UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

UNCONSTRAINED LEARNING MACHINES

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

ROBIN CHARLES GILBERT
Norman, Oklahoma
2010

Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/319132

UNCONSTRAINED LEARNING MACHINES

A DISSERTATION APPROVED FOR THE
SCHOOL OF INDUSTRIAL ENGINEERING

BY

Dr. Theodore B. Trafalis, Chair

Dr. Shivakumar Raman

Dr. S. Lakshmivarahan

Dr. P. Simin Pulat

Dr. Hillel J. Kumin

© Copyright by ROBIN CHARLES GILBERT 2010
All Rights Reserved.

Acknowledgements

I would like to express my deepest gratitude to my adviser, Dr. Theodore Trafalis, for his guidance and providing me with the best atmosphere for doing research. I am heartily thankful to those who made this dissertation possible, especially to Dr. Shivakumar Raman, Dr. Michael Richman and Dr. Lance Leslie who all provided me support for my research.

Naturally, my profound respect also goes to my other committee members, Dr. S. Lakshmivarahan whose advice on Kalman filtering was precious for my research, Dr. Simin Pulat and Dr. Hillel Kumin. I would also like to thank Dr. Randa Shehab for her assistance as well as the staff of the School of Industrial Engineering, notably Amy Piper, Jean Shingledecker and Cheryl Carney.

Lastly, I offer my regards to all of those who helped me in any respect during the completion of this dissertation.

Contents

Acknowledgements iv

List of Tables viii

List of Figures ix

List of Algorithms x

Abstract xi

I Unconstrained Learning Machines for Supervised Learning 1

1 Introduction and Literature Review 2
1.1 Origins of Supervised Learning . . . 2
1.2 The First Generation of Supervised Learning Algorithms . . . 3
1.3 Second Generation (1970s-1980s) . . . 4
1.4 Third Generation (1990s-2000s) . . . 4
1.5 Current Status and ULMs . . . 5

2 Kernel Methods for Supervised Learning 7
2.1 Data . . . 7
2.1.1 Source and Nature of the Processed Data . . . 7
2.1.2 Pre-Processing of Numerical Data . . . 12
2.1.3 Data Thinning . . . 17
2.2 Kernels . . . 20
2.2.1 Definitions and Properties . . . 21
2.2.2 Kernels on Categorical Data . . . 24
2.2.3 Kernels on Numerical Data . . . 27
2.2.4 Kernels for Real-Valued Physical Data . . . 32
2.3 Estimating Functions with Kernels . . . 35
2.3.1 Type of Supervised Learning Problems . . . 36
2.3.2 Representation of Pattern Functions with Kernels . . . 37
2.3.3 Rademacher Complexity . . . 38
2.3.4 Risk Measures . . . 40
2.4 Specialized Models . . . 43
2.4.1 Multiple Output Regression Models . . . 43
2.4.2 Multi-Class Classification Models . . . 44

2.4.3 Model Validation 49

3 Unconstrained Learning Machines 54
3.1 Mathematical Programming Problem 54
3.1.1 Objective Function 54
3.1.2 Constraints 59
3.1.3 Unconstrained Quadratic Programming Problem 60
3.1.4 Optimal Solution 62
3.2 Implementation 63
3.2.1 Formulation of the Linear System 63
3.2.2 Solving the Linear System 64
3.3 Algorithms and Complexities 69
3.3.1 Function Estimation for the General Case 69
3.3.2 Function Estimation for the Symmetric Case 71
3.3.3 Function Estimation with Optimal Coefficients 73

II Applications of Unconstrained Learning Machines 79

4 Form Inspection in Manufacturing Engineering 80
4.1 Introduction, Context and Aims 80
4.2 Mesh Generation 81
4.2.1 Mesh Quality 82
4.2.2 Uniform Grids 85
4.2.3 Measurement Time against Quality of a Uniform Grid 88
4.3 Registration and Parameter Estimation 89
4.3.1 Input Data and Notations 89
4.3.2 Pre-Registration Data Treatment 90
4.3.3 Registration with Implicit Functions 91
4.3.4 Registration with Parametric Functions 93
4.3.5 Getting the Surface Parameters and Normal Deviations 94
4.3.6 Determination of the Features of Solids 97
4.4 Nonlinear Regression and Minimum Zone Estimation with ULMs 99
4.5 Applications 100
4.5.1 Half Cylinder Deformations 100
4.5.2 Half Sphere Deformations 102
4.5.3 Half Torus Deformations 103
4.5.4 Cone Deformations 104
4.5.5 Face-Milled Plates Deformations 105

5 State Forecasts of a Weather System 110
5.1 Introduction, Context and Aims 110
5.2 Assimilation and State Predictions 111

5.2.1 Assimilation using ULMs . . . 111
5.2.2 Predictive Analysis . . . 113
5.2.3 Summary . . . 114
5.3 Lorenz 96 Model . . . 115
5.4 Quasi-Geostrophic Model . . . 118

Conclusions and Recommendations **122**

Bibliography **125**

A Glossary **136**
A.1 Sets . . . 136
A.2 Scalars, Vectors and Matrices . . . 136
A.3 Dot Products, Norms and Other Operators . . . 137

B Additional Algorithms **138**
B.1 Computation of Solutions of Linear Systems . . . 138
B.1.1 Cholesky Decomposition . . . 138
B.1.2 Computation of the Solution of a Triangular System . . . 139
B.1.3 Conjugate Gradient Method . . . 139
B.2 Algorithms for Elementary Linear Algebra . . . 140
B.2.1 Symmetric Matrix Rank-2k Update . . . 140
B.2.2 General Matrix-Vector Multiplication . . . 141
B.2.3 Dot Product . . . 141
B.2.4 Matrix Row Summation . . . 142
B.2.5 Diagonal-Matrix Multiplication . . . 143
B.2.6 Matrix-Diagonal Multiplication . . . 143
B.2.7 Basic Tikhonov Regularization . . . 143

C Elementary Notions of Optimal Control **145**
C.1 Dynamical Systems . . . 145
C.2 Optimal-Control Problems . . . 147

List of Tables

2.1 Error-correcting code scheme 48

4.1 Minimum zones of four face-milled plates. 106
4.2 Minimum zones against the number of nodes of a mesh. 108

List of Figures

2.1 Plot of Fisher's iris data set after a PCA 14
2.2 2-D SOM of Fisher's iris data set 15
2.3 Parallel coordinates plot of Fisher's iris data set 16
2.4 Examples of suitable radial basis functions. 35
2.5 DDAG for a 4-class problem 46

4.1 Schematics of a uniform grid. 85
4.2 Quality of a uniform grid against its number of nodes. 87
4.3 Quality of a uniform grid against against measurement time. 88
4.4 Longitudinal deformations of the half cylinder. 101
4.5 Deformations of the half sphere. 102
4.6 Deformations of the half torus. 103
4.7 Deformations of the cone. 104
4.8 Deformations of plate 2-2. 106
4.9 Deformations of plate 2-3. 107
4.10 Deformations of the Lapmaster optical flat. 109

5.1 Outline of the ULM-based assimilation/prediction scheme 115
5.2 Assimilation results for the Lorenz 96 model 116
5.3 Evolution of RMSDs for the Lorenz 96 model 117
5.4 State assimilation and forecasts for a QG model 120
5.5 Evolution of the forecast RMSDs on the QG model 121

List of Algorithms

2.1 Neighborhood Clearing 19
2.2 Iterated Function Estimate 21
2.3 Hamming Distance 25
2.4 Hamming Distance Kernel 27
2.5 ANOVA Kernel 31
2.6 Cross-Validation Procedure 50
2.7 Bootstrapping Procedure 52

3.1 Function Estimation for the General Case 71
3.2 Function Estimation for the Symmetric Case 72
3.3 Symmetric Case Solution using a Cholesky Decomposition 72
3.4 Function Estimation with Optimal Coefficients 78

4.1 Projection onto a parametric surface 95

B.1 Cholesky Decomposition 138
B.2 Triangular System Solution 139
B.3 Conjugate Gradient Method 140
B.4 Symmetric Matrix Rank-2k Update 141
B.5 General Matrix-Vector Multiplication 141
B.6 Dot Product 142
B.7 Matrix Row Summation 142
B.8 Diagonal-Matrix Multiplication 143
B.9 Matrix-Diagonal Multiplication 143
B.10 Basic Tikhonov Regularization 144

Abstract

With the use of information technology in industries, a new need has arisen in analyzing large scale data sets and automating data analysis that was once performed by human intuition and simple analog processing machines. The new generation of computer programs now has to outperform their predecessors in detecting complex and non-trivial patterns buried in data warehouses. Improved Machines Learning (ML) techniques such as Neural Networks (NNs) and Support Vector Machines (SVMs) have shown remarkable performances on supervised learning problems for the past couple of decades (e.g. anomaly detection, classification and identification, interpolation and extrapolation, etc.).

Nevertheless, many such techniques have ill-conditioned structures which lack adaptability for processing exotic data or very large amounts of data. Some techniques cannot even process data in an on-line fashion. Furthermore, as the processing power of computers increases, there is a pressing need for ML algorithms to perform supervised learning tasks in less time than previously required over even larger sets of data, which means that time and memory complexities of these algorithms must be improved.

The aims of this research is to construct an improved type of SVM-like algorithms for tasks such as nonlinear classification and interpolation that is more scalable, error-tolerant and accurate. Additionally, this family of algorithms must be able to compute solutions in a controlled timing, preferably small with respect to modern computational technologies. These new algorithms should also be versatile enough to have useful applications in engineering, meteorology or quality control.

This dissertation introduces a family of SVM-based algorithms named *Unconstrained Learning Machines* (ULMs) which attempt to solve the robustness, scalability and timing

issues of traditional supervised learning algorithms. ULMs are not based on geometrical analogies (e.g. SVMs) or on the replication of biological models (e.g. NNs). Their construction is strictly based on statistical considerations taken from the recently developed statistical learning theory. Like SVMs, ULMS are using kernel methods extensively in order to process exotic and/or non-numerical objects stored in databases and search for hidden patterns in data with tailored measures of similarities.

ULMs are applied to a variety of problems in manufacturing engineering and in meteorology. The robust nonlinear nonparametric interpolation abilities of ULMs allow for the representation of sub-millimetric deformations on the surface of manufactured parts, the selection of conforming objects and the diagnostic and modeling of manufacturing processes. ULMs play a role in assimilating the system states of computational weather models, removing the intrinsic noise without any knowledge of the underlying mathematical models and helping the establishment of more accurate forecasts.

Part I

Unconstrained Learning Machines for Supervised Learning

Chapter 1

Introduction and Literature Review

1.1 Origins of Supervised Learning

Supervised learning is a machine learning approach that aims to estimate functions which link input observations to pre-determined targets. As the approach itself implies that computers host supervised learning algorithms, the notion of seeking such estimating functions, or patterns, is very old. The first modern cases of pattern searches in data date back from the early days of modern astronomy where empirical laws of motion were deduced from numerical observations. Kepler's laws of planetary motion and Newton's law of universal gravitation are early examples of pattern analysis on observational data [Kepler, 1619; Newton, 1687]. While finding nonlinear patterns was still a matter of good judgment toward the end of the 18th century, several mathematicians re-invented a systematic approach for finding linear relationships between observations: the so-called *method of least-squares* which was introduced to predict the position of celestial objects [Gauss, 1809; Legendre, 1805]. In the 19th century, graphical methods were vastly used to establish many nonlinear empirical laws of chemistry and electricity which are still used nowadays (e.g. Ohm's law [Ohm, 1827]). Log-log and semi-log graphs were used together with nomograms [d'Ocagne, 1885] and specialized slide rules to analyze patterns in observations, and remained used for the purpose of pattern analysis until the early 1960s.

The early 20th century was marked by the emergence of modern statistical science which was due, for a large part, to the English statistician Fisher who also introduced the very first machine learning tools. The *Fisher's linear discriminant* was first described in

1936 to perform what is now referred as linear binary classification [Fisher, 1936]. This method was closely related to the analysis of variance (ANOVA) and the regression analysis which was pioneered the century before, with the exception that targets were now discrete objects and not numerical values. It was also related to Principal Component Analysis (PCA) [Pearson, 1901] and Factor analysis. The method of least squares that spawned regression analysis and eventually the Fisher's linear discriminant was a simple form of what is now called *regularization*. The regularization theory is concerned with the introduction of supplementary information to solve problems highly sensitive to perturbations or *ill-posed problems*. The Tikhonov regularization [Tychonoff, 1963] marked the crown achievement of regression methods at the dawn of computer technology and machine learning.

1.2 The First Generation of Supervised Learning Algorithms

Artificial Intelligence and Machine Learning were born through the work of Rosenblatt on the *perceptron* [Rosenblatt, 1958]. The Perceptron was the first on-line linear binary classifier meant to be powered by computers. It was the earliest example of a feed-forward neural network and its study [Novikoff, 1963] generated today's statistical learning theory. Soon after Rosenblatt's breakthrough, several authors proposed supervised learning algorithms to solve real-life problems such as the *learning matrices* [Steinbuch, 1965] and the *Madaline* [Widrow, 1962].

In the years that followed, decision trees and hidden markov models were also introduced to help computers building logical patterns between observations, although, unlike the perceptron, these methods were not based on neuron models. The search for nonlinear relations hidden in data started to be described with elements of *algorithmic information theory*, in which randomness is simply defined as the absence of patterns in the observations [Chaitin, 1966; Kolmogorov, 1965].

1.3 Second Generation (1970s-1980s)

The research on binary classification led Vapnik and Chervonenkis [1971] to formulate non-asymptotic probabilistic bounds for the rate of convergence of linear binary classifiers, regardless of the distribution of the observations. A decade later, Vapnik generalized the results to nonlinear classifiers [Vapnik, 1982]. These bounds, which are expressed by what is now known as the *VC dimension*, led to the *empirical risk minimization principle* that links empirical risks of learning algorithms to necessary and sufficient conditions for the uniform convergence of their means to their expected values (i.e. a law of large numbers in a functional space). In 1991, the empirical risk minimization principle was finalized with necessary and sufficient condition for the convergence in probability toward the best possible result [Vapnik and Chervonenkis, 1991]. These probabilistic results paved the way to the modeling of supervised learning algorithms as minimization problems that attempt to minimize a risk functional over a set of functions.

During the 1980s, supervised learning algorithms underwent a nonlinear transformation with the sigmoid approximation and the resulting *back-propagation networks* [LeCun, 1986; Rumelhart et al., 1986]. The discovery of nonlinear patterns hidden in classification data was then reduced to the evaluation of gradients using gradient-based optimization techniques that were developed three decades prior (e.g. conjugate gradient method [Hestenes and Stiefel, 1952]). Unfortunately, back-propagation network techniques were plagued by multiple sub-optimal minima and their inability to converge toward a global optimizer. Nevertheless, they helped the launch of modern day data mining and bio-informatics.

1.4 Third Generation (1990s-2000s)

The third generation of supervised learning algorithms is marked by the use of *kernel methods*. The theory of kernels is actually a century old [Mercer, 1909] and can be seen as the

generalization of definite matrices to functions, with an emphasis on positive definite properties. Aronszajn [1950] studied reproducing kernel Hilbert spaces which are centered around positive definite kernels. The resulting notions were eventually used in approximation and regularization theory. It led to the interpretation of kernels as measures of distances and angles in an induced Hilbert space and their use in pattern classification [Aizerman et al., 1964]. The finitely positive definite property of kernels which was a key aspect to the construction of general kernels on exotic objects was introduced a couple of decades later by Saitoh [1988]. In the early 1990s, kernel methods were used in a machine learning context by Girosi et al. [1995] to build new nonlinear neural network architectures.

Learning algorithms such as Support Vector Machines (SVMs) were the direct results of the combination of kernel methods, the empirical risk minimization principle and the maximum margin paradigm [Vapnik, 1995, 1998]. This new family of learning algorithms was fitted for both nonlinear classification and regression tasks. Nevertheless, the VC framework of SVMs provided loose and pessimistic bounds, and therefore Rademacher complexities were soon introduced as empirical estimate of the VC dimension to remedy all these problems [Koltchinskii and Panchenko, 2000]. Several flavors of SVMs were introduced after Vapnik's landmarking work such as the *Least-Squares SVM* (LS-SVM) [Suykens and Vandewalle, 1999] as well as computational improvements such as the Sequential Minimal Optimization (SMO) [Platt, 1999]. Then developments on Bayesian kernel methods followed [Smola and Schölkopf, 2003].

1.5 Current Status and ULMs

Kernel methods and SVMs have been applied to a wide range of problems such as computational biology, bio-informatics and gene analysis [Ding and Dubchak, 2001; Lee et al., 2003; Santosa et al., 2002, 2007], fluid mechanics [Oladunni and Trafalis, 2006; Oladunni et al., 2006; Trafalis et al., 2005], manufacturing engineering [Gilbert et al., 2010, 2009a;

Malyscheff et al., 2002; Prakasvudhisarn et al., 2003; Raman et al., 2005], meteorology [Adrianto et al., 2005; Gilbert et al., 2009b; Mansouri et al., 2007; Trafalis et al., 2007] and even political science [Malyscheff and Trafalis, 2003]. Financial applications of SVMs include short term portfolio management [Ince and Trafalis, 2006a], exchange rate prediction [Ince and Trafalis, 2006b] and stock price prediction [Ince and Trafalis, 2003, 2007]. Other applications are in the area of production [Alenezi et al., 2005], inventory transactions [Beardslee and Trafalis, 2005] and web mining [Chung et al., 2002].

Unconstrained Learning Machines (ULMs) are the natural evolution of the works of Gilbert and Trafalis on error-tolerant SVMs [Trafalis and Gilbert, 2005, 2006, 2007]. The resulting formulations were based either on large linear programming problems or medium-sized second-order cone programming problems, which both presented a computational challenge on large sets of data. The present form of ULMs was born from a deliberate simplification of these underlying mathematical programming problems so that they could retain the error-tolerant properties while allowing the fast computation of optimal solutions [Gilbert and Trafalis, 2009]. This increase in computational speed allowed to embed ULMs in more complex structures such as pattern searches in functional spaces, on-line processing schemes or data thinning procedures.

Chapter 2

Kernel Methods for Supervised Learning

The term “kernel methods” is a generic term that encompasses all treatments that a set of data can receive from a particular category of functions called *kernels* which are discussed in Section 2.2. More exactly, the data is processed using a subset of kernels which include the so-called positive definite kernels. Kernel methods are commonly used by SVMs to process non-trivial supervised learning tasks such as nonlinear binary classification and nonlinear nonparametric regression [Vapnik, 1982, 1995, 1998]. The analysis and pretreatment of the observations is often a crucial step for a successful application of kernel methods to them. It leads to faster and more stable computation of hidden patterns in the data. To this end, a brief review of common data pretreatment techniques is discussed in Section 2.1. Once data treatments and kernel methods are carefully chosen, it then becomes possible to quickly recover complex information from large sets of data. Estimating the patterns that links data features together is the heart of the *function estimation problem* that is reviewed in Sections 2.3 and 2.4.

2.1 Data

2.1.1 Source and Nature of the Processed Data

Source of Data and Mathematical Assumptions

The data to be processed by learning machines comes in various shapes and forms. Modern pattern recognition algorithms have been used to analyze sequences of numbers, letters, images, videos, network graphs and other objects. Despite the sheer amount of possible

data treatments, all these techniques have at least two points in common:

- There exists a non-trivial hidden pattern within the data stream;
- It is possible to quantify similarities between objects found in the data stream.

Many machine learning techniques add further statistical restrictions such that all objects generated by the source of data are distributed according to a well-behaved distribution. Usually these techniques assume a perfect knowledge of the source of data and, whenever observations violate the mathematical restrictions, all outliers are modified or deleted before treatment. Such approaches can be detrimental to the pattern analysis of a data set and can produce partially unreliable results. Therefore, it becomes necessary to adopt machine learning approaches that require the least amount of mathematical assumptions on the source of data while increasing the reliability of the results. Kernel methods can be used to develop learning algorithms that fit these requirements. They only require the existence of a source of data, with no further statistical assumptions on it, the existence of a similarity measure between observations, and the existence of a *pattern* hidden within the data.

Hence, we will assume, in all the following, that the data which is being processed satisfies the following assumptions:

1. The source of data generates a collection of objects $e_1, e_2, \ldots$ that belongs to a *measurable observation space* $(E, \mathfrak{S})$ where $\mathfrak{S}$ is a σ-algebra over the set E.
2. There exists a measure space (Ω, Σ, P) (Σ is a σ-algebra over Ω and $P : \Sigma \to \overline{\mathbb{R}}$ is a measure over Σ such that $P(\Omega) = 1$) and a measurable function $g : \Omega \to E,\ \omega \mapsto e$ which outputs are the observations $e_1, e_2, \ldots$, etc.
3. There exists a *measurable pattern function* $\psi : E \to [0, a]$ with $a > 0$ such that

$$\langle \psi(g) \rangle = \int_\Omega \psi(g) \mathrm{d}P = 0. \tag{2.1}$$

The first assumption ensures that it is always possible to construct a measure of similarity between observations, which is a crucial requirements for kernel methods. The second assumption assumes that all observations were effectively generated according to an unknown measurable function g. In other words, g is the source of data. The last requirement claims that there exists a *pattern* hidden within the observations. It assumes that, out of all possible *events* in Ω, the images by a pattern function ψ of all corresponding observations are null on average. The codomain of ψ is a closed interval of $\mathbb{R}$ in order to avoid seeking patterns which values can explode towards infinity. Vapnik [1995] enumerated several properties regarding the empirical expected values of $\psi(g)$ and its probabilistic bounds. He established guidelines for powerful learning algorithms that detect patterns within data sets which are satisfying the general conditions above. We will use the results on these bounds to generate specialized learning machines that we will name *Unconstrained Learning Machines*.

Data in Supervised Learning

Supervised learning has special requirements depending on the form of the observation space E. Each element of E must include *target* components that will be matched against the rest of the components by the pattern function. The target components are required to be numerical quantities to allow treatments by numerical algorithms. Nevertheless, non-numerical targets can still be represented by real-valued components using an encoding scheme and an appropriate supervised learning algorithm. Target components are necessary to supervised learning methods since they are meant to recover patterns by matching *known* targets with the other components of the observation.

Hence, in all the following, the space E is assumed to be identical to $X \times \mathbb{R}^p$ where X is a measurable space and $p \in \mathbb{N}^*$. Each observation $e \in E$ is decomposed into $e = (x, \mathbf{y})$ where x is an element of X and the vector $\mathbf{y} \in \mathbb{R}^p$ is the target of x. Using Equation 2.1 as a model, supervised learning algorithms can be built to estimate pattern functions $f : X \to [-a, a]^p$

($a > 0$) such that

$$\| f(x) - \mathbf{y} \| \leqslant \varepsilon, \tag{2.2}$$

for all $(x, \mathbf{y}) \in X \times \mathbb{R}^p$, where $\|\cdot\|$ is a norm on $\mathbb{R}^p$ and $\varepsilon > 0$ is an arbitrary value. However, it is possible to simplify this problem by avoiding the use of a norm on $\mathbb{R}^p$. For instance, Unconstrained Learning Machines will be built to find p pattern functions $f_1, \ldots, f_p$ (with $f_i : X \to [-a_i, a_i]$, $a_i > 0$ for all $i \in [\![1, p]\!]$) such that

$$| f_i(x) - y_i | \leqslant \varepsilon, \tag{2.3}$$

for all $(x, y_i) \in X \times \mathbb{R}$ and $i \in [\![1, p]\!]$, where $\varepsilon > 0$ is an arbitrary value.

Numerical Data: Discrete and Continuous Components

The most particular case for the space X is to be isomorphic to $\mathbb{N}^n$, $\mathbb{R}^n$ or a Cartesian product of both. In this case, all observations are *numerical data*, and their processing depends if observations have discrete components (X is equivalent to $\mathbb{N}^n$), continuous components ($X = \mathbb{R}^n$), or mixed components.

Measures of similarity between observations with continuous components can be derived from the usual norms on $\mathbb{R}^n$ such as the Euclidean norm. These components are often measurements with physical dimensions which can be, for example, records of pressure, electrical intensities, frequencies, etc. This type of observations often receive a *pretreatment* that aims to render their components independent of the origin of their physical scale and the choice of their physical units.

Discrete components can be *ordered*, or not. Ordered components correspond to numerical values on a *scale* or *finite amounts* such as a number of days, the cardinality of a set or a magnitude. The measure of similarity between two observations with this type of components can be the same as the measure of similarity between observations with continuous components and they can receive the same type of pretreatment, if any.

On the other hand, non-ordered discrete components are identical to *categorical values* which are elements belonging to a finite countable set that has no binary relation between its elements. For example, categorical values can represent the type of an object, or an indication of magnitude (e.g., red or blue). The measures of similarity between two observations with categorical components are difficult to represent, and they strongly depend how the categorical data has been *encoded* i.e. represented with numbers in $\mathbb{N}$.

However, if $x = (x_1, \ldots, x_m) \in \bigotimes_{i=1}^{m} X_i = X$ is a categorical observation with m categorical components $x_1, \ldots, x_m$ taking values in m *finite* categorical spaces $X_1, \ldots, X_m$, then it is always possible to encode x into a $z \in \{0,1\}^n \subset \mathbb{N}^n$ with $n = \sum_{i=1}^{m} |X_i|$. To do so, each x_i is mapped into $\{0,1\}^{|X_i|}$ such that the j-th bit is equal to 1 and all the others are equal to 0 if x_i is equal to the j-th element of the set X_i. For example, if $X_i = \{\text{'a'}, \ldots, \text{'f'}\}$, then $x_i = \text{'c'}$ is represented by 001000. This representation with binary strings is the most intuitive way to handle categorical data and we will assume, in all the following, that categorical data is always given under a binary form.

Structured Data

The space X can correspond to more abstract objects which are defined by special structures such as time series, matrices, graphs and strings. Although these objects can also be represented via numerical data, the pretreatment and the measures of similarity used on numerical data can be both meaningless and inappropriate for these objects. There is a vast number of methods for representing and pre-processing these objects which cannot be summarized in this dissertation. The reader should be referred to Shawe-Taylor and Cristianini [2004] for indications regarding structured data and the appropriate way to represent such data before treatment with kernel methods.

2.1.2 Pre-Processing of Numerical Data

Missing Values

The records of some numerical observations in X can be sometimes incomplete and some of their components might be missing. Many learning machines were not built to process observations with missing components in order to recover a pattern hidden within a data set. Consequently, several strategies can be adopted in order to circumvent possible lack of records in the observation set.

The first approach is to remove observations with missing components. This method is not detrimental when data sets contain a significant number of observations that are similar to the ones that are being deleted. Furthermore, it has the advantage to be a form of data thinning which can reduce the time needed to process a large number of observations.

The other strategies depend on the nature of the missing values which can be of two kinds:

1. The components are missing because they were not recorded. This case implies that the missing values actually existed and were subject to a pattern, but they are absent from the records because of a failure of the recording equipment.

