} \right] \cdot \widehat{c} = \begin{bmatrix} Y \\ 0_{N_{rU}} \end{bmatrix}. \tag{4.22}$$

Notice that the formulation of the solution came in a form similar to the standard Tikhonov regularization problem (3.10), with the difference that, now, the unsupervised points contribute to the overall estimated function via the matrix $\widetilde{\mathcal{K}}$.

It is now interesting to show a comparison between searching the unknown function, following formulation (4.13), where, respectively, only the Tikhonov or the manifold regularization term is considered. We consider, for clarity, a static function estimation problem, from a finite set of supervised and unsupervised points, using the RKHS framework. The true unknown function $g(x)$ presents a discontinuity point at $x = 0$, in order to show the different properties of smoothness that the two regularization terms induce. The Tikhonov term enforces a global smooth behaviour, while the manifold term strives for local smoothness. The employed kernel is the Gaussian kernel $K(x, z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}}$.

Figure 4.3 shows the results of a regularization network that employs only the Tikhonov regularization as in (3.9), for different values of λ and σ. In this case, the unsupervised points are of no use, and therefore are not depicted. When $\lambda = 0$, also the Tikhonov term is absent, and the estimated function interpolates each one of the supervised points. Choosing a low value of σ, we are defining a function space that admits also non-smooth functions (see Examples 3.1(iii) and 3.2(i)). Because of this, the learned function is composed by a series of sharply peaked Gaussians, centered at the observed points. This is in line with the definition given by the representer theorem (3.2.1), given that, in the case of a Gaussian kernel, the representers (or kernel's sections) are still Gaussian functions. As σ grows, the estimated function gets smoother, fitting worse and worse the high variation regions of the true underlying function. The effect of the regularization hyperparameter λ is that of weighting the regularization and the error cost function. With high values of λ, the estimated function tends to the zero one: this is in line with the parametric approach, where a high λ value makes all parameters' estimates null. In all of these cases, given the global nature of the imposed regularization, the estimated function fails to approximate well the discountinuity region.

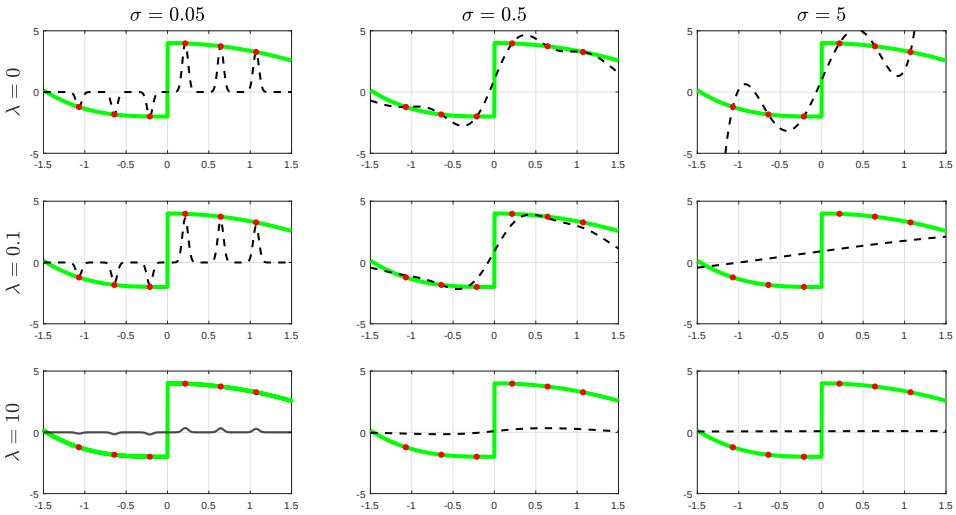


Figure 4.3: Example of hyperparameters' sensitivity when employing only the Tikhonov regularization term. The plots depict the true unknown function (solid green line), the supervised data (red dots), and the estimated function (dotted black line)

The function's estimation example using only the manifold regularization term is depicted in Figure 4.4. Here, we suppose that unsupervised points are available in a neighbourhood of the discontinuity point. The method should

not regularize in this region, in order to allow non-smooth (rapid) variation of the estimated function, and should enforce, instead, smoothness elsewhere. By choosing an appropriate low value of σ, it is possible to fit the function even in the discountinuity region, being not sensible to the variation of λ_M. High values of σ makes the estimate smoother, just like as λ controlling the Tikhonov regularization. Increasing λ_M makes the function as smooth as possible: in this case, this means that the manifold regularization term is weighted much. This, in turns, translates into making each domain point similar to the other, and the estimated function reduces to the mean of the supervised points.

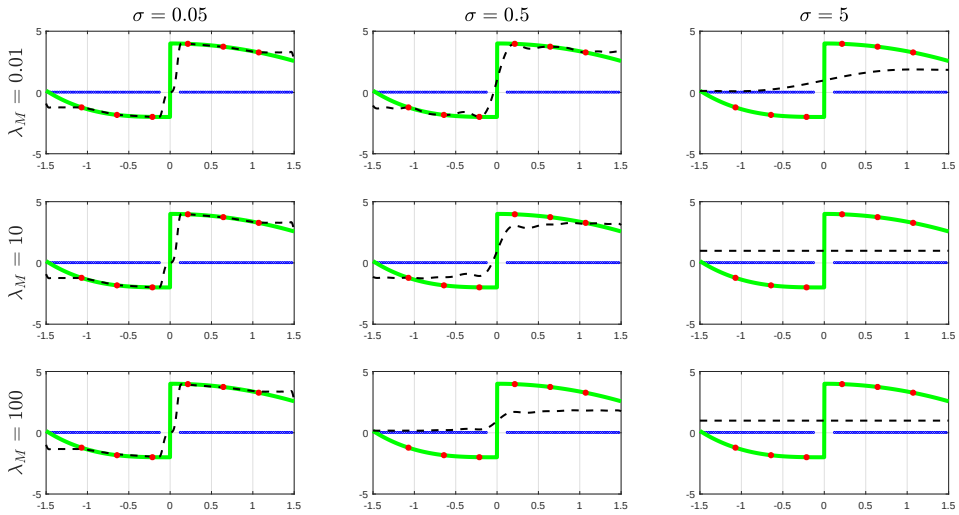


Figure 4.4: Example of hyperparameters' sensitivity when employing only the manifold regularization term. The plots depict the true unknown function (solid green line), the supervised data (red dots), the unsupervised data (blu dots), and the estimated function (dotted black line). The hyperparameter σ_e is fixed to 0.01

4.5 Unsupervised inputs selection

As state in Section 4.1, unlabeled inputs reveal the manifold where the regression variables typically take value. By using the proposed semi-supervised regularization, identification is tuned so as to fit well the data-generation mechanism on such manifold. The use of artificial data to induce regularization is not new. Ridge regression estimates can be obtained by ordinary least squares regression on an augmented dataset, where generated response values are set to zero. In this way, the fitting procedure is forced to shrink also the coefficients toward zero [4]. Authors in [63] formalized the Vicinal Risk

Minimization (VRM) principle. Here, additional virtual examples can be drawn from a defined vicinity distribution of the training examples, to enlarge the support of the training distribution. In this work the authors showed how, using the VRM approach, one can obtain the regularized Ridge regression and Support Vector Machine (SVM) solutions [4],. This method is currently often applied for training deep neural networks, in particular when performing image classification. Infact, it is common to define the vicinity of one image as the set of its horizontal reflections, slight rotations, and mild scalings [64]. In this setting, a recent data augmentation technique has been introduced to alleviate overfitting problems and sensitivity to adversarial examples [65]. A related idea has been presented by [66], where model constraints are implemented by adding artificial data examples that satisfy them. Differently from previously cited methods, here the learning "hints" can be designed by relying only on the independent variables.

In real-world identification of dynamic systems, contrary to the standard semi-supervised problems encountered in machine learning, the unsupervised data set $\mathcal{D}_U$ may not be a problem input as in Figure 4.4, but, instead, a design parameter. In some cases, $\mathcal{D}_U$ may contain some input time series which are likely to excite the system dynamics in future operating conditions (when the model will be used). More often, since this additional data set affects the model quality, $\mathcal{D}_U$ could be chosen to enforce Assumption 4.3.1 to be true. Notice that to obtain such an additional data set, *it is not required to run a new experiment on the plant*. Following Figure 4.4, if the discontinuity region would be known, a possible unsupervised points generation method could be to generate the additional inputs as in the example. If the discontinuity region is not known, then, a more general method has to be devised for the generation of $\mathcal{D}_U$.

Before discussing the choice of $\mathcal{D}_U$, notice that Assumption 4.3.1 requires only that, inside the same high density region, the regressors have a similar corresponding output, namely that their difference is "small". For this reason, the proposed method will generate the unsupervised regressors in the neighborhood of the supervised ones, where, if the system is smooth enough, they should have a similar corresponding output. This approach will generate a regressors set similar to the one shown in Figure 4.5, where it is possible to notice the presence of N_S regions, containing a supervised regressor and some unsupervised ones.

The algorithm used to select $\mathcal{D}_U$ is indicated next. Let $\mathcal{D}_U$ be composed of

p unsupervised datasets $\mathcal{D}_U^i$, $i = 1, \ldots, p$ as:

$$\mathcal{D}_U^i = \left\{ \left(u_U^i \left(t \right) \right) \quad | \quad 1 \leq t \leq N_S \right\} \tag{4.23}$$

where $u_U^i \left(t \right) = u_S \left(t \right) + v^i(t)$, $v^i(t)$ is a random variable and p is a tuning knob of the method. Each one of the p new (unsupervised) datasets contain therefore exactly N_S unsupervised input regressors, see Figure 4.5.

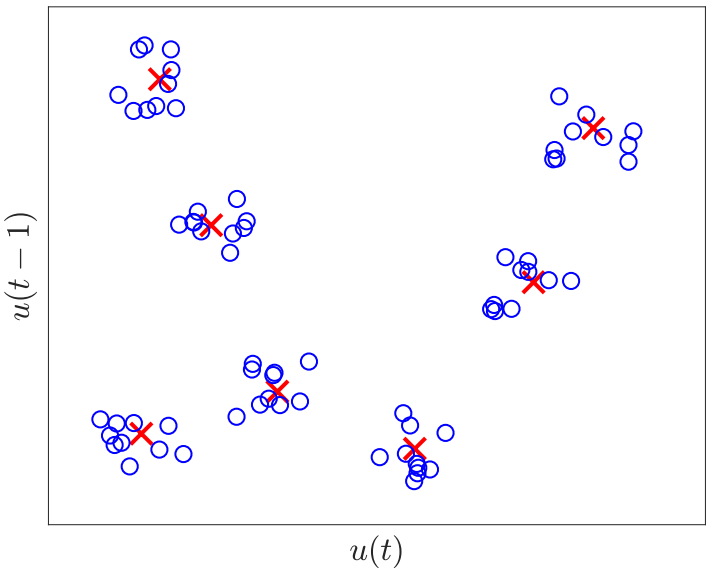


Figure 4.5: An example of unsupervised regressors' selection, for a system with $m = 2$ using $p = 10$. The plot represents the supervised regressors (red crosses) and the unsupervised regressors (blue circles)

The value of $v^i(t)$ determines the distance of the p unsupervised points with respect to the supervised one (proportional to the area of the regressors' regions): therefore, it has to be small enough to guarantee that the system output does not vary significantly inside these regions. The choice of $v^i(t)$ will be discussed later.

From such p datasets, it is possible to determine the quantities defined in Section 4.2. Since the unsupervised points are generated in correspondence of the supervised ones, we have N employable unsupervised regressors for each one of the p datasets. This leads to a total of $N_{rU} = p \cdot N$ unsupervised regressors $\varphi_U^i(t) \in \mathbb{R}^{m \times 1}$, $i = 1, \ldots, p$. Each one of them is such that, according to (4.7):

$$\varphi_U^i \left(t \right) = \left[\begin{array}{ccc} u_U^i \left(t \right) & \cdots & u_U^i \left(t - m + 1 \right) \end{array} \right]^T \quad m \leq t \leq N_S - 1. \tag{4.24}$$

From the unspervised regressors computed in (4.24) using the i-th dataset, it is possible to form the i-th unsupervised regressors' matrix $\Phi_U^i \in \mathbb{R}^{N \times m}$ as in (4.8):

$$\Phi_U^i = \begin{bmatrix} \left(\varphi_U^i(m)\right)^T \\ \cdots \\ \left(\varphi_U^i(N_S - 1)\right)^T \end{bmatrix}. \tag{4.25}$$

The complete (unsupervised) regressors' matrix $\Phi_U \in \mathbb{R}^{N_{rU} \times m}$ can there be therefore composed by stacking the matrices (4.25), $i = 1, \ldots, p$:

$$\Phi_U = \begin{bmatrix} \Phi_U^1 \\ \vdots \\ \Phi_U^p \end{bmatrix}. \tag{4.26}$$

A reasonable criterion for the selection of the random variable $v^i(t)$ is to consider that the regions should not mix with each other, since this would lead to non-smooth functions (e.g., with jumps in certain points). It is then useful to introduce a tuning parameter α, allowing to regulate the regions' maximum area, and that highlights if the regions mix or not. In particular, in the method indicated next, $\alpha = 1$ corresponds to the threshold between mixed regions (achieved using $\alpha < 1$) and completely distinct regions ($\alpha > 1$). In order to use α, it is necessary to define a distribution of $v^i(t)$ that depends on α and guarantees the aforementioned properties. A possible way is to use a uniform distribution:

$$v^i(t) \sim \mathrm{U}(-h, h) \quad 1 \leq t \leq N_S, \ i = 1, \ldots, p \tag{4.27}$$

where $h > 0$ determines the area of the unsupervised points regions. To impose distinct regions, the following inequalities must hold:

$$\left\|\varphi_U^i(t) - \varphi_S(t)\right\| \leq \frac{d}{2} \quad m \leq t \leq N_S - 1, \ i = 1, \ldots, p \tag{4.28}$$

where d is the distance between the two closest supervised regressors. After some computations, it can be shown that (4.28) can be written as:

$$\sum_{j=1}^{m} \left(v^i(t - j + 1)\right)^2 \leq \left(\frac{d}{2}\right)^2 \quad m \leq t \leq N_S - 1, \ i = 1, \ldots, p \tag{4.29}$$

Since $|v^i(t - i + 1)| \leq h$ (it is generated from the random variable (4.27)), the

inequalities (4.29) hold if:

$$\sum_{j=1}^{m} h^2 \leq \left(\frac{d}{2}\right)^2 . \tag{4.30}$$

Recalling that $h \geq 0$, we have that (4.30) corresponds to:

$$h \leq \frac{d}{2\sqrt{m}} \tag{4.31}$$

The condition described in (4.31) imposes a constraint for h to maintain N_S distinct regions. To make such a constraint more or less conservative, it is possible to use α, for examples, as follows:

$$h = \frac{d}{2\alpha\sqrt{m}} \tag{4.32}$$

4.6 Results and discussion

In this section, a numerical example is provided to show the effectiveness of the Semi-Supervised Identification algorithm, presented in the previous sections, that employs the manifold regularization term as in (4.19). The approach is compared with the standard formulation (3.3), where the Tikhonov regularization is considered. We employed the Gaussian kernel to estimate the second order ($m = 2$) NFIR system:

$$\begin{aligned} y(t) =& 1.432 \cdot u(t)^2 + 1.564 \cdot u(t-1)^2 + 3.234 \cdot u(t)\, u(t-1) + \\ &+ 2.145 \cdot u(t)^3 + 3.432 \cdot u(t)^2\, u(t-1) + \\ &+ 2.745 \cdot u(t)\, u(t-1)^2 + 1.034 \cdot u(t-1)^3 . \end{aligned} \tag{4.33}$$

The supervised dataset $\mathcal{D}_S$ generated from (4.33) is composed by very few points, namely $N_S = 15$ measures, corrupted by a Gaussian white noise input of zero mean, unitary variance and signal to noise ratio of 15 dB. The problem is then badly conditioned and is well suited for testing the proposed methodology. The unsupervised input dataset $\mathcal{D}_U$ has been generated according to Section 4.5.

The hyperparameters of the Tikhonov regression method (3.3) are:

1. λ: the regularization coefficient

2. σ: shape parameter of the Gaussian kernel

The hyperparameters of the manifold regression method (4.19) are:

1. λ_M: the regularization coefficient
2. σ: shape parameter of the Gaussian kernel
3. σ_e: shape parameter of the Laplacian Eigenmaps
4. α: the parameter controlling the area of the generated unsupervised points
5. p: the number of additional unsupervised datasets

In order to tune the respective hyperparameters of the methods, an additional supervised dataset $\mathcal{D}_V$ of $N_V = 1000$ points has been generated in the same way as $\mathcal{D}_S$. This has been done in order to assess the method capability and value. For obvious reasons, $\mathcal{D}_V$ should not be available, otherwise the problem is no more ill conditioned, and a standard least-squares approach should be pursed. The empirical Bayes method, introduced in Section A.3, could be employed to estimate the hyperparameters without resorting to $\mathcal{D}_V$. The considered hyperparameters are reported in Table 4.1.

In order to assess the overall performance of the estimation methods, a supervised testing dataset $\mathcal{D}_T$ of $N_T = 10000$ points is employed, generated analogously to $\mathcal{D}_S$. Using $\mathcal{D}_T$ it is possible to evaluate the NRMSE (Normalized Root-Mean Square Error) fitness metric:

$$\text{Fit} = 100 \cdot \left(1 - \frac{\left\| Y_T - \widehat{Y} \right\|}{\| Y_T - \overline{y}_T \cdot \mathbf{1} \|} \right), \tag{4.34}$$

where $\widehat{Y}$ is the vector of the estimated test outputs using the test inputs, Y_T is the true test outputs vector, $\overline{y}_T$ is the mean of Y_T and $\mathbf{1}$ denotes a vector of ones of suitable dimension.

In particular, a Montecarlo test using $N_M = 100$ sets of measures (with different random initializations) is proposed, to show the statistical significance of the method. The notched boxplots in Figure 4.6 depicts that SSI significantly outperform the Tikhonov regularization method, showing a significant difference in the medians. The SSI method, in addition, exhibits a lower estimation variance. In Figure 4.7, one of the 100 realizations is chosen to show how the function estimates evolves in time domain. Again, the SSI estimation is clearly

Table 4.1: Values of the tuning parameters for the nonparametric approaches in the numerical example.

Tikhonov regression	
λ	100 evenly spaced values in $[0, 10^{-3}]$
σ	$3, 6, 10, 11$
Semi-supervised System Identification	
λ_M	100 evenly spaced values in $[10^{-6}, 10^{-1}]$
σ	$3, 6, 10, 11$
σ_e	$0.1, 1, 10, 100$
α	100 evenly spaced values in $[1, 17]$
p	$2, 3, 4$

better than the standard Tikhonov regularization, especially for estimating the peaks of the true system output.

Given the numerical results, the belief is that they clearly show the potential of the semi-supervised approach for nonlinear system identification with respect to state of the art techniques. The price to pay is the fact that, e.g., compared to Tikhonov regression, three additional knobs need to be tuned, namely σ_e, α, p. However, notice that they are characterized by a clear physical interpretation: σ_e controls how much far two points can be considered similar or "connected" (if $\sigma_e \to \infty$ the result is an adiacency graph); p (that is, N_{rU}/N) indicates the relevance of the prior smoothness assumption over the measured data, while α represents a degree of smoothness. Therefore, they can be reasonably tuned with some (mild) prior information on the system dynamics. Moreover, simulations showed that, at least for the considered example, the performance are not sensitive to a fine tuning of such parameters.

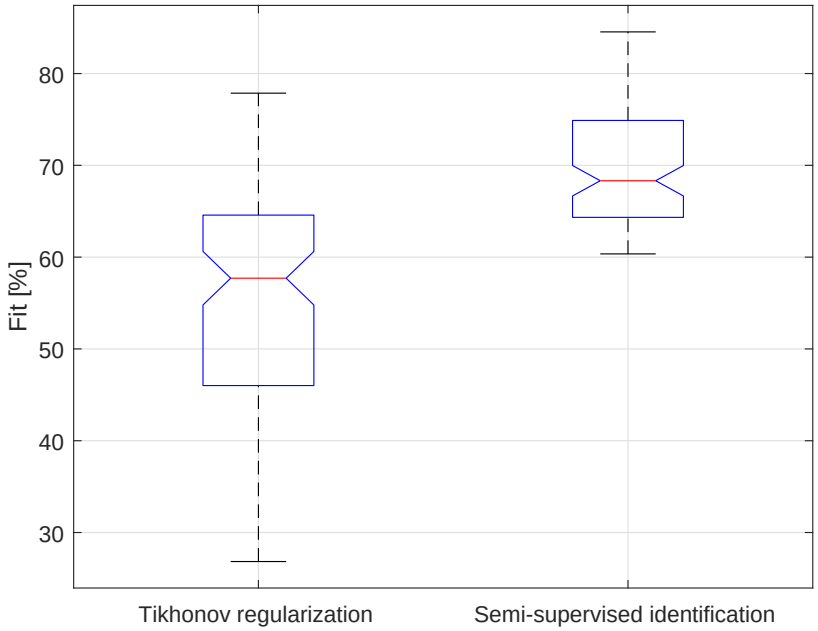


Figure 4.6: Comparison between Tikhonov regression and Semi-Supervised Identification in terms of the NRMSE measure of fitness. The boxplots represent a total of 100 different simulation and estimation trials

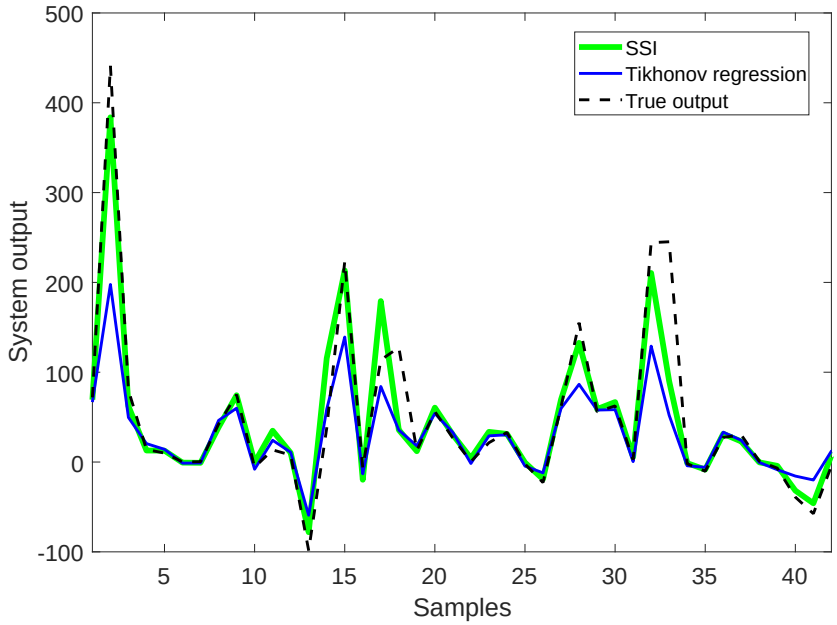


Figure 4.7: Comparison between Tikhonov regression and Semi-Supervised Identification in terms of the time behaviour of one random data realization

4.7 Conclusions and future developments

In this work, a semi-supervised learning approach suited to nonlinear dynamical systems has been developed. The method applies to NFIR models, namely those models where output measurements are not needed for manifold regression, and turns out to be equivalent to a weighted regularization network. The approach has been shown to outperform the statistical performance of Tikhonov regularization, when the additional unsupervised data set is selected as indicated by the method. Future research work will be dedicated to the extension of the semi-supervised paradigm to auto-regressive models and to a comparison with more complex regularization techniques. The estimation of the hyperparameters is still an open problem that have to be tackled. Another possible extension is the ability to put arbitrary constraints to the learned function behaviour. As stated in the beginning of the chapter, this work should be seen also in the light of a first tentative to introduce into system identification the (previously overlooked) framework of semi-supervised learning, which is very active in the machine learning community. The aim is, as previously noticed, to encourage a cross-fertilization between the two research areas.

Part II

Applications of statistical learning methods

CHAPTER 5

Health On Line Monitoring for Electromechanical actuator Safety

The second part of this thesis presents some applications of statistical learning methods to solve practical control and estimation problems. These case studies constitutes the *second contribution* of this work.

The first application regards the development of a health monitoring sofware for mechanical fault detection of Electro-Mechanical Actuators (EMA), employed in the aerospace industry. The HOLMES project was funded by the CleanSky Joint Technology Initiative, within the 7-th Framework Program (FP7) of European Union (EU). Together with project's partners, a massive experimental activity has been carried out, by means of a test bench, on an airliner EMA, equipped with a ballscrew transmission. Several degrees of a specific type of fault have been injected on the actuator, and experimental data were collected. Following chapters will present both a model-based and a data-driven method applied to the HOLMES project. A third application to this fault detection challenge regards an innovative improvement to a known clustering algorithm, known as Principal Direction Divisive Partitioning (PDDP).

This chapter introduces the HOLMES project, reviewing briefly in Section 5.1 the state of art of fault detection on EMA in aerospace environments, and describing its experimental setup in Section 5.2.

5.1 Fault detection and EMA

The development of a More Electrical Aircraft (MEA) is a technological transition applied for almost all the systems in aircrafts and helicopters. In such context, the implementation of Electro-Mechanical Actuators (EMAs) has increased rapidly during the last years [67]. Mechanical systems deployed in aerospace environments require constraints on weigth and robustness. When no hardware redundancy can be afforded, for safety reasons, an actuator must be equipped with a sophisticated diagnostic, prognostic, and recovery system. The monitoring of mechanical components for Fault Detection and Isolation (FDI) purposes is nowadays well known in literature [68].