2. The components were never generated. In this case, the pattern behind this observation is inconsistent with the rest of the observations and these particular observations with missing values must be deleted.

There is a possibility to *infer* the missing components in the scenario where the absence of data is due to a failure to record it. The missing values can be *interpolated* from other observations with similar components. However, it requires to find first the pattern hidden within the data which leads to a causality dilemma similar to the paradox of the chicken and the egg. Nevertheless, practical approaches are using observations with no missing values to estimate a pattern between components, then use this pattern to interpolate the

missing values, then augment the set of observations with the interpolated values and use it to search again for a new pattern.

There are numerous techniques which are used to infer missing components. They vary in sophistication and processing time. When data sets are large, it is highly recommended to select simple techniques with low time complexities such as the calculation of simple statistical measures. For example, a subset of the observation set can be selected such that it contains observations with no missing components which are similar to the observation with missing components. Then a measure of central tendency of the empirical distribution of the components (the median values or the arithmetical means) can then be used to substitute the missing components.

Data Visualization

Patterns in numerical data can be trivial to detect when the data in X belongs to spaces that have at most four dimensions, since we are all gifted with a biological processor which is extremely potent at analyzing patterns in $\mathbb{R}^4$ (i.e. the human brain). Hence, visual aids can be powerful tools to infer good candidate patterns or appropriate pretreatments, even for large and very complex sets of data. However, it becomes extremely difficult to guess patterns when the data belongs to a space with a great number of dimensions.

Fortunately, there are approaches that can be used to visualize multidimensional data, and they often come in two stages:

1. The first stage is about the construction of an optimal map $\phi : X \to \mathbb{R}^n$ so that the new coordinates give a better sense of the geometrical structure of the cloud of observations in the higher dimensional space X.

2. The second stage concerns the plotting of the mapped data into 1-D, 2-D or 3-D graphs or tables, in a way that is most convenient to detect similarities between observations.

Principal Component Analysis (PCA) [Pearson, 1901], Kernel PCA (KPCA) [Diamantaras and Kung, 1996; Schölkopf et al., 1997; Mika et al., 1999] and Self-Organizing Maps (SOMs) [Kohonen, 1982, 2001; Kohonen and Mäkisara, 1986] form the core of the first stage. A PCA aims to map data linearly onto a subspace of X which basis corresponds to the axes of maximum inertia of the cloud of observations. The inertial axes, called principal axes, are ordered by decreasing explained variance. Hence, when the principal components of the mapped observations are truncated to the first one, two, three or four variables, the resulting projection in a lower dimensional space is the best *linear* projection possible, in the sense that it maximizes the explained variance in the lower dimensional space (see Figure 2.1).

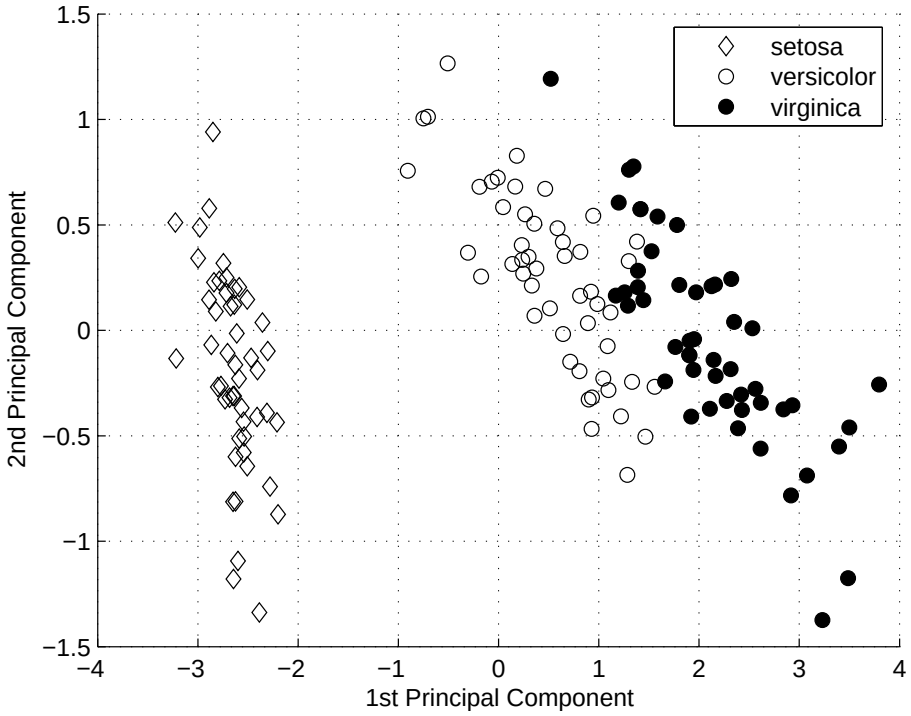


Figure 2.1: Bi-dimensional plot of the first two principal components after a PCA of the observations on made on three distinct species of iris flowers by Fisher [1936]. Observations tend to cluster according to the species the flower belong to.

A KPCA is a PCA that is performed when the observations are initially mapped into an higher dimensional Hilbert space where its metric is induced by the choice a particular

type of function called *kernel*. The inertial axes are warped according to the kernel in order to best fit the mapped cloud of observations. This is, in essence, a nonlinear extension of a PCA using kernel methods.

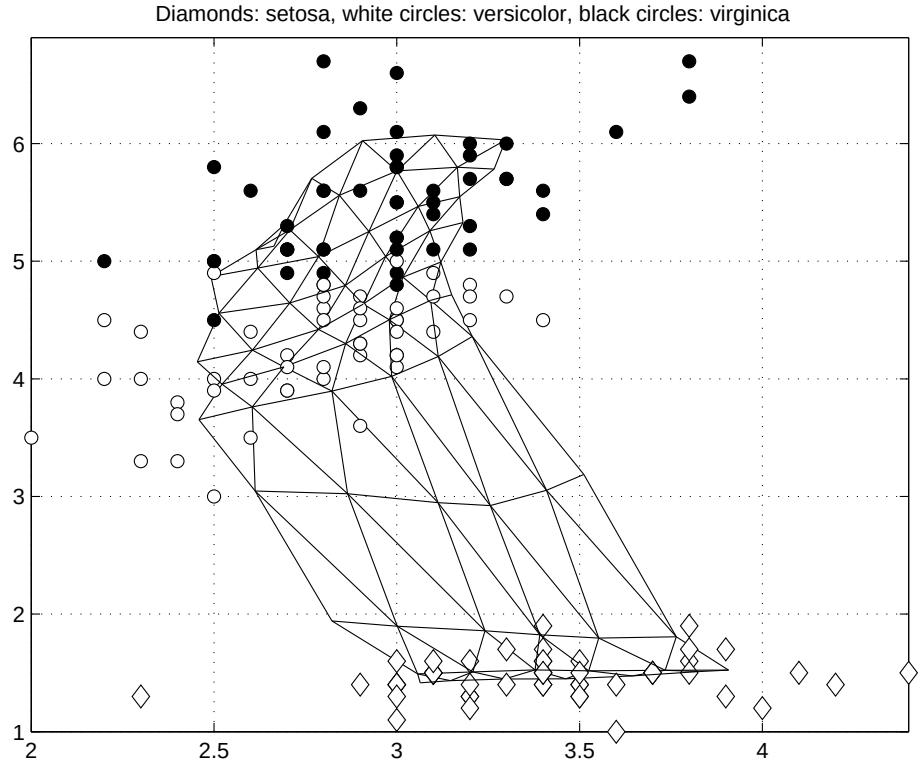


Figure 2.2: 2-D SOM of Fisher's iris data set [Fisher, 1936] and mapped observations. Like for the PCA, observations tend to cluster according to the species the flower belong to.

SOMs is a family of Artificial Neural Networks (ANNs) which maps observations onto a regular two or three dimensional grid. Each node of the grid is associated with an ANN model which is computed with the SOM algorithm (see Figure 2.2). An observation is mapped onto the grid node which model has the smallest distance from the observation, according to some chosen metric (which can be defined by a kernel). Similar to the KPCA, SOMs are also nonlinear generalizations of a PCA [Yin, 2007].

The second stage of treatment for the visualization of multidimensional data consists

in representing the mapped data in a convenient graph or table. If the mapped data has a single dimension, then histogram-related plots are the most suitable to display the data. If the mapped data has two or three dimensions, then 2-D or 3-D scatter plots (see Figures 2.1 and 2.2) are the best. It is possible to visualize up to six dimensions at once with the 3-D plot of a vector field, but, for more dimensions, only parallel coordinates plots [d'Ocagne, 1885] (and related plots such as Andrews plots [Andrews, 1972] and generalized parallel coordinates plots [Moustafa, 2009]) can visualize the mapped data (see Figure 2.3).

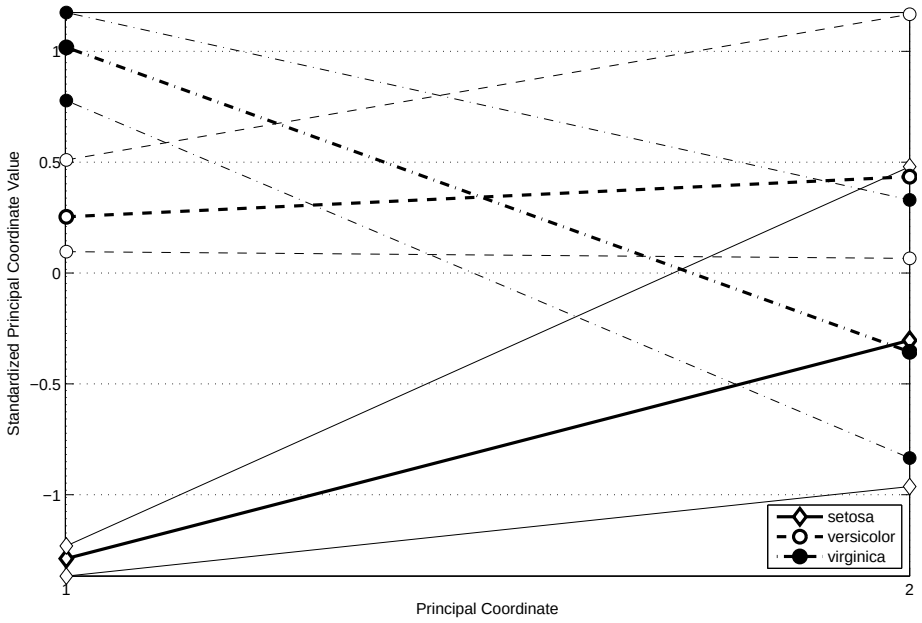


Figure 2.3: Parallel coordinates plot of Fisher's iris data set [Fisher, 1936] over the first two principal coordinates after a PCA and normalization. Only the the first and third quartile envelopes as well as the median values for each flower species are plotted.

Normalization of Continuous Numerical Data

Continuous numerical data is often the result of physical measurements which are linked to the choice of arbitrary physical units and scales. When an observation contains several physical quantities measured with different units and/or different scales, the cloud of observations in the space X becomes arbitrarily stretched along some axes and this can

significantly impact the relevance of other crucial components and make patterns undetectable.

To remedy the problems caused by the choice of arbitrary units and scales, a *normalization* procedure determines statistical measures of central tendency and dispersion (variability) for every component of all the observations. In other words, given ℓ observations $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ in $\mathbb{R}^n$, we compute n measures of central tendency $m_1, \ldots, m_n$ and n measures of dispersion $s_1, \ldots, s_n$ from the empirical distributions of every component. For example, the m_i's can be the means (or medians, if there are too many outliers) of the components and the s_i's can be the standard deviations (or inter-quartile ranges) of the components. Once the statistical measures are determined, all the observations $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ are transformed into vectors $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ in $\mathbb{R}^n$, which components are dimensionless, by using the following formula:

$$(\mathbf{z}_i)_j = \frac{(\mathbf{x}_i)_j - m_j}{s_j}, \tag{2.4}$$

for all i in $[\![1, \ell]\!]$ and all j in $[\![1, n]\!]$. If follows that the components of the vectors $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ have a similar variability around zero, which makes them homogeneous for each observation.

2.1.3 Data Thinning

This section only covers data thinning methods that have been developed to be used with ULMs. A large number of traditional sampling techniques can be used to thin large data sets [Cochran, 1977; He and Garcia, 2009], but the applications that led to investigate ULMs required data thinning methods which keep the overall geometrical structure of the cloud of observations in the space X. A couple of approaches were developed to reduce the burden of processing large amounts of redundant (or quasi-redundant) observations. One approach is independent of the search for a pattern function and is based on the idea of a Voronoi tessellation using metrics defined by kernels. The other approach is a pipe-lining scheme

based on the idea that the properties of the estimate of the pattern function gradually *emerge* as the ULM is sequentially fed with batches of observations.

Neighborhood Clearing

This data thinning approach is based on the use of *kernels* (see Section 2.2) which can define metrics. If we consider a finite set $\mathscr{X} = \{x_1, \ldots, x_\ell\}$ of observations in X, then, given a scalar $\tau > 0$ and a kernel $k_\sigma : X \times X \to \mathbb{R}$, we can define a closed ball of radius τ centered at $x \in \mathscr{X}$ by

$$B_\tau[x] = \left\{ z \in X : k_\sigma(x,x) + k_\sigma(z,z) - 2k_\sigma(x,z) \leqslant \tau^2 \right\}. \tag{2.5}$$

We can notice that, for the metric induced by the kernel k_σ, the observations $z \in \mathscr{X}$ which are in $B_\tau[x]$ are "similar" to $x \in \mathscr{X}$, up to a threshold $\tau > 0$. Hence, these other observations are *quasi*-redundant since they do not differ much from a *representing* observation x for a given similarity measure (the distance induced by the kernel). Consequently, the observations $z \in \mathscr{X}$ which are in $B_\tau[x]$ can be removed from the set $\mathscr{X}$ without inducing great modifications in the shape of the cloud of observations in X. If we repeat this step sequentially, then only a subset of the original set $\mathscr{X}$ remains for which no distance between observations is smaller than τ for the metric induced by the kernel k_σ. Algorithm 2.1 implements this Neighborhood Clearing method.

Iterative Construction of the Pattern Function Estimate

This data thinning approach was based on an experimental result: as the number of observations grows larger, the pattern function estimate given by ULMs quickly stalls and new observations barely bring any change to it. This statement is very accurate if the first batches of observations that are fed to ULMs were chosen so as to keep the geometrical shape of the entire cloud of observations in the space X.

Algorithm 2.1: Neighborhood Clearing

Function $[S]$ = `NeighborhoodClearing` $(\mathscr{X}, k_\sigma, \tau)$
Input: set $\mathscr{X}$, kernel k_σ with parameter σ, threshold $\tau > 0$.
Output: set S.

$S \leftarrow \mathscr{X}$, $L \leftarrow |S|$, $i \leftarrow 1$
while $i < L$ **do**
  for $j \in [\![1, i]\!]$ **do** $v_j \leftarrow s_j$
  $a \leftarrow k_\sigma(s_i, s_i) - \tau$, $l \leftarrow i$
  for $j \in [\![i+1, L]\!]$ **do**
    if $a + k_\sigma(s_j, s_j) > 2k_\sigma(s_i, s_j)$ **then** $l \leftarrow l+1$, $v_l \leftarrow s_j$
  end
  $V \leftarrow \{v_1, \ldots, v_l\}$, $S \leftarrow V$, $L \leftarrow |S|$, $i \leftarrow i+1$
end
return S

Consider a finite set $\mathscr{X} = \{x_1, \ldots, x_\ell\}$ of observations in X and the corresponding set $\mathscr{Y} = \{y_1, \ldots, y_\ell\}$ of targets. Algorithm 2.1 can be used iteratively to build a partition $\mathfrak{X} = \{X_1, \ldots, X_m\}$ of $\mathscr{X}$ $(1 \leqslant m \leqslant \ell)$ and then deduce a partition $\mathfrak{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$ of $\mathscr{Y}$. Any set X_i in $\mathfrak{X}$ will be such that a $x \in X_i$ has "cleared its neighborhood" up to a certain distance and all the sets in $\mathfrak{X}$ have the same geometrical shape overall. If f_1 is pattern function estimate given by an ULM from X_1 and $\mathbf{y}_1$, then, for all $i \in [\![1, |X_m|]\!]$, we have that

$$f_1\big((X_m)_i\big) = (\mathbf{y}_m)_i + r_i, \tag{2.6}$$

where the r_i's in $\mathbb{R}$ are residuals between the outputs of f_1 on a test set X_m and the target $\mathbf{y}_m$. If the residuals are too large, then we are forced to compute a new function f_2 from the set X_2 that accounts for the residuals. However, we can use the results found during the computation of f_1 to describe parts of f_2. For instance, the target vector $\mathbf{y}_2$ is replaced by the residual between $\mathbf{y}_2$ and the outputs of f_1 over X_2, i.e.

$$(\mathbf{y}_2)_i \leftarrow (\mathbf{y}_2)_i - f_1\big((X_2)_i\big), \tag{2.7}$$

for all $i \in [\![1, |X_2|]\!]$. Then a pattern function estimate f_2 is computed by an ULM from X_2

and $\mathbf{y}_2$ and residuals are calculated similarly to Equation 2.6. If the residuals are too large, then the scheme is repeated or until all sets in $\mathfrak{X}$ have been used, or until residuals are deemed sufficiently small. The resulting pattern function estimate f after k iterations will be equal to

$$f = \sum_{i=1}^{k} f_i. \tag{2.8}$$

This approach is both a data thinning method and a method to speed up the computation of pattern function f. Indeed, the computational time is a convex polynomial function π of the size l of the observation set. Hence, if the data sets $X_1, \ldots, X_m$ have reasonably small sizes $l_1, \ldots, l_m$, then we will have

$$\pi\left(\sum_{i=1}^{m} l_i\right) \gg \sum_{i=1}^{k} \pi(l_i), \tag{2.9}$$

where k is the number of iterations needed to terminate the procedure.

Algorithm 2.2 implements the procedure that iteratively constructs a pattern function estimate. The algorithm is fast only if the computational step on line 5 and the iterated summations of the function estimates are fast to compute. It can be noted that, when using ULMs, the different functions $f_1, \ldots, f_k$ can be defined with different kernels each time (see Sub-Section 2.3.2).

2.2 Kernels

Kernels play a major role in Machine Learning. Despite that the techniques using them have been known for more than half a century [Aizerman et al., 1964; Courant and Hilbert, 1953], their utilization by the machine learning community is less than thirty years old. These functions are used to construct, in a implicit manner, simple machine learning frameworks that efficiently handle highly nonlinear patterns in observational data. They provide an implicit way to define new metrics which is a crucial characteristic when two objects

Algorithm 2.2: Iterated Function Estimate

Function $[f]$ = `IteratedFunctionEstimate` $(\mathfrak{X}, \mathfrak{Y}, \varepsilon)$
Data: iteration limit $k_{\max} > 0$.
Input: partitions $\mathfrak{X} = \{X_1, \ldots, X_m\}$ and $\mathfrak{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$ of the sets $\mathscr{X}$ and $\mathscr{Y}$, tolerance $\varepsilon > 0$.
Output: function f.

$k_{\max} \leftarrow \min(k_{\max}, m-1)$, $\mathbf{r} \leftarrow \mathbf{y}_m$, $k \leftarrow 0$
while $\|\mathbf{r}\| > \varepsilon$ *and* $k \leqslant k_{\max}$ **do**
    $k \leftarrow k+1$
    for $i \in [\![1, |X_k|]\!]$ **do** $(\mathbf{y}_k)_i \leftarrow (\mathbf{y}_k)_i - \sum_{j=1}^{k-1} f_j\big((X_k)_i\big)$
    Compute an estimate f_k using X_k and Y_k.
    for $i \in [\![1, |X_m|]\!]$ **do** $r_i \leftarrow r_i - f_k\big((X_m)_i\big)$
end
$f \leftarrow \sum_{i=1}^{k} f_i$
return f

need to be compared. Kernels possess exceptional closure properties that gives them extreme flexibility in many applications.

2.2.1 Definitions and Properties

A kernel is a simple function of two variables over a measurable space with very simple properties. It must be real-valued, symmetric with respect to its argument and square-integrable. The following definition was adapted from Mercer [1909]:

Definition 2.1 (Kernel)**.** *If X is a measurable space, then a* kernel *is a real-valued function over $X \times X$ that is symmetric and square-integrable.*

Kernels need to be real-valued to allow comparisons of measures of similarity between elements that belong to the measurable space X. The symmetry and square-integrability properties of kernels are properties which are necessary, but not sufficient, to the construction of new metrics. The sufficiency aspect comes with the *finite positive definite* property defined below, which was adapted from Saitoh [1988]:

Definition 2.2 (Finitely Positive Semi-Definite Function)**.** *If X is a measurable space, then a function $f : X \times X \to \mathbb{R}$ is* finitely positive semi-definite *if it is symmetric and if*

$$\Lambda = \sum_{i=1}^{m} \sum_{j=1}^{m} f(x_i, x_j)\lambda_i\lambda_j \geqslant 0,$$

for any $m \in \mathbb{N}$, $\lambda_i \in \mathbb{R}$, $x_i \in X$ and $i \in [\![1, m]\!]$. If $\Lambda = 0$ only for $\lambda_1 = \ldots = \lambda_m = 0$, then f is finitely positive definite.

The key aspect of kernel methods is that finitely positive semi-definite kernels can be expressed as inner products in Hilbert spaces. This result was first showed by Mercer in 1909 in a theorem that now bears his name:

Theorem 2.1 (Mercer's Theorem)**.** *Let X be a measurable space and let $k_\sigma : X \times X \to \mathbb{R}$ be a kernel. Then there exists a unique Hilbert space F and a map $\phi : X \to F$ such that*

$$k_\sigma(x, y) = \langle \phi(x), \phi(y) \rangle_F ,$$

for all $(x, y) \in X^2$ if and only if *the function k_σ is a finitely positive semi-definite function.*

Theorem 2.1 implies that, given a finitely positive semi-definite kernel k_σ, the function $\phi : X \to F,\ x \mapsto k_\sigma(x, \cdot)$ is a linear map in an Hilbert space F uniquely defined by k_σ. Hence, the trick used by kernel methods consists into replacing complex nonlinear relations between objects in X (or just dot products) by an expression made of finitely positive semi-definite kernels. By this trick, nonlinear relations become equivalent to linear mappings in Hilbert spaces for which it is unnecessary to properly describe their (complicated) metrics since they are induced by the choice of the kernels. Since processing and interpreting linear maps is trivial, complex nonlinear transformations in X become simple operations in Hilbert spaces and these transformations are easily computed with the help of simple

kernels. The complicated map ϕ from the observation space X (sometimes called the *input space*) into the induced Hilbert space (called the *feature space*) needs not to be described or known, which is saving computational time and resources.

Geometry in the feature space is directly described by the properties of any Hilbert space which are reformulated with the help of kernels. For instance, the Cauchy-Schwarz inequality or the Pythagorean theorem can be rewritten by replacing all inner product expressions with kernel equivalents. This gives an interpretation of angles and distances in the Hilbert space induced by a given kernel, and makes it easier to design novel machine learning processes which are based on the geometrical transformation of the mapped cloud of observations.

Corollary 2.1 (Geometry in the Feature Space)**.** *Let X be a measurable space and let $k_\sigma : X \times X \to \mathbb{R}$ be a finitely positive semi-definite kernel. The kernel k_σ uniquely defines a Hilbert space F and a map $\phi : X \to F$ where there exists a $\theta \in [0, \pi]$ such that*

$$k_\sigma(x,y) = \cos(\theta)\sqrt{k(x,x)k(y,y)},$$

for all x and y in X. Furthermore, we have that

$$\left\| \sum_{i=1}^{m} a_i \phi(x_i) \right\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j k_\sigma(x_i, x_j) = \mathbf{a}^{\mathrm{t}} \mathbf{K} \mathbf{a},$$

for all $a_i \in \mathbb{R}$ and $x_i \in X$, $i \in \mathbb{N}$. The Gramian matrix $\mathbf{K}$ is called the kernel matrix *and is, by the definition of k_σ, a symmetric positive semi-definite matrix.*

Proof. Since F is an Hilbert space where the inner product is defined by k_σ according to Theorem 2.1, the first equality is the direct result of the Cauchy-Schwarz inequality and the second equality is given by the Pythagorean theorem. □

The space of finitely positive semi-definite kernels is closed under some algebraic operations. Hence, it is possible to construct more elaborated kernels from simpler kernels by

using additions, multiplications, limits, etc. The closure properties are listed below:

Theorem 2.2 (Closure Properties of Kernels)**.** *Let X and Z be two measurable spaces and let ϕ be a map from X to Z. Let k (respectively κ) be a finitely positive semi-definite kernel over X^2 (respectively Z^2). Let p be a polynomial with real positive coefficients. Finally, let x and y be two elements of X. The following functions are finitely positive semi-definite kernels:*

- $\prod_{i \geqslant 0} k_i$ *and* $\lim_{i \to \infty} k_i$*, where* $\{k_i\}_{i \in \mathbb{N}}$ *is a sequence of finitely positive semi-definite kernels that converges pointwise.*
- $p(k)$, $\exp(k)$ *and* $(x,y) \mapsto \kappa(\phi(x), \phi(y))$.
- $(x,y) \mapsto \int_X f(x,z) f(y,z) \mathrm{d}z$ *where* f *is a symmetric function.*
- $(\mathbf{x}, \mathbf{y}) \in (\mathbb{R}^n)^2 \mapsto \mathbf{x}^{\mathrm{t}} \mathbf{K} \mathbf{y} \in \mathbb{R}$ *for any positive semi-definite* $n \times n$ *matrix* $\mathbf{K}$.

These closure properties are extremely useful for quickly building simple kernels that measure very complicated similarities between objects belonging to abstract measurable spaces. This allows the processing of very complicated objects by learning algorithms and the easy detection of non-trivial nonlinear patterns hidden within data.

2.2.2 Kernels on Categorical Data

Hamming Distance

Measuring the likeliness between two observations containing an equal number of categorical components is similar to the computation of the number of positions at which the symbols of two strings are different. In other words, the measure of similarity between two such observations is based on the Hamming distance between their string representations [Hamming, 1950].

Given two strings s_1 and s_2, the Hamming distance measures the minimum number of substitutions that are necessary to change s_1 into s_2 and reciprocally i.e., it measures the numbers of categorical components that are different between s_1 and s_2. Hence, since the Hamming distance is a metric on the set of strings of equal length, *any polynomial with positive coefficients or Gaussian function* of the Hamming distance between two strings is a kernel on categorical data. When the data is in binary form, each possible string of length n is a vertex of an n-dimensional unit hypercube and the Hamming distance between two string is equivalent to the Manhattan distance between the vertices.

If two observations s_1 and s_2 with n categorical components are represented in binary form (i.e. $(s_1, s_2) \in \{0,1\}^n \times \{0,1\}^n$), then Algorithm 2.3 given by Wegner [1960] computes the Hamming distance $d_{\mathrm{H}}(s_1, s_2)$ between these two observations. This algorithm is extremely efficient and has a time complexity proportional to the Hamming distance itself rather than the binary length of the inputs.

Algorithm 2.3: Hamming Distance

Function $[d_{\mathrm{H}}]$ = `HammingDistance` (s_1, s_2)
Input: Sequences of bits s_1 and s_2.
Output: Hamming distance d_{H}.

```
1 d_H ← 0
2 s ← s_1 ⊕ s_2                    // Symbol ⊕ stands for exclusive or
3 while s ≠ 0 do
4 |   d_H ← d_H + 1
5 |   s ← s ∧ (s − 1)               // Symbol ∧ stands for logical and
6 end
7 return d_H
```

Hamming Distance Kernel

Couto [2005] provides another kernel for categorical data which is based on the Hamming distance. By defining a mapping of a string into a specially constructed feature space of categorical objects, the author is able to define a new measure of similarity between

categorical data.

Let $v = (v_1, \ldots, v_n) \in \bigotimes_{i=1}^{n} X_i = X$ be a categorical observation with n categorical components $v_1, \ldots, v_n$ taking values in n finite categorical spaces $X_1, \ldots, X_n$. For a coordinate $u \in X$, Couto defines a mapping

$$\phi_u : X \to \mathbb{R},\ v \mapsto \sigma^{d_{\mathrm{H}}(u,v)} = \prod_{i=1}^{n} \sigma^{\delta_{u_i, v_i}}, \tag{2.10}$$

where σ is in $(0,1)$ and where δ is the Kronecker delta. Therefore, given two strings s_1 and s_2 in X, we can define the output of the Hamming distance kernel as the summation of all possible products $\phi_u(s_1)\phi_u(s_2)$ with $u \in X$. This leads to the following definition of a Hamming distance kernel:

Definition 2.3 (Hamming Distance Kernel)**.** *Let X be the Cartesian product of n finite sets of symbols $X_1, \ldots, X_n$ of cardinality $m_1, \ldots, m_n$, and let $(s_1, s_2) \in X^2$. Then the* Hamming distance kernel k_σ *is defined as*

$$k_\sigma : X^2 \to \mathbb{R},\ (s_1, s_2) \mapsto \sum_{u \in X} \prod_{i=1}^{n} \sigma^{\delta_{u_i, (s_1)_i}} \sigma^{\delta_{u_i, (s_2)_i}},$$

where σ is in $(0,1)$ and where δ is the Kronecker delta.

Couto showed that this kernel can be computed recursively with Algorithm 2.4. For two strings s_1 and s_2 in X of length n, this algorithm requires $5(n+1)$ FLOPS to compute the output of the Hamming distance kernel.

Diffusion Kernel for Categorical Data

Kondor and Lafferty [2002] proposed a different mapping than Couto. They considered each categorical object $s \in X$ to be a vertex of a graph such that two vertices are connected by an edge only if their categorical objects s_1 and s_2 differ by the value of one component

Algorithm 2.4: Hamming Distance Kernel

Function $[k]$ = `HammingDistanceKernel` $(s_1, s_2, \sigma, \mathbf{m})$
Input: Strings s_1 and s_2, parameter σ and vector $\mathbf{m}$ with $m_i = |X_i|$ for $i = 1, \ldots, n$.
Output: Kernel output k.

$a \leftarrow \sigma^2$
$b \leftarrow 1 - \sigma$
$c \leftarrow b(1 + \sigma)$
$k \leftarrow 1$
for $i \in [\![1, n]\!]$ **do**
  $k \leftarrow \big(am_i - b\delta_{(s_1)_i,(s_2)_i} + c\big)k$
end
return k

(i.e., $d_{\mathrm{H}}(s_1, s_2) = 1$). Using a bandwidth parameter $\sigma > 0$, they adapted a diffusion kernel on graph for categorical data as follows:

Definition 2.4 (Diffusion Kernel for Categorical Data)**.** *Let X be the Cartesian product of n finite sets of symbols $X_1, \ldots, X_n$ of cardinality $m_1, \ldots, m_n$, let $(s_1, s_2) \in X^2$ and let $\sigma > 0$. Then the* diffusion kernel *for categorical data k_σ is defined as*

$$k_\sigma : X^2 \to \mathbb{R},\ (s_1, s_2) \mapsto \prod_{i=1}^{n} \left(\frac{1 - e^{-m_i\sigma}}{1 + (m_i - 1)e^{-m_i\sigma}} \right)^{\delta_{(s_1)_i,(s_2)_i}},$$

where δ is the Kronecker delta.

2.2.3 Kernels on Numerical Data

There is a wide variety of kernels on numerical data which is blossoming in various families of such functions, each one having desirable properties for certain kind of problems. Kernels can be chosen from a family or built specifically for one particular problem. In the later case, one should keep in mind that kernels represent a measure of distance and angle into some (possibly higher dimensional) Hilbert space. The considerations related to what constitutes the "closeness" of two distinct observations is left to the person modeling the classification problem and the nature of the numerical data.

Kernel on Real Numbers

By using the closure properties of kernels (see Sub-Section 2.2.1), it is possible to build kernels on real numbers in order to obtain kernels between vectors. By direct application of the second point of Theorem 2.2, the simplest kernel on real numbers is

$$k : \mathbb{R}^2 \to \mathbb{R},\ (x,y) \mapsto xy. \tag{2.11}$$

For example, the polynomial kernel (see Definition 2.5) is directly derived from Equation 2.11. But component products inside kernel between vectors (e.g., the polynomial kernel) can be replaced by more elaborated relations in order to form special kernels. If observations belong to $\mathbb{R}_+$ for example, then spline kernels can be directly obtained from polynomial or ANOVA kernels and the function

$$k : \mathbb{R}_+^2 \to \mathbb{R}_+,\ (x,y) \mapsto \mu\big([0,x] \cap [0,y]\big) = \min(x,y), \tag{2.12}$$

where μ is the Lebesgue measure on $\mathbb{R}$. Other constructions are using

$$k_\sigma : (\mathbb{R}_+^*)^2 \to \mathbb{R}_+^*,\ (x,y) \mapsto \mu\big([0,1/x^\sigma] \cap [0,1/y^\sigma]\big) = \frac{1}{\max(x,y)^\sigma}, \tag{2.13}$$

with $\sigma > 0$. Using Theorem 2.2, both Equations 2.12 and 2.13 can be combined to form

$$k : (\mathbb{R}_+^*)^2 \to \mathbb{R}_+^*,\ (x,y) \mapsto \frac{\min(x,y)}{\max(x,y)}. \tag{2.14}$$

Lastly, if there exists a $\sigma > 0$ such that $(x,y) \in [0,\sigma]^2$ then the following function is a positive definite kernel on $[0,\sigma]$:

$$k_\sigma : [0,\sigma]^2 \to [0,\sigma],\ (x,y) \mapsto \sigma - \max(x,y). \tag{2.15}$$

Kernels Based on Polynomials Kernels

The most basic kernels over $(R^n)^2$ are derived from positive definite bilinear forms such as the Euclidean dot product. They are the direct results of Theorem 2.2.

Definition 2.5 (Kernels from Bilinear Forms)**.** *If $\Sigma = \{\sigma_{ij}\}_{i,j=1}^{n}$ is a real symmetric positive definite $n \times n$-matrix, then the following function is a positive definite kernel:*

$$k_\sigma : (\mathbb{R}^n)^2 \to \mathbb{R},\ (\mathbf{x},\mathbf{y}) \mapsto \mathbf{x}^{\mathrm{t}}\Sigma\mathbf{y} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}x_i y_j.$$

In particular, if $\Sigma = \mathbf{I}_n$ then $k_\sigma(\mathbf{x},\mathbf{y}) = k(\mathbf{x},\mathbf{y}) = \mathbf{x}^{\mathrm{t}}\mathbf{y}$ (Euclidean dot-product).

Since any polynomial with positive coefficients of a kernel is also a kernel, we can easily derive the definition of a polynomial kernel from Definition 2.5 and the binomial theorem.

Definition 2.6 (Polynomial Kernel)**.** *Let $\Sigma = \{\sigma_{ij}\}_{i,j=1}^{n}$ be a real symmetric positive definite $n \times n$-matrix, let $\sigma_{n^2+1} \geqslant 0$ and let $\sigma_{n^2+2} \in \mathbb{N}^*$. Then the following function is called the* polynomial kernel*:*

$$k_\sigma : (\mathbb{R}^n)^2 \to \mathbb{R},\ (\mathbf{x},\mathbf{y}) \mapsto \left(\sigma_{n^2+1} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}x_i y_j\right)^{\sigma_{n^2+2}}.$$

Shawe-Taylor and Cristianini [2004] showed that the RKHS with the polynomial kernel as reproducing kernel is the space of all functions $\mathbf{x} \mapsto x_1^{\nu_1}x_2^{\nu_2}\ldots x_n^{\nu_n}$ such that $\sum_{i=1}^{n}\nu_i \leqslant \nu$, which is a space of dimension $\binom{n+\nu}{\nu}$. If the RKHS is further refined so that it becomes the space of all functions $\mathbf{x} \mapsto x_1^{\nu_1}x_2^{\nu_2}\ldots x_n^{\nu_n}$ such that $(\nu_1,\ldots,\nu_n) \in \{0,1\}^n$, then we obtain the all-subsets kernel.

Definition 2.7 (All-Subsets Kernel)**.** *If* $\{\sigma_i\}_{i=1}^n \in \mathbb{R}^n$*, then the following function is called the* all-subsets kernel*:*

$$k_\sigma : (\mathbb{R}^n)^2 \to \mathbb{R},\ (\mathbf{x},\mathbf{y}) \mapsto \prod_{i=1}^{n}(1+\sigma_i x_i y_i).$$

If the RKHS is the space of all functions $\mathbf{x} \mapsto x_1^{\nu_1} x_2^{\nu_2} \ldots x_n^{\nu_n}$ such that $(\nu_1,\ldots,\nu_n) \in \{0,1\}^n$ and $\sum_{i=1}^n \nu_i = \nu$ (which is a space of dimension $\binom{n}{\nu}$), then we obtain the ANOVA kernel of degree ν.

Definition 2.8 (ANOVA kernel of degree σ)**.** *Let* $\{\sigma_i\}_{i=1}^n \in \mathbb{R}^n$ *and let* $\sigma_{n+1} \in \mathbb{N}^*$*. Then the following function is called the* ANOVA kernel of degree σ_{n+1}*:*

$$k_\sigma : (\mathbb{R}^n)^2 \to \mathbb{R},\ (\mathbf{x},\mathbf{y}) \mapsto \sum_{1\leqslant i_1<\cdots<i_{\sigma_{n+1}}\leqslant n}\ \prod_{j=1}^{\sigma_{n+1}} \sigma_{i_j} x_{i_j} y_{i_j}.$$

Shawe-Taylor and Cristianini also showed that the ANOVA kernel can be computed recursively using a dynamic programming evaluation with $2\sigma_{n+1}\big(2n+1-\sigma_{n+1}\big)$ FLOPS (see Algorithm 2.5). A more general and flexible family of such kernels can be constructed from the so-called *graph kernel*.

Kernel Based on Radial Basis Functions

A Radial Basis Function (RBF) is a real-valued function on $\mathbb{R}^n$ of the form $\Phi_{\mathbf{c}} : \mathbb{R}^n \to \mathbb{R},\ \mathbf{x} \mapsto \phi\big(\|\mathbf{x}-\mathbf{c}\|_\Sigma\big)$ where ϕ is a real-valued function on $\mathbb{R}$, Σ is a real symmetric positive definite $n \times n$-matrix, and $\mathbf{c} \in \mathbb{R}^n$. Given $\mathbf{c} = \mathbf{0}$, if the function $k : (\mathbb{R}^n)^2 \to \mathbb{R},\ (\mathbf{x},\mathbf{y}) \mapsto \Phi_{\mathbf{0}}(\mathbf{x}+\varepsilon\mathbf{y})$ with $\varepsilon \in \{-1,1\}$ is positive definite, then this is a positive definite kernel. In the same fashion, we have that:

Algorithm 2.5: ANOVA Kernel

Function $[k]$ = `AnovaKernel` $(\mathbf{x}, \mathbf{y}, \sigma)$
Input: Vectors $\mathbf{x}$, $\mathbf{y}$ in $\mathbb{R}^n$, and vector $\sigma \in \mathbb{R}^{n+1}$.
Output: Kernel output k.

$\mathbf{M} \leftarrow \mathbf{0}_{(\sigma_{n+1}+1)\times(n+1)}$
for $i \in [\![1, n+1]\!]$ **do**
  $M_{1i} \leftarrow 1$
end
for $i \in [\![2, \sigma_{n+1}+1]\!]$ **do**
  $M_{i,i-1} \leftarrow 0$
  for $j \in [\![i, n+1]\!]$ **do**
    $M_{i,j} \leftarrow M_{i,j-1} + \sigma_j x_j y_j A_{i-1,j-1}$
  end
end
return $M_{\sigma_{n+1}+1,n+1}$

- If the function $k : (\mathbb{R}^n)^2 \to \mathbb{R},\ (\mathbf{x}, \mathbf{y}) \mapsto \Phi_{\mathbf{0}}(\mathbf{x} \cdot \mathbf{y})$ (where $\cdot$ is the Hadamard product) is positive definite, then this is a positive definite kernel.
- If the function $k : (\mathbb{R}^n)^2 \to \mathbb{R},\ (\mathbf{x}, \mathbf{y}) \mapsto \phi\big(d_k(\mathbf{x}, \mathbf{y})\big)$ (where $d_k : (\mathbb{R}^n)^2 \to \mathbb{R}_+$ is the norm induced by a positive definite kernel k) is positive definite, then this is a positive definite kernel.

Among all these combinations, only RBF of the form $k : (\mathbb{R}^n)^2 \to \mathbb{R},\ (\mathbf{x}, \mathbf{y}) \mapsto \Phi_{\mathbf{0}}(\mathbf{x} - \mathbf{y})$ are invariant by translation and rotation. This is the case of the widely used Gaussian RBF kernel.

Definition 2.9 (Gaussian RBF kernel)**.** *Let $\Sigma = \{\sigma_{ij}\}_{i,j=1}^n$ be a real symmetric positive definite $n \times n$-matrix and let $\sigma_{n^2+1} \in \mathbb{R}_+^*$. Then the following function is called the* Gaussian RBF kernel*:*

$$k_\sigma : (\mathbb{R}^n)^2 \to (0, 1],\ (\mathbf{x}, \mathbf{y}) \mapsto \exp\left(-\sigma_{n^2+1} \sum_{i=1}^n \sum_{j=1}^n \sigma_{ij}(x_i - y_i)(x_j - y_j)\right).$$

Despite being the most widely used kernel, Chen [2004] showed that the Gaussian RBF kernel is not robust to outliers. The associated map $\mathbf{x} \mapsto k_\sigma(\mathbf{x}, \cdot)$ has images into an infinite-dimensional Hilbert space [Burges, 1998]. The Gaussian RBF kernel can be directly derived by applying Theorem 2.2 to polynomial kernels.

2.2.4 Kernels for Real-Valued Physical Data

For problems that estimate unidentified nonlinear patterns in $\mathbb{R}^n$, it often becomes necessary for kernels to give the same output for the very same pair of real-valued vectors in $\mathbb{R}^n$ that has been translated and/or rotated because of external factors. In other words, nonlinear patterns in $\mathbb{R}^n$ must be estimated the same way even if their position or orientation shifted during the measurement process. This calls for a special family of positive definite kernels that do not have a privileged frame of reference. These kernels must be invariant by translation and rotation.

Distances and Angles in the RKHS

Given a symmetric positive definite kernel $k : (R^n)^2 \to \mathbb{R}$, we have that k is invariant by translation and rotation if, for a pair of vector $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$, the following identity holds for all vectors $\mathbf{t} \in \mathbb{R}^n$ and for all $n \times n$ real orthonormal matrices $\mathbf{Q}$:

$$k(\mathbf{Qx} + \mathbf{t}, \mathbf{Qy} + \mathbf{t}) = k(\mathbf{x}, \mathbf{y}). \tag{2.16}$$

A translation-invariant kernel has additional properties. For instance, if $k(\mathbf{x}+\mathbf{t}, \mathbf{y}+\mathbf{t}) = k(\mathbf{x}, \mathbf{y})$, then $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x}-\mathbf{y}, \mathbf{0})$ and $k(\mathbf{x}, \mathbf{x}) = k(\mathbf{0}, \mathbf{0}) \geqslant 0$, since k is positive definite. This gives a new interpretation of distances and angles in the RKHS $\mathscr{F}$ for which k is the reproducing kernel. The distance between the images of $\mathbf{x}$ and $\mathbf{y}$ by the mapping $\mathbf{x} \mapsto k(\cdot, \mathbf{x})$

(see Corollary 2.1) is such that

$$\|k(\cdot,\mathbf{x}) - k(\cdot,\mathbf{y})\|^2 = k(\mathbf{x},\mathbf{x}) + k(\mathbf{y},\mathbf{y}) - 2k(\mathbf{x},\mathbf{y}). \tag{2.17}$$

If k is translation-invariant then $\|k(\cdot,\mathbf{x}) - k(\cdot,\mathbf{y})\|^2 = 2\big(k(\mathbf{0},\mathbf{0}) - k(\mathbf{x},\mathbf{y})\big)$. Since the above quantity is always positive, we have that $k(\mathbf{x},\mathbf{y}) \leqslant k(\mathbf{0},\mathbf{0})$ for all $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$. Actually, the measure of angles in $\mathscr{F}$ leads to better conclusions regarding the bounds of the quantity $k(\mathbf{x},\mathbf{y})$. The Cauchy-Schwarz inequality is expressed by

$$|k(\mathbf{x},\mathbf{y})| \leqslant \|k(\cdot,\mathbf{x})\|\,\|k(\cdot,\mathbf{y})\| = \sqrt{k(\mathbf{x},\mathbf{x})k(\mathbf{y},\mathbf{y})} = k(\mathbf{0},\mathbf{0}), \tag{2.18}$$

for which we deduce that $k(\mathbf{0},\mathbf{0}) \neq 0$ except for the trivial case $k \equiv 0$. It follows that all measures of angles in the RKHS $\mathscr{F}$ have their values in the interval $[-k(\mathbf{0},\mathbf{0}), k(\mathbf{0},\mathbf{0})]$ and therefore that

$$\|k(\cdot,\mathbf{x}) - k(\cdot,\mathbf{y})\| \leqslant 2\sqrt{k(\mathbf{0},\mathbf{0})}, \tag{2.19}$$

for all $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$. In other words, whatever the distance between two vectors in $\mathbb{R}^n$ is, the distance between their images in the RKHS $\mathscr{F}$ for which the reproducing kernel is translation-invariant is always bounded by the quantity $2\sqrt{k(\mathbf{0},\mathbf{0})}$.

Radial Basis Functions

Consider a continuous RBF $\Phi : \mathbb{R}^n \to \mathbb{R},\ \mathbf{x} \mapsto \phi(\|\mathbf{x}\|_\Sigma)$ where $\|\cdot\|_\Sigma$ is the norm induced by the symmetric positive definite $n \times n$ matrix Σ on $\mathbb{R}^n$. The function $k : (\mathbb{R}^n)^2 \to \mathbb{R},\ (\mathbf{x},\mathbf{y}) \mapsto \Phi(\mathbf{x}-\mathbf{y})$ is real-valued, continuous and symmetric. If k is positive definite, i.e. if

$$\phi(0)\sum_{i=1}^{m}\lambda_i^2 + 2\sum_{i<j}^{m}\phi(x_{ij})\lambda_i\lambda_j \geqslant 0, \tag{2.20}$$

for any $m \in \mathbb{N}$, $\lambda_i \in \mathbb{R}$, $x_{ij} \in \mathbb{R}_+$ and $(i,j) \in [\![1,m]\!] \times [\![1,m]\!]$, then there exists a unique RKHS with k as the reproducing kernel. We can deduce from Equation 2.20 a condition for which k is *not* positive definite. For example, let $m = 2$, $\lambda_1 = -1$ and $\lambda_2 = 1$, the function k is not positive definite if there exists a $x \in \mathbb{R}_+$ such that $\phi(0) < \phi(x)$. If $\lambda_1 = \lambda_2 = 1$, then k is not positive definite if there exists a $x \in \mathbb{R}_+$ such that $\phi(0) < -\phi(x)$. Hence, we conclude that k is not positive definite if there exists a $x \in \mathbb{R}_+$ such that $\phi(0) < |\phi(x)|$. Another trivial deduction is that k is not positive definite if $\phi(0) < 0$ ($m = 1$ and $\lambda_1 = 1$). Consequently, ϕ must be such that $\phi(0) \geqslant 0$ and $|\phi(x)| \leqslant \phi(0)$ for any $x \in \mathbb{R}_+$.

Summary of Properties

A positive definite kernel $k : (R^n)^2 \to \mathbb{R}$ that is suitable for real-valued vectors in physics must be invariant by translation and rotation, and therefore must verify at least the following properties:

- $k(\mathbf{x},\mathbf{x}) = k(\mathbf{0},\mathbf{0}) \geqslant 0$ for any $\mathbf{x} \in \mathbb{R}^n$.
- $k(\mathbf{x},\mathbf{y}) \in [-k(\mathbf{0},\mathbf{0}), k(\mathbf{0},\mathbf{0})]$ for any $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$.
- $\|k(\cdot,\mathbf{x}) - k(\cdot,\mathbf{y})\| = \sqrt{2\big(k(\mathbf{0},\mathbf{0}) - k(\mathbf{x},\mathbf{y})\big)} \leqslant 2\sqrt{k(\mathbf{0},\mathbf{0})}$ for any $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$.

If the kernel k is derived from a radial basis function ϕ taking the Euclidean distance of two vectors as argument, then the positive definite property of k implies that:

- $\phi(0) \geqslant 0$.
- $\phi(x) \in [-\phi(0), \phi(0)]$ for any $x \in \mathbb{R}_+$.
- $\forall (a,b) \in \mathbb{R}^2$ and $\forall x \in \mathbb{R}_+$, $(a^2+b^2)\phi(0) + 2ab\phi(x) \geqslant 0$.

These properties rules out many radial basis functions such as the multi-quadric function $\phi : \mathbb{R} \to \mathbb{R}_+$, $x \mapsto \sqrt{x^2+c^2}$ where $c \neq 0$, and the thin-plate spline function $\phi : \mathbb{R}_+^* \to \mathbb{R}$, $x \mapsto x^2 \log x$.

Examples of Suitable Kernels

The output of a kernel k can be considered as a measure of likeliness between the two input observations. In $\mathbb{R}^n$, the greater the Euclidean distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ is and the lesser their similarity is. Hence, if $k : (\mathbb{R}^n)^2 \to \mathbb{R},\ (\mathbf{x},\mathbf{y}) \mapsto \phi(\|\mathbf{x}-\mathbf{y}\|_\Sigma)$, then $\phi : \mathbb{R}_+ \to \mathbb{R}$ should be a continuous and decreasing function such that $\phi(\mathbb{R}_+) \subseteq [0,\sigma_1]$ with $\sigma_1 > 0$. Additionally, the rate at which ϕ decreases can be controlled with a parameter $\sigma_2 > 0$ (see Figure 2.4).

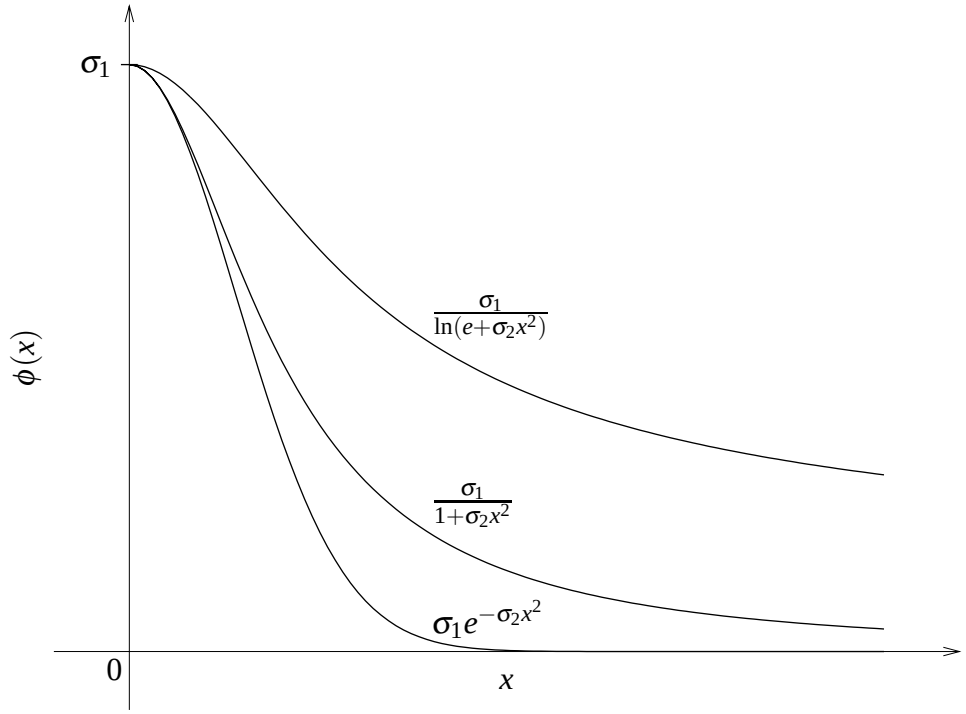


Figure 2.4: Examples of suitable radial basis functions.

2.3 Estimating Functions with Kernels

As mentioned in Sub-Section 2.1.1, the aim of supervised learning is to find an *bounded* pattern function that links an observation x in a measurable space X to a target $\mathbf{y}$ in $\mathbb{R}^p$ such that

$$|f(x) - y_i| \leqslant \varepsilon, \tag{2.21}$$

where $i \in [\![1, m]\!]$ and $\varepsilon > 0$ is an arbitrary value. In this study, the function f is modeled by kernels with the help of the closure properties mentioned in Theorem 2.2. Equation 2.21 represents the need to arbitrarily constrain the deviations of the outputs of f from the targets associated with specific observations. This constraint can be *locally* satisfied by attempting to minimize an empirical error measure over a set of collected observations $\mathscr{X} = \{x_1, \ldots, x_\ell\} \subset X$. However, we also need to guarantee that the pattern function f will also behave as in Equation 2.21 *globally*. To do so, it is necessary to introduce *generalization errors* and the means to minimize them (see Sub-Section 2.3.4). The minimization of the generalization error depends both on the representation of the pattern function f (see Sub-Section 2.3.2) and the notion of Rademacher complexity introduced in Sub-Section 2.3.3.