A fault is defined as a not allowed deviation of at least one system property or parameter, with respect to its nominal operating behaviour [69]. Examples of such malfunctions are the jam of an actuator, a sensor contact loss, or a disconnection of a system component. These faults can lead to unsatisfactory performance, destabilize the process and possible catastrophical events. Fault detection and diagnosis systems implement the following tasks [70]:

- **Fault Detection**: the indication that something deviates from nominal system behaviour
- **Fault Isolation**: the determination of the fault location
- **Fault Identification**: the quantification of the fault magnitude.

The isolation and identification tasks together are referred to as *fault diagnosis*.

Fault detection and diagnosis methods are usually classified into *model-based* and *model-free* ones, see [71] and successive works for comprehensive reviews. Most model-based failure detection and isolation methods rely on the idea of analytical redundancy [72]. In contrast to physical redundancy, where a deviation is detected via comparison of different sensors readings, in the latter method sensory measurements are compared to analytically obtained values of the respective variable. The physical sensor is then replaced by a virtual one. This can be beneficial in some applications. The aerospace industry, as an example, places hard constraints on installed equipment's volume and weight. A general scheme, introduced by [73], highlights the typical components of a model-based fault detection methodology. Faults of different nature can affect both actuators, processes and sensors. Based on system inputs $u(t)$ and outputs $y(t)$, a model of the process under control can then be identified. The feature generation step can be accomplished by one of three different methods:

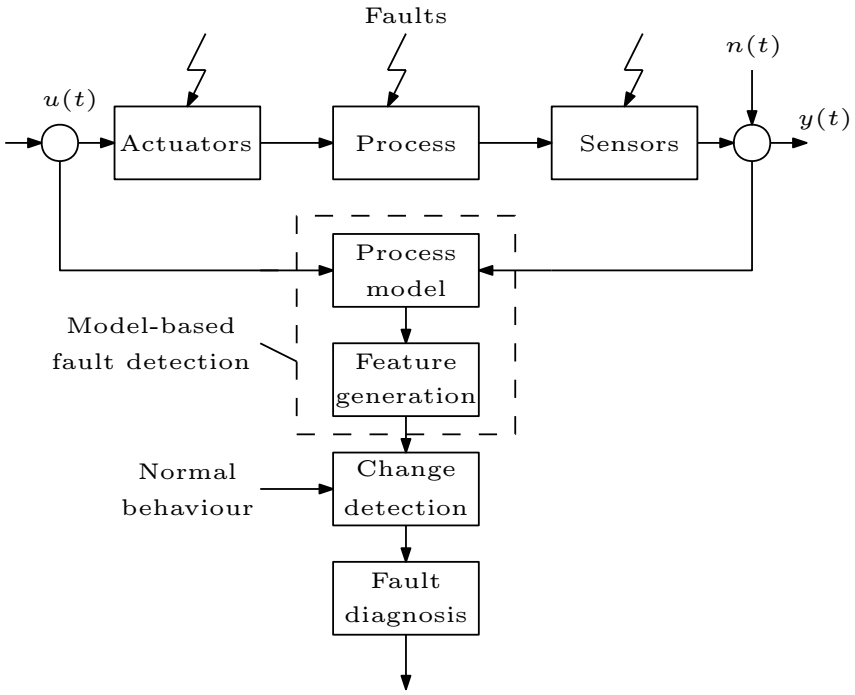


Figure 5.1: Classical model-based fault detection scheme

1. **Parameter estimation**: the system parameters are computed online with recursive identification algorithms. When the estimated values deviate from the starting ones, a fault is detected. This method is suitable for multiplicative faults, which can be modeled as parameter changes [74]

2. **State estimation**: the state of the system is estimated via an observer equipped with the nominal model. In this case the detection can be done tracking the behaviour of the state variables and of the observer innovations [75]

3. **Parity equations**: residuals are computed as $z(t) = y(t) - G(s)u(t)$, where $u(t)$ is the system input, $y(t)$ the measured output and $G(s)$ the process transfer function. These are then subjected to a linear transformation, in order to obtain the desired fault detection and isolation properties [76]. The state estimation and the parity equation methods are suitable for additive faults, that influence a variable by the addition of a fault entity.

Seminal works and surveys can be found in [77, 78], while [79] provides guidelines for a practical use of model-based fault detection and isolation methods. Previous work on model-based fault detection on EMA in aerospace can be

found in [80, 81]. The former paper tested various types of mechanical (spalling on raceway, actuator jam) and sensors faults, using a small scale test bed which can be mounted on an aircraft. On the contrary, tests performed during this project are based on a full scale 1:1 scale actuator. In the latter work, authors focused on simulating failures on transmission gears and bearings. Instead, in this work, real faults were injected on an EMA's ballscrew transmission spheres.

Regarding the recent employment of data-driven methods in the context of electrical motors, in [82] a robust diagnosis technique is presented by iteratively analyzing the patterns of multiple fault signatures in a motor current signal. Similarly, [83] adopts kernel density estimation to evaluate the probability density function of each healthy motor and motor stator fault. A review of pattern recognition techniques for fault detection is given in [84], whereas a panoramic view of features and classifiers for fault diagnosis is employed in [85]. For a combination of model-based and model-free approaches on EMAs in the avionics world, see [86].

The injected faults, test bench structure, experimental campaign and measured data of the HOLMES project are described in the next section.

5.2 Experimental setup

The test rig is designed to represent the actuator of primary flight surfaces of a wide-body airliner. As depicted in Figure 5.2, the test rig is composed by the main EMA under test, equipped with a ballscrew transmission.

The motor consists of a five phases brushless DC motor. This configuration is intrinsically fault tolerant; infact, the motor performance does not decrease even when two phases are open. A nut containing the recirculating spheres moves axially over the screw, transforming the rotation into a linear movement. The EMA position is closed-loop controlled, ranging from 0 mm (home position) to about 400 mm (fully extended). A hydraulic cylinder, modeled in [87], is used to simulate the force which the EMA has to overcome during its motion (mainly due to aerodynamic forces). A load cell is used to close the force control loop. The nut under test has two recirculating circuits, with 80 balls per channel that alternates between steel and ceramic ones, see Figure 5.3.

The fault investigated in this work consists of the damage undergone by the the steel spheres, at different damage severity levels. This fault was chosen for investigation after a Fault Tree Analysis (FTA) and a Failure Mode and

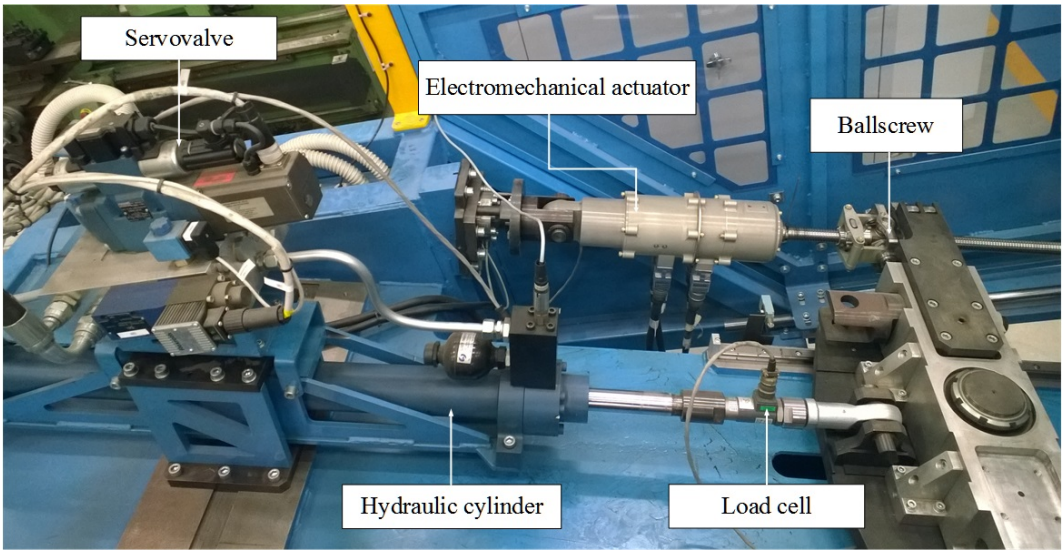


Figure 5.2: Test rig with main components. The load cell is used to close the hydraulic cylinder force loop, controlled by the servovalve

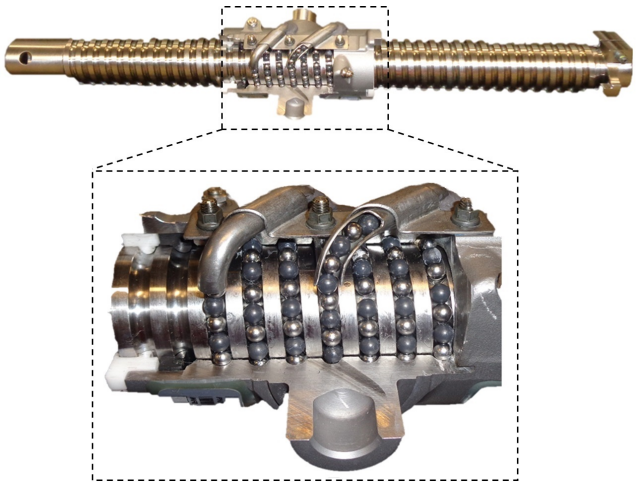
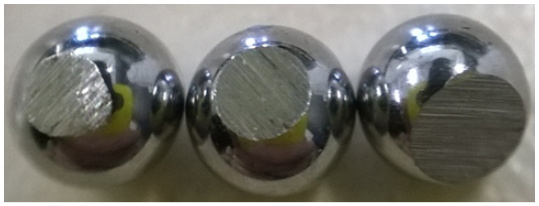


Figure 5.3: Ballscrew transmission and recirculation nut detail. The balls alternates between ceramic and steel ones

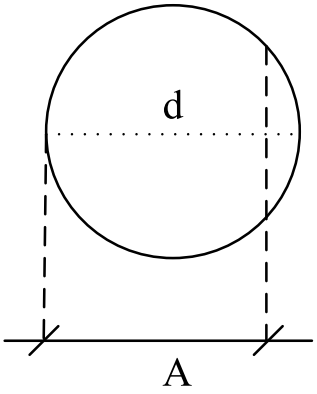
Effect Analysis (FMEA). The RTCA/DO-160 "Environmental Conditions and Test Procedures for Airborne Equipment" standard has been consulted, and low temperature tests were performed. Other test conditions specified in the standard have not been taken into consideration, because the actuator was proved to be robust to them, or because they were impossible to test.

5.2.1 Fault implementation and test conditions

As described previously, the considered mechanical fault conditions regard the spalling of steel balls inside the ballscrew recirculation nut. This type of fault has been deemed representative of a real one by the ballscrew producer. The



(a) Balls damage detail inside the transmission. From left to rigth: light, medium and high damaged balls



(b) Quantification of ballscrew steel balls injected damages

Figure 5.4: Injected faults on ballscrew spheres: qualitative and quantitative views

fault was injected by a Electrical Discharge Machine (EDM), which removes material providing a truncated sphere shape. In this way, both the shape and the dimension of the balls are modified. Three types of damage harshness have been chosen. The fault quantification can be assessed by referring to Figure 5.4: the diameter d for healthy balls is 3.5 mm, while, for defected balls, the entity of the fault is respectively:

- Light damage: $A = 3.3\,\text{mm}$
- Medium damage: $A = 3.2\,\text{mm}$
- High damage: $A = 3.1\,\text{mm}$

With the aforementioned damage levels, four different nuts have been prepared for tests:

1. **Fault condition 0**: no damaged spheres
2. **Fault condition 1**: 6 light damaged + 6 medium damaged + 6 high damaged balls per channel
3. **Fault condition 2**: 20 high damaged balls per channel
4. **Fault condition 3**: 40 high damaged balls per channel

These conditions have been chosen in order to enhance the fault severity, by increasing both the number of damaged balls and their damage level. Tests were performed with both healthy and faulty nuts, by simply replacing one nut

with another. During acquisition sessions, the temperature was controlled by cooling the actuator after each movimentation to its starting temperature, in order to minimize the uprising of temperature-dependent effects. A number of tests have been also performed by letting the motor temperature to raise up, to study the possible effects of heating on actuator performance. The test campaign's prospect included low temperature tests, which were performed by means of a cold chamber, connected to the metal cage which embedded the actuator, see Figure 5.5. This setup allowed to reach temperatures of $-40\,°C$ via liquid nitrogen injection.

Figure 5.5: Low temperature tests setup with motor cage detail

5.2.2 Test profiles

The whole test bench is controlled via a specific PC bench, which permits to select the desired profiles to be executed. The computer is connected to the electric drive through a Serial Peripheral Interface (SPI) connection, and communicates with the hydraulic cylinder via a National Instrument (NI) CompactRIO hardware. The drive deals with the control of the electric motor speed and current loop, while the CompactRIO computes the control law of the hydraulic piston. The position, or speed, profile is sent via the RS232 protocol to an ECU linked to the motor drive, and has the duty to close the position control loop. The profiles used in the experimentation were discussed with the project partners. The nominal load profile used during the tests corresponds to a typical high lift load profile; additional load profiles with constant 12 kN and 15 kN where employed as shown in Figure 5.6, to better assess the fault conditions. Slightly different behaviours in the load response are due to test

bench non-idealities. The position profile has been defined as follows:

1. Position run from 0 mm to 411 mm (100% of the actuator stroke), in 20 s
2. Acceleration of 2 s, from 0 $\frac{\text{mm}}{\text{s}}$ to 21 $\frac{\text{mm}}{\text{s}}$

In order to cope with the second constraint, the motion profile has been implemented as a speed profile, as depicted in Figure 5.7. The experimental tests consisted each in two runs of the aforementioned speed profiles: data are recorded during all runs, but only measurements from the second run are retained for successive processing. This is due to the fact that, during the first motion, test rig's settlements and vibration due to motion starting compromise the validity of acquired data.

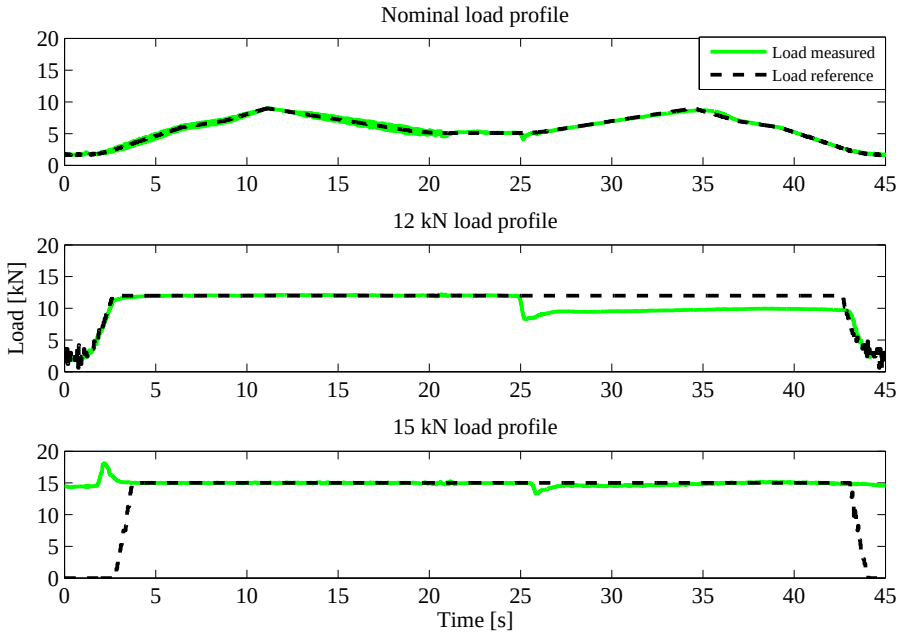


Figure 5.6: Load profiles employed during the test sessions. The non-ideal tracking behaviour is due to test bench limitations

5.2.3 Collected and available measurements

Various measurements have been collected from the test rig's equipped electronic, with the addition of a NI cDAQ device. The modules installed on the cDAQ consisted in a 16 bit voltage module (to acquire the load cell for synchronzation purposes) and current one (to acquire cylinder pressures), along with a 24 bit module used to measure the signals of two piezoelectric accelerometers mounted

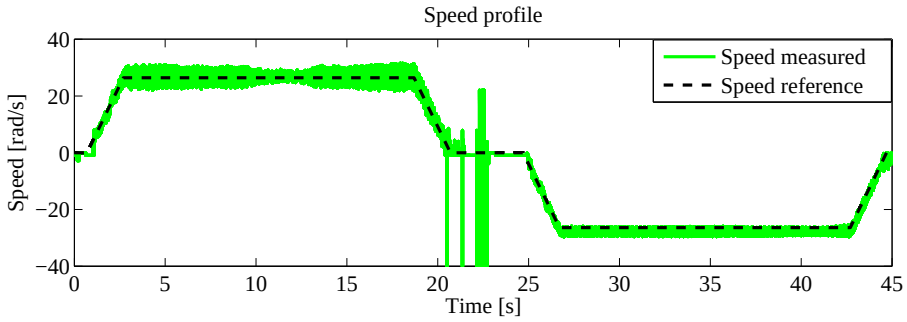


Figure 5.7: Speed profile employed during the test sessions. Spikes and oscillating behaviours are visible

in orthogonal directions on the nut. The acquisition frequency for the cDAQ was set to 20 kHz; already acquired variables, related to the EMA, were acquired at 5 kHz and sent to the PC bench via SPI, while variables related to the hydraulic part were measured at 1 kHz and stored via a NI 6323 16 bit acquisition board. A type K thermocouple was mounted on the motor surface were the magnets lie, and acquired through a Hydra Fluke device.

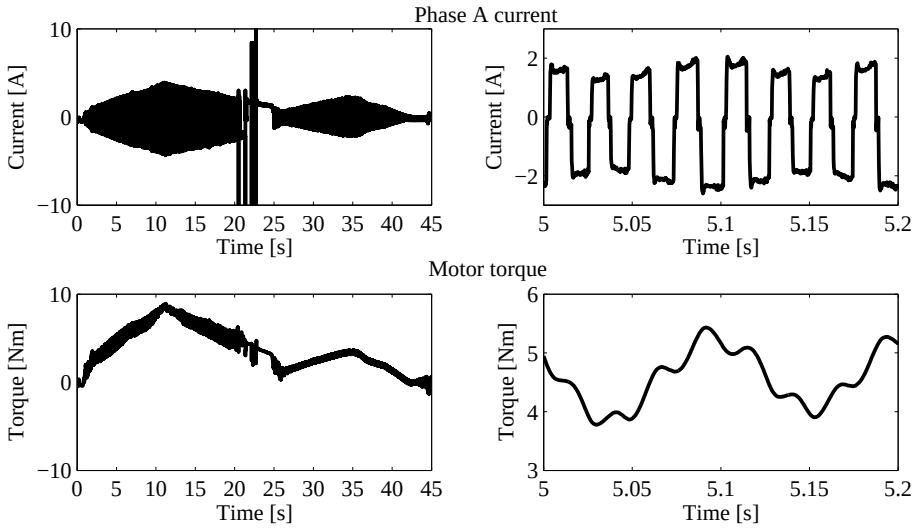


Figure 5.8: Top: motor phase A current with detail on current shapes. Bottom: computed motor torque with detail

Variables that were stored but not used to perform any health monitoring function consist of the motor phase voltage references collected by the motor drive, the motor angular position via a combination of resolver and multiturn encoder measurement, the load and speed profile references.

The variables used for the development of the *data-driven* fault detection algorithms are mainly related to phase current measurements, with current sensors directly installed on the motor drive. The motor's torque, along with

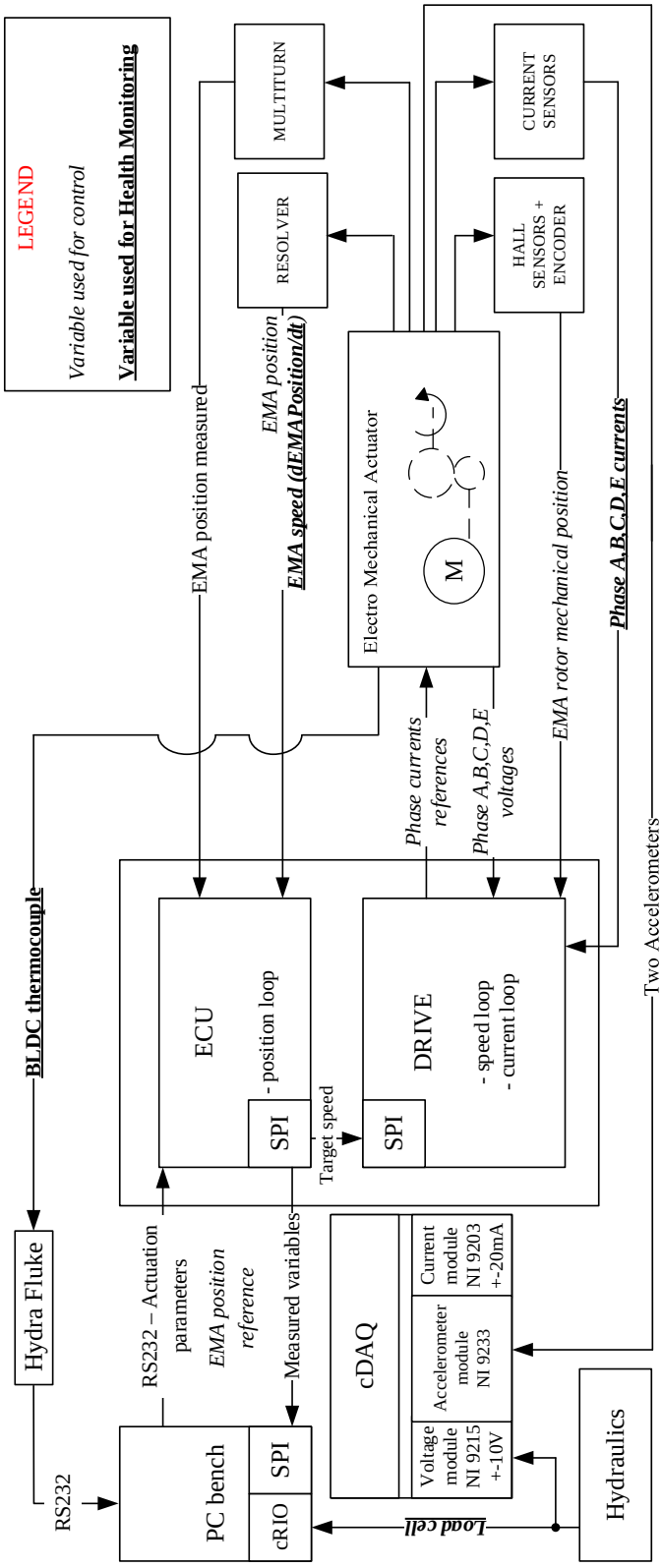


Figure 5.9: Schematic of the test rig components, with interactions and measurements system information

the quadrature current, has been computed from the phase currents and the motor mechanical sectors, Figure 5.8.
The motor's commutation logic is provided by means of hall sensors and an incremental encoder. The motor's torque constant was obtained through bench characterization. Other variables used to compute the fault detection indexes are the load cell and the thermocouple. An overall representation of the measurements' information flow is presented in Figure 5.9.

For the development of the *model-based* fault detection scheme, the main used variables are the phase currents (relating the torque with the motor speed), the load cell, and the motor speed obtained by deriving the position measurement.

The next chapters will present a model-based and a model-free fault detection scheme applied to the HOLMES project.

CHAPTER 6

Holmes project - Model based approach

This chapter presents a modification of the standard particle filter algorithm, applied to the HOLMES project introduced in Chapter 5. The variant, based on a hybrid system interpretation of the health monitoring problem, is known as Observation and Transition Particle Filter (OTPF). By modeling each fault condition as a hybrid system mode, the method is able to assess the most likely regime for each time stamp. A model for each condition was identified and the proposed methodology applied. Simulation results show the superiority of the method with respect to the EKF (Extended Kalman Filter), especially because the distribution of the disturbances that affect the system is usually not gaussian.

6.1 Motivation for the particle filter algorithm

Most model-based methods and schemes referenced in Section 5.1 are based on the assumptions that the sytem is linear and affected by gaussian disturbances. Here, the optimal solution to the filtering problem is given by the Kalman filter, and the residuals are represented by its innovations [72]. In the case of non-linear systems, sub-optimal solutions are employed, such as the Extended Kalman Filter (EKF). In situations where strong non-linearities or non-gaussian disturbances are present, the EKF method is not able to perfom satisfactorily: algorithms based on the Sequential Monte Carlo (SMC) framework, such as the Particle Filter (PF), are candidates for the employement [88]. The use

of these methods in the context of fault detection is not new. In [89], the authors employed the PF for fault detection in a sensor network. The use of a particle filter algorithm called Hybrid Boostrap Particle Filter (HBPF) for fault detection of hybrid systems is introduced in [90]. This approach has the disadvantage that faults with low probability of occurring (often the most serious ones) will be harder to detect because less tracked by the algorithm. To solve this problem, [91] proposed the OTPF (Observation and Transition Particle Filter). Here, the hybrid system's operating mode is interpreted as a particular process status (healthy/faulty). Its estimation permits therefore to solve the detection and diagnosis phase.