2.3.1 Type of Supervised Learning Problems

There exist two main categories of problems in supervised learning: classification problems and regression problems. The observation space X remains the same in both categories, but the target space is different. In *binary* classification problems, targets arbitrarily belong to the set $\{-1, +1\} \subset \mathbb{R}$ without loss of generality. If a set of data encoded targets with other symbols, say 'a' and 'b' for example, then it is always possible to construct a trivial map from this two-element set into $\{-1, +1\}$. Multi-class classification problems are characterized by targets that belong to a finite set of integers between 1 and m with m being the number of classes. Like for the binary classification case, if targets are encoded with different symbols than integers from 1 to m, it is possible to construct a map from that set of symbols into $[\![1, m]\!]$. Targets for regression problems belong either to $\mathbb{R}$ if it is a single-output regression problem, or to $\mathbb{R}^m$ if it is a multiple-output regression problem. In all type of problems, the objective is to find a pattern function f linking observations and targets, whatever the target set may look like.

The division in classification and regression problems may seem arbitrary since a clas-

sification problem *is* a particular kind of regression problem for which the pattern functions take discrete values. Hence the common use of logistic regression methods to solve problems of classification. However, different structural approaches may be considered in classification problems that cannot correspond to similar approaches employed in regression problems. One of these approaches is used in the case of multi-class classification problems (see Sub-Section 2.4.2). Classification problems have also been historically considered separated from regression problems [Vapnik, 1982], with regression problems being a generalization of binary classification problems.

2.3.2 Representation of Pattern Functions with Kernels

Besides the nomenclature of the target set and the corresponding type of supervised learning problems, the choice of the family of pattern functions is the most crucial aspect of the supervised learning algorithms. In the case of ULMs (and SVMs), the pattern functions are built from kernels by using the closure properties of Theorem 2.2. The use of kernels to represent pattern functions is motivated by their geometrical interpretations which can be made in classification and regression problems (see Corollary 2.1) and the theoretical error bounds (see Sub-Sections 2.3.3 and 2.3.4) that are involved with them.

The geometrical interpretations of kernel methods used in classification problems relate to the separation of clusters of observations mapped in the induced Hilbert space by hyperplanes. Linear separation in the feature space has therefore a direct correspondence with the separation of cluster of observations in X by nonlinear manifolds. Similarly, observations mapped in the feature space are being fitted by an hyperplane in the case of nonlinear regression problems.

For reasons which are explained in detail in Sub-Section 3.1.1, the pattern functions are chosen to belong to a family of functions $\mathscr{G}$ which is defined by

$$\mathscr{G} = \left\{ x \in X \mapsto \sum_{i=1}^{\ell} \alpha_i k_\sigma(x, x_i) + b \in \mathbb{R} : \alpha^{\mathrm{t}} \mathbf{K} \alpha \leqslant B^2 \right\}, \tag{2.22}$$

where k_σ is a finitely positive semi-definite kernel, and $B > 0$ is an arbitrary scalar which bounds the norm of the elements of $\mathscr{G}$. Naturally, we may build upon this choice of function space and generalize the type of possible pattern functions a bit further. For example, we may define the function spaces $\mathscr{G}_1, \ldots, \mathscr{G}_m$ induced the choice of m distinct kernels $k_1, \ldots, k_m$ similar to Equation 2.22 and define a pattern function as an element of the space

$$\mathscr{G} = \left\{ \sum_{i=1}^{m} a_i f_i + \beta_i : f_i \in \mathscr{G}_i,\ i \in [\![1, m]\!] \right\}. \tag{2.23}$$

The point behind the formulation of Equation 2.23 is to define a pattern function with multiple kernels instead of a single one. This allows the use of different similarity measures between observations to be taken into account and yield more fitting patterns. ULMs can directly support the use of multiple kernels at once, provided that the vectors $\mathbf{a}$ and β in $\mathbb{R}^m$ are given beforehand. The tuning of theses vectors can however be made using a non-convex pattern search (see Sub-Section 2.4.3) but it might result in a computationally intensive approach. To circumvent this timing problem, one can refer to the method of iterative construction of the pattern function estimate that is detailed in Sub-Section 2.1.3.

2.3.3 Rademacher Complexity

The Rademacher complexity of a class of function $\mathscr{G}$, such as the classes formulated in Sub-Section 2.3.2, measures the capacity, with respect to a probability distribution, of the functions of $\mathscr{G}$ to fit random data [Bartlett and Mendelson, 2001, 2002]. Many useful risk measures associated with kernel methods are relying on the expression of the empirical value of the Rademacher complexity. These risk measures are probabilistic measures that represent the capability of a pattern function to return outputs which are arbitrarily close to their targets (known or unknown), and this for any possible observation in X regardless that if it was generated or not. This kind of risk measure, which is different than the empirical error, is known as *generalization error*.

Vapnik and Chervonenkis [1971] were the firsts to investigate the link between the generalization error of $\{0,1\}$-indicator functions f in $\mathscr{G}$ (such as the ones used in binary classification) and the empirical Rademacher complexity of this class. Namely, they showed that, whatever the observation $x \in X$ might be, the error between the output of the pattern function $f(x)$ and the actual target $y \in \{0,1\}$ is upper-bounded, with a certain probability, by a function of the *Vapnik-Chervonenkis dimension* of $\mathscr{G}$ which is the cardinality of the largest set of points that can be *shattered*[1] by the functions in $\mathscr{G}$. It was later shown that the empirical Rademacher complexity of $\mathscr{G}$ is instead bounded by this function of the Vapnik-Chervonenkis dimension and that the upper bound of Vapnik and Chervonenkis is indeed pessimistic. The Rademacher complexity of a class of function then goes further by extending the expression of upper bounds of the generalization errors for any type of pattern functions, which also means for any type of supervised (or semi-supervised) learning problems (and not just binary classification problems).

The Rademacher complexity of a class of functions $\mathscr{G}$ over the measurable space X is defined below:

Definition 2.10 (Empirical Rademacher complexity)**.** *Let $\mathscr{G}$ be a class of real-valued functions over a measurable space X and let $\mathscr{X} = \{x_1, \ldots, x_\ell\} \subset X$ be a set of sample observations in X. The* empirical Rademacher complexity *of $\mathscr{G}$ is defined as*

$$\hat{R}(\mathscr{G}) = \left\langle \frac{2}{\ell} \sup_{f \in \mathscr{G}} \left| \sum_{i=1}^{\ell} \sigma_i f(X_i) \right| \middle| X_i = x_i,\ i \in [\![1, \ell]\!] \right\rangle,$$

where the σ_i's, with $i \in [\![1,\ell]\!]$, are independent uniform $\{\pm 1\}$-valued random variables. The Rademacher complexity *of $\mathscr{G}$ is the expected value of $\hat{R}(\mathscr{G})$ taken over an identically and independently distributed sample $\{X_1, \ldots, X_\ell\}$ of random variables.*

Empirical Rademacher complexities have closure properties than can be useful in the

[1] A parameterized binary classification model is said to shatter a set of observations if, for all assignments of target labels to those observations, there exists a parameter such that the model makes no classification errors.

establishment of upper bounds on generalization errors. These properties are defined in the following theorem which was derived Bartlett and Mendelson [2001, 2002]:

Theorem 2.3. *Let $\mathscr{F}, \mathscr{F}_1, \ldots, \mathscr{F}_m$ be classes of real-valued functions over a measurable space X. Let $\mathscr{X} = \{x_1, \ldots, x_\ell\} \subset X$ be a set of sample observations in X. We have:*

- *If $\mathscr{F}_1 \subseteq \mathscr{F}_2$ then $\hat{R}(\mathscr{F}_1) \leqslant \hat{R}(\mathscr{F}_2)$.*
- $\hat{R}\left(\sum_{i=1}^{m} \mathscr{F}_i\right) \leqslant \sum_{i=1}^{m} \hat{R}(\mathscr{F}_i)$.
- *For every λ in $\mathbb{R}$, $\hat{R}(\lambda\mathscr{F}) = |\lambda|\,\hat{R}(\mathscr{F})$.*
- *If $f : \mathbb{R} \to \mathbb{R}$ is Lipschitz with constant C and if the condition $f(0) = 0$ is satisfied, then $\hat{R}\big(f(\mathscr{F})\big) \leqslant 2C\hat{R}(\mathscr{F})$.*
- *For any function $f : X \to \mathbb{R}$, we have*

$$\hat{R}(\mathscr{F} + f) \leqslant \hat{R}(\mathscr{F}) + \frac{2}{\ell}\sqrt{\sum_{i=1}^{\ell} f^2(x_i)}.$$

- *Let $p \in \mathbb{N}^*$, $g \in \mathscr{F}$ and $\mathscr{L}_{\mathscr{F},g,p} = \{|f - g|^p : f \in \mathscr{F}\}$. If $\|f - g\|_\infty \leqslant 1$ for all $f \in \mathscr{F}$, then*

$$\hat{R}(\mathscr{L}_{\mathscr{F},g,p}) \leqslant 2p\left(\hat{R}(\mathscr{F}) + \frac{2}{\ell}\sqrt{\sum_{i=1}^{\ell} g^2(x_i)}\right).$$

The last point of Theorem 2.3 is a key inequality for the establishment of a probabilistic upper bound on the generalization error of ULMs.

2.3.4 Risk Measures

A risk measure for a given pattern function f in a function class $\mathscr{G}$ (as defined in Sub-Section 2.3.2) and a finite set of observations $\mathscr{X} = \{x_1, \ldots, x_\ell\} \subset X$ is a quantity that

evaluates that discrepancy between the outputs of f for an x in the measurable space X and its associated target. A straightforward risk measure for $f \in \mathscr{G}$ given observations $x_1, \ldots, x_\ell$ that belongs to the set $\mathscr{X}$ is the *empirical error* which is defined by

$$\hat{E}(f) = \frac{1}{\ell} \sum_{i=1}^{\ell} d\big(f(x_i), y_i\big), \tag{2.24}$$

where d is a distance between $f(x)$ and y in $\mathbb{R}$. Typically, d is based on a p-norm and we have

$$\hat{E}_p(f) = \frac{1}{\ell} \sum_{i=1}^{\ell} |f(x_i) - y_i|^p. \tag{2.25}$$

If $p = 2$, then $\hat{E}_p(f)$ is the *mean square deviation*. Nevertheless, finding $f \in \mathscr{G}$ such that $\hat{E}_p(f)$ is minimized gives no guarantee that

$$|f(x) - y| \leqslant \varepsilon, \tag{2.26}$$

for any $(x,y) \in X \times \mathbb{R}$, where $\varepsilon > 0$ is an arbitrary scalar. However, there exists an expression of a probabilistic bound for quantities expressed in Equation 2.26. If we consider the notions introduced in Sub-Section 2.1.1, there exist a measure space (Ω, Σ, P) and a measurable function $g : \Omega \to X \times \mathbb{R},\ \omega \mapsto (x,y)$ that generates independently and identically distributed observations. If $z : \Omega \to X$ is the measurable function that associates $\omega \in \Omega$ to $x \in X$, and if $v : \Omega \to \mathbb{R}$ is the measurable function that associates $\omega \in \Omega$ to $y \in \mathbb{R}$, then Bartlett and Mendelson [2001, 2002] and Koltchinskii and Panchenko [2000] showed that there exists a scalar $\delta \in (0,1)$ such that

$$P\left(\left\langle \left| \frac{f(z) - v}{C} \right|^p \right\rangle \leqslant \frac{\hat{E}_p(f)}{C^p} + \hat{R}(\mathscr{L}_{\mathscr{G}/C, y/C, p}) + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}} \right) \geqslant 1 - \delta, \tag{2.27}$$

where:

- The quantity C is equal to $2\rho(B + |b|)$ where $\rho > 0$ is the radius of the smallest ball

centered at the origin in X which contains the observations in $\mathscr{X} = \{x_1, \ldots, x_\ell\}$ i.e. $\mathscr{X} \subset B_\rho(0) = \{x \in X : d_X(x,0) \leqslant \rho\}$;

- The function f is in the function class

$$\mathscr{G} = \left\{ x \in X \mapsto \sum_{i=1}^{\ell} \alpha_i k_\sigma(x, x_i) + b \in \mathbb{R} : \alpha^{\mathrm{t}} \mathbf{K} \alpha \leqslant B^2 \right\}, \tag{2.28}$$

  hence $\|f\|_\infty \leqslant B + |b|$;

- The function class $\mathscr{L}_{\mathscr{G}/C,y/C,p}$ is equal to $\left\{ \left| \frac{f-y}{C} \right|^p : f \in \mathscr{G} \right\}$ and $\|(f-y)/C\|_\infty \leqslant 1$ for all $f \in \mathscr{G}$.

The quantity $\langle |f(z) - v|^p \rangle \geqslant 0$ is the *generalization error* and Equation 2.27 shows that it is upper-bounded by the empirical error plus an expression of the Rademacher complexity of $\mathscr{G}$ denoted by $h(B,b)$. The term $h(B,b)$ is such that

$$h(B,b) = \big(2\rho(B+|b|)\big)^p \left(\hat{R}(\mathscr{L}_{\mathscr{G}/C,y/C,p}) + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}} \right). \tag{2.29}$$

Theorem 2.3 can be used to establish the relation

$$\hat{R}(\mathscr{L}_{\mathscr{G}/C,y/C,p}) \leqslant \frac{p}{\rho(B+|b|)} \left(\hat{R}(\mathscr{G} - b) + \frac{2|b|}{\sqrt{\ell}} + \frac{2\|\mathbf{y}\|_2}{\ell} \right). \tag{2.30}$$

The following result, showed notably by Shawe-Taylor and Cristianini [2004], allows the numerical computation of an upper bound of the Rademacher complexity $\hat{R}(\mathscr{G} - b)$. We have

$$\hat{R}(\mathscr{G} - b) \leqslant \frac{2B}{\ell} \sqrt{\mathrm{tr}(\mathbf{K})}, \tag{2.31}$$

and hence,

$$\hat{R}(\mathscr{L}_{\mathscr{G}/C,y/C,p}) \leqslant \frac{2p}{\ell\rho(B+|b|)} \Big(B\sqrt{\mathrm{tr}(\mathbf{K})} + |b|\sqrt{\ell} + \|\mathbf{y}\|_2 \Big). \tag{2.32}$$

If $p = 1$, then we have

$$h(B,b) \leqslant B\frac{4\sqrt{\mathrm{tr}(\mathbf{K})}}{\ell} + 6(B+|b|)\rho\sqrt{\frac{\ln(2/\delta)}{2\ell}} + \frac{4|b|}{\sqrt{\ell}} + \frac{4\|\mathbf{y}\|_2}{\ell}. \tag{2.33}$$

If the data and the kernel matrix $\mathbf{K}$ were normalized, then

$$h(B,b) \leqslant \frac{4}{\sqrt{\ell}}\left(1 + \left(1 + \frac{3}{2\sqrt{2}}\sqrt{\ln\left(\frac{2}{\delta}\right)}\right)(B+|b|)\right). \tag{2.34}$$

Therefore, we have, with a probability of at least 95%, that

$$\langle |f(z) - v| \rangle \leqslant \frac{1}{\ell}\sum_{i=1}^{\ell} |f(x_i) - y_i| + \frac{4}{\sqrt{\ell}}\big(1 + 3(B+|b|)\big). \tag{2.35}$$

It follows that, given a function $f \in \mathscr{G}$, the minimization of the generalization error is linked to the minimization of the empirical error of f for a given observation set $\mathscr{X}$ and an upper bound of the infinity norm of f. This is an important point that will serve into the construction of ULMs (see Sub-Section 3.1.1). It is also useful to notice that the larger the radius ρ in Equation 2.33 is and the larger the upper bound on the generalization error becomes. This shows that if an observation $x \in X$ is outside $B_\rho(0)$ then the pattern function is unlikely to properly fit the datum. Consequently, ULMs will guarantee the upper bound of the generalization error only for observations contained in $B_\rho(0)$.

2.4 Specialized Models

2.4.1 Multiple Output Regression Models

Cases may arise where each observation x in X is associated with a target vector $\mathbf{y} \in \mathbb{R}^p$ rather than a single real value $y \in \mathbb{R}$. The pattern function f that links observations to targets is therefore a function over X with values in $\mathbb{R}^p$. Most supervised learning algorithms that search for patterns (including ULMs) are designed to accept only single-value targets and

cannot be inherently modified for multiple output regression models. Nevertheless, the alternative to non-multiple output regression techniques is to embed several single-value regression models into a coherent structure that returns multiple output values.

The simplest way to construct such a multiple output regression model is to represent the outputs of the pattern function f as a vector of outputs of p single-valued pattern functions $f_1, \ldots, f_p$ over X. In other words the function f is defined by

$$f : X \to \mathbb{R}^p,\ x \mapsto \begin{pmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_p(x) \end{pmatrix}. \tag{2.36}$$

Each individual pattern function f_i, with $i \in [\![1, p]\!]$, is then computed separately with the help of a supervised learning algorithm. As usual, this computation is made using a set of ℓ observations $\mathscr{X} = \{x_1, \ldots, x_\ell\}$ for which their associated single value targets are the i-th components of the vectors $\mathbf{y}_1, \ldots, \mathbf{y}_\ell$. The drawback of this approach is that the pattern computation is repeated p times without the possibility of reducing the size of each individual sub-problem.

2.4.2 Multi-Class Classification Models

Vapnik [1995] initially proposed a simple *ad hoc* multi-class classification scheme, the so-called one-against-all scheme, in which several binary classifiers are combined to form a unique multi-class classifier. Despite that this scheme was developed for Support Vector Machines, the approach is general enough to allow the scheme to be used with any type of binary classifier. Later, Platt et al. [2000] generalized the approach with a decision-tree-based scheme which overcomes some shortcomings of Vapnik's scheme. The same year, Allwein et al. [2000] presented an error-correcting code scheme that forms a compromise between performances of the one-against-all scheme and the decision-tree-based scheme.

In this section, a binary classifier is represented by a pattern function $f : X \to \{-1, +1\}$ which is itself computed by learning algorithms such as ULMs. Each observation x in the measurable space X has a target y that can take p distinct values that we arbitrarily choose to be in the set $[\![1, p]\!]$ without loss of generality. The following multi-class classification schemes are valid for any type of binary classifier regardless of the learning algorithm that generated it. They are all based on the establishment of classification structures that organize different binary classifiers together. However "structure-less" multi-class classification schemes based on logistic regression are also possible. These alternate schemes are briefly mentioned in Equation 3.24.

One-Against-All Scheme

A simple multi-class classification scheme for p classes was proposed by Vapnik [1995] when a set of p binary classifiers is available. The i-th binary classifier is associated to a real-valued "confidence" function g_i derived from the pattern function f_i that is positive if an observation $x \in X$ belongs to the i-th class, and negative if it does not. The larger the output value of g_i is, the more reliable the output value of f_i is. In this scheme, a total of p binary classifiers are computed using learning algorithms, with each one of these classifiers determining if a given observation belongs to a certain class or all the other classes; hence the name *one-against-all*. Suppose that p confidence functions $g_1, \ldots, g_p$ are obtained, then the function f that determines the one class among p for which a given observation $x \in X$ belongs to, is defined by

$$f : X \to [\![1, p]\!], \; x \mapsto \arg\max\{g_i(x) : i \in [\![1, p]\!]\}. \tag{2.37}$$

The output of the function f is the number of the class of the observation x. The determination of the class of an observation requires p evaluations of confidence functions which can be done quickly if each individual confidence function is not too complicated.

Despite the fast determination of the class of an observation, the one-against-all scheme has a couple of very inconvenient drawbacks. The first disadvantage is the relevance of comparing p confidence functions that have extremely close output values in most cases. The second disadvantage is that each one of the p confidence functions is computed from ℓ observations, which can be a massive computational bottleneck if ℓ is very large since the evaluation of a binary classifier is repeated p times.

Decision-Tree-Based Scheme

Platt et al. [2000] introduced a decision-tree-based architecture for binary classifiers that returns Boolean values. This architecture is a *Decision-Directed Acyclic Graph* (DDAG) that is, as the name suggests, a structure based on a directed graph with no cycles. Suppose that, for a p-class classification problem, a total of $p(p-1)/2$ pair-wise binary classifiers are computed i.e. we obtained pattern functions f_{ij} discriminating between class $i \in [\![1, p]\!]$ and class $j \in [\![1, p]\!]$ with $i < j$.

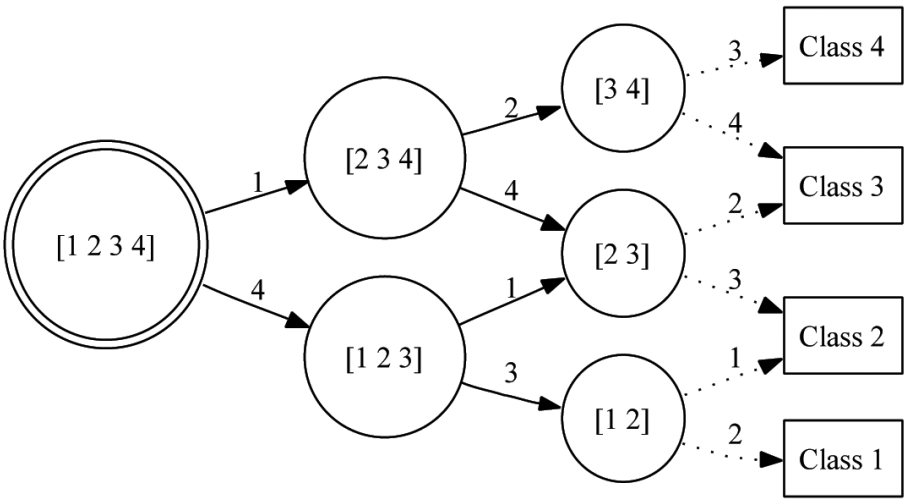


Figure 2.5: Decision-directed acyclic graph for a 4-class problem. Each edge is labeled with the class number that has been removed from the list associated with the source node. The root node (on the left), which contains the full list of all possible classes, is connected to the leaf nodes (on the right), which are associated with lists of only two classes.

The DDAG classifier has a pyramidal structure in which every node but the root node

has two direct successors, and every node but the leaf nodes has two direct predecessors. A path from the root node to one of the leafs is computed by updating an ordered list of classes to be tested, each node successively removing one element from that list. The removing process works by picking up the pattern function f_{ij} discriminating the class at the top of the list, denoted by i, from the one at the bottom, denoted by j, and testing if the observation $x \in X$ belongs to class i or class j. If, for example, x belongs to class i, then the next node on the path would be the node which list does not contain j and reciprocally; hence the reason for two direct successors at each node. The path starts at the root node with initial list $\{1,\ldots,p\}$ and goes to one leaf node with a list containing only two classes. Once the leaf node has been reached, the last binary classification is performed and the last class is removed from the list, the remaining one being the class the observation x belongs to (see figure 2.5).

Determining the class of of an observation x requires $p-1$ decision evaluations and each pattern function f_{ij} is trained on $\ell_i + \ell_j$ observations, ℓ_i being the number of training observations belonging to class i and ℓ_j being the number of training observations belonging to class j. The number of pattern functions to compute is one order of magnitude greater than for the one-against-all scheme, but the dimension of each binary classification sub-problem is much smaller than the original data set size. The DDAG scheme also overcomes problems related to confidence functions having output values too similar.

Error-Correcting Code Scheme

Suppose m ($1 \leqslant m \leqslant p$) Boolean pattern functions $f_1,\ldots,f_m$ are given for an p-class problem. A function f_i, $i \in [\![1,m]\!]$, will return either $+1$ or -1 depending on which class the input belongs to. If the function f_i was not designed to discriminate one particular class against another then, by convention the output will be zero for that specific class. An example is given in table 2.1 where all outputs are arranged into a decision matrix.

Let $\mathbf{D}$ be an $p \times m$ decision matrix built from $f_1,\ldots,f_m$. We can notice that each row of

Table 2.1: Example of error-correcting codes for a 4-class problem and five pattern functions. The table lists the values the pattern functions return for elements belonging to the classes listed in the rows.

Class	f_1	f_2	f_3	f_4	f_5
1	0	-1	-1	-1	$+1$
2	$+1$	-1	0	$+1$	$+1$
3	$+1$	-1	-1	-1	0
4	-1	$+1$	$+1$	0	-1

D represents a code that we seek to be unique for each class. If it is the case, then, for an observation $x \in X$, the outputs for every f_i, $i \in [\![1,m]\!]$, are put in order so it forms a code for that specific observation. The code is then stored in an $m \times 1$ vector $\mathbf{c}(x)$ and then compared to each row of the decision matrix **D** by computing a distance between $\mathbf{c}(x)$ and the codes on the rows of **D**. The class of the observation x will be the one with the minimum distance between $\mathbf{c}(x)$ and the code on the row corresponding to that class. The distance proposed by Allwein et al. [2000] between x and the i-th class is

$$d\big(\mathbf{c}(x), \mathbf{D}_{i\cdot}\big) = \frac{1}{2}\left(m - \sum_{j=1}^{m} \operatorname{sign}\big((\mathbf{c}(x))_j D_{ij}\big)\right), \tag{2.38}$$

and the final pattern function is given by

$$f : X \to [\![1,p]\!]\,,\ x \mapsto \arg\min\Big\{d\big(\mathbf{c}(x), \mathbf{D}_{i\cdot}\big) : i \in [\![1,p]\!]\Big\}. \tag{2.39}$$

The error-correcting code scheme is, in a way, a generalization of the one-against-all scheme and the decision-tree-based scheme for which a potentially lesser number of decision functions is needed in order to determine the class an observation belongs to. Furthermore, depending on the codes contained in the decision matrix **D**, the size of the programming problems to be solved can be significantly reduced. The contents of **D** also suggest that there exist optimal codes that combine an high discriminating power and a short number of pattern functions trained on smaller problems.

2.4.3 Model Validation

Selecting a suitable kernel is one of the few tasks needed to be performed before an ULM can search for patterns in the provided set of data. The very act of selecting kernels and correct algorithm parameters is nothing else than the building of a mathematical model of the given data. Naturally, a chosen model must be tested and validated to ensures that it properly predicts non-trivial hidden patterns within, of course, a certain margin of error.

While selecting an appropriate model is often based on considerations that are problem-specific, the validation of prediction models is a procedure that is now standardized. These validation procedures rely on repeated tests on the very same set of observations $\mathscr{X} = \{x_1, \ldots, x_\ell\}$ in X upon which the pattern functions were generated by the learning algorithms. These procedures can be divided into two families:

- Cross-validation procedures which are partition-based.
- Bootstrapping procedures which are sampling-based.

Both families have their pros and cons which are often related to the size of the data sets. Furthermore, repeated selection-validation procedures can be embedded into a generalized pattern search approach which can search for the optimal choice of parameters for a mathematical model. This pattern search can be time consuming if the chosen learning algorithms are not efficient at handling large sets of data.