The employement of the OTPF algorithm in place of the more known EKF is advocated by simulation results. Specifically, when strong non-linearities and non-gaussian disturbances affect the system, the added computational and tuning complexity of the proposed approach is justified. In particular, in aerospace applications, phenomena as atmospheric turbulences [92] and exhaust plume disturbance [93] are typical sources of non-gaussianity.

The remainder of the chpater is organized as follows. Section 6.2 develops the continous mathematical model of the mechanical system under study. In Section 6.3, along with a brief description of the standard particle filter method, the OTPF algorithm is described. Section 6.4 discusses results and comparisons with the Extended Kalman Filter technique, under non-gaussian disturbance distributions simulations. Lastly, Section 6.5 is devoted to concluding remarks and future developments.

6.2 System modeling

A schematic representation of the developed model is depicted in Figure 6.1. The test bench is equipped with two motors but only one is considered for this work, since the other one is disconnected. Indicating with t the *continous time index*, the model inputs consist of:

1. Motor torque T_t^m - computed by measuring the phase currents
2. Load force F_t^h - measured from the hydraulic cylinder

The output of the model is:

1. Motor angular speed ω_t - computed from the resolver's position measurement

The *known parameters* are:

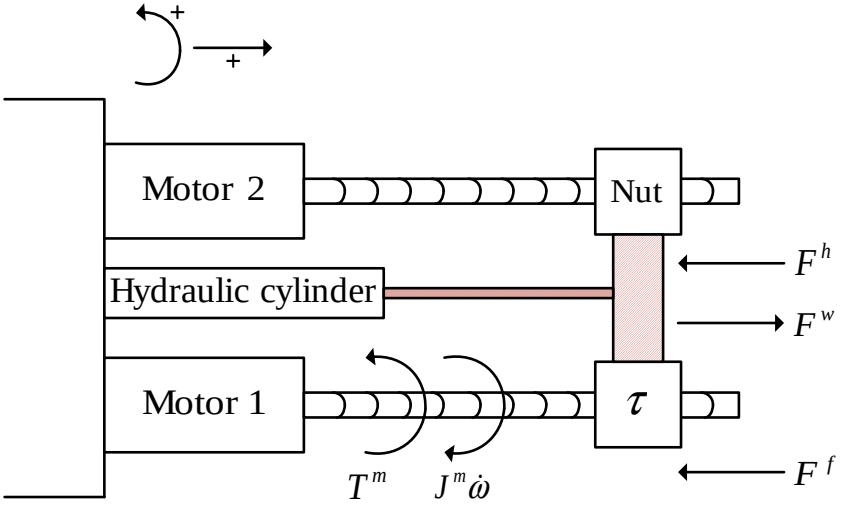


Figure 6.1: Diagram of the system dynamic quantities

1. Transmission ratio $\tau = \frac{0.005}{2\pi}\,[\frac{\text{m}}{\text{rad}}]$
2. Weight force $F^w = 210\,\text{N}$ - measured experimentally

The *unknown parameters* are:

1. Motor inertia J^m
2. Coulomb f^c, Stribeck f^s, and viscous friction parameters $c_1\,, c_2$, along with the Stribeck velocity ω^s. These coefficients control the friction force F_t^f

Given measured data, a continous time system can be developed and the six parameters identified. From the relations in Figure 6.1, it is possible to obtain the following dynamic balance, where the load inertia has been neglected because of the low acceleration of the speed profile used for the performed tests, as discussed in Section 5.2:

$$T_t^m - J^m\ddot{\theta}_t = \tau\left[F_t^h - F^w + F_t^f\right]. \tag{6.1}$$

The model in (9.1) can be casted into state-space form by introducing the state variable $x_t = \omega_t$:

$$\begin{cases} \dot{x}_t &= \dfrac{T_t^m - T_t^h + T_t^w - T_t^f}{J^m}\,, \\ y_t &= x_t \end{cases} \tag{6.2}$$

where $T_t^h\,, T^w\,, T_t^f$ are respectively the torques of load, weight and friction referred to the motor-side. The friction torque, depending on the motor angular

speed, is modeled with a Tustin friction model [94]:

$$T_t^f = \left[f^c + (f^s - f^c) e^{-\left(\frac{x_t}{\omega^s}\right)^2} + c_2 \cdot x_t^2 \right] \cdot sign(x_t) + c_1 \cdot x_t \tag{6.3}$$

Tests were performed at various constant speeds to better characterize the friction behaviour. The quadratic term $c_2 \cdot x_t^2$ has been shown to experimentally better represent the data, as discussed in [95]. In order to simulate the model, the function *sign* has been replaced with the hyperbolic tangent. The model was then discretized using the backward Euler method, in order to implement the filtering procedures described hereafter.

6.3 Fault detection via particle filters

The particle filter is a special implementation of a SMC method [96]. These techniques make use of simulations to generate weighted samples, in such a way to approximate a target distribution. In the context of dynamic systems filtering, this boils down into estimating the state probability density function (pdf). The KF and EKF algorithms estimate the state pdf as a gaussian one: this is correct if the system is linear and the disturbances have normal distribution. The state point estimate is then the mean of that gaussian. In other cases, the employement of SMC algorithms enable the use of samples to compute *any* distribution's moment, and empirically approximate the state pdf, without resorting to a specific distribution's mathematical model. The practical generation of the samples is made possible by the combined actions of importance sampling and resampling schemes, which prevent the *weight degeneracy* issue, [97]. Consider a generic state-space model, under the usual Markov assumption, where k *is the discrete time stamp*:

- System model

$$x_k = a(x_{k-1}, e_k) \leftrightarrow \overbrace{f(x_k|x_{k-1})}^{\text{Transition density}} \tag{6.4}$$

- Measurement model

$$y_k = b(x_k, v_k) \leftrightarrow \overbrace{g(y_k|x_k)}^{\text{Observation density}} \tag{6.5}$$

Variables x_k and y_k represent respectively the dynamic system state and output, $a(\cdot)$ and $b(\cdot)$ are generic non-linear functions, f and g are probability

density functions, e_k and v_k are disturbances with known pdf. The system can be equivalently represented by means of the transition and observation densities. The filtering problem consists into estimating the state distribution $\pi_{k|0:k}(x_k|y_{0:k})$ given observations up to time k, assuming that the initial state x_0 is distributed according to $\pi_0(x_0)$. The full state posterior can be obtained by the following recursion [98]:

- Prediction

$$\pi_{0:k|0:k-1}(x_{0:k}|y_{0:k-1}) = \tag{6.6}$$

$$= \pi_{0:k-1|0:k-1}(x_{0:k-1}|y_{0:k-1})\overbrace{f(x_k|x_{0:k-1}, y_{0:k-1})}^{f(x_k|x_{k-1})}, \tag{6.7}$$

- Correction

$$\pi_{0:k|0:k}(x_{0:k}|y_{0:k}) = \frac{g(y_k|x_k)\pi_{0:k|0:k-1}(x_{0:k}|y_{0:k-1})}{l_{k|0:k-1}(y_k|y_{0_k-1})}, \tag{6.8}$$

where $l_{k|0:k-1}$ is the predictive distribution of y_k given the past observations $y_{0:k-1}$, and $f(x_k|x_{0:k-1}, y_{0:k-1}) = f(x_k|x_{k-1})$ thanks to the Markov property of dynamic systems. For a fixed data realization, this term is a normalization constant, that makes the quantity in (6.8) to be a correct probability density function. The aim is now to sample from $\pi_{0:k|0:k}(x_{0:k}|y_{0:k})$. Since it is generally impossible to sample directly from this distribution, a sequential version of the importance sampling is employed. Conceptually, N_p particle paths, $\widetilde{x}_{0:k}^{(i)}$ $i = 1 \dots N_p$, are sampled from a convenient importance distribution $q_{0:k}(x_{0:k}|y_{0:k})$, and the unnormalized importance weights $\widetilde{\omega}_k^{(i)}$ are computed:

$$\widetilde{\omega}_k^{(i)} = \frac{\pi_{0:k|0:k}(\widetilde{x}_{0:k}^{(i)}|y_{0:k})}{q_{0:k}(\widetilde{x}_{0:k}^{(i)}|y_{0:k})}. \tag{6.9}$$

Using the weighted sample $\{\widetilde{x}_{0:k}^{(i)}, \widetilde{\omega}_k^{(i)}\}$, and having normalized the weights, it is possible to compute any distribution's statistic. The trick behind the sequential importance sampling lies in choosing an importance distribution which factorizes as the target posterior distribution (the state pdf). In this way, both particles and weights can be computed recursively. In order to prevent the weights degeneracy problem, a *resampling* scheme is applied to remove particles with low weight and replicate the most important ones. In this work, the

state transition density $f(x_k|x_{k-1})$ is chosen as importance distribution: with this choice, new particles are obtained via the system dynamic equations, and weighted according to their likelihood of belonging to the output density (which makes use of the current observation). This algorithm is know as *boostrap particle filter*. For further details and implementation issues, see [99].

6.3.1 Observation and Transition Particle Filter

The idea behind the OTPF method [91] is to frame the fault detection problem into an hybrid systems formulation [100]. When, at a certain discrete time instant k, the hybrid system is in the mode $\nu_k \in \{1, \dots V\}$, with V the number of discrete possible process modes, its behaviour is described by:

$$\begin{cases} x_k &= a_{\nu_k}(x_{k-1}, e_k) \\ y_k &= b_{\nu_k}(x_k, v_k) \end{cases}, \tag{6.10}$$

where $a_{\rho_k}(\cdot)$ and $b_{\nu_k}(\cdot)$ are generic functions which describe the system dynamics in the mode ν_k. It is assumed that the initial mode and initial system state are known. A matrix Θ, known a-priori, is such that Θ_{ij} defines the transition probabilities from mode i to mode j. The problem of state estimation in hybrid systems and the fault detection one are strictly interconnected. In fact, assuming that different modes describe the system dynamics against different faults, once the most probable mode at current time is estimated, automatically the detection and identification of the fault are assessed. While much work has been devoted to the filtering of hybrid linear system, see [101], less is known about non-linear models. Proposed solutions to this issue, based on particle filter methods, have been shown to perform well in practice [102]. In the OTPF method, the hybrid system mode (which can vary at each time stamp) is considered an unknown parameter to be estimated. A step is added to the particle filter procedure to obtain the most likely estimation of the mode from the particles. Once the most probable mode is found, it is assumed that the hybrid system will follow the relative dynamics. Then, the subsequent steps are the same as the standard particle filter. In order to estimate the mode, the OTPF combines the current observation y_k with the mode transition probabilities. This permits to monitor even low occurring modes.

The logic behind the fault detection filtering algorithm, depicted graphically in Figure 6.2, is as follows. Given an hybrid system with V possible modes, known process and observation models, N_p particles are generated from the

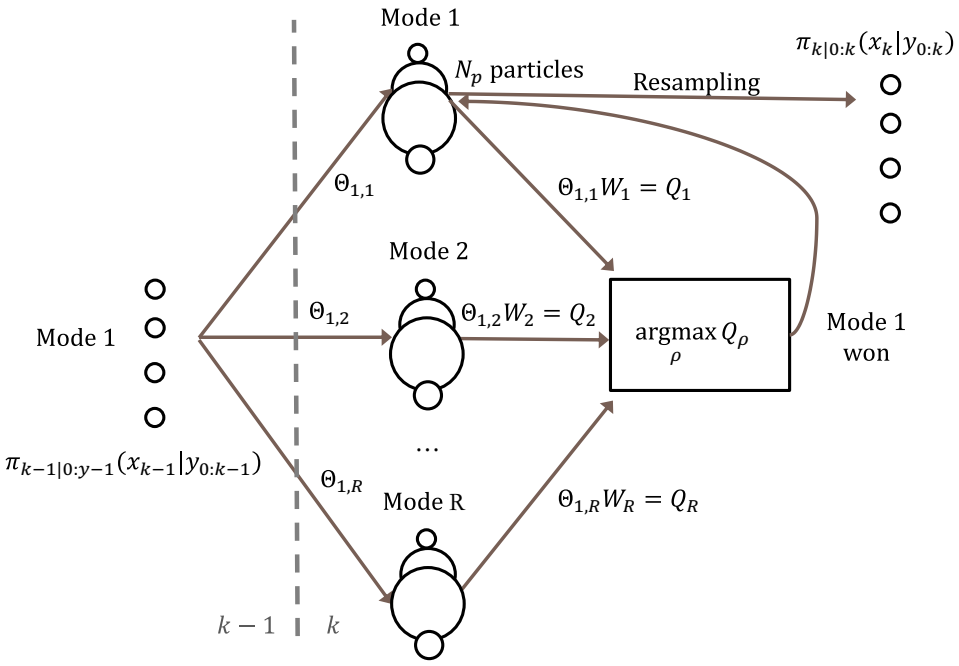


Figure 6.2: OTPF single-step procedure

assumed known initial state. Considering the number R of modes for which the previous mode has not-null transition probability, particles are propagated through each the R modes. The outputs of this step are R sets of N_p particles. Then, for each mode $\rho \in \{1, \dots, R\}$, the mean weight W_ρ of all particles in that mode is computed, leveraging on the current measurement y_k. Then, the quantity $Q_\rho = \Theta_{\nu_{k-1},\rho} \cdot W_\rho$ is computed, with ν_{k-1} the selected mode at time $k-1$ and ρ the hypothesized mode at time k. The quantity Q_ρ encodes information about both the transition probability and the data likelihood given by W_ρ. In this way, even modes occurring with low probability can be selected if supported by the data. The mode $\nu_k = \arg\max_\rho Q_\rho$ is chosen as the most plausible one, and its particles resampled and propagated through its dynamic.

6.4 Results and discussion

Data were collected from the test bench described in Section 5.2. A model, in the form described in Section 6.2, has been identified for each of the four fault conditions (using data measured with fault injected components), focusing only on the nominal load condition. Simulated data were generated by feeding the four models with the same inputs, and their outputs concatenated. Performances on the fault detection and identification problem are shown,

comparing the OTPF with the EKF, under gaussian and uniform distributions. Results show how the OPTF method outperforms the EKF, especially when the noise pdf is not gaussian. The employement of these methods is mandatory given the non-linear form of the models.

The EKF fault detection approach is as follows. The ν-th Kalman filter tracks the dynamics of the ν-th mode, with $\nu \in 1, \ldots, V$. The ν-th residual at time k is then $z_{k,\nu} = y_k - \widehat{y}_{k,\nu}$, with y_k the observed output and $\widehat{y}_{k,\nu}$ the predicted output from the mode ν filter. If the state estimation is correct, the innovations are gaussian with zero mean and covariance $\Sigma_{k,\nu} = [\bar{H}_{k,\nu} P_{k|k-1} \bar{H}^T_{k,\nu} + V_2]$, where $\bar{H}_{k,\nu}$ represents the linearized output matrix, $P_{t|t-1}$ is the recursively computed variance matrix of the state filtering error, and V_2 is the covariance of the output disturbances. A fault can be detected by a variation (significative deviation from zero) of the Weighted Squared Residual (WSR): $\xi_{k,\nu} = z_{k,\nu} \Lambda^{-1}_{k,\nu} z^T_{k,\nu}$. A more robust measure, used in this work, is the Weighted Sum Squared Residual (WSSR): $\Xi_{k,\nu} = \sum_{j=k-s+1}^{k} \xi_{j,\nu}$, where s is the window length in which residuals are added [72, 103]. In the simulations, a value $s = 20$ was chosen. The estimated mode ν_k at time k are the one for which the WSSRs are minimum: $\nu_k = \arg\min_{\nu} \Xi_{k,\nu}$.

6.4.1 Simulation results

This section reports the simulation results comparing the OTPF with the EKF under different disturbances distributions. The selected number of particles was $N_p = 100$, while the transition matrix was defined as:

$$\Theta = \begin{bmatrix} 1-\epsilon & \epsilon & 0 & 0 \\ 0 & 1-\epsilon & \epsilon & 0 \\ 0 & 0 & 1-\epsilon & \epsilon \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.11}$$

where $\epsilon = 10^{-8}$ is chosen as probability of the investigated ballscrew fault, and each column and row reflects the following order: (Fault condition 0, Fault condition 1, Fault condition 2, Fault condition 3). As an example, $\Theta_{1,2} = \epsilon$ is the probability of transitioning in the "Fault condition 1" mode, given that the current mode is the "Fault condition 0" one.

Gaussian disturbances

The first experiment added gaussian disturbances to the simulated models states and outputs, both with variance 0.01. The transition and observation densities of the OTPF consisted in gaussian distributions with variance 0.01. The covariances V_1, V_2 of the state and output noises of the EKFs were set to 0.01. Results are reported in Figure 6.3. Binary accuracy is the percentage of corrected classifications made by trying to distinguish the healthy class vs. the faulty ones. Multiclass accuracy refers to the multiclass classification task, that is, to exactly assign each data in its own class.

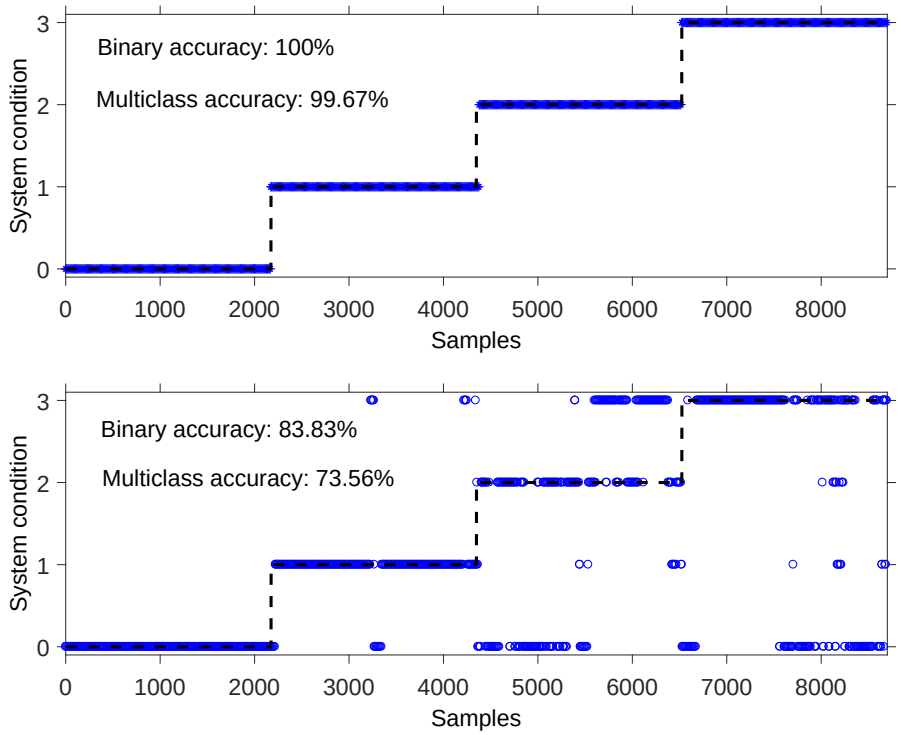


Figure 6.3: Fault detection with gaussian disturbances. Top: OTPF estimated system mode (blue dots) vs. real system mode (black dashed line). Bottom: EKF estimation

Uniform disturbances

The second experiment added disturbances distributed according to a uniform distribution $\mathrm{U}(-3, 3)$ distribution to the simulated models states, and gaussian noise to outputs, with variance 0.01. The transition density of the OTPF consisted in an $\mathrm{U}(-3, 3)$ pdf, while the output one was a gaussian with mean 0

and variance 0.01. The covariances V_1 , V_2 of the state and output noises of the EKFs were set to 0.01. Results are reported in Figure 6.4.

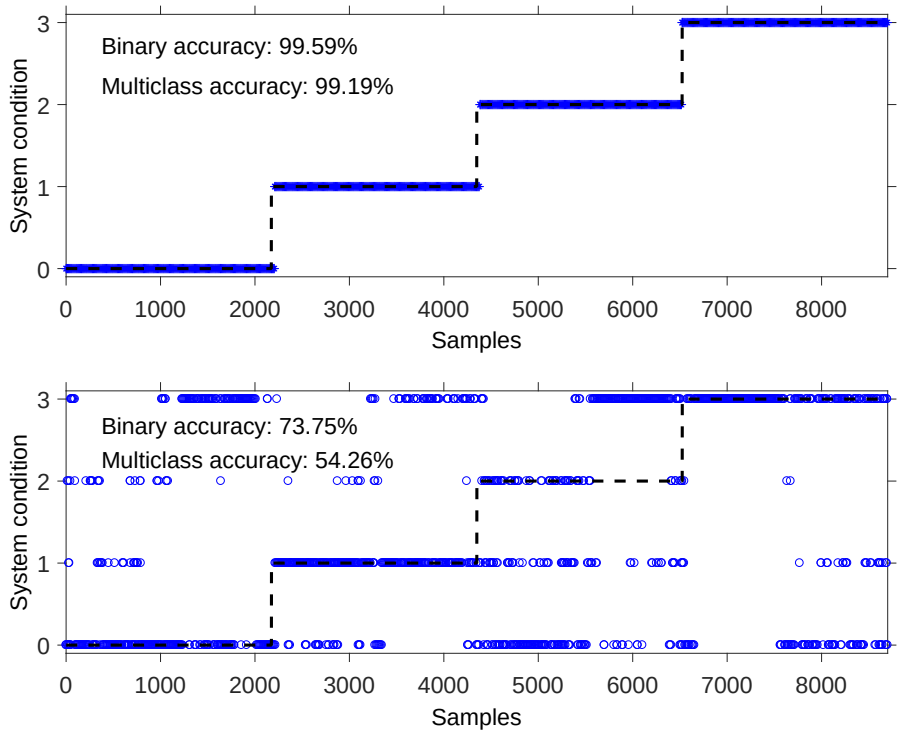


Figure 6.4: Fault detection with uniform disturbances. Top: OTPF estimated system mode (blue dots) vs. real system mode (black dashed line). Bottom: EKF estimation

6.4.2 Discussion

As it is possible to observe, the OPTF outperforms the EKF in both experiments. While the latter approach still keeps the pace when the disturbances are gaussian, it performs poorly when non-gaussian distribution comes into play. This suggest that, in presence of non standard disturbances and strong non linearities, the employement of a particle filter algorithm can be beneficial. Binary accuracy refers to the classification error considering the "fault condition 0" data as the "healthy" class, and the data acquired in the other fault conditions as the "faulty" class. The RMSE (Root Mean Square Error) on the filtering of the state variable was computed for each simulation. Regarding results of Figure 6.3, the RMSE for the OPTF was 0.07 while for the EKF was 0.73. As concerns results from Figure 6.4, the RMSE for the OPTF was 0.13 while for the EKF was 1.15. For the first experiment, the elapsed time for the OTPF and EKF

was 2.27 s and 0.91 s respectively, on a i7-2.30 GHz processor. For the second experiments, the OTPF took 31 s, while the EKF 1.15 s.

6.5 Conclusions and future developments

In this work, a variant of the particle filtering algorithm, known as Observation and Transition Particle Filter, has been applied to fault detection of an electro-mechanical actuator. The actuator, deployed in aerospace environments, was injected with real faults on the ballscrew transmission spheres. By collecting data in each fault condition, four different models were identified, where the friction component included a non-linear aspect. The compared non-linear filtering techniques, in terms of fault detection capability, were the OPTF and the EKF methodologies. In order to do this, data were simulated from the four different models. When gaussian noise was added to the data, both techniques performed good. However, when a non-gaussian noise was introduced, the OTPF method clearly won. In aerospace applications, where phenomena such as atmospheric turbulence and exhaust plume disturbance are present, the gaussianity of the disturbances could be a strong assumption. Future research consists of the application of the method on real data and the integration of a model-free approach.

CHAPTER 7

Holmes project - Data driven approach

This chapter presents a data-driven approach to the fault detection problem of Chapter 5. The methodology is based on a machine learning pipeline, on the assumption that data collected from different system conditions belong to different "classes" that the algorithm learns to discern. This assumption came from the fact that non-ideal mechanical behaviours can be detected by inspecting suitable measurements, such as motor phase currents [104]. Features, belonging to different domains, have been extracted from the measured signals. These indexes are based largely on the motor driving currents, in order to avoid the installation of new sensors. A comparison of different classification algorithms is presented, and the chosen one is a Gradient Tree Boosting classifier. Furthermore, the most promising features for a classification point of view are reported. Methods and results are validated by means of experimental tests.

In Section 7.1, the steps involving the design of a model-free fault detection algorithm are outlined. Section 7.2 shows a comparison between different classifiers, with indications about the choices made, and a graphical visualization of the most important features for fault detection is given. Section 7.3 is devoted to concluding remarks and future developments.

7.1 Data-driven fault detection strategy

This section presents the logical steps adopted in order to develop the machine learning based model-free solution. The process pipeline is sketched in Figure 7.1,

where details about each phase are described. The steps consists into feature extraction, feature selection with classifier design, and classifier evaluation.

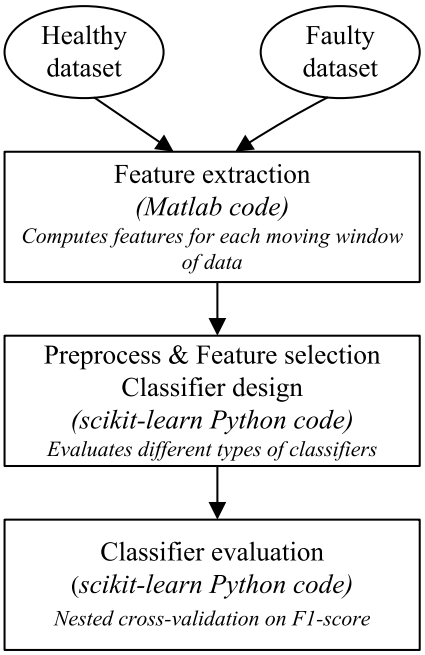


Figure 7.1: Model-free methodology flowchart

The motivations behind the data-driven solution has to be sought into the possibility to perform many experimental tests with different fault conditions. The model-free approach, which is independent of any physical modeling, takes a higher vision on the system at hand. The proposed methodology consists in computing features on data obtained through a sliding window, which runs on the entire measurement vectors, selecting each time a portion of the data. The length of the sliding window has been chosen, after a sensitivity analysis and guided by a trade-off between computational time and quantity of data on which to compute the features, to be of 1.5 s, with an overlapping factor of 0.75 s. These hyperparameters have to be tuned for the application at hand. Preprocessing of data consisted in filtering noisy signals. The features that are extracted for each data window are described next.

7.1.1 Feature extraction

In this work, up to 15 features were computed, spanning time and frequency domain. The indexes are:

1. Torque-load ratio
2. Root Mean Square value

3. Kurtosis
4. Skewness
5. Frequency power via FFT transform
6. Peak-to-valley
7. Energy operator
8. Crest factor
9. Shape factor
10. Mean frequency
11. Frequency center
12. Root Mean Square frequency
13. Standard deviation frequency
14. Sixth central moment
15. Mean temperature

The use and computation of these indexes has been advocated in many previous fault detection applications (see for details the work done by [85], [105], [106], [107], [108]). Feature 1 is computed by taking the ratio of the computed motor torque over the load measured by the load cell mounted on the hydraulic cylinder. Features from 2 to 14 are computed on the motor quadrature current signal, while feature 15 is computed from the thermocouple measurements. The considered spectrum in Feature 6 is $(0\,\text{Hz} - 50\,\text{Hz}]$, since the major frequency content of the quadrature current lies in that range. Then, the total frequency power in that range is used as a feature. The output of this stage is a feature matrix $\Phi \in \mathbb{R}^{N \times m}$, where $N = 5359$ is the number of observations (depending on the length of the performed tests), and $m = 15$ is the number of features. This choice of measurements has been demanded by the application: the aim was indeed to rely mainly on electrical variables to perform the health monitoring.

7.1.2 Feature selection and classifier design

The data were randomly shuffled and then divided into train (80%) and test (20%) set. This lead to a total of 4287 training data and 1072 test data. After the splitting, the classes were almost equally represented, as concerns the number of points belonging to each class, in the train and test set. The train data were then scaled via a robust standardization procedure [109], which, for each feature, removes the median and divides for the interquantile range (the interval between the 25th quantile and the 75th quantile). This standardization was chosen because it is more robust to outliers in the data. The transformation, with parameters fitted on the training set, is then applied to the test set. Then, various types of classification algorithms were tested, such as: Logistic Regression (LR), Support Vector Machine (SVM), Naïve Bayes (NB) and Gradient Tree Boosting (GTB) [4]. All chosen classifiers are discriminative, except for the Naïve Bayes one. The choice is dictated by the fact that the classification result is of most interest with respect to understand the data-generating process. However, it is useful to test both classifier types, given that, under certain conditions, generative classifiers can reach faster their maximum accuracy bound with respect to discriminative algorithms [110]. The hyperparameters of each algorithm have been found by using a 5-fold cross-validation (cv) on the train set. The selected model is then trained on the training data. The logistic regression classifier was equipped with a $L2-$regularization term (another name for the ridge regularization), and the relative hyperparameter was tuned. The Support Vector Machine classifier used a Radial Basis Function kernels which required to find the proper hyperparameters' value. Regarding the Naïve Bayes algorithm, the Gaussian likelihood was assumed. The tuning parameters of the Gradient Tree Boosting method were the number of tree estimators, the subsample percentage and the learning rate.

7.1.3 Classifier evaluation

The evaluation of each classifier is done through two different procedures. As a first performance check, the classifiers were evaluated on the test set, and the mean F1-score is reported [111]. A value of 1 indicates perfect classification, while a value of 0 indicates a completely wrong result. Since the F1-score is defined for a binary classification problem, we end up with four F1-scores, because in this formulation there are four classes into which classify the data

(Fault condition 0, Fault condition 1, Fault condition 2, Fault condition 3). This score is computed by taking the weighted mean of the four F1-scores. The weights are the percentage of observations for a specific class over the total number of tests points. This choice of metric is due to the fact that it better assesses cases of imbalanced classes as opposed to classification accuracy. In our case, since the classes are well balanced, there is no high difference with the classification accuracy performance.

To check the stability of the training procedure, including also the steps performed to find the best hyperparameters, a nested cross-validation can be employed. It has been shown in [112] that this method better assesses the true algorithm performance, giving a less biased estimation with respect to the stardard cross-validation with fixed parameters, which would lead to an optimistic evaluation. With this method, each train/test fold may get different hyperparameter settings, resulting in an algorithm that internally finds the best parameters for each data set it gets. The results of this procedure are then reported as estimation of the model true performance. In this work a 5-fold nested-cv has been used on all the data (training + test dataset). As before, the weighted F1-score has been applied as performance metric. The feature scaling is fit on the training fold and applied on the test ones, for each training/testing folds combinations. The output is a vector of 5 weighted F1-scores, and the mean and standard deviation of this vector is taken as performance metric for classifiers comparison. This leads to an estimation of the mean F1-score with associated standard error. If the standard error is high, it means that the discovered hyperparameters are not reliable, and the learned model, with hyperparameters selected via cross-validation on the training set (or on all available data) can't be deployed into production.

7.2 Results and discussion

The final comparison results are reported in Table 7.1. The best performing classifier is the Gradient Tree Boosting algorithm, with a weigted mean F1-score obtained through nested cross-validation of 0.82. The Logistic Regression and Naïve Bayes algorithm failed to properly capture most of the data traits, not being enough flexible in their decision boundaries. The low standard error of the mean F1-score obtained by nested cross-validation indicates that the procedure used to select the classifiers hyperparameters is stable, not exibiting large variations when different datasets are used to tune them. Figure 7.2 depicts

the importance of each feature used, as considered by the GTB algorithm. The most informative indexes, as concerns the classification point of view, are the cage temperature, the torque to load ratio, and the computed frequency content.

Table 7.1: Classifiers comparison summary

Classifier	**Mean Test set F1-score**	**Mean Nested cv F1-score**	**Std. error Nested cv F1-score**
LR	0.25	0.21	0.024
SVM	0.70	0.70	0.005
NB	0.13	0.12	0.006
GTB	0.83	**0.82**	0.009

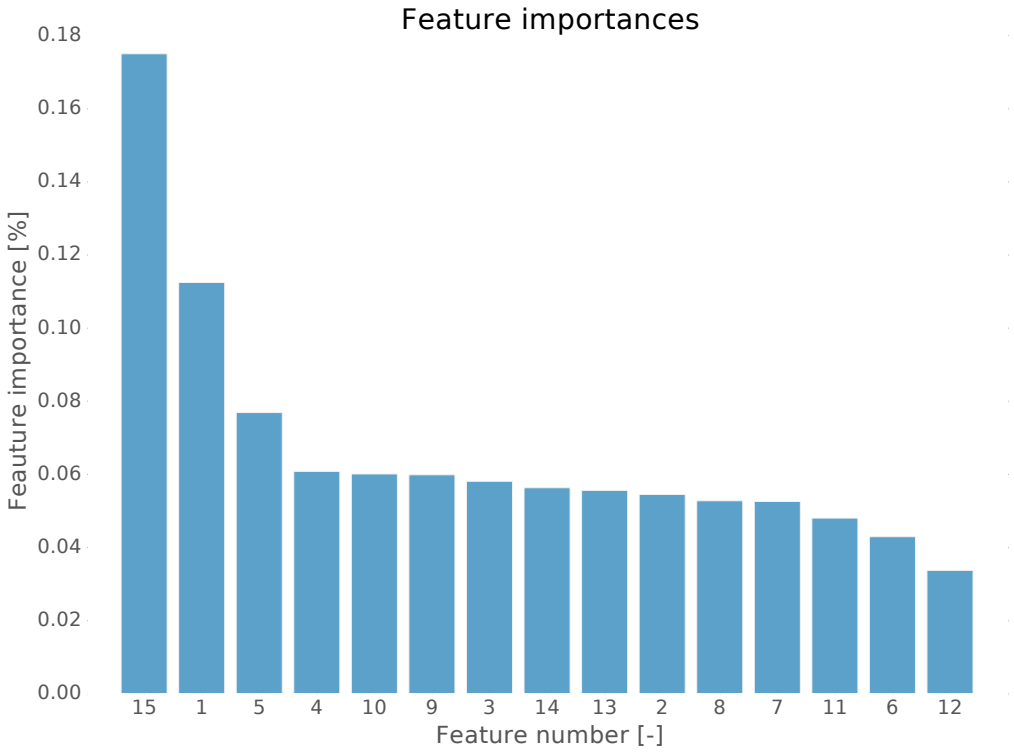


Figure 7.2: Features importance

7.3 Conclusions and future developments

This work presented a practical approach to the fault detection and identification problem. The application described regarded the health monitoring of mechanical components for electro-mechanical actuators deployed in an airliners applications. Future developments include the combination of the proposed approach with a model-based methodology, and applications of the framework to other fault types and conditions.

CHAPTER 8

Holmes project - Clustering

In this application, the use of the Principal Direction Divisive Partitioning (PDDP) method for unsupervised learning is discussed and analyzed with a focus on fault detection applications. Specifically, a geometric limit of the standard algorithm is highlighted by means of a simulation example and a modified version of PDDP is introduced. Such a method is shown to correcly perform data clustering also when the standard algorithm fails. The modified strategy is based on the use of a Chi-squared statistical test, and offers more guarantees in terms of detection of a wrong functioning of the system. The proposed algorithm is finally experimentally tested in the same framework of Chapter 5, but on different test conditions. A comparison with k-means and fuzzy k-means approaches is also provided.

8.1 Introduction

The Principal Direction Divisive Partitioning (PDDP) [113] is a clustering algorithm originally proposed to solve a text document classification. Various research activities have worked towards improvements in this direction [114, 115, 116]. The PDDP approach presents many distinctive features. First of all, its computational cost is roughly linear in the number of non-zeros in the feature matrix and only weakly (logarithmic) scaled with the number of generated clusters. Therefore, despite of its simplicity, it has been proven to produce high quality clusters, especially when the dimensionality of the data is high [115].

This is possible by stopping the singular value decomposition (SVD) at the first singular value/vector and makes PDDP significantly less computationally demanding than other widely known text-mining methods like, e.g., Latent Semantic Indexing (LSI) algorithm [117], especially if the data-matrix is sparse and the principal singular vector is computed by resorting to the Lanczos' technique [118, 119]. In this work, it will be shown that Fault Detection and Isolation (FDI) using the standard PDDP approach may yield some problems. More specifically, it will be illustrated by means of a motivating simulation example that PDDP may intrinsically split the data along the first principal component even when two clusters are slender and narrow, and their length is greater than the distance between their centroids. This fact constitutes a great limit, as it may significantly jeopardize the quality of clustering. To overcome this problem a modified version of PDDP - called mPDDP - is proposed, in which the choice of the cluster to split is based on Chi-squared goodness of the data fitting. It should also be said that many other variations of the original PDDP algorithm have been proposed to enhance its performance. In [120] the authors developed a non-greedy variant of the algorithm, which tries all the possible choices of partition on a specified number of clusters and principal components, by evaluating the variance within the cluster. The choice of the number of clusters, if not specified a priori, is discussed in [121] with the use of a BIC criterion (see Appendix A). The works in [122, 123] focus instead on the choice of which cluster to split. However, as far as the author is aware, none of the above variations of the PDDP has directly dealt with the problem object of this work. The remainder of the chapter is as follows. In Section 8.2, the standard PDDP method is briefly recalled. The proposed modification of the clustering method is illustrated in Section 8.3, where a simulation example is used to show the limits of the standard approach and visually explain the main idea behind the new algorithm. The modified PDDP is then applied on real data acquired from the test bench described in Chapter 5, where a significant increase in clustering performance is highlighted. Section 9.5 is then related to some concluding remarks.

8.2 Principal Direction Divisive Partitioning

The task of unsupervised learning is to reveal the organization of patterns into "sensible" clusters (groups). Similar patterns, in the sense of a defined similarity measure, will be grouped into the same cluster by a clustering algorithm. Many

clustering algorithms have been proposed throughout the years. The oldest ones are based on the Basic Sequential Algorithmic Scheme (BSAS), where each new point is said to belong to a group of points, depending on its distance from the existing clusters [124]. Then, several other categories of methods have been proposed: optimization-based, among which the celebrated k-means algorithm [125]; density-based, *e.g.* the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [126] and hierarchical clustering, like Principal Direction Divisive Partitioning (PDDP) [113].

PDDP belongs to another important class of data-processing techniques: SVD-based (Singular Value Decomposition) methods, in which the Latent Semantic Indexing algorithm (LSI) [117], and the LSI-related Linear Least Square Fit (LLSF) algorithm [127] are also included. The considered clustering approach is the bisecting divisive clustering: the problem to be solved is the splitting of the data matrix $\Phi \in \mathbb{R}^{N \times m}$ (where N is the number of data and m is the data dimensionality) into two sub-matrices (or sub-clusters) $X_L \in \mathbb{R}^{N_L \times m}$ and $X_R \in \mathbb{R}^{N_R \times m}$, with $N_L + N_R = N$. The data matrix Φ is therefore composed by stacking all the observations $x_i \in \mathbb{R}^{m \times 1}$, $i = 1, \ldots, N$, in row. PDDP is mainly based on Principal Components Analysis (PCA), thus involving the eigenvector decomposition of the data covariance matrix, or equivalently a Singular Value Decomposition of the data matrix after mean centering. The principal trend in data can be considered in two ways. In PCA, the direction of principal trend is taken as the direction in which the variance (or "spread") of the data is maximum. The second way to define the principal trend is by means of least squares, in which case the trend is along a line l for which the total sum of squares of orthogonal deviations from l is minimal among all lines in $\mathbb{R}^m$.

The PDDP algorithm is very popular, mainly for its low computational requirements. As a matter of fact, PDDP has a computational cost roughly linear in the number of non-zeros in the feature matrix and it is characterized by a weak scaling, which is logarithmic with the number of generated clusters. Moreover, it provides a "one-shot" deterministic solution, unlike the initialization dependent solution given, e.g., by the k-means. A thorough comparative analysis of bisecting k-means and PDDP is given in [128].

The PDDP algorithm can be formalized as follows.

PDDP clustering algorithm

1. Compute the centroid $w \in \mathbb{R}^{m\times 1}$ of Φ
2. Compute the auxiliary matrix $\breve{\Phi} = \Phi - e \cdot w^T$, where $e \in \mathbb{R}^{N\times 1}$ is column vector of ones, namely $e = [1, 1, \ldots, 1]^T$
3. Compute the Singular Value Decompositions (SVD) of $\breve{\Phi}$, $\breve{\Phi} = U\Sigma V^T$, where Σ is a diagonal $N \times m$ matrix, and $U \in \mathbb{R}^{N\times N}$ and $V \in \mathbb{R}^{m\times m}$ are orthonormal unitary square matrices
4. Take the first column vector $v_1 \in \mathbb{R}^{m\times 1}$ of V (the first principal component), and divide Φ into two sub-clusters Φ_L and Φ_R, according to the following rule:
$$\begin{cases} x_i \in \Phi_L & (x_i - w)^T \cdot v_1 \leq 0 \\ x_i \in \Phi_R & (x_i - w)^T \cdot v_1 > 0, \end{cases} \tag{8.1}$$
where $x_i \in \mathbb{R}^{m\times 1}$, $i = 1, \ldots, N$, represents the i-th observation
5. Iterate until the desired number of clusters is reached.

8.3 Modified PDDP based on statistical test

In this section, a limit of the PDDP algorithm in some practical situations is highlighted. Precisely, it will be shown that the splitting along the first principal component is not always the best choice when two clusters are slender and narrow, and their length is greater than the distance between their centroids. To overcome this problem, a modified version of the PDDP method is proposed. In what follows, first the Chi-squared test will be briefly outlined, then the modified version of PDDP method will be described and illustrated by means of a simulation example.

8.3.1 Chi-squared goodness of fit test

The chi-squared goodnees of fit test performs a statistical test to assess whether the data is drawn from a Gaussian probability density function (pdf). The situations are then two: either the data was drawn from a normal distribution (assumption H_0) or from another distribution (assumption H_1).

Given an histogram, the question is whether it is consistent with a given pdf. If the histogram has κ bins, let $b_0, b_1, \ldots, b_\kappa$ be the $\kappa + 1$ boundaries. So,

x belongs to the i-th bin if $b_{i-1} \leq x \leq b_i$, $i = 1, \ldots, \kappa$. Let o_i be the counts (numbers of points) for the i-th bin. Since there could be many experiments, o_i is a particular outcome of the random variable O_i. The expected counts e_i are computed from the distribution with parameters estimated on the data. To measure the discrepancy between the observed histogram o_i and the histogram e_i computed under the null hypothesis, it is then natural to use

$$\overline{\chi}^2 = \sum_{i=1}^{\kappa} \frac{(o_i - e_i)^2}{e_i}. \tag{8.2}$$

Since the observed bin values o_i are outcomes of the random variables O_i, the value $\overline{\chi}^2$ is itself an outcome of the random variable

$$\chi^2 = \sum_{i=1}^{\kappa} \frac{(O_i - e_i)^2}{e_i}, \tag{8.3}$$

which is distributed according to a Chi-squared probability density function, with $d = \kappa - 1$ degrees of freedom, $p_{\chi^2_d}(x)$, because of the constraint $\sum_{i=1}^{\kappa} O_i = N$.

The p-value is defined as the probability of obtaining a test statistic at least as extreme as the one that was actually observed, assuming that the null hypothesis is true:

$$p = \mathbb{P}\left[\chi^2 \geq \overline{\chi}^2\right] = \int_{\overline{\chi}^2}^{\infty} p_{\chi^2_d}(x)dx \tag{8.4}$$

If $p \leq \alpha$, the hypothesis H_0 is rejected at a significance level α, and the result is consistent with the null hypothesis with probability $1 - \alpha$.

8.3.2 Modified PDDP

To exemplify the problem, consider the case depicted in Figure 8.1. The projection on the first principal component, according to the PDDP rule, splits the cluster into two groups: the first cluster is composed by the top half of the blue and the red clusters, while the second cluster is composed by the bottom halves. Projecting on the second component, the two resulting clusters are the blue one and the red one, as an external observer would have suggested.

Consider now the simulation example illustrated in Figure 8.2. The first and the second big clusters are recognized by the PDDP algorithm and the cluster selected for further splitting is chosen to be the top one. However, at the second step of the algorithm, the red cluster is split into the green and red

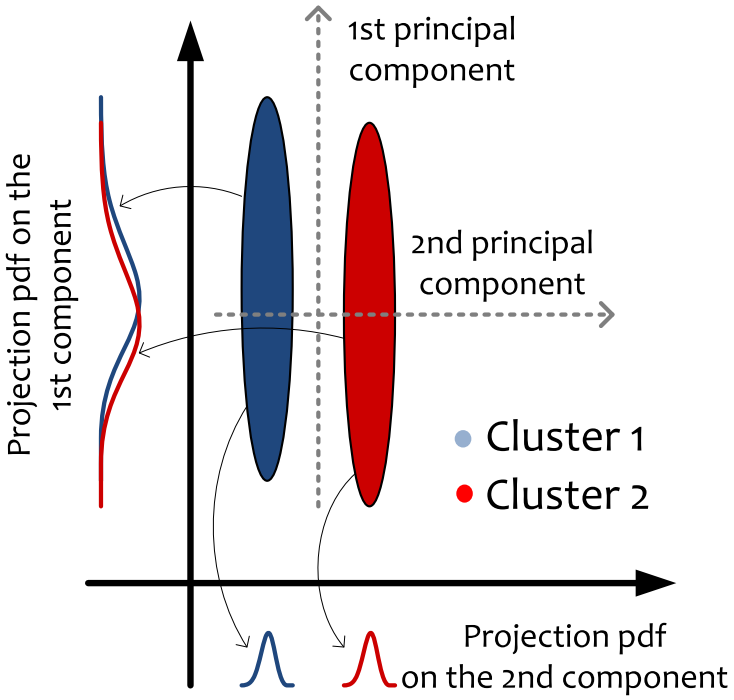


Figure 8.1: Dataset projection on the first two principal components

groups, which is obviously the uncorrect choice.

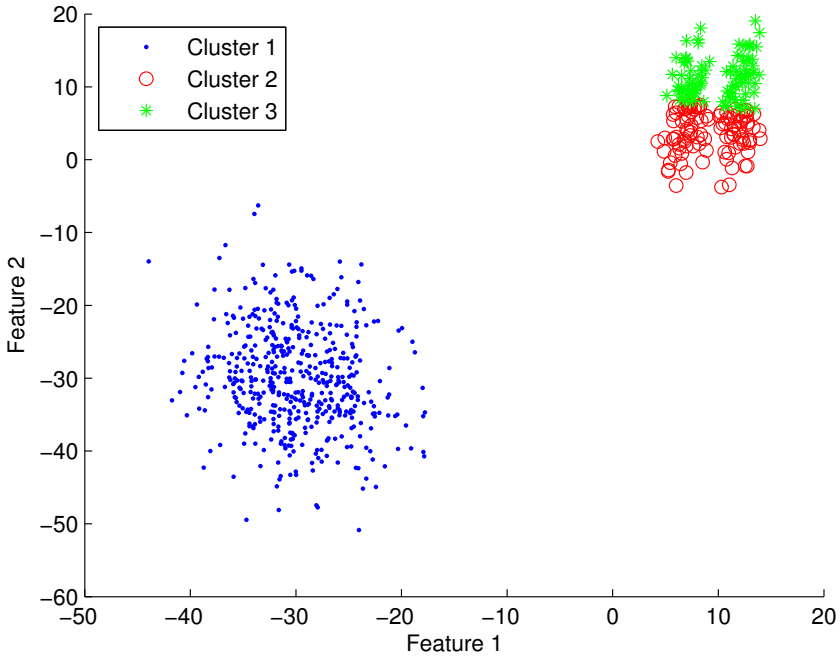


Figure 8.2: Clustering produced by the standard PDDP algorithm at the second step

To address this problem, it should first be noticed that the distributions of the data projected on the first component (see again Figure 8.1) are significantly

overlapped. Moreover, the sum of the two distributions approximates the normal distribution better than the sum of the distributions of the projections on the second principal component. The idea is then to adopt the statistical hypothesis test of Section 8.3.1 to check if the data really follow a normal distribution. By relying on the previous arguments, the data have to be projected on the direction for which their distribution is *less similar* to a Gaussian. Then, a statistical goodness of fit test can be performed on the data projected on each direction. The direction which has generated the data for which the hypothesis test gives the smallest p-value is chosen as the direction where to apply the PDDP. Notice how the choice of the α level is irrelevant to our analysis, since we rely only on the p-value, regardless of the fact that the test has confirmed or not the null hypotesis. The number C of principal directions to be evaluated is a trade-off between computational complexity and performance. In this application, the choice of C is made once at the beginning, but it can also vary at every step. Investigations about this topic are still ongoing.

The first three steps of the so-built algorithm (we call it the mPDDP algorithm) are then equal to the old ones, whereas the others need to be reformulated. The overall procedure looks as follows.

mPDDP clustering algorithm

1. Compute the centroid $w \in \mathbb{R}^{m\times 1}$ of Φ
2. Compute the auxiliary matrix $\breve{\Phi} = \Phi - e \cdot w^T$, where $e \in \mathbb{R}^{N\times 1}$ is column vector of ones, namely $e = [1, 1, \ldots, 1]^T$
3. Compute the Singular Value Decompositions (SVD) of $\breve{\Phi}$, $\breve{\Phi} = U\Sigma V^T$, where Σ is a diagonal $N \times m$ matrix, and $U \in \mathbb{R}^{N\times N}$ and $V \in \mathbb{R}^{m\times m}$ are orthonormal unitary square matrices
4. Choose the number C of principal components to evaluate, $1 \leq C \leq m$. Take the first C columns of V, $v_c \in \mathbb{R}^{m\times 1}$, $c = 1, \ldots, C$. Compute the projections of all the data onto this principal directions, $z_c = \breve{\Phi} \cdot v_c$, $z_c \in \mathbb{R}^{N\times 1}$
5. Perform a Chi-squared goodness of fit test on the z_c vector of projected data, with a significance level α and compute the p-value p_c
6. Find $j = \arg\min_c p_c$ to find the index of the test which gave the lowest p-value.

7. Divide the data into two sub-clusters X_L and X_R, according to the following rule:

$$\begin{cases} x_i \in \Phi_L & (x_i - w)^T \cdot v_j \leq 0 \\ x_i \in \Phi_R & (x_i - w)^T \cdot v_j > 0, \end{cases} \tag{8.5}$$

8. Iterate until the desired number of clusters is reached.

In Figure 8.3, the new strategy is applied on the simulation example. Notice that now, as expected, the cluster is split according to the second principal component, unlike using standard PDDP.