Cross-Validation Procedure

In a cross-validation, the observation set $\mathscr{X}$ is *partitioned* into q subsets $\{\mathscr{X}_1, \ldots, \mathscr{X}_q\}$ which are alternatively used to build a classifier to be tested on the remaining subsets. In other words, the set $\mathscr{X}$ is iteratively divided into an actual *training* set, for which a classifier is built, and a *testing* set, for which the classifier is tested to confirm the validity of the mathematical model. At each iteration of the cross-validation, an error statistic is

constructed for the results of the testing set. The error statistics are then merged after all training sets have been used which evaluate the validity of the chosen mathematical model. The cross-validation procedure is outlined in Algorithm 2.6.

Algorithm 2.6: Cross-Validation Procedure

Function [μ] = `CrossValidation` ($M, \mathscr{P}(\mathscr{X}), \mathscr{P}(\mathbf{y})$)
Input: model M, partition of the observation set $\mathscr{P}(\mathscr{X}) = \{\mathscr{X}_1, \ldots, \mathscr{X}_q\}$ and partition of the target vector set $\mathscr{P}(\mathbf{y}) = \{\mathbf{y}_1, \ldots, \mathbf{y}_q\}$.
Output: error statistic μ.

```
μ ← 0
for i ∈ ⟦1, q⟧ do
    Compute a pattern function f using the model M and the set 𝒳 \ 𝒳_i.
    Update μ by comparing f(𝒳_i) and y_i.
end
return μ
```

For classification problems, the error statistic is often based on a *confusion matrix* $\mathbf{C}$. Given an p-class problem, the matrix $\mathbf{C}$ is an $p \times p$ matrix for which each i-th row represents the number of observations that belong to class i and each j-th column represents the number of observations that were predicted to belong to class j. A common error statistic based on the confusion matrix is

$$\mu = \min \left\{ \frac{C_{ii}}{\mathbf{1}_p^{\mathrm{t}} \mathbf{C}_{i\cdot}} : i \in [\![1, p]\!] \right\}, \tag{2.40}$$

with $\mathbf{C}_{i\cdot}$ being the i-th row of the matrix $\mathbf{C}$. In other words this error statistic is the minimum fraction of correctly classified samples.

For regression problems, errors are based on many different statistics such as:

- Mean square deviations between functional outputs and targets,
- Rank correlation coefficients between functional outputs and targets (e.g., Spearman's rank correlation coefficient),

- Statistics computed the empirical distribution of the residuals (e.g., from normality tests, etc.).

The partition of $\mathscr{X}$ can be chosen randomly but some samples may never be used in a training or testing set. Additionally, a poorly chosen partition can fail to yield correct patterns, hence introducing a bias during the testing of the mathematical model. Furthermore, it should be noticed that the computed pattern functions $f_1, \ldots, f_q$ can be all significantly different if the partition is ill-chosen. In such case, the cross-validation procedure becomes totally irrelevant since each individual pattern function should be similar to the one that is obtained by training on the whole observation set.

Improved cross-validation schemes have been designed over the years to curb the partitioning problems. The most commonly used schemes are the *q-fold cross-validation* and its particular case, the *Leave-One-Out* (LOO) cross-validation. In a q-fold cross-validation, or more exactly in a *stratified* q-fold cross-validation, the set $\mathscr{X}$ is partitioned into q subsets of equal size that contain the same proportion of class labels as in the whole observation set $\mathscr{X}$. During each iteration of the method, $q-1$ subsets are used for training and the remaining subset is used for testing. Every subset is used for testing only once, hence a maximum of q iterations.

The LOO cross-validation is an extreme case of the q-fold cross-validation in which only a single sample is taken out at each iteration to be in the testing set. This last approach usually needs modifications in order to be efficiently implemented for large data sets, but it is often the sole validation approach that can be used with very small sets of data.

Bootstrapping Procedures

Bootstrapping is a re-sampling technique for inferring sample statistics (such as error statistics) by drawing randomly, *with replacement*, several observations from $\mathscr{X}$ and testing them, multiple times, with a pattern function initially computed from $\mathscr{X}$. The repeated tests allow the computation an estimate of the distribution of the error statistic which is the

main advantage of this validation procedure. The knowledge of such a distribution is in fact a key factor for comparing the different choice of kernels and model parameters.

Algorithm 2.7 outlines the bootstrapping procedure for a given mathematical model and an observation set $\mathscr{X} = \{x_1, \ldots, x_\ell\}$ of ℓ observations. The sampling and testing cycles are repeated q times until there is a sufficient number of error statistics to allow the construction of a well-shaped empirical distribution of errors.

Algorithm 2.7: Bootstrapping Procedure

Function $[\mu]$ = `Bootstrapping` $(q, M, \mathscr{X}, \mathbf{y})$
Input: number of rounds $q \geqslant 1$, model M, observation set $\mathscr{X} = \{x_1, \ldots, x_\ell\}$ and associated target vector $\mathbf{y} \in \mathbb{R}^\ell$.
Output: error statistic μ.

$\mu \leftarrow 0$
Compute a pattern function f using the model M and the set $\mathscr{X}$.
for $i \in [\![1, q]\!]$ **do**
    Sample ℓ_i observations with replacement from the set $\mathscr{X}$ (observations are stored in $\mathscr{X}_i$ and their targets in $\mathbf{y}_i$).
    Update μ by comparing $f(\mathscr{X}_i)$ and $\mathbf{y}_i$.
end
return μ

Bootstrapping is often computationally intensive but it has none of the disadvantages associated to cross-validation procedures. Furthermore, an empirical distribution of the error statistic is immediately available. This allows the construction of a confidence interval on the error statistic and the statistical comparison of the performances of several mathematical model.

Pattern Search for Optimal Model Parameters

If a finite number of kernels and model parameters are available for a given problem, then cross-validation or bootstrapping schemes can be looped inside a pattern search method to find the most suitable kernel and the best parameters. The objective is to minimize the measure of error that is returned returned by the validation procedures over the search

domain.

The main disadvantage of such an approach is that the resulting optimization problem is not necessarily a convex optimization problem. Hence deterministic pattern searches for convex problems are no longer guaranteed to successfully reach an optimal solution. Therefore, meta-heuristics such as genetic algorithms, simulated annealing, tabu search or ant colony optimization might be more suitable to minimize error measures.

Chapter 3

Unconstrained Learning Machines

3.1 Mathematical Programming Problem

Consider a source of data which provides a finite collection of $\ell \in \mathbb{N}^*$ observations $x_1, \ldots, x_\ell$ that belong to a measurable space X of any kind and that are uniquely associated with real-valued targets $y_1, \ldots, y_\ell$. All observation-target pairs are distributed according to an *unknown* distribution. The function $f : X \to \mathbb{R}$ that links any observation x to a target y is assumed to be unknown, non-trivial and continuous. The objective of this chapter is to determine an estimate of the function f using kernels and the collection of observation-target pairs. This problem is called in all the following the *function estimation problem* and it is formulated as a mathematical programming problem. Within the context of the function estimation problem, Unconstrained Learning Machines (ULMs) are a family of learning algorithms which use kernel methods and optimization techniques to estimate functions from a given set of observation-target pairs. The part related to the training (or learning phase) of ULMs is discussed in this chapter. The following explains how the function estimation problem is formulated as an unconstrained Quadratic Programming problem (hence the choice for the name of ULMs) and efficiently solved using methods from linear algebra.

3.1.1 Objective Function

Let k_σ be a continuous real-valued symmetric positive definite kernel parameterized by σ i.e. $k_\sigma : X \times X \to \mathbb{R}$ is a continuous function such that, for any pair $(x, y) \in X^2$, we have

that $k_\sigma(x,y) = k_\sigma(y,x)$ and such that

$$\sum_{i=1}^{m}\sum_{j=1}^{m} k_\sigma(x_i,x_j)\lambda_i\lambda_j \geqslant 0, \tag{3.1}$$

for any $m \in \mathbb{N}$, $\lambda_i \in \mathbb{R}$, $x_i \in X$ and $i \in [\![1,m]\!]$. The Moore-Aronszajn theorem [Aronszajn, 1950] states that there exists a unique Hilbert space $\mathscr{F}$ of real-valued functions on X such that $k_\sigma(\cdot,x) \in \mathscr{F}$ for any $x \in X$ and such that

$$\langle g, k_\sigma(\cdot,x)\rangle_{\mathscr{F}} = g(x), \tag{3.2}$$

for all $g \in \mathscr{F}$ (this is called the *reproducing property*) with $\langle\cdot,\cdot\rangle_{\mathscr{F}}$ being the dot product in $\mathscr{F}$. The kernel k_σ is said to be the *reproducing kernel* of $\mathscr{F}$ and the space $\mathscr{F}$ is called a Reproducing Kernel Hilbert Space (RKHS).

Suppose that the unknown function f that has to be estimated belongs to a RKHS $\mathscr{F}$ with k_σ as the reproducing kernel. From the reproducing property of the RKHS $\mathscr{F}$, we have for all given observations $x_1,\ldots,x_\ell$ in X

$$\langle f, k_\sigma(\cdot,x_i)\rangle_{\mathscr{F}} = f(x). \tag{3.3}$$

Let $\mathscr{S}$ be the linear space embedded in $\mathscr{F}$ spanned by the $k_\sigma(\cdot,x_i)$-images of the observations x_i, $i \in [\![1,\ell]\!]$. Since f is assumed to belong to $\mathscr{F}$, we can rewrite the function f as

$$f = \sum_{i=1}^{\ell} \alpha_i k_\sigma(\cdot,x_i) + f^\perp, \tag{3.4}$$

where the function $f^{\perp} \in \mathscr{F}$ is orthogonal to $\mathscr{S}$ and $\alpha_i \in \mathbb{R}$ for all $i \in [\![1, \ell]\!]$. Then

$$\begin{aligned} f(x_i) &= \langle k_\sigma(\cdot, x_i), f \rangle_{\mathscr{F}} \\ &= \left\langle k_\sigma(\cdot, x_i), \sum_{j=1}^{\ell} \alpha_j k_\sigma(\cdot, x_j) + f^{\perp} \right\rangle_{\mathscr{F}} \\ &= \left\langle k_\sigma(\cdot, x_i), \sum_{j=1}^{\ell} \alpha_j k_\sigma(\cdot, x_j) \right\rangle_{\mathscr{F}} \\ &= \sum_{j=1}^{\ell} \alpha_j \left\langle k_\sigma(\cdot, x_i), k_\sigma(\cdot, x_j) \right\rangle_{\mathscr{F}} \end{aligned} \tag{3.5}$$

implies that

$$f(x_i) = \sum_{j=1}^{\ell} \alpha_j k_\sigma(x_i, x_j) = \sum_{j=1}^{\ell} \alpha_j \mathbf{K}_{ij} = \mathbf{K}_{i.}\alpha, \tag{3.6}$$

where $\mathbf{K}_{ij}$ is the (i, j)-th element of the $\ell \times \ell$ Gram matrix $\mathbf{K}$ made of all the dot products $\left\langle k_\sigma(\cdot, x_i), k_\sigma(\cdot, x_j) \right\rangle_{\mathscr{F}} = k(x_i, x_j)$ for every i and j in $[\![1, \ell]\!]$. The Gram matrix $\mathbf{K}$ is, by the properties of k_σ, a positive semi-definite matrix which is called the *kernel matrix*. The equality

$$f(x_i) = \mathbf{K}_{i.}\alpha, \tag{3.7}$$

for all $i \in [\![1, \ell]\!]$ is only valid if the function f actually belongs to the RKHS $\mathscr{F}$ with k_σ as the reproducing kernel. Equation 3.7 does not hold in the general case, however, if the kernel k_σ is carefully chosen, then there exist a scalar $b \in \mathbb{R}$ such that the approximation

$$f(x_i) \approx \mathbf{K}_{i.}\alpha + b, \tag{3.8}$$

is acceptable for any continuous function f. There are a couple of reasons for introducing the scalar b in Equation 3.8. The first reason is to provide a correcting term since the choice of k_σ may not be completely suitable for estimating the function f. The second reason is to make the correction linear since the introduction of a nonlinear term, while possible, would imply that the correcting term is not a kernel-based function (since the sum of kernels is a

kernel) and hence void this analysis based on kernel methods. Consequently, we will search in all the following an estimate of the function f in a subspace $\mathscr{G}$ of bounded functions of the translated space $\mathscr{F}+b$ after a suitable reproducing kernel k_σ is chosen. The reason for choosing a set of bounded functions is motivated by a common requirement of all physical problems: the outputs of all physical systems are never infinite and hence the norm of the function that estimates the outputs of a physical system must be finite i.e. bounded. The set of functions $\mathscr{G}$ is therefore defined by

$$\mathscr{G} = \left\{ x \in X \mapsto \sum_{i=1}^{\ell} \alpha_i k_\sigma(x, x_i) + b \in \mathbb{R} : \alpha^{\mathrm{t}} \mathbf{K} \alpha \leqslant B^2 \right\}, \tag{3.9}$$

with $B \in \mathbb{R}^*$. Bartlett and Mendelson [2001, 2002] and Koltchinskii and Panchenko [2000] showed that, given a probability $p \in (0,1)$, the generalization error of $\hat{f} \in \mathscr{G}$ is bounded with probability p by the sum of an expression of the empirical Rademacher complexity $\hat{R}(\mathscr{G})$ of the function class $\mathscr{G}$ and the empirical error of $\hat{f}$. The smaller the quantity $\hat{R}(\mathscr{G})$ is, the smaller the generalization error becomes. This complexity is bounded by elements that defines the class $\mathscr{G}$ according to the following formula

$$\hat{R}(\mathscr{G}) \leqslant \frac{4}{\ell} \left(B \sqrt{\mathrm{tr}(\mathbf{K})} + |b| \sqrt{\ell} + \|\mathbf{y}\|_2 \right). \tag{3.10}$$

Consequently, minimizing both the upper bound of $\hat{R}(\mathscr{G})$ and the empirical error of $\hat{f}$ is a way to obtain a function $\hat{f}$ in $\mathscr{G}$ that estimates f and that satisfies Approximation 3.8 for any given observation $x \in X$. The aim of the function estimation problem is therefore to minimize the quantities $|b|$ and B, as well as the empirical error of $\hat{f}$.

The minimization of $|b|$ can be replaced by the minimization of the quantity

$$\delta_b b^2 + c_b b, \tag{3.11}$$

which provides the objective function of the mathematical minimization problem with a

quadratic formulation rather than a nonlinear one or additional constraints. This formulation is weighted with coefficients $\delta_b > 0$ and $c_b \in \mathbb{R}$ in order to contrast this quantity with the other terms of the objective function. The coefficient c_b can be negative and b can still be minimized since the coefficient δ_b is always greater than zero.

The empirical error of $\hat{f}$ is a function of the quantities $\hat{f}(x_i) - y_i = \xi_i$ which are slack variables. Like for the minimization of $|b|$, the minimization of the empirical error can easily be replaced by the minimization of the quantity

$$\xi^{\mathrm{t}} \Delta_\xi \xi + \mathbf{c}_\xi^{\mathrm{t}} \xi, \tag{3.12}$$

where Δ_ξ is a positive definite matrix and where $\mathbf{c}_\xi \in \mathbb{R}^\ell$. Since the norm of ξ induced by Δ_ξ on $\mathbb{R}^\ell$ is equivalent to the Euclidean norm of ξ on $\mathbb{R}^\ell$, the minimization of the quantity $\xi^{\mathrm{t}}\xi$ is similar to the minimization of $\xi^{\mathrm{t}}\Delta_\xi\xi$ with the exception that the later brings a supplementary degree of control over the ξ_i's. The contributions of each ξ_i to the objective function can now be tuned individually.

From Equation 3.9, the minimization of B can be achieved by minimizing $\alpha^{\mathrm{t}}\mathbf{K}\alpha$ for any $\alpha \in \mathbb{R}^\ell$. By using the Cauchy-Schwarz inequality, we obtain

$$\alpha^{\mathrm{t}}\mathbf{K}\alpha = \langle \alpha, \mathbf{K}\alpha \rangle \leqslant \|\alpha\|_2 \, \|\mathbf{K}\alpha\|_2 \,. \tag{3.13}$$

However, since the Euclidean norm $\|\cdot\|_2$, like all vector norms induced on the space of $\ell \times \ell$ matrices, is a consistent norm, we can immediately derive that

$$\alpha^{\mathrm{t}}\mathbf{K}\alpha \leqslant \|\mathbf{K}\|_2 \, \|\alpha\|_2^2, \tag{3.14}$$

where $\|\mathbf{K}\|_2$ is the spectral norm of $\mathbf{K}$. Moreover, given a positive definite matrix Δ_α, the Euclidean norm is equivalent to the norm induced by Δ_α on $\mathbb{R}^\ell$. In other words, there exists

a constant $C_\alpha > 0$ such that

$$\|\alpha\|_2 \leqslant C_\alpha \sqrt{\langle \alpha, \Delta_\alpha \alpha \rangle}. \tag{3.15}$$

Consequently, the minimization of B can be replaced by the minimization of

$$\alpha^{\mathrm{t}} \Delta_\alpha \alpha + \mathbf{c}_\alpha^{\mathrm{t}} \alpha, \tag{3.16}$$

where $\mathbf{c}_\alpha \in \mathbb{R}^\ell$.

Equations 3.11, 3.12 and 3.16 provide the expression of the quadratic objective function of the mathematical programming problem that solves the function estimation problem. This objective function is

$$(\alpha, \xi, b) \in \mathbb{R}^{2\ell+1} \mapsto \alpha^{\mathrm{t}} \Delta_\alpha \alpha + \xi^{\mathrm{t}} \Delta_\xi \xi + \delta_b b^2 + \mathbf{c}_\alpha^{\mathrm{t}} \alpha + \mathbf{c}_\xi^{\mathrm{t}} \xi + c_b b \in \mathbb{R}. \tag{3.17}$$

3.1.2 Constraints

Given a function $\hat{f} \in \mathscr{G}$ (the function class $\mathscr{G}$ is defined in Sub-Section 3.1.1), the constraints of the mathematical programming problem consist into matching the outputs of $\hat{f}$ for every x_i with their corresponding targets y_i. In other words, we must have

$$\hat{f}(x_i) = \mathbf{K}_{i\cdot} \alpha + b \approx y_i, \tag{3.18}$$

for all $i \in [\![1, \ell]\!]$. However, the quantity $\hat{f}(x_i) - y_i$ was defined in Sub-Section 3.1.1 as the slack variable ξ_i, for which the quantity $\xi^{\mathrm{t}} \Delta_\xi \xi + \mathbf{c}_\xi^{\mathrm{t}} \xi$ has to be minimized. Consequently, Approximation 3.18 can be reformulated into an equality which is

$$\mathbf{K}_{i\cdot} \alpha + b - y_i = \xi_i, \tag{3.19}$$

for all $i \in [\![1, \ell]\!]$. This enforces that the quantities ξ_i in the equation

$$\hat{f}(x_i) - \xi_i = y_i \tag{3.20}$$

must be minimized for all $i \in [\![1, \ell]\!]$, and hence that Approximation 3.18 is matched as closely as possible. Equation 3.20 illustrates that slack variables act as an error-tolerant term in the formulation of the constraints. Non-negligible slack variables bridge the gap between what can be fitted with f and the desired target outputs, and hence can account for the influence observational outliers. This is a primitive form of adaptable robustness.

The set of ℓ equalities in Equation 3.19 is the entire set of constraints for the mathematical programming problem solving the function estimation problem since there are no other binding constraints on the variables α, ξ and b due to the quadratic formulation of the objective function in Equation 3.17.

3.1.3 Unconstrained Quadratic Programming Problem

The objective function in Equation 3.17 and the set of constraints in Equation 3.19 are all that is needed to write the complete formulation of the mathematical programming problem that solves the function estimation problem. This problem is

$$\begin{aligned} \min \quad & \alpha^{\mathrm{t}} \Delta_\alpha \alpha + \xi^{\mathrm{t}} \Delta_\xi \xi + \delta_b b^2 + \mathbf{c}_\alpha^{\mathrm{t}} \alpha + \mathbf{c}_\xi^{\mathrm{t}} \xi + c_b b, \\ \text{with} \quad & \begin{cases} \mathbf{K}\alpha + b\mathbf{1} - \xi = \mathbf{y}, \\ \alpha \in \mathbb{R}^\ell,\ \xi \in \mathbb{R}^\ell,\ b \in \mathbb{R}. \end{cases} \end{aligned} \tag{3.21}$$

This problem is an unconstrained Quadratic Programming problem with equality constraints which is the source of the name for the Unconstrained Learning Machines [Gilbert and Trafalis, 2009].

The optimal solution (α^*, b^*) is then used to formulate $\hat{f} \in \mathscr{G}$ which is the estimate of the unknown continuous function f that forms the pattern between the observations $x_i \in X$

and their corresponding targets $y_i \in \mathbb{R}$, for all i in $[\![1,\ell]\!]$. This function $\hat{f}$ is given by

$$\hat{f} : X \to \mathbb{R},\ x \mapsto \sum_{i=1}^{\ell} \alpha_i^* k_\sigma(x, x_i) + b^*. \tag{3.22}$$

This estimate can be further improved depending of the specificity attached to a particular function estimation problem. For example, if some coefficients α_i^* are null or negligible, then a subset $I \subseteq [\![1,\ell]\!]$ of integers can be stored so that the α_i's for all $i \in I$ are not negligible. This also gives the possibility to reduce the number of kernel evaluations and summations when computing the outputs of the function $\hat{f}$ and, additionally, it represents a form of *data thinning* (see Sub-Section 2.1.3) since it sorts out all observations that are unnecessary for estimating the pattern f between observations and targets. The set $\{x_i\}_{i\in I}$ is called the set of *support vectors* due to the analogy with SVMs.

Moreover, the function $\hat{f}$ can be combined with step functions to form complex logical rules and patterns. For example, if we are given $l > 1$ disjoint semi-open intervals $A_1 = [a_1, b_1), \ldots, A_l = [a_l, b_l)$ of $\mathbb{R}$ such that

$$\bigcup_{i=1}^{l} A_i = \mathbb{R}, \tag{3.23}$$

and a set of l distinct coefficients β_i, $i \in [\![1,\ell]\!]$, then the step function

$$\hat{g} : X \to \mathbb{R},\ x \mapsto \sum_{i=1}^{l} \beta_i \chi_{A_i}\big(\hat{f}(x)\big), \tag{3.24}$$

where χ_A is the indicator function of the interval $A \subset \mathbb{R}$, and represents a discrete nonlinear classification rule that maps the outputs of $\hat{f}$ into a discrete set $\{\beta_i\}_{i=1}^{l}$.

3.1.4 Optimal Solution

The programming problem in Equation 3.21 is a Quadratic Programming problem with equality constraints of the form

$$\min\left\{\frac{1}{2}\mathbf{x}^{\mathrm{t}}\mathbf{H}\mathbf{x}+\mathbf{c}^{\mathrm{t}}\mathbf{x} : \mathbf{A}\mathbf{x}=\mathbf{b},\ \mathbf{x}\in\mathbb{R}^n\right\}. \tag{3.25}$$

The $n\times n$-matrix $\mathbf{H}$ is real, symmetric and positive definite, the vector $\mathbf{c}$ is in $\mathbb{R}^n$, the vector $\mathbf{b}$ is in $\mathbb{R}^m$ and the matrix $\mathbf{A}$ is a real $m\times n$-matrix. Bazaraa et al. [2006] showed that the Lagrangian dual problem of (3.25) is

$$\max\left\{\inf\left\{\frac{1}{2}\mathbf{x}^{\mathrm{t}}\mathbf{H}\mathbf{x}+\mathbf{c}^{\mathrm{t}}\mathbf{x}+\mathbf{v}^{\mathrm{t}}(\mathbf{A}\mathbf{x}-\mathbf{b}) : \mathbf{x}\in\mathbb{R}^n\right\} : \mathbf{v}\in\mathbb{R}^m\right\}. \tag{3.26}$$

Given a vector $\mathbf{v}\in\mathbb{R}^m$, the function $\mathbf{x}\mapsto\frac{1}{2}\mathbf{x}^{\mathrm{t}}\mathbf{H}\mathbf{x}+\mathbf{c}^{\mathrm{t}}\mathbf{x}+\mathbf{v}^{\mathrm{t}}(\mathbf{A}\mathbf{x}-\mathbf{b})$ is convex and therefore a necessary and sufficient condition for a minimum is

$$\mathbf{H}\mathbf{x}+\mathbf{A}^{\mathrm{t}}\mathbf{v}+\mathbf{c}=\mathbf{0}, \tag{3.27}$$

i.e its first derivative vanishes. From equation (3.27) we derive $\mathbf{c}^{\mathrm{t}}\mathbf{x}+\mathbf{v}^{\mathrm{t}}\mathbf{A}\mathbf{x}=-\mathbf{x}^{\mathrm{t}}\mathbf{H}\mathbf{x}$ which, once substituted in equation (3.26), leads to the following problem

$$\min\left\{\frac{1}{2}\mathbf{x}^{\mathrm{t}}\mathbf{H}\mathbf{x}+\mathbf{b}^{\mathrm{t}}\mathbf{v} : \mathbf{H}\mathbf{x}+\mathbf{A}^{\mathrm{t}}\mathbf{v}=-\mathbf{c},\ \mathbf{x}\in\mathbb{R}^n,\ \mathbf{v}\in\mathbb{R}^m\right\}. \tag{3.28}$$

The matrix $\mathbf{H}$ is invertible, hence we can derive from equation (3.27) an identity that unambiguously links the primal variable $\mathbf{x}$ to its dual variable $\mathbf{v}$. This identity is

$$\mathbf{x}=-\mathbf{H}^{-1}(\mathbf{c}+\mathbf{A}^{\mathrm{t}}\mathbf{v}). \tag{3.29}$$

Replacing $\mathbf{x}$ in equation (3.28) by its expression in equation (3.29), we then obtain

$$\min\left\{\frac{1}{2}\mathbf{v}^{\mathrm{t}}\mathbf{Q}\mathbf{v}+\mathbf{d}^{\mathrm{t}}\mathbf{v}:\mathbf{v}\in\mathbb{R}^{m}\right\}, \tag{3.30}$$

where $\mathbf{Q}=\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^{\mathrm{t}}$ and $\mathbf{d}=\mathbf{A}\mathbf{H}^{-1}\mathbf{c}+\mathbf{b}$. Problem (3.30) is an unconstrained quadratic programming problem for which $\mathbf{Q}$ is a real, symmetric and positive definite $m\times m$-matrix. Therefore problem (3.30) has a unique optimal solution $\mathbf{v}^{*}$ given by $\mathbf{v}^{*}=-\mathbf{Q}^{-1}\mathbf{d}$ which is the value of the dual variable such that the first derivative of the objective function of (3.30) vanishes. Using equation (3.29), we can derive an analytical expression of the solution of the quadratic programming problem (3.25) which is

$$\mathbf{x}=\mathbf{H}^{-1}\big(\mathbf{A}^{\mathrm{t}}(\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^{\mathrm{t}})^{-1}(\mathbf{A}\mathbf{H}^{-1}\mathbf{c}+\mathbf{b})-\mathbf{c}\big). \tag{3.31}$$

3.2 Implementation

3.2.1 Formulation of the Linear System

Consider the notations of Sub-Section 3.1.3 and let $\mathbf{x}^{\mathrm{t}}=(\alpha^{\mathrm{t}},\xi^{\mathrm{t}},b)$. For the function estimation problem, the Hessian matrix of the objective function is a $(2\ell+1)\times(2\ell+1)$ block diagonal matrix with diagonal elements $(2\Delta_{\alpha},2\Delta_{\xi},2\delta_{b})$. The linear term of the objective function is defined by $\mathbf{c}^{\mathrm{t}}=(\mathbf{c}_{\alpha}^{\mathrm{t}},\mathbf{c}_{\xi}^{\mathrm{t}},c_{b})$. Constraints are of the form $\mathbf{A}\mathbf{x}=\mathbf{y}$ with $\mathbf{A}=(\mathbf{K},-\mathbf{I}_{\ell},\mathbf{1}_{\ell})$.

Proposition 3.1. *The optimal parameters $\alpha\in\mathbb{R}^{\ell}$ and $b\in\mathbb{R}$ of the estimated function $f:\mathbb{R}^{n}\to\mathbb{R},\ \mathbf{x}\mapsto\sum_{i=1}^{\ell}\alpha_{i}k(\mathbf{x}_{i},\mathbf{x})+b$ and the slack vector $\xi\in\mathbb{R}^{\ell}$ are given by $\alpha=\Delta_{\alpha}^{-1}(\mathbf{K}\mathbf{a}-\mathbf{c}_{\alpha}/2)$, $b=(\mathbf{1}_{\ell}^{\mathrm{t}}\mathbf{a}-c_{b}/2)/\delta_{b}$, and $\xi=-\Delta_{\xi}^{-1}(\mathbf{a}+\mathbf{c}_{\xi}/2)$ where $\mathbf{a}\in\mathbb{R}^{\ell}$ is the solution of the linear system $(\mathbf{K}\Delta_{\alpha}^{-1}\mathbf{K}+\Delta_{\xi}^{-1}+\mathbf{1}_{\ell}\mathbf{1}_{\ell}^{\mathrm{t}}/\delta_{b})\mathbf{a}=\mathbf{u}$ with $\mathbf{u}=\mathbf{y}+\big(\mathbf{K}\Delta_{\alpha}^{-1}\mathbf{c}_{\alpha}-\Delta_{\xi}^{-1}\mathbf{c}_{\xi}+(c_{b}/\delta_{b})\mathbf{1}_{\ell}\big)/2$.*

Proof. Using the notations of Sub-Section 3.1.3, the inverse of the Hessian matrix $\mathbf{H}$ is a diagonal matrix with diagonal elements $\big(\Delta_\alpha^{-1}/2, \Delta_\xi^{-1}/2, 1/(2\delta_b)\big)$. Equation (3.31) gives the optimal solution of the problem which is $\mathbf{x} = \mathbf{H}^{-1}\big(\mathbf{A}^{\mathrm{t}}(\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^{\mathrm{t}})^{-1}(\mathbf{A}\mathbf{H}^{-1}\mathbf{c} + \mathbf{y}) - \mathbf{c}\big)$. Since $\mathbf{H}$ is symmetric we have $\mathbf{H}^{-1}\mathbf{A}^{\mathrm{t}} = (\mathbf{A}\mathbf{H}^{-1})^{\mathrm{t}}$, and $\mathbf{A}\mathbf{H}^{-1} = (\mathbf{K}\Delta_\alpha^{-1}, -\Delta_\xi^{-1}, \mathbf{1}_\ell/\delta_b)/2$. Hence the term $\mathbf{A}\mathbf{H}^{-1}\mathbf{c} + \mathbf{y}$ is equal to $\big(\mathbf{K}\Delta_\alpha^{-1}\mathbf{c}_\alpha - \Delta_\xi^{-1}\mathbf{c}_\xi + (c_b/\delta_b)\mathbf{1}_\ell\big)/2 + \mathbf{y} = \mathbf{u}$. Since $\mathbf{K}$ is a real, symmetric positive semi-definite matrix and $\mathbf{H}$ is a real, symmetric and positive definite matrix, the matrix $\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^{\mathrm{t}} = (\mathbf{K}\Delta_\alpha^{-1}\mathbf{K} + \Delta_\xi^{-1} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b)/2$ is a real, symmetric and positive definite matrix and is therefore invertible. Consequently, if $\mathbf{a} \in \mathbb{R}^\ell$ is the solution of the linear system $(\mathbf{K}\Delta_\alpha^{-1}\mathbf{K} + \Delta_\xi^{-1} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b)\mathbf{a} = \mathbf{u}$, then $\mathbf{x} = \mathbf{H}^{-1}\big(\mathbf{A}^{\mathrm{t}}(2\mathbf{a}) - \mathbf{c}\big) = 2\mathbf{H}^{-1}(\mathbf{A}^{\mathrm{t}}\mathbf{a} - \mathbf{c}/2)$. Since $\mathbf{x}^{\mathrm{t}} = (\alpha^{\mathrm{t}}, \xi^{\mathrm{t}}, b)$, it follows that $\alpha = \Delta_\alpha^{-1}(\mathbf{K}\mathbf{a} - \mathbf{c}_\alpha/2)$, $\xi = -\Delta_\xi^{-1}(\mathbf{a} + \mathbf{c}_\xi/2)$ and $b = (\mathbf{1}_\ell^{\mathrm{t}}\mathbf{a} - c_b/2)/\delta_b$ at optimality. □

3.2.2 Solving the Linear System

Condition Number of the System Matrix

Proposition 3.1 shows that the optimal parameters of the estimated function are the solutions of a linear system. However there are some concerns regarding the stability of the solution of this linear system due to the condition number κ of the matrix $\mathbf{K}\Delta_\alpha^{-1}\mathbf{K} + \Delta_\xi^{-1} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b$. The quantity $\log_d \kappa$ is an estimate of how many base-d digits are lost when solving the linear system $(\mathbf{K}\Delta_\alpha^{-1}\mathbf{K} + \Delta_\xi^{-1} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b)\mathbf{a} = \mathbf{u}$. Therefore if an upper bound on κ is numerically large then the estimated function f would be unreliable.

Theorem 3.1. *Let $\lambda_{\max}$ and $\lambda_{\min}$ be the maximum and the minimum eigenvalues of the kernel matrix $\mathbf{K}$ respectively. Let $\delta_\alpha^{\max}$ and $\delta_\alpha^{\min}$ be the minimum and the maximum elements of the diagonal matrix Δ_α respectively, and let $\delta_\xi^{\max}$ and $\delta_\xi^{\min}$ be the minimum and the maximum elements of the diagonal matrix Δ_ξ respectively. Let η be a scalar defined by $\eta = \min\left\{\ell \left\|\mathbf{K}\right\|_{\max}, \sqrt{\ell}\left\|\mathbf{K}\right\|_1, \sqrt{\left\|\mathbf{K}\right\|_1 \left\|\mathbf{K}\right\|_\infty}, \left\|\mathbf{K}\right\|_{\mathrm{F}}\right\}$. Then the condition number κ_0 of the matrix $\mathbf{K}\Delta_\alpha^{-1}\mathbf{K} + \Delta_\xi^{-1} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b$ is such that:*

$$1 \leqslant \kappa_0 \leqslant \frac{\lambda_{\max}^2/\delta_\alpha^{\min} + 1/\delta_\xi^{\min} + \ell/\delta_b}{\lambda_{\min}^2/\delta_\alpha^{\max} + 1/\delta_\xi^{\max}} \leqslant \delta_\xi^{\max}(\eta^2/\delta_\alpha^{\min} + 1/\delta_\xi^{\min} + \ell/\delta_b).$$

Proof. We have $\left\|\mathbf{K}\Delta_\alpha^{-1}\mathbf{K} + \Delta_\xi^{-1} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b\right\|_2 \leqslant \left\|\mathbf{K}\right\|_2^2 \left\|\Delta_\alpha^{-1}\right\|_2 + \left\|\Delta_\xi^{-1}\right\|_2 + \left\|\mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}\right\|_2/\delta_b = \lambda_{\max}^2/\delta_\alpha^{\min} + 1/\delta_\xi^{\min} + \left\|\mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}\right\|_2/\delta_b$. The matrix $\mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}$ is of rank 1 therefore it has two eigenvalues, one of them being 0 which has multiplicity $\ell - 1$. We also have $\mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}\mathbf{1}_\ell = \ell\mathbf{1}_\ell$, thus ℓ is the other eigenvalue and $\left\|\mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}\right\|_2 = \ell$. The matrix $\mathbf{K}$ is semi-positive definite and the matrices Δ_α and Δ_ξ are positive definite, hence a lower bound of the smallest eigenvalue of the matrix $\mathbf{K}\Delta_\alpha^{-1}\mathbf{K} + \Delta_\xi^{-1} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b$ is $\lambda_{\min}^2/\delta_\alpha^{\max} + 1/\delta_\xi^{\max} > 0$. Consequently $\left\|(\mathbf{B} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b)^{-1}\right\|_2 \leqslant 1/(\lambda_{\min}^2/\delta_\alpha^{\max} + 1/\delta_\xi^{\max})$ with $\mathbf{B} = \mathbf{K}\Delta_\alpha^{-1}\mathbf{K} + \Delta_\xi^{-1}$. The condition number of $\mathbf{B} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b$ is defined by $\kappa_0 = \left\|\mathbf{B} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b\right\|_2 \left\|(\mathbf{B} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b)^{-1}\right\|_2$ hence the first upper bound is immediately obtained. The lower bound, 1, is derived when the maximum eigenvalue of $\mathbf{B} + \mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b$ is equal to its minimum eigenvalue. The second upper bound is given by noticing that $\lambda_{\min} \geqslant 0$ so this term can vanish in the expression of the first upper bound to give a looser upper bound. The value of η is the minimum of several classic upper bounds on the 2-norm of $\mathbf{K}$ with $\left\|\mathbf{K}\right\|_{\max} = \max_{(i,j)\in[\![1,\ell]\!]^2}\left|K_{ij}\right|$, $\left\|\mathbf{K}\right\|_1 = \max_{j\in[\![1,\ell]\!]}\sum_{i=1}^{\ell}\left|K_{ij}\right|$, $\left\|\mathbf{K}\right\|_\infty = \max_{i\in[\![1,\ell]\!]}\sum_{j=1}^{\ell}\left|K_{ij}\right|$ and $\left\|\mathbf{K}\right\|_{\mathrm{F}} = \sqrt{\sum_{i,j=1}^{\ell}K_{ij}^2}$. □

The Normal Equations

Considering the potentially large value that the condition number κ_0 can take (cf. Theorem 3.1), it is necessary to develop a numerically stable way to compute the solution of the linear system $(\mathbf{K}\Delta_\alpha^{-1}\mathbf{K}+\Delta_\xi^{-1}+\mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b)\mathbf{a}=\mathbf{u}$ despite how large the kernel matrix $\mathbf{K}$ might be. The first step is to write an equivalent of the inverse of the system matrix.

Lemma 3.1. *The solution of the linear system* $(\mathbf{K}\Delta_\alpha^{-1}\mathbf{K}+\Delta_\xi^{-1}+\mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b)\mathbf{a}=\mathbf{u}$ *is given by* $\mathbf{a}=\mathbf{d}-\big(\sum_{i=1}^{\ell}d_i\big)\mathbf{e}/\big(\delta_b+\sum_{i=1}^{\ell}e_i\big)$ *where* $\mathbf{d}=\Delta_\xi(\mathbf{u}-\mathbf{K}\Delta_\alpha^{-½}\mathbf{v})$ *and* $\mathbf{e}=\Delta_\xi(\mathbf{1}_\ell-\mathbf{K}\Delta_\alpha^{-½}\mathbf{w})$. *The vector* $\mathbf{v}$ *is the solution of the linear system* $(\mathbf{I}_\ell+\mathbf{G}\mathbf{G}^{\mathrm{t}})\mathbf{v}=\mathbf{G}\Delta_\xi^{½}\mathbf{u}$ *where* $\mathbf{G}=\Delta_\alpha^{-½}\mathbf{K}\Delta_\xi^{½}$. *The vector* $\mathbf{w}$ *is the solution of the linear system* $(\mathbf{I}_\ell+\mathbf{G}\mathbf{G}^{\mathrm{t}})\mathbf{w}=\mathbf{G}\Delta_\xi^{½}\mathbf{1}_\ell$.

Proof. Using the binomial inverse theorem, the inverse of the matrix $\mathbf{B}+\mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b$ (with $\mathbf{B}=\mathbf{K}\Delta_\alpha^{-1}\mathbf{K}+\Delta_\xi^{-1}$) is $(\mathbf{B}+\mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b)^{-1}=\mathbf{B}^{-1}-\mathbf{B}^{-1}\mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}\mathbf{B}^{-1}/(\delta_b+\mathbf{1}_\ell^{\mathrm{t}}\mathbf{B}^{-1}\mathbf{1}_\ell)$. Let $\mathbf{d}$ be the solution of the linear system $\mathbf{B}\mathbf{d}=\mathbf{u}$ and let $\mathbf{e}$ be the solution of the linear system $\mathbf{B}\mathbf{e}=\mathbf{1}_\ell$. With these notations, the solution $\mathbf{a}$ of the linear system $(\mathbf{B}+\mathbf{1}_\ell\mathbf{1}_\ell^{\mathrm{t}}/\delta_b)\mathbf{a}=\mathbf{u}$ is then $\mathbf{a}=\mathbf{d}-\mathbf{e}(\mathbf{1}_\ell^{\mathrm{t}}\mathbf{d})/(\delta_b+\mathbf{1}_\ell^{\mathrm{t}}\mathbf{e})$. Furthermore, we can develop the expression of the inverse of the matrix $\mathbf{B}$ further using the Woodbury matrix identity [Woodbury, 1950]. For instance, $\mathbf{B}^{-1}=\big(\Delta_\xi^{-1}+\mathbf{K}\Delta_\alpha^{-1}\mathbf{K}\big)^{-1}=\Delta_\xi-\Delta_\xi\mathbf{K}\Delta_\alpha^{-½}\big(\mathbf{I}+\Delta_\alpha^{-½}\mathbf{K}\Delta_\xi\mathbf{K}\Delta_\alpha^{-½}\big)^{-1}\Delta_\alpha^{-½}\mathbf{K}\Delta_\xi$ since that the matrices Δ_α and Δ_ξ are symmetric and positive definite and that the matrix $\mathbf{K}$ is symmetric and positive semi-definite. If $\mathbf{G}=\Delta_\alpha^{-½}\mathbf{K}\Delta_\xi^{½}$, then $\mathbf{d}=\mathbf{B}^{-1}\mathbf{u}=\Delta_\xi\mathbf{u}-\Delta_\xi\mathbf{K}\Delta_\alpha^{-½}\mathbf{v}$ and $\mathbf{e}=\mathbf{B}^{-1}\mathbf{1}_\ell=\Delta_\xi\mathbf{1}_\ell-\Delta_\xi\mathbf{K}\Delta_\alpha^{-½}\mathbf{w}$, where $\mathbf{v}$ is the solution of the linear system $(\mathbf{I}_\ell+\mathbf{G}\mathbf{G}^{\mathrm{t}})\mathbf{v}=\mathbf{G}\Delta_\xi^{½}\mathbf{u}$ and $\mathbf{w}$ is the solution of the linear system $(\mathbf{I}_\ell+\mathbf{G}\mathbf{G}^{\mathrm{t}})\mathbf{w}=\mathbf{G}\Delta_\xi^{½}\mathbf{1}_\ell$. □

The trained eye can immediately recognize that a linear system of the form $(\mathbf{I}_\ell+\mathbf{G}\mathbf{G}^{\mathrm{t}})\mathbf{v}=\mathbf{G}\Delta_\xi^{½}\mathbf{u}$ is nothing else than a Tikhonov regularization [Tychonoff, 1963] of the normal equations. In our case, the transformation was beneficial for two reasons. The first reason is the (slight) improvement of the condition number of the matrix of the linear system, as show in Corollary 3.1. The second reason is the existence of efficient and

numerically stable large scale conjugate gradient methods that solve such linear systems with a computational complexity of the order of $O(\ell^2)$ [Bai and Zhang, 2002; Chen and Shen, 2007; Hestenes and Stiefel, 1952]. The condition number can be further improved, but at the price of a costly Cholesky decomposition (which computational complexity is of the order of $O(\ell^3)$). If such a decomposition is affordable, then the linear system can be quickly solved by back-substitution (that has a computational complexity of the order of $O(\ell^2)$) and needs not to be solved using a conjugate gradient method anymore.

Corollary 3.1. *The condition number κ_1 of the matrix $\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}}$ is such that:*

$$1 \leqslant \kappa_1 \leqslant \frac{1 + \delta_\xi^{\max}\lambda_{\max}^2/\delta_\alpha^{\min}}{1 + \delta_\xi^{\min}\lambda_{\min}^2/\delta_\alpha^{\max}} \leqslant 1 + \delta_\xi^{\max}\eta^2/\delta_\alpha^{\min}.$$

If $\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}} = \mathbf{L}\mathbf{L}^{\mathrm{t}}$ is the Cholesky decomposition of $\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}}$ where $\mathbf{L}$ is a lower triangular matrix, then the condition number $\tilde{\kappa}_1$ of the matrix $\mathbf{L}$ is such that $\tilde{\kappa}_1 \leqslant \sqrt{1 + \delta_\xi^{\max}\eta^2/\delta_\alpha^{\min}}$.

Proof. We have that $\|\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}}\|_2 \leqslant 1 + \left\|\Delta_\alpha^{-½}\right\|_2^2 \left\|\Delta_\xi^{½}\right\|_2^2 \|\mathbf{K}\|_2^2 = 1 + \delta_\xi^{\max}\lambda_{\max}^2/\delta_\alpha^{\min}$. The matrix $\mathbf{K}$ is semi-positive definite and the matrices Δ_α and Δ_ξ are positive definite, hence a lower bound of the smallest eigenvalue of the matrix $\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}}$ is $1 + \delta_\xi^{\min}\lambda_{\min}^2/\delta_\alpha^{\max} > 0$. Consequently, $\left\|(\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}})^{-1}\right\|_2 \leqslant 1/(1 + \delta_\xi^{\min}\lambda_{\min}^2/\delta_\alpha^{\max})$. The rest of the proof for the lower and upper bounds of κ_1 can be found in the proof of Theorem 3.1. The upper bound on the condition number of the lower triangular matrix $\mathbf{L}$ is derived by noticing that, since $\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}}$ is positive definite, if $\sigma^{\max} > 0$ is the maximum eigenvalue of the matrix $\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}}$ (and hence its 2-norm), then the maximum eigenvalue of $\mathbf{L}$ is $\sqrt{\sigma^{\max}}$. The minimum eigenvalue of $\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}}$ is $\sigma^{\min} > 0$ and hence the condition number $\tilde{\kappa}_1$ is such that $\tilde{\kappa}_1 \leqslant \sqrt{\sigma^{\max}/\sigma^{\min}}$. The upper bound $\sqrt{1 + \delta_\xi^{\max}\eta^2/\delta_\alpha^{\min}}$ is derived from the upper bound of $\tilde{\kappa}_1$, which is $\sqrt{\left(1 + \delta_\xi^{\max}\lambda_{\max}^2/\delta_\alpha^{\min}\right)/\left(1 + \delta_\xi^{\min}\lambda_{\min}^2/\delta_\alpha^{\max}\right)}$ since $\sigma^{\max} \leqslant 1 + \delta_\xi^{\max}\lambda_{\max}^2/\delta_\alpha^{\min}$ and $\sigma^{\min} \geqslant 1 + \delta_\xi^{\min}\lambda_{\min}^2/\delta_\alpha^{\max}$. □

Corollary 3.1 provides a easy way to control the condition number of the system matrix by tuning the matrices Δ_α and Δ_ξ appropriately. For example, choosing these matrices such that $\delta_\alpha^{\min}/\delta_\xi^{\max} \approx \eta^2$ will yield an upper bound for κ_1 approximately equal to 2, which correspond to a single binary digit lost during the computation of the solution of the linear system. Low condition numbers also provide a rapid convergence of conjugate gradient methods.

Special Case for Symmetric Matrices

In case the matrix $\mathbf{G}$ is symmetric (e.g. $\Delta_\xi^{½} = \tau\Delta_\alpha^{-½}$ with $\tau > 0$), then it becomes possible to solve the regularized normal equations above with a series of linear systems which matrices are much more well-conditioned than in the general case. This method can provide an even better conditioning if the computational cost of the Cholesky decomposition of the matrix $\mathbf{I}_\ell + \mathbf{G}$ is not too high. If such a decomposition is affordable, then solving the linear system of Corollary 3.1 is equivalent of solving triangular systems by back-substitution which is extremely fast. Otherwise a conjugate gradient algorithm can be used iteratively if the matrix $\mathbf{I}_\ell + \mathbf{G}$ is too large for a fast decomposition.

Theorem 3.2. *If the matrix* $\mathbf{G}$ *is symmetric, then the sum* $\sum_{i\in\mathbb{N}} \mathbf{v}_i$ *with principal term defined by* $(\mathbf{I}_\ell+\mathbf{G})^2\mathbf{v}_0 = \mathbf{G}\Delta_\xi^{½}\mathbf{u}$, $(\mathbf{I}_\ell+\mathbf{G})\tilde{\mathbf{v}}_i = \mathbf{v}_i$, *and* $(\mathbf{I}_\ell+\mathbf{G})\mathbf{v}_{i+1} = 2(\mathbf{v}_i - \tilde{\mathbf{v}}_i)$ *for* $i \in \mathbb{N}$ *is finite and is the solution of the linear system* $(\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}})\mathbf{v} = \mathbf{G}\Delta_\xi^{½}\mathbf{u}$.

Proof. We have $\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}} = \mathbf{I}_\ell + \mathbf{G}^2 = (\mathbf{I}_\ell + \mathbf{G})^2 - 2\mathbf{G}$ since $\mathbf{G}$ is symmetric. Furthermore, since $\mathbf{G}$ is positive semi-definite, the matrix $\mathbf{C} = \mathbf{I}_\ell + \mathbf{G}$ is invertible and we have that $\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}} = \mathbf{C}^2\big(\mathbf{I}_\ell + \mathbf{C}^{-2}(-2\mathbf{G})\big)$. It follows that we have $(\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}})^{-1} = \big(\mathbf{I}_\ell + \mathbf{C}^{-2}(-2\mathbf{G})\big)^{-1}\mathbf{C}^{-2}$. Now, given an $\ell \times \ell$ matrix $\mathbf{U}$ such that $\mathbf{U}^i \to \mathbf{0}$ for $i \to \infty$, we have the Taylor expansion $(\mathbf{I}_\ell + \mathbf{U})^{-1} = \sum_{i\geqslant 0}(-1)^i\mathbf{U}^i$. Or, since $\mathbf{G}$ is symmetric, if $\mathbf{G} = \mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{t}}$ is the eigen-decomposition of the matrix $\mathbf{G}$ where $\mathbf{Q}$ is orthonormal and Λ is diagonal, then we have $\big\|\mathbf{C}^{-2}(-2\mathbf{G})\big\|_2 = 2\big\|(\mathbf{I}_\ell + \Lambda)^{-2}\Lambda\big\|_2 = 2\max_{i\in[\![1,\ell]\!]}\big\{\lambda_i/(1+\lambda_i)^2\big\}$. The function

$x \mapsto \frac{x}{(x+1)^2}$ is continuously differentiable on $\mathbb{R}_+$ and it reaches its unique maximum on $\mathbb{R}_+$ at $x^* = 1$ for which its point value is $\frac{1}{4}$. Hence $\left\|\mathbf{C}^{-2}(-2\mathbf{G})\right\|_2 \leqslant 1/2$, $\left\|\left(\mathbf{C}^{-2}(-2\mathbf{G})\right)^i\right\|_2 \leqslant 1/2^i \to 0$ when $i \to \infty$ and $(\mathbf{I}_\ell + \mathbf{G}\mathbf{G}^{\mathrm{t}})^{-1} = \mathbf{C}^{-2} + \sum_{i\geqslant 1}\left(2\mathbf{C}^{-2}\mathbf{G}\right)^i\mathbf{C}^{-2}$. Noticing that $\mathbf{C}^{-2}\mathbf{G} = (\mathbf{G}+\mathbf{I}_\ell)^{-2}(\mathbf{G}+\mathbf{I}_\ell-\mathbf{I}_\ell) = \mathbf{C}^{-1}-\mathbf{C}^{-2}$, then solving $(\mathbf{I}_\ell+\mathbf{G}\mathbf{G}^{\mathrm{t}})\mathbf{v} = \mathbf{G}\Delta_\xi^{\frac{1}{2}}\mathbf{u}$ is equivalent to solve $\mathbf{C}^2\mathbf{v}_0 = \mathbf{G}\Delta_\xi^{\frac{1}{2}}\mathbf{u}$ and to compute $\mathbf{v}_0 + \sum_{i\geqslant 1}\left(2(\mathbf{C}^{-1}-\mathbf{C}^{-2})\right)^i\mathbf{v}_0$. The sequence with principal term $\mathbf{v}_i = \left(2(\mathbf{C}^{-1}-\mathbf{C}^{-2})\right)^i\mathbf{v}_0$ for $i \in \mathbb{N}^*$ can be derived in a recursive form. We have $\mathbf{v}_{i+1} = 2(\mathbf{C}^{-1}-\mathbf{C}^{-2})\mathbf{v}_i = 2(\mathbf{C}^{-1}\mathbf{v}_i - \mathbf{C}^{-2}\mathbf{v}_i)$ i.e. $\mathbf{C}\mathbf{v}_{i+1} = 2(\mathbf{v}_i - \tilde{\mathbf{v}}_i)$ with $\tilde{\mathbf{v}}_i$ being the solution of the linear system $\mathbf{C}\tilde{\mathbf{v}}_i = \mathbf{v}_i$. □

We can now write another corollary of Theorem 3.1 concerning the condition number of the new system matrix $\mathbf{I}_\ell + \mathbf{G}$. The upper bound of the condition number can be further improved if a Cholesky decomposition of the system matrix is used.

Corollary 3.2. *The condition number κ_2 of the matrix $\mathbf{I}_\ell + \mathbf{G}$ is such that:*

$$1 \leqslant \kappa_2 \leqslant \frac{1+\delta_\xi^{\max}\lambda_{\max}/\delta_\alpha^{\min}}{1+\delta_\xi^{\min}\lambda_{\min}/\delta_\alpha^{\max}} \leqslant 1+\delta_\xi^{\max}\eta/\delta_\alpha^{\min}.$$

Given that $\mathbf{G}$ is symmetric, if $\mathbf{I}_\ell + \mathbf{G} = \mathbf{L}\mathbf{L}^{\mathrm{t}}$ is the Cholesky decomposition of $\mathbf{I}_\ell + \mathbf{G}$ where $\mathbf{L}$ is a lower triangular matrix, then the condition number $\tilde{\kappa}_2$ of the matrix $\mathbf{L}$ is such that $\tilde{\kappa}_2 \leqslant \sqrt{1+\delta_\xi^{\max}\eta/\delta_\alpha^{\min}}$.