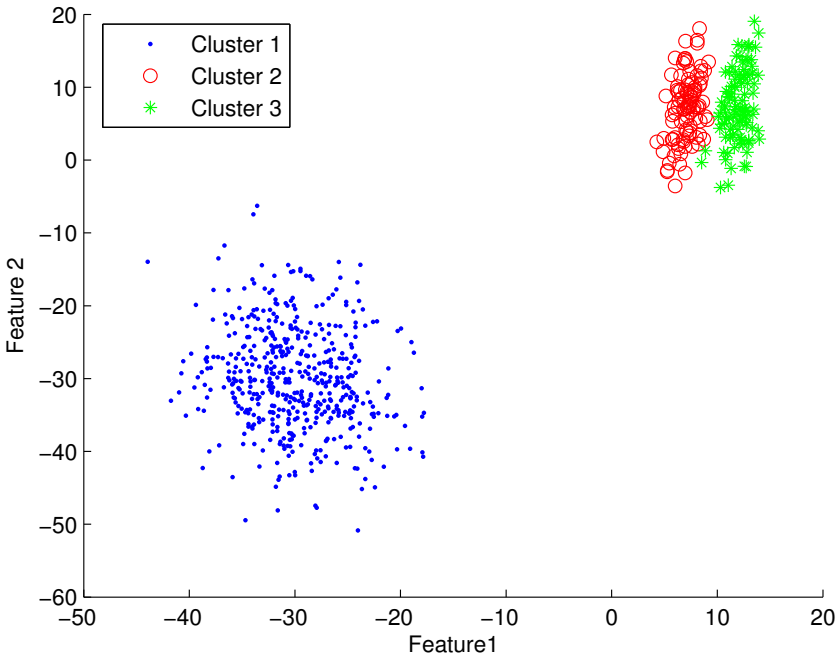


Figure 8.3: Clustering produced by the mPDDP algorithm at the second step

8.4 Application to fault detection

In the following Section, the mPDDP algorithm is applied to data measured from the test bench describen in Chapter 5. It should be noticed that this is only a benchmark on which we want to test the mPDDP method, and it is not meant to provide a complete solution to the aforementioned problem. In particular, the injected faults were different from the ones described in Chapter 5, and consisted in:

- One raceway slightly clogged
- Two raceways slightly clogged
- Worn balls

The adopted approach was a data-driven one, similar to that of Chapter 7 The training data were collected for different types of input position profiles, with the aim to excite the system in all its components. The type of input profiles was trapezoidal one, with different amplitudes and speed. For each input, the data were collected for every fault conditions, with a sampling frequency of $f_s = 10\,\text{kHz}$. The variables available for the acquisitions were:

1. Position set-point
2. Position measured
3. One phase current (phase A)

The load profiles were set to zero for each experiment.

Recent studies [129, 130] report that existing current and position/speed sensors equipping aerospace EMA are a promising tool for health monitoring of electromechanical actuators based on screw systems. Therefore, features to be used for FDI are computed based on the current signals from the various fault types and profiles. Each feature sample is computed via an overlapping moving window, with length of 3 s and overlapping length of 1.5 s.

In this application, up to 21 features have been computed (on the phase current signal if not specified), spanning from time domain, frequency domain, and time-frequency domain. Time domain features include general purpose indexes, like Root Mean Square (RMS) value, skewness, kurtosis, sixth central moment, shape and crest factors, peak-to-valley value, energy operator [131, 108, 85], and application specific indexes, like position error [132] and the torque-speed ratio. Frequency domain features consist mainly of the magnitude of the Fast Fourier Transform (FFT) over 3 sets of frequencies. Three indexes of these type have been extracted, based on the value of the magnitude at different frequency bands. Other features are mean frequency, frequency center, RMS and standard deviation in the frequency domain [85]. The remaining features in time-frequency domain consisted in the magnitude of a current's spectrogram at different frequency bands.

After the computation of the features, any point in the feature space is m-dimesional, with $m = 21$. In order to be able to visualize the clustering

results, a feature selection step is performed. Specifically, the method of the Linear Discriminant Analysis (LDA), which goes back to the pioneering work of Fisher [133], is used. The LDA is a supervised method, which means that the class which data belong is given to the algorithm; the dimension reduction is achieved via a linear combinations of the existing features, by seeking the direction in the m-dimensional space along which the classes are best separated. This can be done by maximizing the Fisher Discriminant Ratio, which, for the two-classes case, it is equal to:

$$F = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2},$$

where μ_1 and μ_2 are the mean of the class one and two after the projection along the best direction, respectively, and with σ_1^2, σ_2^2 are the variances of the class one and two after the projection. These parameters are scalar values after the projection along the best direction. Notice that F is large if the classes are well separated. The method can be straightforwardly extended to be used in the multi-class case [134]. The peculiarity of this technique is that it produces a number of features which is at most equals to the number of classes minus one. So, in this case, since there are 3 classes, after this step the feature space changes from 21-dimensional to 2-dimensional, and the new features, which we can call "Feature1" and "Feature2", are linear combination of the previous ones.

After that the feature extraction and selection phases have been performed, we can apply and observe the clustering results. A point belongs to the cluster at minimum Euclidean distance, computed respectively to the cluster center. Here, it is possible to compare the standard and modified PDDP algorithms. Figure 8.4 and Figure 8.5 show the real bounds (solid) and the boundaries found by the clustering algorithms (dashed). The three considered faults are highlighted using three different colors and the misclassified points are put in evidence with surrounding circles. It can be noted that the boundaries found by the new algorithm are closer to the true ones than the stripes selected by the standard algorithm. This fact produces a much better detection rate and a much smaller number of misclassified points, as summarized in Table 8.1.

For the sake of completeness, also the performance of the k-means [135] and the fuzzy k-means [136] algorithms are evaluated. The results with such methods are shown in Figure 8.6 and 8.7, whereas the main quality indeces are summarized in Table 8.1. Notice that the proposed mPDDP algorithm

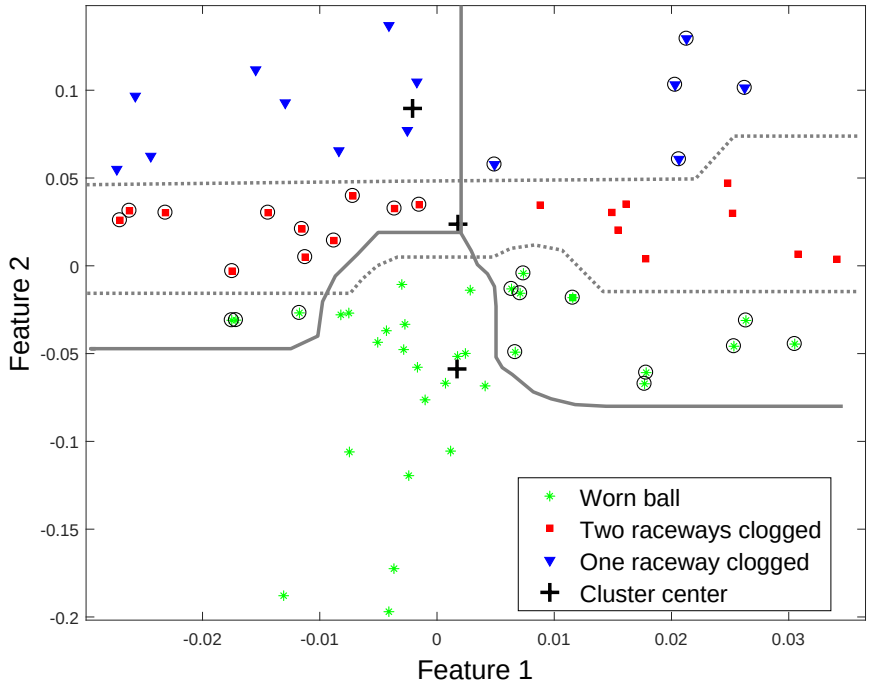


Figure 8.4: Performance of the standard PDDP algorithm. True boundaries (solid lines), obtained boundaries (dotted lines), misclassified points (circled)

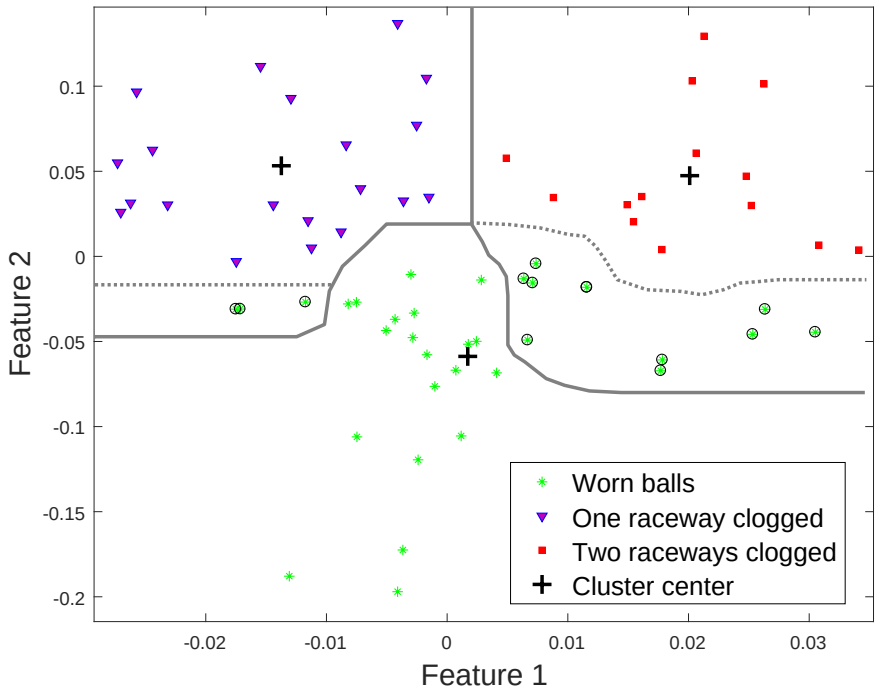


Figure 8.5: Performance of the proposed mPDDP algorithm. True boundaries (solid lines), obtained boundaries (dotted lines), misclassified points (circled)

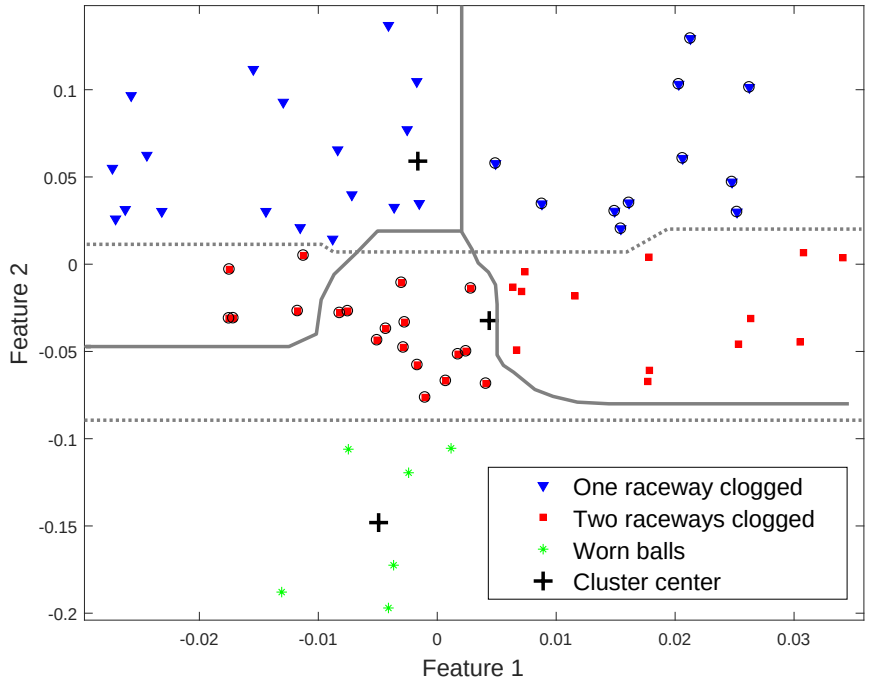


Figure 8.6: Performance of the k-means algorithm. True boundaries (solid lines), obtained boundaries (dotted lines), misclassified points (circled)

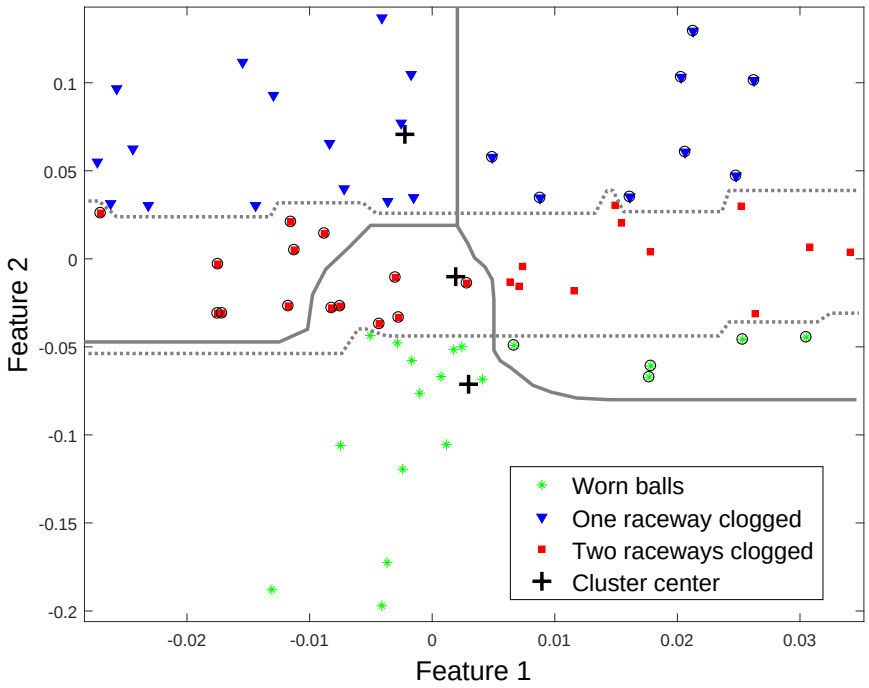


Figure 8.7: Performance of the fuzzy k-means algorithm. True boundaries (solid lines), obtained boundaries (dotted lines), misclassified points (circled)

Table 8.1: Results of classification

Algorithm	**Misclassified points**	**Detection rate**
mPDDP	13/67	0.1940
PDDP	29/67	0.4328
k-means	30/67	0.4478
fuzzy k-means	27/67	0.4030

outperforms also this different clustering approach.

8.5 Conclusions and future developments

Principal Direction Divisive Partitioning is among the most popular techniques in the clustering framework. However, in this work, it is shown that PDDP may provide wrong results in case of particular distribution of the data in the features space. Therefore, a modified version of the PDDP algorithm is proposed, called mPDDP, based on a Chi-squared statistical test. The proposed method showed to be very effective when applied on experimental data taken from an aerospace EMA setup (where suitable faults were simulated ad-hoc). Future work will be dedicated to the theoretical analysis of the proposed approach. Moreover, different applications will be addressed to better understand the potential of the proposed approach with respect to other existing methods.

CHAPTER 9

Control-oriented modeling of SKU-level demand in retail food market

In food market, modeling the dynamics of Stock-Keeping Unit (SKU) requests is of fundamental importance, not only to understand the market, but also for optimization and control purposes. In fact, standing on model-based predictions of future demand, an efficient planning of the promotional calendar can be devised. Moreover, better inventory management can be achieved, by reducing losses due to expired aliments remained unsold and improving distribution operations. In this work, a data-driven control-oriented modeling of such a demand is discussed and a novel switching dynamical strategy is proposed. When applied to experimental data from a real food company, the above strategy is shown to accurately predict future sales under fixed promotion events.

9.1 Introduction

The ability to accurately forecast the future demands of goods and services gives a clear competitive advantage over competitors, leading to higher profits by constantly optimizing operation management tasks, such as inventory management, planning and scheduling [137]. In the context of food retail market, Stock-Keeping Unit (SKU) request prediction is of paramount importance to reduce losses caused by expired aliments that remained unsold, enhance customer satisfaction and improve distribution operations. One of the main

factors contributing to sold quantity variation is the employment of promotional events by the retailers. Usually these are planned collaboratively by them and manufacturers, who jointly agree on the products, types, price reduction and the timing of promotions. Products are typically on promotion for a limited period of time, in the order of days, during which demand is usually substantially higher than during periods without promotions [138]. Here, the authors also identified that the display location of items in retail stores, weather and holiday periods had a positive impact on sales. This dramatical change in sales behaviour (*uplift*) with respect to the quantity sold during non-promotional days (*baseline*) makes standard inventory management and replenishment techniques (e.g. based on reorder point/order quantity policies) not suitable for an efficient management. There is therefore a need for more intense collaboration, such as Collaborative Planning, Forecasting and Replenishment (CPFR) procedures [139, 140].

However, despite the benefits of CFPR, collaboration and information sharing does not prevail [141]. When collaborative information is not available, the supplier must rely on historical data and qualitative knowledge of the marketplace to build the forecast. The effectiveness of data-driven decisions on firm performance has been assessed in [142]. To tackle this problem, two basic approaches have been considered: judgemental and quantitative forecasting [143]. A combined approach, aiming at supporting human decisions on predictions provided by the statistical model, has been taken into consideration in [144]. Some of the most used models in literature consist of the Simple Exponential Smoothing (SES) [145], the "last-like promotion" [146] and the Autoregressive Distributed Lag (ADL) model [145]. The SES model, shown to be efficient in capturing the level component in demand over time, does not make use of any promotional information: when the characteristics of the demand series change due to special events such as promotions or holidays, the fitted parameters can no longer describe the series. The so-called "last-like promotion" model is one of the simplistic models that use exogenous variables. This method first generates a baseline forecast using a model, such as the SES, for non-promoted time periods. A "lift effect" is then added to the baseline forecast during the promoted periods. The ADL model considers instead various types of predictors, such as: past values of demand of the considered product, price of the considered product, promotional index of the considered product, price of the competitors products, promotional indices of the competitors products, monthly indicator variable and calendar events. Many studies have focused on evaluating the effect of promotions on sales, using

store or market level data [147, 148]. Authors in [149] proposed a hierarchial model at single Universal Product Code (UPC) level, while [150] investigated the presence of outliers in developing promotional sale forecasting methods. The work of [151] was focused on predicting the sales of a new launched product, and [152] simulated customers behavior to estimate the possible cart content and assess the changing in sales under different price conditions.

The contribution of this work contribution is twofold: (i) a switching dynamic model is proposed to perform both baseline and uplift forecasting; (ii) an imputation methodology, based on a similarity metric, is employed when promotional information are not present for a particular customer. The development of a switching model encompasses the benefits of both the last-like promotion and ADL models. The benefits of the proposed approach are the identification of a parametric model which is able to perform forecasts at any prediction horizon, encapsulating the different aspect of each promotion. Furthermore, when historical data are few, the ADL model structure can be too complex: the proposed method is shown to be reliable even with a low number of data, and domain specific KPIs (Key Performance Indexes) are introduced. The comparison of clients via a similarity measure can be used to produce visualizations and dashboards, enhancing the company understanding and decision making by revealing hidden groups and structures. In this view, the use of customer clustering is not new [153]. However, in the aforementioned work clustering is used as a solution to build a lower number of models. Nowadays, provided that technological resources are available, it is not uncommon to deploy thousands of models into production [154]. In this work, stores comparisons are instead used to recostruct missing promotional information at customer level.

The remainder of the chapter is organized as follows. In Section 9.2, the business problem tackled in this work is stated. In Section 9.3, the steps involving the predictive model design and missing data imputation are presented. Section 9.4 highligths the main performance indexes used to assess the approach validity, and a comparison with known methods is performed. Section 9.5 is devoted to concluding remarks and future developments.

The whole analysis and method development will be supported by the use of experimental data taken from a real food company. For confidentiality reasons, data will be normalized and some details on the specific business will be omitted.

9.2 Problem statement

The aim of this work is to evaluate the effect, that is, the sold quantity, generated by a different set of promotion types, for different products, for every customer of a food producer company. Forecasts are relative to the total delivered quantity in a working week. Correct predictions are beneficial for both the customers and producer. In the first case, there will be a more focused planning of the promotional calendar, while for the producer side, better inventory management and improved line production efficiency will be of direct conseguence.

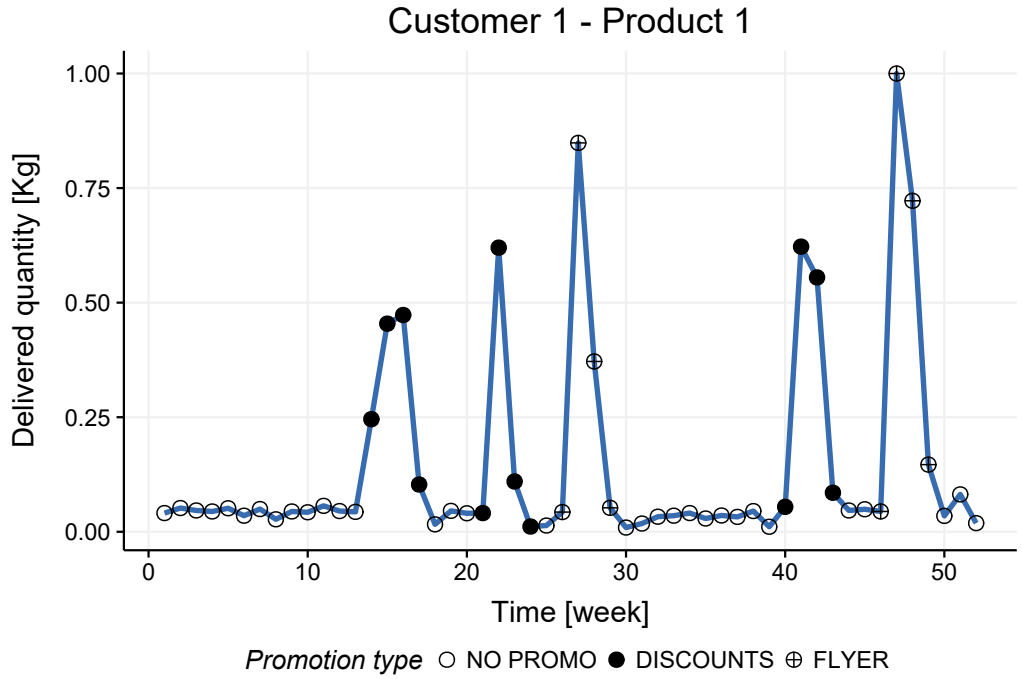


Figure 9.1: Sales over a one year period. Baseline and uplifts phases are highlighted. The promotion of each week is represented by different symbols

The dataset employed in this work consists of 185 customers, 3 products (referred as Product 1, Product 2 and Product 3) and 8 promotion types (No Promo, Discount, Flyer, Collection Points, Flyer 1+1, Flyer discounted, Discount 1+1, Other). In each week, only a promotion can be present. A tipical behaviour of a sold quantity, normalized to protect sensible information, with and without promotional events, is depicted in Figure 9.1. The promotion type "No Promo" indicates the absence of a promotion. The company needed to plan the working 2 weeks in advance, due to constraints on logistics, stocks and production management. The ultimate goal of a forecasting algorithm is to

take optimal actions, which have a positive impact on the company economy, as discussed in the introduction. In this view, this work can be stated in a control system framework, where the focus is on the control component in Figure 9.2.

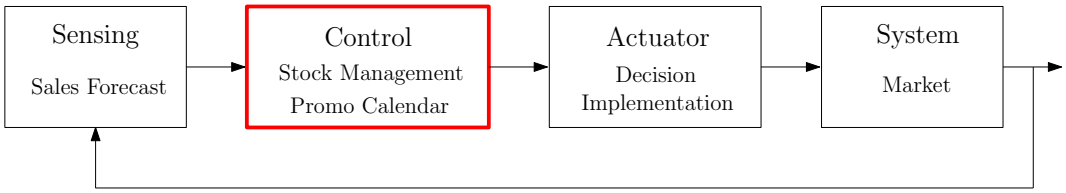


Figure 9.2: Recasting of the business question as a control design problem: standing on the prediction of future requests, a promotion calendar can be modified or the inventory can be efficiently managed

This boils down to the identification of a proper sales model (considered as the process under control), which will be used to infer future sales. This, accordingly, results in specific actions undertaken on production lines (the "Actuator" block). Once the forecasted quantity has been observed ("Sensing" block), corrective actions can be taken, both in the form of automatic model updating or human-in-the-loop interventions (the "Control" component). The feedback is generated by the fact that operational actions (select when and what promotion to apply, inventory replenishment decisions) affect the market. By looking at the market responses, new information about the effectiveness of certain promotion types and period of their application are gathered, improving the decision-making process. This framework permits an efficient, self-improving management of stocks and production operations. Similar control-oriented approaches applied to the business sector, such as inventory management, can be found in [155, 156].

In order to evaluate the goodness of the proposed approach, three different KPIs were defined:

- **Integral Error during Promotion (IEP)**: to quantify the total error during promotional weeks
- **Mean Baseline Error (MEB)**: to check the error committed during non-promotional weeks
- **Maximum Uplift Error (MUE)**: to evaluate the maximum error committed in forecasting a single promotion effect

The proposed performance measures are considered more representative of the forecasting problem facets, with respect to standard evaluation metrics, such as

the Mean Absolute Error (MAE) [157, 146], or the Mean Average Percentage Error (MAPE) [149]. In [158], the author showed the drawbacks of using such standard indicators, and proposed a metric based on the logarithm of the ratio between predicted and actual demand, called Log Accuracy Ratio (LAR). Nonetheless, MAE and MAPE metrics are reported for clarity, along with the LAR index. Information regarding customers sales were gathered from the company business intelligence software.