Proof. The proof is similar to the proof of Corollary 3.1. □

3.3 Algorithms and Complexities

3.3.1 Function Estimation for the General Case

The procedure that determines the optimal parameters α and b of an estimating function f (see Section 2.3) for any choice of trade-off coefficients is derived from the results of

Lemma 3.1. Given a finite observation set $\mathscr{X} \subset X$ and a target vector $\mathbf{y}$ which elements are the targets of each observation in $\mathscr{X}$, one must initially choose an appropriate kernel k_σ operating on $X \times X$ with a parameter vector σ as well as the trade-off coefficients $(\delta_\alpha, \delta_\xi, \delta_b, \mathbf{c}_\alpha, \mathbf{c}_\xi, c_b)$ of the Quadratic Programming problem in Equation 3.21. The choice of an appropriate kernel is discussed in Section 2.2 while the choice of the coefficients is presented in Sub-Section 2.4.3 and 3.3.3. Once all these elements are collected, an estimating function can finally be computed using Algorithm 3.1.

If ℓ is the total number of observations contained in $\mathscr{X}$ and if c_k is the number of floating point operations required to compute the kernel output between two observations, then Algorithm 3.1 requires $4\ell^3/3+(36+c_k)\ell^2/2+(100+3c_k)\ell/6+4$ FLOPS to compute the solutions α, ξ and b. In other words, this algorithm has a time complexity in $O(\ell^3)$ which is due to the operations on lines 11 (matrix-matrix multiplication) and 13 (Cholesky decomposition).

Matrix-matrix multiplication algorithms such as the Strassen algorithm [Strassen, 1969] or the Coppersmith-Winograd algorithm [Coppersmith and Winograd, 1990] have a smaller time complexity, but their numerical stability is weaker and the time gain is only noticeable for large non-sparse matrices. In the case of the Coppersmith-Winograd algorithm, the matrices in question are larger than what modern computer hardware can process.

If the kernel matrix is large and sparse, then the Cholesky decomposition on line 13 can be removed and the computation of the solutions of the linear systems can be performed using a Conjugate Gradient method (Algorithm B.3). Such a method has a time complexity in $O(\ell^2)$ but the time gain is noticeable for large systems and outputs are approximations. Additionally, the system matrix may be ill-conditioned (see Corollary 3.1) which will contribute to even poorer approximations if a Conjugate Gradient method is used.

If the number ℓ of observations is an issue, then one might use data thinning approaches (see Sub-Section 2.1.3) to decrease the size of the kernel matrix while keeping a good sample to learn from. This way, a Cholesky decomposition becomes a better choice than a

Conjugate Gradient method, and the number of FLOPS of Algorithm 3.1 and the condition number of the system matrices can be efficiently controlled.

Algorithm 3.1: Function Estimation for the General Case

Function $[\alpha, \xi, b]$ = GEFEST $(\mathscr{X}, \mathbf{y}, k_\sigma, \delta, \mathbf{c})$

Input: set $\mathscr{X}$, observation vector $\mathbf{y} \in \mathbb{R}^\ell$, kernel k_σ with parameter vector σ, coefficient vectors $\delta = (\delta_\alpha; \delta_\xi; \delta_b)$ and $\mathbf{c} = (\mathbf{c}_\alpha; \mathbf{c}_\xi; c_b)$ in $\mathbb{R}^{2\ell+1}$.

Output: parameters $\alpha \in \mathbb{R}^\ell$ and $b \in \mathbb{R}$, slack vector $\xi \in \mathbb{R}^\ell$.

1 $\mathbf{K} \leftarrow$ KERMAT $(\mathscr{X}, k_\sigma)$ // Kernel matrix computation

2 $\mu_\alpha \leftarrow 1/\delta_\alpha$, $\mu_\xi \leftarrow 1/\delta_\xi$, $\mu_b \leftarrow 1/\delta_b$ // Symbol / is entry-wise division

3 $\nu_\alpha \leftarrow \sqrt{\mu_\alpha}$ // Symbol $\sqrt{\ }$ is entry-wise square root

4 $\bar{\mathbf{c}}_\alpha \leftarrow 0.5(\mu_\alpha \cdot \mathbf{c}_\alpha)$, $\bar{\mathbf{c}}_\xi \leftarrow -0.5(\mu_\xi \cdot \mathbf{c}_\xi)$ // Symbol $\cdot$ is Hadamard product

5 $\bar{c}_b \leftarrow 0.5\mu_b c_b$

6 $\mathbf{u} \leftarrow \mathbf{y} +$ GEMV $(\mathbf{K}, \bar{\mathbf{c}}_\alpha) + \bar{\mathbf{c}}_\xi + \bar{c}_b$ // Last term implies a scalar expansion

7 $\mathbf{G} \leftarrow \mathbf{K}$, $\mathbf{G} \leftarrow$ DIMM $(\nu_\alpha, \mathbf{G})$ // $\mathbf{G} = \Delta_\alpha^{-½}\mathbf{K}$

8 $\mathbf{H} \leftarrow \mathbf{G}$, $\mathbf{H} \leftarrow$ MDIM $(\mathbf{H}, \delta_\xi)$ // $\mathbf{H} = \Delta_\alpha^{-½}\mathbf{K}\Delta_\xi$

9 $\mathbf{p} \leftarrow$ GEMV $(\mathbf{H}, \mathbf{u})$ // $\mathbf{p} \leftarrow \mathbf{H}\mathbf{u}$

10 $\mathbf{q} \leftarrow$ ROWSUM $(\mathbf{H})$ // $\mathbf{q} \leftarrow \sum_{i=1}^{\ell} \mathbf{H}_{\cdot i}$

11 $\mathbf{G} \leftarrow$ SYR2K $(\mathbf{H}, \mathbf{G})$ // $\mathbf{G} \leftarrow \mathbf{H}\mathbf{G}^{\mathrm{t}}$

12 $\mathbf{G} \leftarrow$ TIKREG $(\mathbf{G}, 1)$ // Tikhonov regularization

13 $\mathbf{L} \leftarrow$ CHOLDC $(\mathbf{G})$ // Cholesky decomposition

14 $\mathbf{v}^* \leftarrow$ CHOLSL $(\mathbf{L}, \mathbf{p})$, $\mathbf{w}^* \leftarrow$ CHOLSL $(\mathbf{L}, \mathbf{q})$ // Solve by back-substitution

15 $\mathbf{H} \leftarrow \mathbf{H}^{\mathrm{t}}$, $\mathbf{d} \leftarrow \delta_\xi \cdot \mathbf{u} -$ GEMV $(\mathbf{H}, \mathbf{v}^*)$, $\mathbf{e} \leftarrow \delta_\xi -$ GEMV $(\mathbf{H}, \mathbf{w}^*)$

16 $\mathbf{a} \leftarrow \mathbf{d} - \big($ SUM $(\mathbf{d})/\big(\delta_b +$ SUM $(\mathbf{e})\big)\big)\mathbf{e}$

17 $\alpha \leftarrow \mu_\alpha \cdot$ GEMV $(\mathbf{K}, \mathbf{a}) - \bar{\mathbf{c}}_\alpha$, $\xi \leftarrow \bar{\mathbf{c}}_\xi - \mu_\xi \cdot \mathbf{a}$, $b \leftarrow \mu_b$ SUM $(\mathbf{a}) - \bar{c}_b$

18 **return** (α, ξ, b)

3.3.2 Function Estimation for the Symmetric Case

When trade-off coefficients are chosen such that the matrix **G** of Theorem 3.2 is symmetric, it then becomes possible to obtain an optimal solution to the function estimation problem by a numerically stable iterative procedure. The approach slightly differs from Algorithm 3.1 in the sense that the matrix-matrix multiplication at line 11 is no longer needed and that the operations on line 14 are replaced by iterative procedures (see Algorithm 3.2).

Algorithm 3.2: Function Estimation for the Symmetric Case

Function $[\alpha, \xi, b]$ = SYFEST $(\mathscr{X}, \mathbf{y}, k_\sigma, \delta, \mathbf{c})$
Input: set $\mathscr{X}$, observation vector $\mathbf{y} \in \mathbb{R}^\ell$, kernel k_σ with parameter vector σ, coefficient vectors $\delta = (\delta_\alpha; \delta_\xi; \delta_b)$ and $\mathbf{c} = (\mathbf{c}_\alpha; \mathbf{c}_\xi; c_b)$ in $\mathbb{R}^{2\ell+1}$.
Output: parameters $\alpha \in \mathbb{R}^\ell$ and $b \in \mathbb{R}$, slack vector $\xi \in \mathbb{R}^\ell$.

1 $\mathbf{K} \leftarrow$ KERMAT $(\mathscr{X}, k_\sigma)$
2 $\mu_\alpha \leftarrow 1/\delta_\alpha$, $\mu_\xi \leftarrow 1/\delta_\xi$, $\mu_b \leftarrow 1/\delta_b$, $\nu_\alpha \leftarrow \sqrt{\mu_\alpha}$, $\nu_\xi \leftarrow \sqrt{\delta_\xi}$
3 $\bar{\mathbf{c}}_\alpha \leftarrow 0.5(\mu_\alpha \cdot \mathbf{c}_\alpha)$, $\bar{\mathbf{c}}_\xi \leftarrow -0.5(\mu_\xi \cdot \mathbf{c}_\xi)$, $\bar{c}_b \leftarrow 0.5\mu_b c_b$
4 $\mathbf{u} \leftarrow \mathbf{y} +$ GEMV $(\mathbf{K}, \bar{\mathbf{c}}_\alpha) + \bar{\mathbf{c}}_\xi + \bar{c}_b$
5 $\mathbf{G} \leftarrow \mathbf{K}$, $\mathbf{G} \leftarrow$ DIMM $(\nu_\alpha, \mathbf{G})$, $\mathbf{H} \leftarrow \mathbf{G}$, $\mathbf{H} \leftarrow$ MDIM $(\mathbf{H}, \delta_\xi)$
6 $\mathbf{p} \leftarrow$ GEMV $(\mathbf{H}, \mathbf{u})$, $\mathbf{q} \leftarrow$ ROWSUM $(\mathbf{H})$
7 $\mathbf{G} \leftarrow$ MDIM $(\mathbf{G}, \nu_\xi)$, $\mathbf{G} \leftarrow$ TIKREG $(\mathbf{G}, 1)$
8 $\mathbf{L} \leftarrow$ CHOLDC $(\mathbf{G})$, $\mathbf{v}^* \leftarrow$ SFECD $(\mathbf{L}, \mathbf{p})$, $\mathbf{w}^* \leftarrow$ SFECD $(\mathbf{L}, \mathbf{q})$
9 $\mathbf{H} \leftarrow \mathbf{H}^{\mathrm{t}}$, $\mathbf{d} \leftarrow \delta_\xi \cdot \mathbf{u} -$ GEMV $(\mathbf{H}, \mathbf{v}^*)$, $\mathbf{e} \leftarrow \delta_\xi -$ GEMV $(\mathbf{H}, \mathbf{w}^*)$
10 $\mathbf{a} \leftarrow \mathbf{d} - \big($ SUM $(\mathbf{d})/\big(\delta_b +$ SUM $(\mathbf{e})\big)\big)\mathbf{e}$
11 $\alpha \leftarrow \mu_\alpha \cdot$ GEMV $(\mathbf{K}, \mathbf{a}) - \bar{\mathbf{c}}_\alpha$, $\xi \leftarrow \bar{\mathbf{c}}_\xi - \mu_\xi \cdot \mathbf{a}$, $b \leftarrow \mu_b$ SUM $(\mathbf{a}) - \bar{c}_b$
12 **return** (α, ξ, b)

The implementation of the SFECD procedure which is called on line 8 of Algorithm 3.2 is described in Algorithm 3.3. This procedure requires $4(1+k_{\max})\ell^2 + (2+5k_{\max})\ell - 1$ FLOPS in the worst-case scenario (i.e. early termination) where $k_{\max}$ is the maximum number of iterations allowed.

Algorithm 3.3: Symmetric Case Solution using a Cholesky Decomposition

Function $[\mathbf{x}]$ = SFECD $(\mathbf{L}, \mathbf{b})$
Data: tolerance $\varepsilon > 0$, iteration limit $k_{\max} > 0$.
Input: $n \times n$ lower triangular matrix $\mathbf{L}$, right-hand vector $\mathbf{b} \in \mathbb{R}^n$.
Output: vector $\mathbf{z} \in \mathbb{R}^n$.

1 $\mathbf{y} \leftarrow$ CHOLSL $(\mathbf{L}, \mathbf{b})$, $\mathbf{x} \leftarrow$ CHOLSL $(\mathbf{L}, \mathbf{y})$, $\mathbf{z} \leftarrow \mathbf{x}$, $\rho \leftarrow$ DOT $(\mathbf{x}, \mathbf{x})$, $k \leftarrow 0$
2 **while** $\rho > \varepsilon$ *and* $k \leqslant k_{\max}$ **do**
3 $\quad \mathbf{y} \leftarrow$ CHOLSL $(\mathbf{L}, \mathbf{x})$, $\mathbf{x} \leftarrow$ CHOLSL $\big(\mathbf{L}, 2(\mathbf{x} - \mathbf{y})\big)$, $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{x}$, $\rho \leftarrow$ DOT $(\mathbf{x}, \mathbf{x})$
4 $\quad k \leftarrow k + 1$
5 **end**
6 **return z**

The analysis of Algorithm 3.3 gives the computational cost of Algorithm 3.2 for the worst-case scenario. The computation of a solution to the function estimation problem

takes at most $\ell^3/3 + (c_k + 16k_{\max} + 45)\ell^2/2 + (3c_k + 60k_{\max} + 133)\ell/6 + 2$ FLOPS where c_k is the number of floating point operations required to compute the kernel output between two observations. Asymptotically, Algorithm 3.2 requires four times less floating point operations than Algorithm 3.1 with the same memory requirements and improves the conditioning by one order of magnitude, providing a greater numerical stability.

If the kernel matrix is large and sparse, then the operations on line 8 of Algorithm 3.2 can be replaced by iterative procedures based on a Conjugate Gradient method (see Algorithm B.3). This will render the asymptotic time complexity of the algorithm $O(\ell^2)$ but at the cost of a slightly lower numerical stability (see Corollary 3.2).

3.3.3 Function Estimation with Optimal Coefficients

The automated choice of the coefficients of the objective function of the quadratic programming problem in Equation 3.21 has three objectives:

- Improve the numerical stability of the resolution methods;
- Speed-up the computation of the solutions;
- Simplify the modeling of the estimating function.

These choices are motivated by different reasons that depend of the nature of the coefficients (coefficients for the quadratic terms or coefficients for the linear terms).

Coefficients of the Quadratic Terms

It was shown in Sub-Section 3.2.2 that, if the matrix $\mathbf{G}$ is symmetric then we can derive an algorithm which solves the function estimation problem with four times less floating point operations and a greater numerical stability. The symmetry of $\mathbf{G}$ is directly dependent on the appropriate choice of the matrices Δ_α and Δ_ξ since $\mathbf{G} = \Delta_\alpha^{-½}\mathbf{K}\Delta_\xi^{½}$ and the kernel matrix $\mathbf{K}$ is, by definition, symmetric. Hence a first requirement for the coefficients of the

quadratic terms is that

$$\Delta_{\xi}^{½} = \tau \Delta_{\alpha}^{-½}, \tag{3.32}$$

with $\tau > 0$. Since both matrices Δ_{α} and Δ_{ξ} are diagonal and positive definite, Equation 3.32 can be rewritten as

$$(\delta_{\alpha})_i (\delta_{\xi})_i = \tau^2 > 0 \ \text{ for all } i \in [\![1, \ell]\!], \tag{3.33}$$

with $(\delta_{\alpha})_i$ and $(\delta_{\xi})_i$ being the i-th components of the diagonals of Δ_{α} and Δ_{ξ}. If Equation 3.33 is satisfied, then the matrix $\mathbf{G}$ is symmetric.

If we assume that Equation 3.33 holds, then we can further improve the numerical stability of Algorithm 3.2 using the results of Corollary 3.2. This corollary provides an upper bound for the number of base-d digits that are lost during the computation of the solutions of the linear systems in Algorithm 3.2. If $p \in \mathbb{N}^*$ is the maximum number of base-d digits which can be lost during computation, then we must find $\delta_{\alpha}^{\min} = \min_{i \in [\![1,\ell]\!]} (\delta_{\alpha})_i$ and $\delta_{\xi}^{\max} = \max_{i \in [\![1,\ell]\!]} (\delta_{\xi})_i$ such that

$$\log_d \left(\sqrt{1 + \delta_{\xi}^{\max} \eta / \delta_{\alpha}^{\min}} \right) \leqslant p, \tag{3.34}$$

where $\eta = \min \left\{ \ell \, \|\mathbf{K}\|_{\max}, \sqrt{\ell} \, \|\mathbf{K}\|_1, \sqrt{\|\mathbf{K}\|_1 \, \|\mathbf{K}\|_{\infty}}, \|\mathbf{K}\|_{\mathrm{F}} \right\}$. In other words, we must have

$$\delta_{\xi}^{\max} \leqslant \frac{d^{2p} - 1}{\eta} \delta_{\alpha}^{\min}. \tag{3.35}$$

The construction of the objective function in Sub-Section 3.1.1 provides the final requirements for the coefficients of the quadratic terms. The minimization of the upper bound of the generalization error provides a reason for minimizing both the Δ_{α}-norm of α and the quantity $\delta_b b^2$ in an equal fashion since none of these two terms can minimize the bound

separately. Hence we should have an equality of the form $\|\Delta_\alpha\|_2 = \delta_b$ which gives

$$\delta_\alpha^{\max} = \delta_b > 0, \tag{3.36}$$

with $\delta_\alpha^{\max} = \max_{i \in [\![1,\ell]\!]} (\delta_\alpha)_i$. Furthermore we should balance the minimization of the generalization error with the minimization of the empirical error (which is a function of the variable ξ). By the same argument that led to Equation 3.36, we should have a positive trade-off coefficient $C \in \mathbb{R}_+^*$ such that $\|\Delta_\alpha\|_2 = C \left\|\Delta_\xi\right\|_2$, i.e.

$$\delta_\alpha^{\max} = C \delta_\xi^{\max}. \tag{3.37}$$

From Equations 3.33, 3.35, 3.36 and 3.37, we can see that the choice of δ_α and a trade-off coefficient $C > 0$ controls the choice of δ_ξ and δ_b. This shows that δ_α can be fixed arbitrarily and that the solution to the function estimation problem can be controlled with only a single parameter $C > 0$ for the coefficients of the quadratic terms. Additionally, if all the components of δ_α are equal to one, then Algorithm 3.2 requires much less floating point operations to compute a solution. Therefore a strategic choice for the matrix Δ_α is to have $\Delta_\alpha = \mathbf{I}_\ell$. Consequently, Equations 3.33 and 3.37 implies that $\Delta_\xi = C\mathbf{I}_\ell$ while Equation 3.36 implies that $\delta_b = 1$. Equation 3.35 gives a lower and an upper bound for the trade-off coefficient C, leading to

$$0 < C \leqslant \frac{d^{2p} - 1}{\eta}. \tag{3.38}$$

Coefficients of the Linear Terms

The linear coefficients of the objective function have a significant impact on the constraints of the optimization problem. They correspond to a translation in the solution space as it is

proved in the equation below:

$$\frac{1}{2}(\mathbf{x}+\mathbf{t})^{\mathrm{t}}\mathbf{H}(\mathbf{x}+\mathbf{t}) = \frac{1}{2}\mathbf{x}^{\mathrm{t}}\mathbf{H}\mathbf{x}+\mathbf{c}^{\mathrm{t}}\mathbf{x}+C, \tag{3.39}$$

where $\mathbf{c} = \mathbf{H}\mathbf{t}$ and $C = \mathbf{t}^{\mathrm{t}}\mathbf{H}\mathbf{t}/2$ are constant terms. While such a translation provides no advantage to the objective function, it greatly impact the expression of the constraints of the mathematical programming problem. In our case, the coefficients $\mathbf{c}_\alpha$, $\mathbf{c}_\xi$ and c_b impact the values of the vector $\mathbf{u}$ as defined in Proposition 3.1. In other words, given such coefficients, we have

$$u_i = y_i + \frac{1}{2}\left(\mathbf{K}_{i\cdot}\Delta_\alpha^{-1}\mathbf{c}_\alpha - \frac{(\mathbf{c}_\xi)_i}{(\delta_\xi)_i} + \frac{c_b}{\delta_b}\right), \tag{3.40}$$

for all $i \in [\![1,\ell]\!]$. This equation shows that the linear coefficients contribute to the *scaling* of the vector $\mathbf{u}$ which, in return, greatly impacts the quality of the numerical solution given by Algorithms 3.1 and 3.2.

Both coefficients $\mathbf{c}_\alpha$ and $\mathbf{c}_\xi$ modify the vector $\mathbf{y}$ on a *per element* basis while the coefficient c_b influence all the components of $\mathbf{y}$ at once. Since there are no other factor that discriminate $\mathbf{c}_\alpha$ and $\mathbf{c}_\xi$, then one of these vectors can be set equal to $\mathbf{0}$ and the other can be left to control the variations of the y_i's. The choice $\mathbf{c}_\alpha = \mathbf{0}$ saves on floating point operations (a matrix-vector product can be removed) and, hence, is a computationally better choice than $\mathbf{c}_\xi = \mathbf{0}$. A null coefficient $\mathbf{c}_\alpha$ also brings an additional form of robustness to ULMs since that observational outliers slightly change the value of the kernel matrix $\mathbf{K}$. Naturally, if the entries of $\mathbf{K}$ are perturbed, then the solution described in Theorem 3.2 will be perturbed as well. However, the choice $\mathbf{c}_\alpha = \mathbf{0}$ avoids the additional perturbation of the right-hand term of the linear systems. Hence, the removal of one of the terms most vulnerable to observational errors leads to more stable and reliable optimal solutions.

By choosing $\delta_\xi = C\mathbf{1}_\ell$ with $C > 0$, $\delta_b = 1$ (i.e. an optimal choice for the coefficients of

the quadratic terms) and $\mathbf{c}_\alpha = \mathbf{0}$, we obtain the following identity

$$u_i = y_i - \frac{(\mathbf{c}_\xi)_i}{2C} + \frac{c_b}{2}, \tag{3.41}$$

for all $i \in [\![1, \ell]\!]$. The values of the components of $\mathbf{u}$ can be chosen such that

$$-1 \leqslant u_i \leqslant 1, \tag{3.42}$$

for all $i \in [\![1, \ell]\!]$, in order to improve the numerical stability of Algorithm 3.2. These inequalities are satisfied if we have

$$c_b = 2\left(\frac{2y_{\min}}{y_{\min} - y_{\max}} - 1\right), \tag{3.43}$$

and

$$(\mathbf{c}_\xi)_i = 2C\left(\frac{2}{y_{\max} - y_{\min}} - 1\right)y_i, \tag{3.44}$$

with $y_{\min} = \min_{i \in [\![1,\ell]\!]} y_i$ and $y_{\max} = \max_{i \in [\![1,\ell]\!]} y_i$. This leads to a simplified expression for the u_i's which is

$$u_i = 2\frac{y_i - y_{\min}}{y_{\max} - y_{\min}} - 1. \tag{3.45}$$

Algorithm with the Optimal Coefficients

Algorithm 3.4 is an adaptation of Algorithm 3.2 with the optimal choice for the coefficients of the objective function of the quadratic optimization problem of Equation 3.21. Given a trade-off coefficient $C > 0$ that satisfies Inequality 3.38, Algorithm 3.4, so far, is the fastest and most stable method investigated that computes a solution to the function estimation problem. As discussed in Sub-Section 3.3.2, the Cholesky decomposition of line 6 can be replaced by an asymptotically faster Conjugate Gradient method if the kernel matrix is large and sparse. However, it comes at the price of a less accurate solution.

Algorithm 3.4: Function Estimation with Optimal Coefficients

Function $[\alpha, \xi, b]$ = `OPTFEST` $(\mathscr{X}, \mathbf{y}, k_\sigma, C)$

Input: set $\mathscr{X}$, observation vector $\mathbf{y} \in \mathbb{R}^\ell$, kernel k_σ with parameter vector σ, trade-off coefficient $C > 0$.

Output: parameters $\alpha \in \mathbb{R}^\ell$ and $b \in \mathbb{R}$, slack vector $\xi \in \mathbb{R}^\ell$.

1 $\mathbf{K} \leftarrow$ `KERMAT` $(\mathscr{X}, k_\sigma)$
2 $A \leftarrow C^2$, $B \leftarrow \sqrt{C}$, $D \leftarrow 1/C$
3 $y_{\max} \leftarrow$ `MAX` $(\mathbf{y})$, $y_{\min} \leftarrow$ `MIN` $(\mathbf{y})$, $E \leftarrow 2/(y_{\max} - y_{\min})$, $F \leftarrow E y_{\min} + 1$
4 $\mathbf{z} \leftarrow E\mathbf{y}$, $\mathbf{u} \leftarrow \mathbf{z} - F$, $\mathbf{G} \leftarrow$ `TIKREG` $(B\mathbf{K}, 1)$
5 $\mathbf{p} \leftarrow$ `GEMV` $(\mathbf{K}, \mathbf{u})$, $\mathbf{q} \leftarrow$ `ROWSUM` $(\mathbf{K})$
6 $\mathbf{L} \leftarrow$ `CHOLDC` $(\mathbf{G})$, $\mathbf{v}^* \leftarrow$ `SFECD` $(\mathbf{L}, \mathbf{p})$, $\mathbf{w}^* \leftarrow$ `SFECD` $(\mathbf{L}, \mathbf{q})$
7 **for** $i \in [\![1, \ell]\!]$ **do** $e_i \leftarrow D$
8 $\mathbf{e} \leftarrow \mathbf{e} -$ `GEMV` $(\mathbf{K}, \mathbf{w}^*)$, $\mathbf{d} \leftarrow D\mathbf{u} -$ `GEMV` $(\mathbf{K}, \mathbf{v}^*)$
9 $\mathbf{a} \leftarrow \mathbf{d} - \big($ `SUM` $(\mathbf{d}) / \big(D^2 +$ `SUM` $(\mathbf{e})\big)\big)\mathbf{e}$
10 $\alpha \leftarrow A$ `GEMV` $(\mathbf{K}, \mathbf{a})$, $\xi \leftarrow \mathbf{y} - \mathbf{z} - C\mathbf{a}$, $b \leftarrow A$ `SUM` $(\mathbf{a}) + F$
11 **return** (α, ξ, b)

If c_k is the number of floating point operations required to compute the kernel output between two observations and if $k_{\max}$ is the iteration limit of the procedure `SFECD`, then Algorithm 3.4 requires at most $\ell^3/3 + (c_k + 16k_{\max} + 37)\ell^2/2 + (3c_k + 60k_{\max} + 85)\ell/6 + 7$ FLOPS to compute a solution. This is an improvement of $4\ell^2 + 8\ell - 5$ FLOPS with respect to Algorithm 3.2 which, for an average size of $\ell = 5000$ observations, represents a gain of 100039995 floating point operations.