9.3 Sales prediction

This section describes the proposed approach to solve the forecast of promotional events' effect, illustrating first the parametric model identification procedure undertaken, and then the missing promotional information imputation strategy.

9.3.1 Covariate selection

For each customer-product couple, its relative dataset consisted in 104 weeks of historical data, with aggregated sales for each week. The data covered the 2014 and 2015 years. The data sampling time can therefore be considered as *one week*. The dependent variable is the sold quantity, while chosen regressors consist of the promotion at each week (to evaluate the promotion effect on sales) and the period each week belong to (in order to obtain a stagionality effect). The variable "Period of the year" is computed by subdividing the weeks in a year into 6 periods (January-February, March-April, May-June, September-october, November-December). Given the assumption that previous sales level and promotions have an effect on the current delivered quantity, two additional covariates where computed from the existing ones. The new variables consist of the quantity delivered 2 weeks before the current time (in order to ensure the 2 weeks forecast horizon), and the promotion present in the week before the current one. Each week t is then described by the following set of variables:

1. Delivered quantity: $y(t)$ (*quantitative dependent variable*)
2. Delivered quantity lagged of 2 timestamps: $y(t-2)$
3. Promotion type: $p(t)$ (*categorical variable, 8 levels*)
4. Promotion type lagged of 1 timestamp: $p(t-1)$

5. Period of the year: $d(t)$ (*categorical variable, 6 levels*)

Given the low number of training points with respect to the number of regressors, a linear model structure was chosen to perform evaluations. The identification of a linear model is supported by the literature reviewed in Section 9.1: however, proposed methods like the ADL model, requires too many information which can not be available when a company has just started its business or does not have enough historical information. The proposed methodology consists in a switching model formulation, in order to estimate correctly both uplifts and baseline behaviours. Referring again to Figure 9.1, it can be observed that the dependence of baseline periods on previous sales quantity is not evident, as opposite to uplift peaks. For this reason, the switching behaviour includes a static model which is more thrifty, being able to manage the low data available for the identification. During promotional events, a dynamic predictor is employed, which makes use of past information. During weeks with no promotions, the static model estimates the baseline value:

$$\hat{y}(t) = \begin{cases} \alpha_d \cdot y(t-2) + \beta_d\big(p(t)\big) + \gamma_d\big(p(t-1)\big) + \mu_d\big(d(t)\big) + \\ +\kappa \cdot [y(t-2) - \hat{y}(t-2)] \quad \text{if } p(t) \neq \text{NO PROMO} \\ \\ \beta_s\big(p(t)\big) + \mu_s\big(d(t)\big) \qquad\qquad \text{if } p(t) = \text{NO PROMO} \end{cases} \tag{9.1}$$

This switching strategy is made possible because the promotion calendar is known since the beginning of the year. The developed tool will help to improve the definition of the time and type of promotions for each week of the year, depending on the specific customer and product. Practical benefits of the proposed methodology are: i) independent estimation of baseline and uplift scenarios; ii) linear parametric structure, which makes simple the imputation of missing data; iii) effective even when the number of historical data is scarse.

The variable $\hat{y}(t)$ represents the predicted delivered quantity at time t, with α_d, β_d, γ_d, μ_d the dynamic system estimated coefficients, and β_s, μ_s the static system estimated coefficients. The notation of the promotion coefficients β_d, γ_d, β_s as function of the promotions at a certain time, indicates that various model with the same slope but different intercepts are being fitted by the linear model method. Variables $p(t)$ and $d(t)$ were coded as factor variables with defined levels, and $y(t)$ is a numerical quantity. The parameter κ, chosen heuristically given the impossibility to perform proper cross-validation with the small data sample at disposal, controls the entity of the correction based on

the previous 2-step prediction error. The model parameters are then estimated via least squares.

The model is trained incrementally each week, as more data are at disposal. This ensures that all available information are used and taken into account when performing forecasts. The initial training data set consists of the year 2014 data (52 observations), and predictions for the year 2015 in its entirety are performed. Then, the model is updated with data from the year 2015, and forecasts are recasted with the up to date sales information. As depicted in Figure 9.3, it is possible to observe how the model does a good job into estimating both uplifts and baseline values. It is to be noticed that the error on the promotion "Discount", represented by a filled circle, decreases as more data on this particular promotion type become available. The system is therefore obviously expected to perform even better with the arrival of new data.

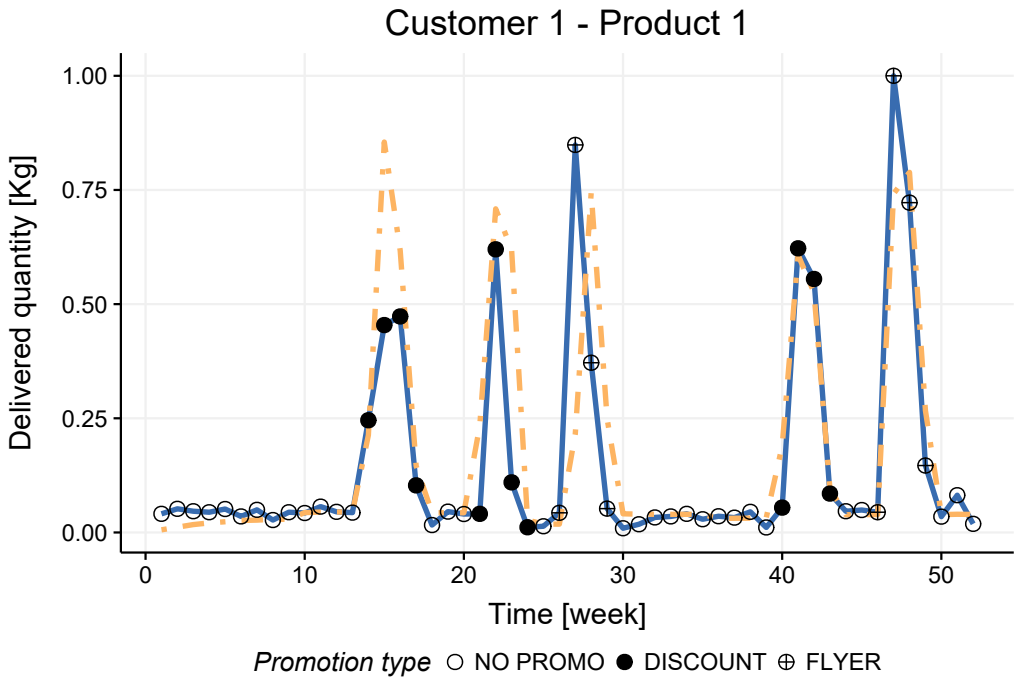


Figure 9.3: Product 1 delivered quantity (continous blue line) and forecast (dot-dashed orange line). The large error on the first promotion uplift is due to the lack of information on that particular promotion in the training data (i.e., before week 13 of 2015).

The behaviour of the Product 2 for a specific customer 2, see Figure 9.4, it is somewhat different with respect to the sales profile showed in Figure 9.3. Although uplift peaks are still visibile, period with no promotion does not seem to differentiate much with respect to promotional weeks. Nevertheless, the

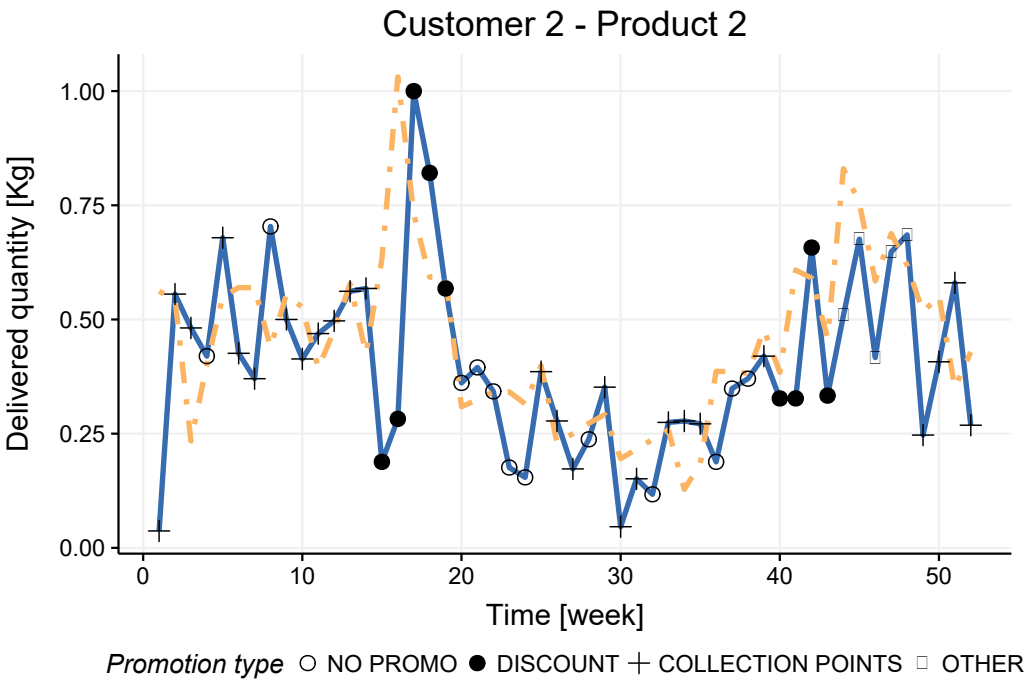


Figure 9.4: Product 2 delivered quantity (continous blue line) and forecast (dot-dashed orange line).

forecasts are still able to capture the sales general trend. The plot in Figure 9.5 reflects how the feedback correction acts, allowing a higher prediction at week 21 to compensate the underestimation of sales at week 19. From week 30, the Product 3 was no more sold in Customer 3 salespoints. The algorithm is able to cope with this sudden sales interruption. An interesting aspect is that the promotion "Flyer" in Figure 9.3 is not present in the training data prior to week 28 of year 2015. To overcome this problem and being able to perform the required predictions, an imputation procedure based on a similarity measure between customers has been employed.

9.3.2 Missing data imputation

In order to retrieve the missing promotional information, customers were compared by the euclidean distance measure, in the vector space defined by the following two features:

- **Mean baseline value**: the mean quantity sold during baseline periods
- **Mean uplift value**: the average increase in sales during promotional events. This feature is computed by taking the delivered quantity values

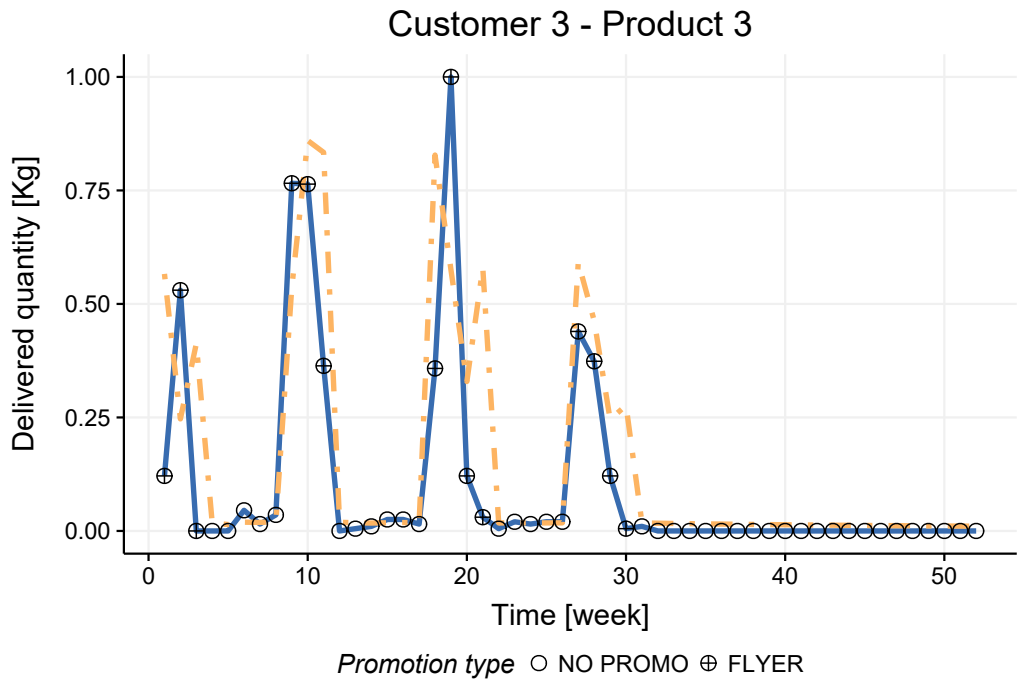


Figure 9.5: Product 3 delivered quantity (continous blue line) and forecast (dot-dashed orange line).

at the beginning and at the end of the promotion, and their mean computed. Then, this value is subtracted from the maximum value of sold quantity during the promotion period. The procedure is repeated for each promotion of that particular client, giving raise to a number of indexes equal to the number of promotions of that client. Finally, their mean is computed and taken as a feature

The adopted representation of clients in a 2-dimensional space permits a rapid and effective distance computations, given the number of stores (185 in this study). An higher dimensional space would have led to less significative comparisons due to the curse of dimensionality. As can be seen in Figure 9.6 , the clients form one large cluster, with the upper-right customer considerable as another category. In order to impute missing promotional information for a client, its nearest customer (which has an evaluation of the missing promotion), is taken. Once the searched client is found, the model in Equation 9.1 is identified, and the coefficients corresponding to the missing promotions are used to evaluate the information not present. If the nearest customer does not have all required information (but only a subset of the missing information), the second nearest store is used, and so on. The results of this mapping can

be used to interpret how a client is positioned into different market segments. The procedure has therefore been useful to develop a deeper understanding of the customer portfolio for the company.

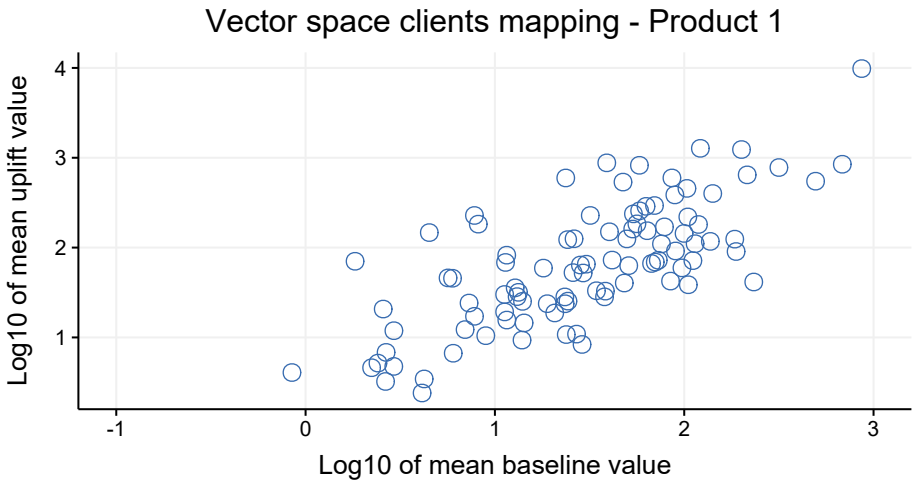


Figure 9.6: Clustering induced on Product 1 clients.

9.4 Performance assessement

In this section, the proposed model performances are compared with standard benchmark systems.

9.4.1 Last-like promotion benchmark model

The first benchmark consists in the last-like promotion model, as implemented in [146], with the minor modification of taking values two steps in the past. This method, also known as exponential smoothing with lift adjustment, contains a switching logic like the algorithm proposed in this paper. The philosopy behind the benchmark model is the following. If there is no promotion in the incoming week, the forecast is the smoothed value of the non promotional weeks in the past; else, the last observed lift amount is added to the smoothed value to obtain the prediction:

$$\hat{y}_l(t) = \begin{cases} z(t) + L(t) & \text{if } p(t) \neq \text{NO PROMO} \\ \\ z(t) & \text{if } p(t) = \text{NO PROMO} \end{cases} \tag{9.2}$$

and

$$z(t) = \begin{cases} z(t-2) & \text{if } p(t) \neq \text{NO PROMO} \\ \\ (1-\tau)z(t-2) + \tau y(t-2) & \text{if } p(t) = \text{NO PROMO} \end{cases} \tag{9.3}$$

with $\hat{y}_l(t)$ denoting the forecast at time t of the last-like promotion model, $z(t)$ refers to the smoothed number of items sold up to week t, based on non promotional weeks. The value of τ used is 0.2, based on the traditional default value for exponential smoothing. The lift amount $L(t)$ is computed as the difference of the actual sales at time t and the smoothed non-promotion sales at the time of the most recent promotion, $L(t) = y(t-2) - z(t-2)$. The model behaves as a "persistency" one, replicating the uplift behaviour two weeks in the future.

9.4.2 ARMAX benchmark model

The ARMAX (Autoregressive Moving Average with eXogenous inputs) model provides a general framework for modeling short-memory time series equipped with an external inputs $u(t)$:

$$A(q)y(t) = B(q)u(t) + C(q)e(t) \tag{9.4}$$

where $e(t)$ is a white noise error term, q is the lag operator such that $q \cdot y(t) = y(t+1)$, and $A(q)$, $B(q)$, $C(q)$ are the polinomials of the autoregressive, exogenous and moving average part respectively. In this work, the most suitable model structure was found to be an ARMA(1,1) model, with indicators variable for period of the year and promotion type. The model was retrained each week, and a 2-step forecast computed.

9.4.3 Comparison of results

A comparative overview of the proposed approach performance (on a client without missing promotion), with respect to the benchmarks described in the previous sections, is reported in Figure 9.7. As can be seen, the last-like promotion model behaves like a persistence model, repeating the delivered quantity of two week before. The ARMAX model correctly detected the promotion periods, but it is not able to "reset" its prediction to the baseline value when the promotion ends. The proposed model is able to correctly detect

the timings of a promotion, its effect, and rapidly switch to the baseline trend. A quantitative comparison of the described methods is reported in Table 9.1 and Table 9.2, where the best results are highlighted in bold text. The proposed method is the best for 6 out of 7 KPIs. Major improvements lie in the baseline and uplifts value estimation. The ARMAX model instead obtains a better result on the integral error. Notice however that the proposed model is not designed to optimize such a measure but for a pointwise estimation.

Table 9.1: Comparison results: standard metrics

Method	**MAE**	**MAPE**	**LAR**
Proposed	**0.171**	**195%**	**12.655**
Last-like	0.418	249%	35.404
ARMAX(1,1)	0.248	358%	18.781

Table 9.2: Comparison results: proposed metrics

Method	**IEP**	**MEB**	**MUE**
Proposed	0.083	**0.005**	**-0.186**
Last-like	-0.094	0.038	-0.756
ARMAX(1,1)	**0.019**	0.056	-0.514

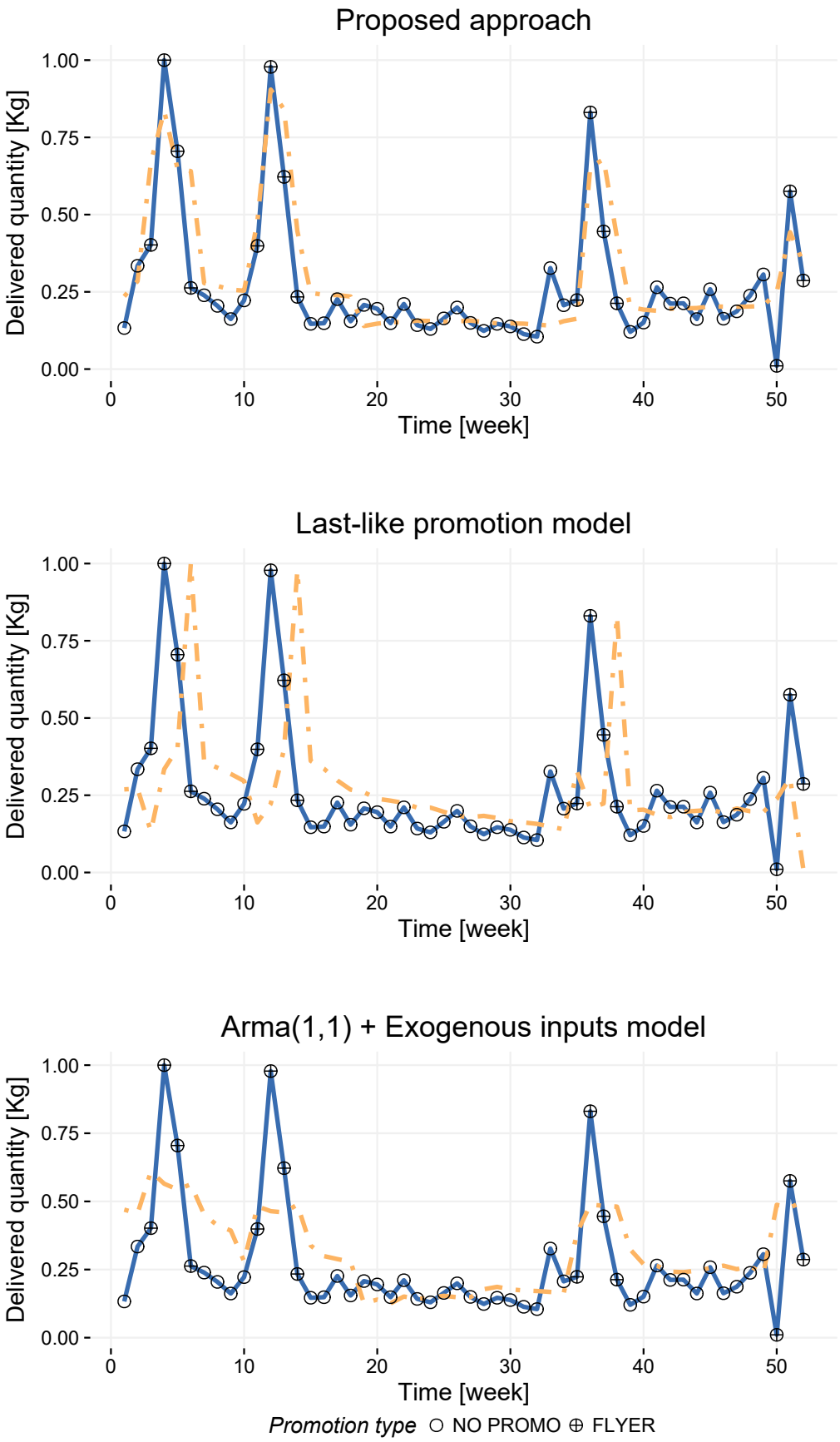


Figure 9.7: Comparison of delivered quantity (continous blue line) and forecasts (dot-dashed orange line) with the proposed method and the benchmarks models.

9.5 Conclusions and future developments

This work presented an innovative approach to tackle the forecast of sales delivery in presence of promotional events. The proposed methodology consists in a switching linear structure, which is able to correctly predict both baseline and uplifts sale periods. The identified model has been shown to be particularly effective in presence of a low amount of training data for the identification stage. When no prior information on the effect of a promotion are available, an imputation procedure based on customer similarity has been proposed and employed with success. Furthermore, a performance assessment and a comparison with state of the art methods has been carried out, introducing new and domain specific KPIs which better reflects the forecasts accuracy from different points of view. Future research will be focused on the integration of the proposed modeling approach within a closed-loop business decision process, and the use of hierarchical models to better capture the correlations between clients and products.

CHAPTER 10

Conclusions

In Part I of this thesis, we took a journey from the classical landscapes of parametric system identification (Chapter 1) to modern valleys of nonparametric functions' estimation techniques (Chapter 2). The bridge that made this crossing possible is the interest that the system identification community started to cultivate for the related statistical learning one. Leveraging on this fruitful fertilization, this work investigated one, until now, hidden scenario for system identification: the semi-supervised learning framework. In this view, additional inputs points with no corresponding output data can be employed to better conditionate the learning problem. Chapter 3 introduced a new regularization term, called manifold regularization, to the estimation of Nonlinear Finite Impulse Response (NFIR) systems. The method is shown to outperform standard Tikhonov regularization, and has to be seen a first step to incentivate even more the communication and sharing of knowledge between the system identification and the statistical learning fields. In Part II, we discussed some applications of statistical learning methods, mostly in the field of fault detection for electro-mechanical actuators. Motived by the HOLMES european project (Chapter 5), a model-based (Chapter 6) and a data-driven (Chapter 7) methods were proposed. The former methodology was based on a particular variation of the particle filter algorithm. This method treats the fault detection problem as a hybrid system state estimation one, were the system condition (healthy, faulty) is seen a state to be estimated. The latter approach treated the problem as a pure black-box, computing different

types of features on measured signals, and finally training a classifier on the extracted features in order to assign to each data point a particular system condition. In Chapter 8, we developed a modification of a pre-existing clustering algorithm, the Principal Direction Divisive Partitioning (PDDP) clustering strategy. The modified PDDP is shown to generate better cluster with respect to standard PDDP, k-means and fuzzy k-means clustering methodologies, by a fault detection application to data coming from the same experimental setup of previous applicative chapters. In Chapter 9 we faced a different problem, that is, the forecasting of delivered food quantity in the retail market. The aim was to correctly predict the sold quantity during promotional events. In order to solve this, a switching dynamic model has been designed, which is able to forecast promotional peaks and baseline sales, even with the few amount of data from a real food company.

Statistical learning methods are ubiquitous in the challenges poses in the automatic control areas, both the theoretical ones as in Part I, and in the applicative context as in Part II.

As a final note, if this thesis succeeds into light the reader interest for learning methodologies, I will consider it to be a success.

Appendices

APPENDIX A

Topics in learning parametric models

A.1 Bias and variance

The bias-variance tradeoff is related to the approximation-generalization dilemma. The ultimate goal is to have a small E_{out} for a particular identified model, that is, a good approximation of $\mathcal{S}_0$ on new and unseen data. Suppose that the model family $\mathcal{M}$ has been fixed. A particular model is then $M \equiv \mathcal{M}(\theta)$. The out of sample error of a model M is indicated with $E_{\text{out}}(M)$. Since the identified model M depends on the identification (training) dataset $\mathcal{D}$, it is convenient to explicitate this dependence, leading to the notation $E_{\text{out}}(M^{(\mathcal{D})})$ for the out of sample error of model M trained on $\mathcal{D}$. The bias-variance analysis decomposes $\mathbb{E}_{\mathcal{D}}\left[E_{\text{out}}(M^{(\mathcal{D})})\right]$, i.e. the expected out of sample error of the models $M^{(\mathcal{D})}$, each identified on a different realizations of $\mathcal{D}$, into two terms, Figure A.1:

- **Bias**: How well $\mathcal{M}$ can approximate $\mathcal{S}_0$, that is, how much the model $M^* \in \mathcal{M}$ that best approximates $\mathcal{S}_0$ differs from the true system.
- **Variance**: How much the identified model $M^{(\mathcal{D})}$ deviates from M^*, on average over different training sets $\mathcal{D}$.

We can think about the bias as how much our hypothesis space $\mathcal{M}$ is biased away from the true target. Infact, the best model M^* in $\mathcal{M}$ is only limited in its ability to approximate $\mathcal{S}_0$ by the limitations of the model structure $\mathcal{M}$ itself. The bias expresses both *model bias* (due to structural mismatch between model and true system) and *estimation bias* (due to incorret parameters'

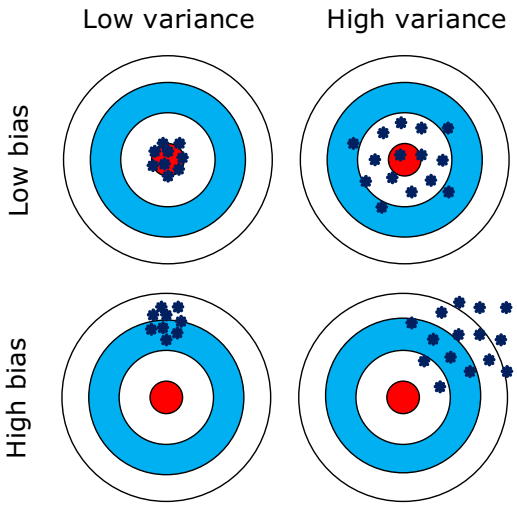


Figure A.1: Graphical representation of bias and variance

value estimation) [4]. The variance of the parameters indicates how much a different training set $\mathcal{D}$ leads to a different learned model $M^{(\mathcal{D})}$. The learned models have the same structure but a different value of the parameters. A more flexible model family has typically smaller bias, since it is easier to be closer to the true system. The tradeoff is that this model will have generally higher variance, since a minimal variation in the data can led to a completely different parameters' estimate. In the case where $\mathcal{S}_0 \in \mathcal{M}$, if the mean of the parameters' estimator coincides with the true parameters' values, the estimator is said to be *unbiased*. In the aforementioned case, the PEM method leads to unbiased models. For Gaussian noise sources, the PEM estimate is also *asymptotically efficient*, provided that the model family $\mathcal{M}$ contains the true system's description. This means that, as $N \to \infty$, the covariance matrix of $\hat{\theta}$ will approach the Cramér-Rao limit, and so no other unbiased estimate can be better than the PEM estimate. If $\mathcal{S}_0 \notin \mathcal{M}$, an unbiased estimate of the $\frac{B_0(q)}{A_0(q)}$ part can still be obtained. Suppose that $\mathcal{M}$ is an ARX model and $\mathcal{S}_0$ is ARMAX, with $\frac{B_0(q)}{A_0(q)} \in \frac{B(q,\theta)}{A(q,\theta)}$. In this setting, the *instrumental variable* method [3] can be employed to obtain unbiased estimates of the input-output transfer function. The phenomenon of *overfitting* happens when decreasing the in-sample error leads to increasing out-of sample error, Figure A.2. Overfitting is primarly caused by the model's variance, and can happen even if there is no noise in the data, due to an excessive higher complexity of the unknown function that has to be found, with respect to the chosen model's complexity [10]. In order to balance the bias and the variance components, different methods have been developed, mainly by penalizing overly complex models.

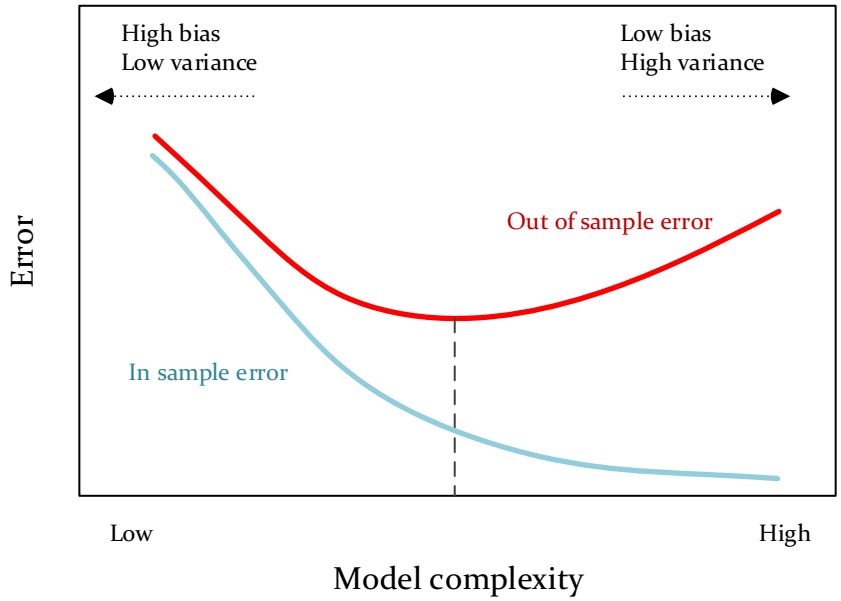


Figure A.2: Representation of the overfitting phenomenon. In the region to the rigth of the vertical dashed line, the model is overfitting the data

A.2 Model order selection

Suppose that the model's structure has been fixed. Model order selection is a way to prevent overfitting, by selecting the optimal model's order, as defined by a specific measure. In the case where there is a sufficient number of data, cross-validation is the best method to assess the model's performance. When the data is limited, a number of measures have been introduced to estimate the out-of-sample error using only training data. The simplest method for defining this metric is to penalize the in sample error with a term that is proportional to the complexity of the model. A common penalty term introduced to penalize model's flexibility is the Akaike Information Criterion (AIC) [5]:

$$\text{AIC} = \ln\Big(J_N\big(\hat{\theta}; m\big)\Big) + 2\frac{m}{N}, \tag{A.1}$$

where $\hat{\theta} \in \mathbb{R}^{m\times 1}$ is the parameters' estimate using the model order m. To use AIC for model selection, we simply choose the model giving smallest AIC over the set of models considered. Another similar penalty is (under Gaussian disturbances) the Bayesian Information Criterion (BIC) [2, 159] or Rissanen's Minimum Description Length (MDL) criterion [160]:

$$\text{BIC} = \ln\Big(J_N\big(\hat{\theta}; m\big)\Big) + \ln(N)\frac{m}{N}, \tag{A.2}$$

which tends to penalize complex models more heavily than AIC, when $\ln(N) > 2 \Rightarrow N > 8$ (condition which is almost always met). A third technique is the Final Prediction Error (FPE) method [3]:

$$\text{FPE} = \frac{N+m}{N-m} \cdot J_N\big(\hat{\theta}; m\big), \tag{A.3}$$

where, under the hypothesis that $m \ll N$, we have that $\ln(\text{FPE}) = \text{AIC}$ and the two methods lead to the same result.

The AIC and BIC criteria are based on *in-sample* computation: the model goodness is tested on the errors made on the same training data used to fit the model. While this leds often to accurate model selection [4], it is not likely that future data will be the same as the training ones. In order to test the model's performance on *unseen* data, i.e., its generalization capability, the *cross-validation* method is preferable. The goal is to obtain an estimate of the prediction capability of future data of the model in correspondence with different choices of m. The selection of the parameters' number is thus performed by optimizing the estimated prediction score. Holdout validation is the simplest form of CV: the available data are split in two parts, where one of them (estimation set) is used to estimate the model, and the other one (validation set) is used to assess the prediction capability. By ensuring independence of the model fit from the validation data, the estimate of the prediction performance is approximately unbiased. More advanced cross-validation techniques include the $k-$fold cross-validation, where data are split in k parts instead of only two parts. The model is trained for each combination of $k-1$ folds and tested on the remaining one. Then, the average error over k folds is computed. This gives also an indication about the standard error of the mean estimate.

Bootstrap methods are an alternative to cross-validation. In particular, the ".632+" estimator [161] can be used to assess the generalization performance of the learned model. In both CV and bootstrap cases, the estimated error is *not* strictly the out of sample error $E_{\text{out}}(M^{(\mathcal{D})})$, but rather the *expected* out of sample error $\mathbb{E}_{\mathcal{D}}\left[E_{\text{out}}(M^{(\mathcal{D})})\right]$, see [4].

A.3 Empirical Bayes

By casting the regularized regression problem into the Bayesian framework, we have defined in (2.44) and (2.45) the variables γ^2 and σ^2. These variables are defined as hyperparameters, since they govern how the unknown parameters

are estimated. In most of the cases, the exact value of these hyperparameters is not defined, since we do not know the noise's variance or the prior's variability. It is then intuitive to put a prior also on the hyperparameters, leading to a *hierarchical model*, where each layer is conditionally dependent from the layers above [162]. This is called a *Full Bayes* approach. In this setting, the prediction y^* at a new point x^* is given by the *predictive distribution*:

$$p(y^*|x^*, Y) = \iiint p(y^*|x^*, \theta, \sigma)p(\theta|Y, \gamma, \sigma)p(\gamma, \sigma|Y)\ d\theta\ d\gamma\ d\sigma, \tag{A.4}$$

where $p(y^*|x^*, \theta, \sigma) = \mathcal{N}(x^{*T}\theta, \sigma^2)$ is the conditioned distribution of the data (independent of γ and Y), the parameters' posterior $p(\theta|Y, \gamma, \sigma) = \mathcal{N}(\mu_{\theta|Y}, \Sigma_{\theta|Y})$ with $\mu_{\theta|Y}, \Sigma_{\theta|Y}$ given by (2.50) and (2.51) respectively, and the posterior distribution $p(\gamma, \sigma|Y)$ that acts as the prior distribution on the hyperparameters. The complete marginalization of the distribution (A.4) with respect to θ, γ, σ is often analytically intractable. For this reason, a number of computational methods have been developed to approximate the solution, such as Markov Chain Monte Carlo (MCMC) [163] and Integrated nested Laplace approximation (INLA) approaches [164]. An alternative approach is given by the *Empirical Bayes* method [162, 165]. This approach is also known as *type-2 maximum likelihood* [166], or *generalized maximum likelihood* [167], and in the machine learning literature is also called the *evidence approximation* [168, 169]. If the posterior distribution $p(\gamma, \sigma|Y)$ is sharply peaked at values $\hat{\gamma}$ and $\hat{\sigma}$, then the predictive distribution is obtained simply by marginalizing over θ, in which γ and σ are fixed to the values $\hat{\gamma}$ and $\hat{\sigma}$, so that:

$$p(y^*|x^*, Y) \simeq p(y^*|x^*, Y, \hat{\gamma}, \hat{\sigma}) = \int p(y^*|x^*, \theta, \hat{\sigma})p(\theta|Y, \hat{\gamma}, \hat{\sigma})\ d\theta. \tag{A.5}$$

The aim is therefore to find the hyperparameters' values $\hat{\gamma}$ and $\hat{\sigma}$ which maximize $p(\gamma, \sigma|Y)$, and use them to compute the approximation (A.5). In order to do this, it is possible to notice that from Bayes' theorem:

$$p(\gamma, \sigma|Y) \propto p(Y|\gamma, \sigma)p(\gamma, \sigma). \tag{A.6}$$

Then, if the prior is quite flat, the values $\hat{\gamma}$ and $\hat{\sigma}$ are obtained by maximizing the marginal likelihood function $p(Y|\gamma, \sigma)$. This function directly assesses the relative goodness of a model with respect to another one, expressing the preference shown by the data for the different models. Two models may differ, as an example, for a different value of their hyperparameters.

The marginal likelihood can be effectively used as a model selection tool, because it automatically incorporates a trade-off between model fit to data and model complexity. By maximizing this function, we are able to find the hyperparameters' values that best balance these two conflicting aspects. The marginal likelihood can be expressed as:

$$p(Y|\gamma,\sigma) = \int p(Y|\theta,\sigma)p(\theta|\gamma)\, d\theta, \tag{A.7}$$

and can be computed analytically in the linear Gaussian model case (2.19), by referring to the relations (2.37) - (2.44) - (2.46). The minimization is then performed by iterative schemes [30]. In equation (A.7), the term $p(Y|\gamma,\sigma)$ is nothing more than the marginal distribution of the data Y computed in the bottom part of (2.37), where the dependence on the hyperparameters γ and σ have been made explicit. The factor $p(Y|\theta,\sigma)$ is the conditional distribution of the data (2.46) that depends on σ. The term $p(\theta|\gamma)$ is the prior distribution on the parameters (2.44), which depends on its standard deviation γ. Therefore, we are given $p(\theta)$ and $p(Y|\theta)$, and we want to find $p(Y)$. Using (2.37), the marginal likelihood (as a function of the hyperparameters' vector $\eta = [\gamma,\sigma]^T$) is found to be:

$$p(Y|\gamma,\sigma) = \mathcal{N}\big(0,\Sigma_y(\eta)\big) \tag{A.8}$$

$$= \frac{1}{(2\pi)^{N/2}} \cdot \frac{1}{\big(\det\Sigma_y(\eta)\big)^{1/2}} \cdot e^{-\frac{1}{2}\left(Y^T\Sigma_y^{-1}(\eta)Y\right)}, \tag{A.9}$$

where $\Sigma_y(\eta) = \sigma^2 I_N + \Phi\gamma^2 I_m \Phi^T$. The equivalent solution which maximises the evidence (A.8) can be found by minimizing the negative log-likelihood:

$$-\ln\big[p(Y|\gamma,\sigma)\big] = -\left[-\frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln\big[\det\Sigma_y(\eta)\big] - \frac{1}{2}\big(Y^T\Sigma_y(\eta)^{-1}Y\big)\right], \tag{A.10}$$

and the parameters' estimate is obtained as in [21, 20]:

$$\hat{\eta} = \arg\min_{\eta}\, Y^T\Sigma_y(\eta)^{-1}Y + \ln\big[\det\Sigma_y(\eta)\big] \tag{A.11}$$

When it is not possible to compute analytically the marginal likelihood (A.7), the Expectation Maximization (EM) method can be employed [30].

APPENDIX B

Functional analysis fundamentals

B.1 Vector spaces and linear operators

Definition B.1.1 *A vector space over the real field* $\mathbb{R}$ *is a set* V *endowed with two operations:*

1. *sum:* $\underset{(u,v)\ \mapsto\ u+v}{V \times V \to V}$
2. *inner product:* $\underset{(\lambda,v)\ \mapsto\ \lambda v}{\mathbb{R} \times V \to V}$

and that satisfies the following axioms:

- $u + (v + w) = (u + v) + w, \ \ \forall u, v, w \in V$
- $u + v = v + u$
- $\exists\, 0 \in V \ : \ v + 0 = v, \ \ \forall v \in V$
- $\exists\, (-v) \in V \ : \ v + (-v) = 0, \ \ \forall v \in V$
- $\lambda_1 (\lambda_2 v) = (\lambda_1 \lambda_2) v, \ \ \forall \lambda_1, \lambda_2 \in \mathbb{R}, v \in V$
- $\exists\, 1 \in V \ : \ 1 \cdot v = v, \ \ \forall v \in V$
- $\lambda (u + v) = \lambda u + \lambda v, \ \ \forall \lambda \in \mathbb{R}, v, u \in V$
- $(\lambda_1 + \lambda_2) v = \lambda_1 u + \lambda_2 v, \ \ \forall \lambda_1 \lambda_2 \in \mathbb{R}, v, u \in V$

Suppose now that V and W are two vector spaces. Then, the following definitions hold:

Definition B.1.2 *$F : V \to W$ is a linear operator if, $\forall \lambda_1, \lambda_2 \in \mathbb{R}$ one has that $F(\lambda_1 u_1 + \lambda_2 u_2) = \lambda_1 F(u_1) + \lambda_2 F(u_2), \ \ \forall u_1, u_2 \in V$*

Definition B.1.3 *A linear operator $F : V \to \mathbb{R}$ is called linear form or linear functional*

Definition B.1.4 *A bilinear form is a function $a : V \times V \to \mathbb{R}$ that is linear in both the arguments, that is:*

- *$a(u, \cdot) : \underset{v \mapsto a(u,v)}{V \to \mathbb{R}}$ is a linear functional, $\forall u \in V$ (u is fixed)*
- *$a(\cdot, v) : \underset{u \mapsto a(u,v)}{V \to \mathbb{R}}$ is a linear functional, $\forall v \in V$ (v is fixed)*

Definition B.1.5 *A bilinear form a is said to be symmetric if $a(u, v) = a(v, u)$ $\forall v, u \in V$*

Example: 2.1 (i) *Linear operators*

1. Be $V = \mathbb{R}^n$, $W = \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, then
   $f : \underset{v \mapsto Av}{\mathbb{R}^n \to \mathbb{R}^m}$ is a linear operator. If $m = 1$, it is a linear form.
2. Be $V = \mathbb{R}^n$, then
   $a : \underset{(u,v) \mapsto u^T A v}{\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}}$ is a bilinear form; a is a symmetric form $\iff$ A is symmetric.

B.1.1 Banach spaces

Definition B.1.6 *Let V be a vector space. A norm over V is a function $\| \cdot \| : V \to \mathbb{R}$ s.t. $\forall u, v \in V$ and $\forall \lambda \in \mathbb{R}$ the following properties hold:*

a) $\|u\| \geq 0; \ \ \|u\| = 0 \iff u = 0$

b) $\|\lambda u\| = |\lambda| \cdot \|u\|$

c) $\|u + v\| \leq \|u\| + \|v\|$

In this case we say that $(V, \| \cdot \|)$ is a normed space.

Example: 2.1 (ii) *Normed spaces*

1. $V = \mathbb{R}^n$, with the Euclidean norm: $\| (u_1, \ldots, u_n) \|_2 = \sqrt{\sum_{i=1}^n u_i^2}$
2. $V = C([a,b])$ with the max norm: $\|u\|_\infty = \underset{x\in[a,b]}{\text{Max}} |u(x)|$
3. $V = C([a,b])$ with the L^p norm: $\|u\|_p = \left(\int_a^b |u(x)|^p \, dx\right)^{\frac{1}{p}}$

Definition B.1.7 *Let $(V, \|\cdot\|)$ a normed space. A sequence $\{u_n\}$ is said to be:*

- *convergent, if $\exists u \in V$ s.t. $u_n \to u$, that is, $\lim_{n\to\infty} \|u_n - u\| = 0$*
- *the sequence is a Cauchy sequence if $\|u_n - u_m\|$ when $n, m \to \infty$*

Proposition B.1.1 *In a normed space, every convergent sequence is a Cauchy sequence.*

Proposition B.1.2 *In $\mathbb{R}^n$ (with the euclidean norm), every Cauchy sequence is convergent. There exist however normed spaces in which not all Cauchy sequences are convergent.*

Definition B.1.8 *A normed space is said to be complete if every Cauchy sequence converges (in the same space). A Banach space is a complete normed space.*

Example: 2.1 (iii) *Complete spaces*

1. $V = \mathbb{R}^n$, with the Euclidean norm: $\| (u_1, \ldots, u_n) \|_2 = \sqrt{\sum_{i=1}^n u_i^2}$
2. $V = C([a,b])$ with the max norm: $\|u\|_\infty = \underset{x\in[a,b]}{\text{Max}} |u(x)|$
3. $V = C([a,b])$ with the L^p norm: $\|u\|_p = \left(\int_a^b |u(x)|^p \, dx\right)^{\frac{1}{p}}$
4. $V = L^p$ with the L^p norm

Examples 1., 2. and 4. are complete (and so they are Banach spaces), while 3. is not complete. Notice that 2. and 3. are the same space but with a different norm, while 3. and 4. are different spaces with the same norm.

B.1.2 Hilbert spaces

Definition B.1.9 *A pre-Hilbert space is a vector space V endowed with a inner product, that is, of a bilinear symmetric form $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$, $(u,v) \mapsto \langle u,v \rangle$ s.t. $\langle u, u \rangle > 0 \ \ \forall u \neq 0$.*

The induced norm by the inner product is:

$$\|u\| = \sqrt{\langle u, u \rangle}. \tag{B.1}$$

Proposition B.1.3 *Let V be a pre-Hilbert space. Then, the following propositions hold:*

- *Schwarz inequality: $|\langle u, v \rangle| \leq \|u\| \cdot \|v\| \ \ \forall u, v \in V$.*
- *The induced norm is effectively a norm.*

Definition B.1.10 *A Hilbert space is a complete pre-Hilbert space (with respect to the induced norm).*

Example: 2.1 (iv) *Hilbert spaces*

1. Let $V = \mathbb{R}^n$, $\langle u, v \rangle = \sum_{i=1}^{n} u_i \cdot v_i$. The induced norm is the euclidean one. The space is complete, therefore V is a Hilbert space.
2. $V = C([a,b])$, $\langle u, v \rangle = \int_a^b u(x) \cdot v(x)\, dx$. The induced norm is the $\| \cdot \|_2$ norm. The space is not complete (see previous example). Then, V is a pre-Hilbert space but not a Hilbert one.

Definition B.1.11 *Let V be a Hilbert space and $F : V \to \mathbb{R}$ a linear functional. We say that F is continous in a point $u \in V$, if, for every sequence $\{u_n\} \subset V$ that converges to u, we have that $F(u_n)$ converges to $F(u)$. In formulas:*

$$\lim_{n \to \infty} \|u_n - u\| = 0 \Longrightarrow \lim_{n \to \infty} |F(u_n) - F(u)| = 0. \tag{B.2}$$

It can be shown that F is continous in every $u \in V$ if and only if $\exists M \in \mathbb{R}$ s.t. $|F(u)| \leq M \cdot \|u\|$.

Let now be V' the set of the continous and linear functional from V to $\mathbb{R}$.

Example: 2.1 (v) *Continous linear functional*
Let $V = \mathbb{R}^2$ with inner product $\langle u, v\rangle = \langle (u_1, v_1), (u_2, v_2)\rangle = u_1v_1 + u_2v_2$. Then, $F : \underset{(u_1,u_2) \mapsto 2u_1 - u_2}{\mathbb{R}^2 \to \mathbb{R}}$ is a linear form, continous over $\mathbb{R}^2$.

Definition B.1.12 *V' is called the dual space of V. If $F \in V'$ we can define its dual norm:*

$$\|F\|_{V'} = \sup_{\|u\|=1} |F(u)| \tag{B.3}$$

Let now V be a Hilbert space, with fixed $v \in V$. Let $F_v(u) = \langle u, v\rangle$, that is $F_v = \langle \cdot, v\rangle$. Then, $F_v : \underset{u \mapsto \langle u,v\rangle}{V \to \mathbb{R}}$ is linear (because $\langle \cdot, \cdot\rangle$ is bilinear) and continous. Infact $|\langle u, v\rangle| \leq \|u\| \cdot \|v\|$ (for the Schwarz inequality), and so $|F_v(u)| \leq M \cdot \|u\|$ (where $M = \|v\|$). Therefore, we have that $\forall v \in V$, $F_v \in V'$, i.e. there is a function $\Phi : \underset{v \mapsto F_v}{V \to V'}$ that maps each element of the space V to a linear functional in V'.

Example: 2.1 (vi) *Mapping from V to V'*
Let $v_1, v_2 \in V$. Then, the following relations hold:

$$\Phi(v_1) = F_{v_1}(u) = \langle u, v_1\rangle$$
$$\Phi(v_2) = F_{v_2}(u) = \langle u, v_2\rangle$$

It can be shown that Φ is a linear injective operator. A fundamental result is the following:

Theorem B.1.1 ***Riesz's representation theorem***
The fuction Φ is surjective. This means that $\forall F \in V'$ there exists $v \in V$ s.t. $F = F_v$.

The function Φ is both injective and surjective, then it is a bijection. The results of the Riesz's representation theorem make possible to express every $F \in V'$ in the form F_v, that is, every $F(u)$ returns the same result as $F_v(u) = \langle u, v\rangle$.

Bibliography

[1] L. Ljung and T. Glad, *Modeling of dynamic systems*. PTR Prentice Hall Englewood Cliffs, 1994.

[2] L. Ljung, *System Identification: Theory for the User*. Pearson Education, 1998.

[3] T. Söderström and P. Stoica, *System identification*. Prentice-Hall, Inc., 1988.

[4] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[5] H. Akaike, "A new look at the statistical model identification," *IEEE transactions on automatic control*, vol. 19, no. 6, pp. 716–723, 1974.

[6] T. Chen, H. Ohlsson, and L. Ljung, "On the estimation of transfer functions, regularizations and gaussian processes - revisited," *Automatica*, vol. 48, no. 8, pp. 1525–1535, 2012.

[7] G. Pillonetto, A. Chiuso, and G. De Nicolao, "Prediction error identification of linear systems: a nonparametric gaussian regression approach," *Automatica*, vol. 47, no. 2, pp. 291–305, 2011.

[8] G. Pillonetto and G. De Nicolao, "A new kernel-based approach for linear system identification," *Automatica*, vol. 46, no. 1, pp. 81–93, 2010.

[9] H. Leeb and B. M. Pötscher, "Model selection and inference: Facts and fiction," *Econometric Theory*, vol. 21, no. 1, pp. 21–59, 2005.

[10] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H. T. Lin, *Learning from data*. AMLBook New York, NY, USA, 2012.

[11] D. L. Phillips, "A technique for the numerical solution of certain integral equations of the first kind," *Journal of the ACM (JACM)*, vol. 9, no. 1, pp. 84–97, 1962.

[12] A. N. Tikhonov and V. Y. Arsenin, *Solutions of ill-posed problems*, vol. 14. Winston, 1977.

[13] P. C. Hansen, "The truncated svd as a method for regularization," *BIT Numerical Mathematics*, vol. 27, no. 4, pp. 534–553, 1987.

[14] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[15] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[16] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[17] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*, vol. 1. MIT press Cambridge, 2006.

[18] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[19] B. Efron and T. Hastie, *Computer Age Statistical Inference*, vol. 5. Cambridge University Press, 2016.

[20] G. Prando, D. Romeres, G. Pillonetto, and A. Chiuso, "Classical vs. bayesian methods for linear system identification: Point estimators and confidence sets," in *Control Conference (ECC), 2016 European*, pp. 1365–1370, IEEE, 2016.

[21] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, "Kernel methods in system identification, machine learning and function estimation: A survey," *Automatica*, vol. 50, no. 3, pp. 657–682, 2014.

[22] G. Pillonetto, M. H. Quang, and A. Chiuso, "A new kernel-based approach for nonlinear system identification," *IEEE Transactions on Automatic Control*, vol. 56, no. 12, pp. 2825–2840, 2011.

[23] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory-Implementation-Applications*. Springer Science & Business Media, 2012.

[24] M. Verhaegen and V. Verdult, *Filtering and system identification: a least squares approach*. Cambridge university press, 2007.

[25] R. Pintelon and J. Schoukens, *System identification: a frequency domain approach*. John Wiley & Sons, 2012.

[26] M. A. H. Darwish, J. Lataire, and R. Toth, "Bayesian frequency domain identification of LTI systems with OBFs kernels," in *20th World Congress*, IFAC, 2017.

[27] N. Wiener, *Extrapolation, interpolation, and smoothing of stationary time series*, vol. 7. MIT press Cambridge, MA, 1949.

[28] L. Ljung, "Prediction error estimation methods," *Circuits, Systems and Signal Processing*, vol. 21, no. 1, pp. 11–21, 2002.

[29] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[30] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[31] F. P. Carli, "On the maximum entropy property of the first-order stable spline kernel and its implications," in *Control Applications (CCA), 2014 IEEE Conference on*, pp. 409–414, IEEE, 2014.

[32] G. Pillonetto and G. De Nicolao, "Kernel selection in linear system identification part i: A gaussian process perspective," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pp. 4318–4325, IEEE, 2011.

[33] F. P. Carli, A. Chiuso, and G. Pillonetto, "Efficient algorithms for large scale linear system identification using stable spline estimators," *Proceedings of IFAC SYSID symposium*, vol. 45, no. 16, pp. 119–124, 2012.

[34] T. Chen and L. Ljung, "Implementation of algorithms for tuning parameters in regularized least squares problems in system identification," *Automatica*, vol. 49, no. 7, pp. 2213–2220, 2013.

[35] W. Rudin, *Real and complex analysis*. Tata McGraw-Hill Education, 1987.

[36] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American mathematical society*, vol. 68, no. 3, pp. 337–404, 1950.

[37] F. Cucker and S. Smale, "On the mathematical foundations of learning," *Bulletin of the American mathematical society*, vol. 39, no. 1, pp. 1–49, 2002.

[38] G. Wahba, *Spline models for observational data*. SIAM, 1990.

[39] R. Vert and J.-P. Vert, "Consistency and convergence rates of one-class svms and related algorithms," *Journal of Machine Learning Research*, vol. 7, no. May, pp. 817–854, 2006.

[40] G. Kimeldorf and G. Wahba, "Some results on tchebycheffian spline functions," *Journal of mathematical analysis and applications*, vol. 33, no. 1, pp. 82–95, 1971.

[41] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, 1990.

[42] J. A. Suykens, T. Van Gestel, and J. De Brabanter, *Least squares support vector machines*. World Scientific, 2002.

[43] C. Runge, "Über empirische funktionen und die interpolation zwischen äquidistanten ordinaten," *Zeitschrift für Mathematik und Physik*, vol. 46, no. 224-243, p. 20, 1901.

[44] G. S. Kimeldorf and G. Wahba, "A correspondence between bayesian estimation on stochastic processes and smoothing by splines," *The Annals of Mathematical Statistics*, vol. 41, no. 2, pp. 495–502, 1970.

[45] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural computation*, vol. 7, no. 2, pp. 219–269, 1995.

[46] C. R. Rojas and H. Hjalmarsson, "Sparse estimation based on a validation criterion," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pp. 2825–2830, IEEE, 2011.

[47] C. R. Rojas, B. Wahlberg, and H. Hjalmarsson, "A sparse estimation technique for general model structures," in *Control Conference (ECC), 2013 European*, pp. 2410–2414, IEEE, 2013.

[48] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen, "Bayesian inference and learning in gaussian process state-space models with particle mcmc," in *Advances in Neural Information Processing Systems*, pp. 3156–3164, 2013.

[49] R. Frigola and C. E. Rasmussen, "Integrated pre-processing for bayesian nonlinear system identification with gaussian processes," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 5371–5376, IEEE, 2013.

[50] J. Hall, C. Rasmussen, and J. Maciejowski, "Modelling and control of nonlinear systems using gaussian processes with partial model information," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pp. 5266–5271, IEEE, 2012.

[51] X. Zhu, "Semi-supervised learning," in *Encyclopedia of machine learning*, pp. 892–897, Springer, 2011.

[52] O. Chapelle, B. Schlkopf, and A. Zien, *Semi-Supervised Learning*. The MIT Press, 1st ed., 2010.

[53] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[54] D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1. Athena scientific Belmont, MA, 1995.

[55] H. Ohlsson and L. Ljung, "Semi-supervised regression and system identification," in *Three Decades of Progress in Control Sciences*, pp. 343–360, Springer, 2010.

[56] X. Yang, H. Fu, H. Zha, and J. Barlow, "Semi-supervised nonlinear dimensionality reduction," in *Proceedings of the 23rd international conference on Machine learning*, pp. 1065–1072, ACM, 2006.

[57] M. Belkin, P. Niyogi, and V. Sindhwani, "On manifold regularization.," in *AISTATS*, p. 1, 2005.

[58] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of machine learning research*, vol. 7, no. Nov, pp. 2399–2434, 2006.

[59] X. Zhu and A. B. Goldberg, "Semi-supervised regression with order preferences," *Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep*, vol. 1578, p. 10, 2006.

[60] V. Castelli and T. M. Cover, "The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter," *IEEE Transactions on information theory*, vol. 42, no. 6, pp. 2102–2117, 1996.

[61] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[62] L. Cayton, "Algorithms for manifold learning," *Univ. of California at San Diego Tech. Rep*, vol. 12, pp. 1–17, 2005.

[63] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik, "Vicinal risk minimization," in *Advances in neural information processing systems*, pp. 416–422, 2001.

[64] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[65] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[66] Y. S. Abu-Mostafa, "Learning from hints," *Journal of Complexity*, vol. 10, no. 1, pp. 165–178, 1994.

[67] A. Isturiz, J. Vinals, S. Fernandez, R. Basagoiti, E. d. l. Torre Arnanz, and J. Novo, "Development of an aeronautical electromechanical actuator with real time health monitoring capability," 2010.

[68] G.-A. Capolino, J. A. Antonino-Daviu, and M. Riera-Guasp, "Modern diagnostics techniques for electrical machines, power electronics, and drives," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1738–1745, 2015.

[69] D. van Schrick, "Remarks on terminology in the field of supervision, fault detection and diagnosis," in *Preprints of the 3rd IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS*, pp. 959–964, 1997.

[70] J. Gertler, *Fault Detection and Diagnosis in Engineering Systems*. Electrical Engineering and Electronics, Taylor & Francis, 1998.

[71] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part i: Quantitative model-based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 293–311, 2003.

[72] A. S. Willsky, "A survey of design methods for failure detection in dynamic systems," *Automatica*, vol. 12, no. 6, pp. 601–611, 1976.

[73] R. Isermann, "Model-based fault-detection and diagnosis–status and applications," *Annual Reviews in control*, vol. 29, no. 1, pp. 71–85, 2005.

[74] R. Isermann, "Process fault detection based on modeling and estimation methods - a survey," *Automatica*, vol. 20, no. 4, pp. 387–404, 1984.

[75] P. M. Frank and X. Ding, "Survey of robust residual generation and evaluation methods in observer-based fault detection systems," *Journal of process control*, vol. 7, no. 6, pp. 403–424, 1997.

[76] J. Gertler, "Fault detection and isolation using parity relations," *Control engineering practice*, vol. 5, no. 5, pp. 653–661, 1997.

[77] J. J. Gertler, "Survey of model-based failure detection and isolation in complex plants," *IEEE Control systems magazine*, vol. 8, no. 6, pp. 3–11, 1988.

[78] P. M. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy: A survey and some new results," *automatica*, vol. 26, no. 3, pp. 459–474, 1990.

[79] R. Patton, J. Chen, and S. Nielsen, "Model-based methods for fault diagnosis: some guide-lines," *Transactions of the Institute of Measurement and Control*, vol. 17, no. 2, pp. 73–83, 1995.

[80] E. Balaban, A. Saxena, S. Narasimhan, I. Roychoudhury, and K. Goebel, "Experimental validation of a prognostic health management system for electro-mechanical actuators," in *Infotech@ Aerospace 2011*, p. 1518, 2011.

[81] M. A. Ismail, E. Balaban, and H. Spangenberg, "Fault detection and classification for flight control electromechanical actuators," in *Aerospace Conference, 2016 IEEE*, pp. 1–10, IEEE, 2016.

[82] S. Choi, E. Pazouki, J. Baek, and H. R. Bahrami, "Iterative condition monitoring and fault diagnosis scheme of electric motor for harsh industrial application," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1760–1769, 2015.

[83] A. Giantomassi, F. Ferracuti, S. Iarlori, G. Ippoliti, and S. Longhi, "Electric motor fault detection and diagnosis by kernel density estimation and kullback–leibler divergence based on stator current measurements," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1770–1780, 2015.

[84] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis: Part iii: Process history based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 327–346, 2003.

[85] Y. Lei, M. J. Zuo, Z. He, and Y. Zi, "A multidimensional hybrid intelligent method for gear fault diagnosis," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1419–1430, 2010.

[86] S. Narasimhan, I. Roychoudhury, E. Balaban, and A. Saxena, "Combining model-based and feature-driven diagnosis approaches-a case study on electromechanical actuators," 2010.

[87] A. Cologni, M. Mazzoleni, and F. Previdi, "Modeling and identification of an electro-hydraulic actuator," in *Control and Automation (ICCA), 2016 12th IEEE International Conference on*, pp. 335–340, IEEE, 2016.

[88] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[89] J. Blesa Izquierdo, F. Le Gall, C. Jauberthie, and L. Travé-Massuyès, "State estimation and fault detection using box particle filtering with stochastic measurements," in *DX 2015-26th International Workshop on Principles of Diagnosis, 31 August-1 Septembrer, Paris (France)*, pp. 67–73, 2015.

[90] X. Koutsoukos, J. Kurien, and F. Zhao, "Monitoring and diagnosis of hybrid systems using particle filtering methods," in *International Symposium on Mathematical Theory of Networks and Systems*, 2002.

[91] S. Tafazoli and X. Sun, "Hybrid system state tracking and fault detection using particle filters," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 6, pp. 1078–1087, 2006.

[92] P. M. Reeves, G. Campbell, V. Ganzer, and R. Joppa, "Development and application of a non-gaussian atmospheric turbulence model for use in flight simulators," 1974.

[93] Z. Korona and M. M. Kokar, "A fusion and learning algorithm for landing aircraft tracking: Compensating for exhaust plume disturbance," *IEEE transactions on aerospace and electronic systems*, vol. 31, no. 3, pp. 1210–1215, 1995.

[94] B. Armstrong-Hélouvry, P. Dupont, and C. C. De Wit, "A survey of models, analysis tools and compensation methods for the control of machines with friction," *Automatica*, vol. 30, no. 7, pp. 1083–1138, 1994.

[95] A. Niglis and P. Öberg, "Modelling high-fidelity robot dynamics," 2015.

[96] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 174–188, 2002.

[97] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *IEE Proceedings F-Radar and Signal Processing*, vol. 140, pp. 107–113, IET, 1993.

[98] Y. Ho and R. Lee, "A bayesian approach to problems in stochastic estimation and control," *IEEE Transactions on Automatic Control*, vol. 9, no. 4, pp. 333–339, 1964.

[99] O. Cappé, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential monte carlo," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 899–924, 2007.

[100] J. Lygeros, C. Tomlin, and S. Sastry, "Hybrid systems: modeling, analysis and control," *preprint*, 1999.

[101] Q. Zhang, "Optimal filtering of discrete-time hybrid systems," *Journal of Optimization Theory and Applications*, vol. 100, no. 1, pp. 123–144, 1999.

[102] N. De Freitas, R. Dearden, F. Hutter, R. Morales-Menendez, J. Mutch, and D. Poole, "Diagnosis by a waiter and a mars explorer," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 455–468, 2004.

[103] D. Zhou, Y. Xi, and Z. Zhang, "Non-linear adaptive fault detection filter," *International journal of systems science*, vol. 22, no. 12, pp. 2563–2571, 1991.

[104] H. Henao, G.-A. Capolino, M. Fernandez-Cabanas, F. Filippetti, C. Bruzzese, E. Strangas, R. Pusca, J. Estima, M. Riera-Guasp, and S. Hedayati-Kia, "Trends in fault diagnosis for electrical machines: A review of diagnostic techniques," *IEEE industrial electronics magazine*, vol. 8, no. 2, pp. 31–42, 2014.

[105] M. E. H. Benbouzid, "A review of induction motors signature analysis as a medium for faults detection," *IEEE transactions on industrial electronics*, vol. 47, no. 5, pp. 984–993, 2000.

[106] C. Combastel, S. Lesecq, S. Petropol, and S. Gentil, "Model-based and wavelet approaches to induction motor on-line fault detection," *Control Engineering Practice*, vol. 10, no. 5, pp. 493–509, 2002.

[107] T. W. Rauber, E. M. do Nascimento, E. D. Wandekokem, and F. M. Varejão, *Pattern recognition based fault diagnosis in industrial processes: review and application.* INTECH Open Access Publisher, 2010.

[108] J. Zarei, "Induction motors bearing fault detection using pattern recognition techniques," *Expert systems with Applications*, vol. 39, no. 1, pp. 68–73, 2012.

[109] P. J. Rousseeuw and C. Croux, "Explicit scale estimators with high breakdown point," *L1-Statistical analysis and related methods*, vol. 1, pp. 77–92, 1992.

[110] A. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, vol. 14, p. 841, 2002.

[111] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[112] G. C. Cawley and N. L. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *Journal of Machine Learning Research*, vol. 11, no. Jul, pp. 2079–2107, 2010.

[113] D. Boley, "Principal direction divisive partitioning," *Data mining and knowledge discovery*, vol. 2, no. 4, pp. 325–344, 1998.

[114] D. Boley, M. Gini, R. Gross, E.-H. S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore, "Partitioning-based clustering for web document categorization," *Decision Support Systems*, vol. 27, no. 3, pp. 329–341, 1999.

[115] D. Boley, "Hierarchical taxonomies using divisive partitioning," tech. rep., Technical Report TR-98-012, Department of Computer Science, University of Minnesota, Minneapolis, 1998.

[116] E.-H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore, "Webace: a web agent for document categorization and exploration," in *Proceedings of the second international conference on Autonomous agents*, pp. 408–415, ACM, 1998.

[117] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *JASIS*, vol. 41, no. 6, pp. 391–407, 1990.

[118] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office, 1950.

[119] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3. JHU Press, 2012.

[120] M. Nilsson, "Hierarchical clustering using non-greedy principal direction divisive partitioning," *Information Retrieval*, vol. 5, no. 4, pp. 311–321, 2002.

[121] C. Kruengkrai, V. Sornlertlamvanich, and H. Isahara, "Refining a divisive partitioning algorithm for unsupervised clustering.," in *HIS*, pp. 535–542, 2003.

[122] S. M. Savaresi, D. L. Boley, S. Bittanti, and G. Gazzaniga, "Choosing the cluster to split in bisecting divisive clustering algorithms," tech. rep., Technical Report TR-00-53, 2002.

[123] S. Tasoulis and D. Tasoulis, "Improving principal direction divisive clustering," in *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008), Workshop on Data Mining using Matrices and Tensors, Las Vegas, USA*, 2008.

[124] J. A. Hartigan and J. Hartigan, *Clustering algorithms*, vol. 209. Wiley New York, 1975.

[125] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, p. 14, California, USA, 1967.

[126] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *KDD*, vol. 96, pp. 226–231, 1996.

[127] C. G. Chute and Y. Yang, "An overview of statistical methods for the classification and retrieval of patient events.," *Methods of information in medicine*, vol. 34, no. 1-2, pp. 104–110, 1995.

[128] S. M. Savaresi and D. L. Boley, "A comparative analysis on the bisecting k-means and the pddp clustering algorithms," *Intelligent Data Analysis*, vol. 8, no. 4, pp. 345–362, 2004.

[129] A. Isturiz, J. Vinals, J. M. Abete, and A. Iturrospe, "Health monitoring strategy for electromechanical actuator systems and components. screw backlash and fatigue estimation," in *Recent Advances in Aerospace Actuation Systems and Components*, vol. 5, 2012.

[130] A. Picot, Z. Obeid, J. Régnier, S. Poignant, O. Darnis, and P. Maussion, "Statistic-based spectral indicator for bearing fault detection in permanent-magnet synchronous machines using the stator current," *Mechanical Systems and Signal Processing*, 2014.

[131] T. W. Rauber, E. M. do Nascimento, E. D. Wandekokem, and F. M. Varejão, "Pattern recognition based fault diagnosis in industrial processes: Review and application," in *Pattern Recognition Recent Advances* (A. Herout, ed.), ch. 25, Springer, 2010.

[132] E. Balaban, P. Bansal, P. Stoelting, A. Saxena, K. F. Goebel, and S. Curran, "A diagnostic approach for electro-mechanical actuators in aerospace systems," in *Aerospace conference, 2009 IEEE*, pp. 1–13, IEEE, 2009.

[133] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[134] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 2009.

[135] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[136] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE Transactions on Systems, Man and Cybernetics*, no. 4, pp. 580–585, 1985.

[137] D. Samson and P. J. Singh, *Operations management: An integrated approach*. Cambridge University Press, 2008.

[138] L. G. Cooper, P. Baron, W. Levy, M. Swisher, and P. Gogos, "Promocast: A new forecasting method for promotion planning," *Marketing Science*, vol. 18, no. 3, pp. 301–316, 1999.

[139] R. K. Ireland and C. Crum, *Supply chain collaboration: How to implement CPFR and other best collaborative practices*. J. Ross Publishing, 2005.

[140] K. L. Ailawadi, J. P. Beauchamp, N. Donthu, D. K. Gauri, and V. Shankar, "Communication and promotion decisions in retailing: a review and directions for future research," *Journal of retailing*, vol. 85, no. 1, pp. 42–55, 2009.

[141] M. Holweg, S. Disney, J. Holmström, and J. Småros, "Supply chain collaboration:: Making sense of the strategy continuum," *European management journal*, vol. 23, no. 2, pp. 170–181, 2005.

[142] E. Brynjolfsson, L. M. Hitt, and H. H. Kim, "Strength in numbers: How does data-driven decisionmaking affect firm performance?," *Available at SSRN 1819486*, 2011.

[143] J. S. Armstrong, *Principles of forecasting: a handbook for researchers and practitioners*, vol. 30. Springer Science & Business Media, 2001.

[144] W. Y. Lee, P. Goodwin, R. Fildes, K. Nikolopoulos, and M. Lawrence, "Providing support for the use of analogies in demand forecasting tasks," *International Journal of Forecasting*, vol. 23, no. 3, pp. 377–390, 2007.

[145] T. Huang, R. Fildes, and D. Soopramanien, "The value of competitive information in forecasting fmcg retail product sales and the variable selection problem," *European Journal of Operational Research*, vol. 237, no. 2, pp. 738–748, 2014.

[146] Ö. G. Ali, S. Sayın, T. Van Woensel, and J. Fransoo, "Sku demand forecasting in the presence of promotions," *Expert Systems with Applications*, vol. 36, no. 10, pp. 12340–12348, 2009.

[147] R. C. Blattberg and A. Levin, "Modelling the effectiveness and profitability of trade promotions," *Marketing Science*, vol. 6, no. 2, pp. 124–146, 1987.

[148] M. M. Abraham and L. M. Lodish, "Promoter: An automated promotion evaluation system," *Marketing Science*, vol. 6, no. 2, pp. 101–123, 1987.

[149] D. Yang, G. S. Goh, S. Jiang, A. N. Zhang, and O. Akcan, "Forecast upc-level fmcg demand, part ii: Hierarchical reconciliation," in *Big Data (Big Data), 2015 IEEE International Conference on*, pp. 2113–2121, IEEE, 2015.

[150] M. Banek, D. Osrecki, M. Vranic, and D. Pintar, "Outlier detection as the primary step for promotion planning in retail," in *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on*, pp. 1405–1410, IEEE, 2015.

[151] S. Meeran, K. Dyussekeneva, and P. Goodwin, "Sales forecasting using combination of diffusion model and forecast market–an adaption of prediction/preference markets," *7th IFAC Conference on Manufacturing Modelling, Management, and Control (MIM)*, vol. 46, no. 9, pp. 87–92, 2013.

[152] C. Schwenke, J. Ziegenbalg, K. Kabitzsch, and V. Vasyutynskyy, "Simulation based forecast of supermarket sales," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, pp. 1–8, IEEE, 2012.

[153] P. W. Murray, B. Agard, and M. A. Barajas, "Forecasting supply chain demand by clustering customers," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 1834–1839, 2015.

[154] T. Raeder, O. Stitelman, B. Dalessandro, C. Perlich, and F. Provost, "Design principles of massive, robust prediction systems," in *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1357–1365, ACM, 2012.

[155] V. N. Azarskov, V. I. Skurikhin, L. S. Zhiteckii, and R. O. Lypoi, "Modern control theory applied to inventory control for a manufacturing system," *7th IFAC Conference on Manufacturing Modelling, Management, and Control (MIM)*, vol. 46, no. 9, pp. 1200–1205, 2013.

[156] R. Abbou, C. Moussaoui, and J. J. Loiseau, "Effects of inventory control on bullwhip in logistic systems under demand and lead time uncertainties," *15th IFAC Symposium on Information Control Problems in Manufacturing (INCOM)*, vol. 48, no. 3, pp. 266–271, 2015.

[157] J. R. Trapero, N. Kourentzes, and R. Fildes, "On the identification of sales forecasting models in the presence of promotions," *Journal of the Operational Research Society*, vol. 66, no. 2, pp. 299–307, 2014.

[158] C. Tofallis, "A better measure of relative prediction accuracy for model selection and model estimation," *Journal of the Operational Research Society*, vol. 66, no. 8, pp. 1352–1362, 2015.

[159] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[160] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[161] B. Efron and R. Tibshirani, "Improvements on cross-validation: the .632+ bootstrap method," *Journal of the American Statistical Association*, vol. 92, no. 438, pp. 548–560, 1997.

[162] J. M. Bernardo and A. Smith, "Bayesian theory, vol. 405," *JohnWiley & Sons*, 2009.

[163] J. Kruschke, *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press, 2014.

[164] M. Blangiardo and M. Cameletti, *Spatial and spatio-temporal Bayesian models with R-INLA*. John Wiley & Sons, 2015.

[165] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*, vol. 2. CRC press Boca Raton, FL, 2014.

[166] J. O. Berger, *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

[167] G. Wahba, "A comparison of gcv and gml for choosing the smoothing parameter in the generalized spline smoothing problem," *The Annals of Statistics*, pp. 1378–1402, 1985.

[168] S. F. Gull, "Developments in maximum entropy data analysis," in *Maximum entropy and Bayesian methods*, pp. 53–71, Springer, 1989.

[169] D. J. MacKay, "Bayesian interpolation," *Neural computation*, vol. 4, no. 3, pp. 415–447, 1992.