Part II

Applications of Unconstrained Learning Machines

Chapter 4

Form Inspection in Manufacturing Engineering

4.1 Introduction, Context and Aims

Manufacturing processes leave very specific patterns on part surfaces, which provide a good basis for inspecting them. In some cases, it is possible to quantify the manufacturing errors and their effect on the product, based on the model of the processing [Badar et al., 2003, 2005a,b]. However, when multiple processes are applied on the same feature, the net effect of these processing errors is far too difficult to model and compute. In this chapter, we apply ULMs to quantify process errors on parts and thereby provide a basis for adaptive sampling and form inspection.

Although previous works regarding the use of machine learning techniques for determining the size of the manufacturing errors exist [Malyscheff et al., 2002; Prakasvudhisarn et al., 2002, 2003; Balakrishna et al., 2008], the results were plagued by misalignment problems and sampling issues, and only concerned with simple geometric shapes such as plates [Hopp, 1993; Hulting, 1992]. It was also suggested that the manufacturing errors in such cases should be captured using experimental analysis, even though that sampling the part throughout, uniformly or randomly, may miss some process characteristics [Kim and Raman, 2000].

This chapter develops a general approach that attempts to circumvent all these problems. The objectives are: to re-align the measurements into their canonical frame of reference (following the work of Besl and McKay [1992] on *registration*); to assess if each surface that is probed has its form contained within a certain range from a nominal plane (i.e.

to determine the size of the *minimum zone* [ASME, 1995a,b; Carr and Ferreira, 1995a,b; Hurt and Colwell, 1980; Kurfess and Banks, 1995; Murthy and Abdin, 1980; Requicha, 1993; Roy and Zhang, 1992; Samuel and Shunmugam, 1999; Shunmugam, 1987]); and to optimize the probing procedure (sampling size, mesh geometry [Hocken et al., 1993; Woo and Liang, 1993], measurement path, accuracy, etc.) to accelerate the measurement process and to reduce time-induced errors.

Section 4.2 covers mesh issues while Section 4.3 details registration problems when knowledge about the canonical frame of reference has been partially lost. Section 4.4 introduces a nonlinear nonparametric regression approach based on ULMs that finds the size of the minimum zone (which is defined as the part of space where the actual surface form is different from its ideal nominal form). In Section 4.5, measurements were collected on several types of surfaces with a *Coordinate Measuring Machine* (CMM). ULMs were then used to fit test models of these surfaces and estimates of the size of the minimum zones.

4.2 Mesh Generation

The form inspection of manufactured parts naturally implies to collect surface measurements which are to be compared against a nominal profile. These measurements can be collected in an organized fashion with highly accurate probe-type CMMs, which is a certain advantage over optical CMMs. On the other hand, optical CMMs collect large amount of coordinate measurements quickly, if the part to be inspected can reflect the beam. Given that probe-type CMMs were not rendered obsolete by optical ones, it is then still crucial to design suitable meshes of contact points to retrieve observations samples with the right properties for the registration (see Section 4.3) and ULM-based regression (see Section 4.4) procedures.

The generalization error, which is one of the core concept of ULMs (see Sub-Section

2.3.4), indicates that deformation patterns on the surface of the parts will be successfully estimated if:

- Contact points (observations) are spread evenly on the entire portion of the surface to be inspected;
- There is a fair number of contact measurements collected.
- The surface model (i.e. the choice of the kernel and the other parameters needed by ULMs) is suitable.

While the last point is a matter left to the modeler and the validation procedures (see Sub-Section 2.4.3), the first two points are narrowing the type of meshes that are suitable for form inspection. Furthermore, a physical requirement of probe-type CMMs is to favor smooth and regular trajectories of the probe so a path that minimizes the time required to collect measurements can easily be found (and hence reduce the likelihood of time-induced errors). Smooth trajectories aim to minimize the sudden lateral displacements of the probe which are both time consuming and error-producing operations. This rules out fractal trajectories for the probe (e.g. space-filling curves such as Z-curves or Hilbert curves) despite that the induced meshes satisfy the first two requirements of the list above.

The construction of suitable meshes and the comparison of their respective properties necessitate the establishment of measures of the quality of the different meshes. Sub-Section 4.2.1 covers this issue while Sub-Section 4.2.2 proposes a candidate mesh for collecting measurements: a uniform grid. Sub-Section 4.2.3 discusses issues related to the trade-off between the quality of a mesh and the time needed to collect measurements.

4.2.1 Mesh Quality

The quality of a mesh has to give a proper measure of what constitutes a large enough number of nodes (hence a large enough number of contact points), and an even spread of

the nodes over a compact domain of the observation space (which is, in this case, $\mathbb{R}^2$). Additionally, the measure can be augmented with considerations based on the minimum time it takes for the probe to visit all the nodes.

Assume that we have n distinct nodes contained in a compact domain $\mathscr{D} \subset \mathbb{R}^2$ of total surface S and let ρ be the diameter of the smallest ball that contains $\mathscr{D}$. Nodes are represented by n points $\mathbf{a}_1, \ldots, \mathbf{a}_n$ in $\mathbb{R}^2$. These points are inducing a Voronoi tessellation of the compact domain $\mathscr{D}$ [Voronoi, 1907], and we will assume that the Dirichlet domain of each point $\mathbf{a}_i$ has a surface S_i, for all $i \in [\![1, n]\!]$. In all applications, the surfaces which need to be inspected can be described by a set of two parameters varying in close intervals of $\mathbb{R}$. In other words, the nodes of a mesh need to cover a rectangle in $[a, b] \times [c, d] = \mathscr{D} \subset \mathbb{R}^2$. Hence, meshes can be built to cover a rectangle and then the contact points can be projected onto the surface of interest.

Spreading the points $\mathbf{a}_1, \ldots, \mathbf{a}_n$ evenly over $\mathscr{D}$ is akin to maximize the following quantity:

$$\sum_{j \in \mathscr{N}_i} \frac{\left\| \mathbf{a}_i - \mathbf{a}_j \right\|_2^2}{\rho^2}, \tag{4.1}$$

for all $i \in [\![1, n]\!]$ and where $\mathscr{N}_i$ is the index set of nodes adjacent to the i-th node. This represents the need to maximize the distances between adjacent points in order to avoid them to agglomerate somewhere on the domain $\mathscr{D}$. The term ρ^2 is a scaling factor for the inter-node distances.

Having a large enough number of points distributed over the surface $\mathscr{D}$ is similar to maximizing the amount of contact information per unit of surface. This can be represented by the maximization of the quantity

$$S \sum_{i=1}^{n} \frac{1}{S_i}, \tag{4.2}$$

where S is a scaling factor. If each term of the sum in Equation 4.2 is weighted by the terms

of Equation 4.1, we then obtain the quantity

$$\sum_{i=1}^{n}\left(\frac{S}{S_i}\sum_{j\in\mathcal{N}_i}\frac{\left\|\mathbf{a}_i-\mathbf{a}_j\right\|_2^2}{\rho^2}\right), \tag{4.3}$$

which is growing with the number of nodes n. However, if the number of nodes grows, then the time spent to collect measurements grows as well. Hence, the quantity in Equation 4.3 is averaged by the number of nodes. Namely, we obtain a quantity Q which is defined by

$$Q=\frac{1}{n}\sum_{i=1}^{n}\sum_{j\in\mathcal{N}_i}\frac{S\left\|\mathbf{a}_i-\mathbf{a}_j\right\|_2^2}{S_i\rho^2}. \tag{4.4}$$

The quantity Q is chosen as the measure of the quality of a given mesh. The goal is to build a mesh that maximizes this quantity while satisfying time constraints related to the CMM probe visiting each node. The minimum length path (i.e. the path through all nodes requiring the less time, assuming that the probe has a constant displacement speed) is an issue that is separated from the concept of mesh quality. Usually a minimum-length path is computed from a given mesh but does not serve in the establishment of the mesh itself. Furthermore, the length of the path is dependent on which type of surface the mesh is being projected onto. This surface dependency is additionally complicated by the fact that the normal to the surface at the contact point is not co-linear with the axis of the probe. The reaction force when the probe is making contact with the surface pushes the tip aside and inserts a tiny, but not necessarily negligible, bias in the contact point measurements. All these factors are the main reasons why paths through nodes are not taken into account in the evaluation of the mesh quality. Measurement paths and probe angles are considerations made by the operator of the CMM which have marginal impact on the registration and regression procedures.

4.2.2 Uniform Grids

Grid Uniform grids were the chosen meshes for the experiments on face-milled plates (see Sub-Section 4.5.5). The choice is not claimed to be optimal with respect to the quality measure of Equation 4.4, but it was retained because of a few key properties:

- The simplicity of the structure that allows for the formulation of a close form of Equation 4.4 and the derivation of several useful measures;
- The conformance with the requirements of a suitable mesh (contact points spread evenly with each Dirichlet domain bringing an equal amount of information about the surface).

Meshes with contact points generated by low-discrepancy sequences (e.g. van der Corput sequence, Hammersley sequence) are an alternative to rigid structures such as uniform grids. They allow the use of a more flexible number of contact points, but Equation 4.4 has no close form and formulas predicting the measuring time with respect to the quality to be achieved cannot be properly established.

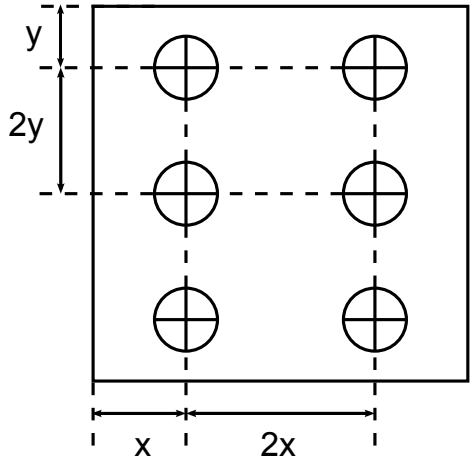


Figure 4.1: Schematics of a uniform grid.

An uniform grid on a rectangle domain $\mathscr{D} = [a,b] \times [c,d]$ is a grid with n columns and m rows such that the nodes are spaced equally row-wise and column-wise. The nodes

adjacent to the border of the rectangle domain are located half an inter-node distance from the border (see Figure 4.1). The surface of the domain is $S = (b-a)(d-c)$, the surface of the Dirichlet domains are all equal to $S_i = S/\big((n+1)(m+1)\big)$ for $i \in [\![1, nm]\!]$, and the diameter ρ is such that $\rho^2 = (b-a)^2/(n+1)^2 + (d-c)^2/(m+1)^2$. The grid has four types of nodes

- 4 corner nodes with

$$\sum_{j \in \mathcal{N}_i} \left\| \mathbf{a}_i - \mathbf{a}_j \right\|_2^2 = 2 \left(\frac{(b-a)^2}{(n+1)^2} + \frac{(d-c)^2}{(m+1)^2} \right). \tag{4.5}$$

- $2(m-2)$ nodes adjacent to the vertical borders with

$$\sum_{j \in \mathcal{N}_i} \left\| \mathbf{a}_i - \mathbf{a}_j \right\|_2^2 = 3 \frac{(b-a)^2}{(n+1)^2} + 4 \frac{(d-c)^2}{(m+1)^2}. \tag{4.6}$$

- $2(n-2)$ nodes adjacent to the horizontal borders with

$$\sum_{j \in \mathcal{N}_i} \left\| \mathbf{a}_i - \mathbf{a}_j \right\|_2^2 = 4 \frac{(b-a)^2}{(n+1)^2} + 3 \frac{(d-c)^2}{(m+1)^2}. \tag{4.7}$$

- $m(n-2) - 2n + 4$ other nodes with

$$\sum_{j \in \mathcal{N}_i} \left\| \mathbf{a}_i - \mathbf{a}_j \right\|_2^2 = 6 \left(\frac{(b-a)^2}{(n+1)^2} + \frac{(d-c)^2}{(m+1)^2} \right). \tag{4.8}$$

Using Equation 4.4, the quality of an $m \times n$ uniform grid is therefore

$$Q = \frac{2(n+1)(m+1)}{nm\big(\lambda^2 + \mu^2\big)} \left(\frac{\lambda^2 (n-1)(3m-2)}{(n+1)^2} + \frac{\mu^2 (m-1)(3n-2)}{(m+1)^2} \right), \tag{4.9}$$

where $\lambda = b - a$ and $\mu = d - c$. If we assume that $r = \lambda/\mu = n/m \geqslant 1$ is the *aspect ratio* of the domain $\mathscr{D}$, then Equation 4.9 is rewritten as

$$Q(n,r) = \frac{2r(n+r)(n+1)}{n^2(1+r^2)} \left(\frac{(n-1)(3n-2r)}{(n+1)^2} + \frac{(n-r)(3n-2)}{(n+r)^2} \right). \tag{4.10}$$

For a given aspect ratio $r \geqslant 1$, we have $L(r) = \lim_{n\to\infty} Q(n,r) = 12r/(1+r^2) \in (0,6]$, with $\arg\max\{L(r) : r \in [1,+\infty)\} = r^* = 1$ and $L(r^*) = 6$. We can use $L(r)$ to normalize the value $Q(n,r)$ and then obtain the normalized quality of an $m \times n$ uniform grid:

$$\tilde{Q}(n,r) = \frac{(n+r)(n+1)}{6n^2} \left(\frac{(n-1)(3n-2r)}{(n+1)^2} + \frac{(n-r)(3n-2)}{(n+r)^2} \right) \in [0,1]. \tag{4.11}$$

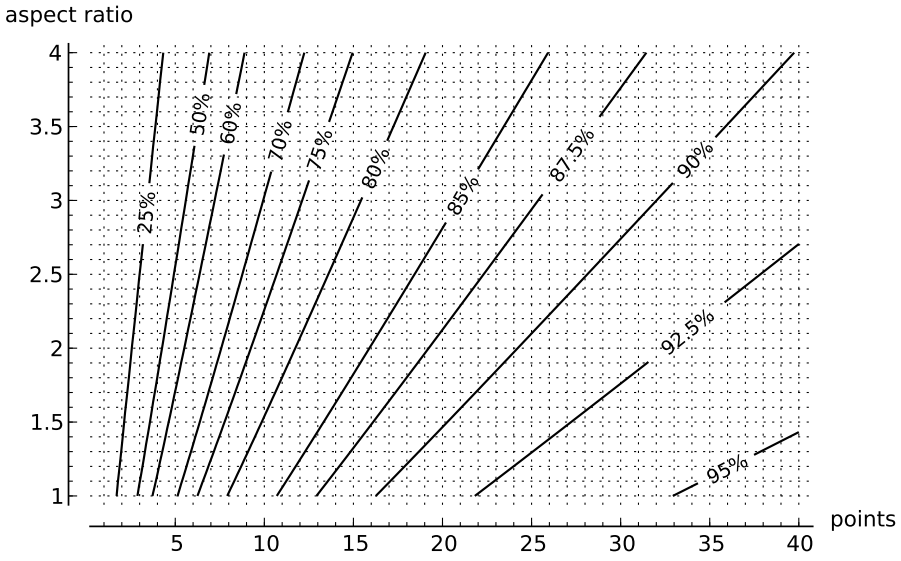


Figure 4.2: Quality of a uniform grid against its number of nodes.

Figure 4.2 shows the contour plot of $\tilde{Q}(n,r)$ for $n \in [\![1,40]\!]$ and $r \in [1,4]$. It may be used to determine the size that a uniform grid should have if there is a minimum quality to be matched. For example, a rectangle domain with aspect ration $r = 2$ should be covered by a 12×24 uniform grid to reach a normalized quality of 90%.

4.2.3 Measurement Time against Quality of a Uniform Grid

Assume that the CMM was set to collect contact point measurements at a constant rate of $v > 0$ contact points per unit of time. At such a rate, the *nm* points of the uniform grid will be measured in vt time units,i.e.

$$vt = nm = \frac{n^2}{r}. \tag{4.12}$$

It is then possible to compute the quality of a uniform grid with respect to the measurement time instead of the number of nodes on the grid. Namely, we replace the expression $\tilde{Q}(n,r)$ by $\tilde{q}(t,v,r) = \tilde{Q}(\sqrt{rvt},r)$.

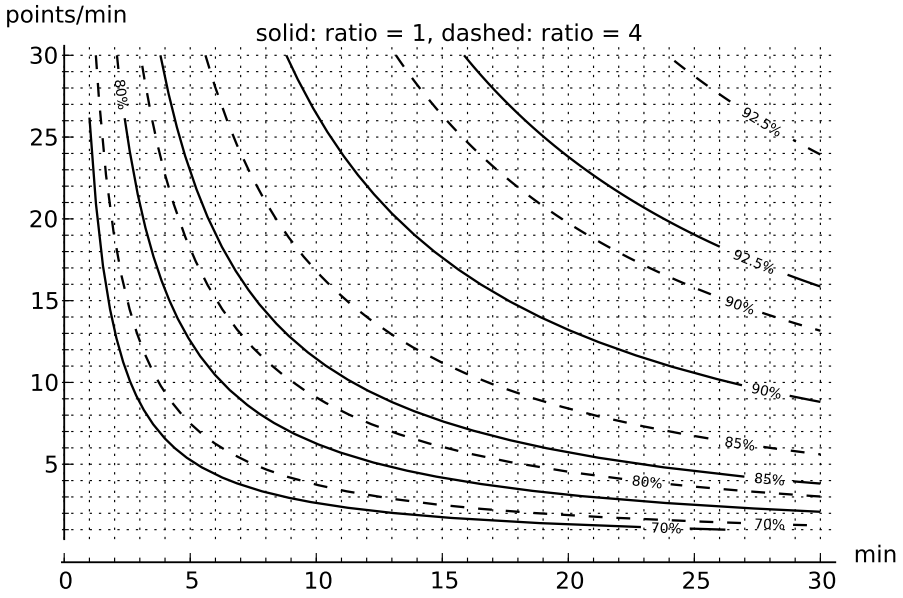


Figure 4.3: Quality of a uniform grid against against measurement time.

Figure 4.3 displays the contour plots of $\tilde{q}(t,v,r)$ for $r \in \{1,4\}$. It may be used to choose the rate v for given time and quality constraints. For example, one would need to choose a rate $v = 18$ points/min to collect measurements on a 90% quality square grid under 15 minutes. Figure 4.2 also indicates that this is a 16×16 uniform grid.

4.3 Registration and Parameter Estimation

This section covers a method of *registration*, or re-alignment, between a cloud of points and the theoretical shape it must assume. Methods of registrations have been investigated thoroughly for the past two decades in which a landmark paper by Besl and McKay [1992] was published. In this section, we are applying a few modifications to the registration approach described by Besl and McKay to serve the specificity of our problem in which we must match contact points given by a probe-type CMM to known 3-D shapes described by implicit or parametric functions. The method was modified to work with non-derivative optimization techniques and without involving approximations such as pairwise point registration. The approach was also slightly simplified to fasten computations and to allow the search of the optimal features defining the 3-D shapes (e.g. the actual radius of a sphere).

4.3.1 Input Data and Notations

Suppose we are given a set of $\ell \in \mathbb{N}^*$ vectors $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ in $\mathbb{R}^3$ which represent Cartesian coordinate measurements. We know that a surface in $\mathbb{R}^3$ can be described by an implicit function $\phi : \mathbb{R}^3 \to \mathbb{R}$ or by a parametric function $\psi : \mathbb{R}^2 \to \mathbb{R}^3$, and that the alignment problem consists into properly rotate and translate the coordinate measurements $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ such that they coincide at best with the form of the theoretical surface described by ϕ or ψ. This alignment procedure is a necessary pretreatment step for surface inspection since the measurement process is introducing a bias into the experimental coordinates of every points.

In our notations, the translation to be found is denoted by a vector $\mathbf{t} \in \mathbb{R}^3$ and the rotation is denoted by a 3×3 matrix $\mathbf{R} \in \mathscr{SO}_3(\mathbb{R})$ where $\mathscr{SO}_3(\mathbb{R})$ is the real special orthogonal group in three dimensions. In our case, we are searching the group $\mathscr{SO}_3(\mathbb{R})$ using a coordinate chart formed by the *Euler angles* $\alpha \in [0, 2\pi]$, $\beta \in [0, \pi]$ and $\gamma \in [0, 2\pi]$ that describe a product of rotations around the *z*, *x* and *z* axes. Let $\theta = (\alpha, \beta, \gamma)$ and let $\mathbf{R}_\theta$

be an element of $\mathscr{SO}_3(\mathbb{R})$ described by θ, then the general representation of $\mathbf{R}_\theta$ is given by

$$\mathbf{R}_\theta = \begin{pmatrix} c_\alpha c_\gamma - s_\alpha c_\beta s_\gamma & -c_\alpha c_\beta s_\gamma - s_\alpha c_\gamma & s_\beta c_\gamma \\ c_\alpha s_\gamma + s_\alpha c_\beta c_\gamma & c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma & -s_\beta c_\gamma \\ s_\alpha c_\beta & c_\alpha s_\beta & c_\beta \end{pmatrix}, \tag{4.13}$$

where $s_\alpha = \sin(\alpha)$, $c_\alpha = \cos(\alpha)$, etc.

4.3.2 Pre-Registration Data Treatment

The first step of the alignment procedure is to center the cloud of points $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ by subtracting to every point the centroid of the cloud defined by

$$\mathbf{x}_c = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbf{x}_i. \tag{4.14}$$

It is also possible to modify the orientation of the cloud such that the variance of the projections on each coordinate axis is maximized. The first coordinate corresponds to the greatest variance, the second coordinate corresponds to the second greatest variance, etc. In other words, it is possible to perform a complete Principal Component Analysis (PCA) [Pearson, 1901] on the cloud of points instead of a simple centering procedure. Adjusting the orientation along the axes corresponding to the maximum inertia of the cloud using a PCA may reduce the computational effort in the next steps of the alignment procedure.

Let $\mathbf{X}$ be a $\ell \times 3$ matrix such that the i-th row contains the vector $\mathbf{x}_i - \mathbf{x}_c$, then a PCA is performed by computing the 3×3 matrix $\mathbf{V}$ such that $\mathbf{U}\Sigma\mathbf{V}^{\mathrm{t}}$ is the Singular Value Decomposition (SVD) of $\mathbf{X}$. Once the SVD performed, the new data matrix is obtained by computing $\hat{\mathbf{X}} = \mathbf{X}\mathbf{V}$. Alternatively, a PCA can be realized by performing a partial SVD in which only the first three columns of $\mathbf{U}$ (stored in the $\ell \times 3$ matrix $\mathbf{U}_3$) are computed. The new data matrix is then given by computing $\hat{\mathbf{X}} = \mathbf{U}_3\Sigma$.

Nevertheless, the aim of the pretreatment phase of the alignment procedure is to make

the cloud of experimental points roughly coincide with the theoretical surface. A centering and a PCA might actually fail to do so, therefore it is often inevitable that manual translations and rotations have to be introduced in order to achieve the goal of the pretreatment phase.

4.3.3 Registration with Implicit Functions

Consider the case where the theoretical surface is described by a implicit function $\phi : \mathbb{R}^3 \to \mathbb{R}$. A point $\hat{\mathbf{x}} \in \mathbb{R}^3$ lies on the theoretical surface if it belongs to the zero set of ϕ, otherwise there exists a $r \in \mathbb{R}^*$ such that $\hat{\mathbf{x}}$ belongs to the zero set of $\phi - r$. The idea behind the alignment of the points $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_\ell$ is to find an orthonormal matrix $\mathbf{R}_\theta$ and a vector $\mathbf{t}$ that minimize simultaneously the quantities $|\phi(\mathbf{z}_i)|$, $i \in [\![1, \ell]\!]$, where $\mathbf{z}_i = \mathbf{R}_\theta \hat{\mathbf{x}}_i + \mathbf{t}$. It will then force every transformed point $\mathbf{z}_1, \ldots, \mathbf{z}_\ell$ to be at a minimum distance to the zero set of ϕ, matching the cloud of points at best with the theoretical surface. The problem is simplified by minimizing the quantity

$$\Lambda = \sum_{i=1}^{\ell} |\phi(\mathbf{z}_i)|, \tag{4.15}$$

with respect to $\theta \in [0, 2\pi] \times [0, \pi] \times [0, 2\pi]$ and $\mathbf{t} \in \mathbb{R}^3$. The domain of θ is complicating the minimization procedure since it brings inequality constraints to the optimization problem. However, this problem can actually be circumvented by extending the search of an optimal θ on the entire space $\mathbb{R}^3$. This approach is valid since we are not interested by unique chart coordinates for $\mathbf{R}_\theta$ but simply by *some* chart coordinates yielding minimum value for the objective function. This leads to the following unconstrained optimization problem

$$\min \left\{ \sum_{i=1}^{\ell} |\phi(\mathbf{z}_i)| : (\theta, \mathbf{t}) \in \mathbb{R}^6 \right\}. \tag{4.16}$$

The objective function of this problem is in most cases not convex on $\mathbb{R}^6$ and therefore the uniqueness of the optimal solution is not guaranteed. Furthermore, algorithms for convex minimization might fail to converge toward a global minimum if the pretreatment step

failed to align the cloud of points at a close proximity to the theoretical surface. Nevertheless, if we assume the convexity of the objective function on the search domain, the minimization can be performed with the help of a variety of optimization algorithms. Since ϕ might be fairly complicated, or even not differentiable, it is safer to rely on derivative-free optimization methods. Among such methods there are the method of Rosenbrock [1960], the method of Hooke and Jeeves [1961], the simplex search method [Nelder and Mead, 1964, 1965; Spendley et al., 1962] and the method of Zangwill [1967]. The preferred method is the one that requires the least amount of functional evaluations, which is the simplex search method.

In some cases the number of experimental points is very large (outputs of an optical Coordinate Measuring Machine often range in tens of thousand points). Consequently, the computational time required to perform the optimization of the quantity Λ in Equation 4.15 would be too long for practical applications of the registration method. One solution is to truncate the sum and to compute a mean value of the absolute values of the deviations from the theoretical surface. This approach is valid since that, after a certain rank, the sequence of means will converge toward a finite value. The sum might be truncated by choosing (wisely) a representative sample S of data points which is changing the previous unconstrained optimization problem into

$$\min \left\{ \frac{1}{|S|} \sum_{\mathbf{z}_i \in S} |\phi(\mathbf{z}_i)| : (\theta, \mathbf{t}) \in \mathbb{R}^6 \right\}. \tag{4.17}$$

The goodness of the solution given by the above mathematical problem depends entirely on the "goodness" of the sample S. Thankfully, several sampling techniques can be used to help accelerating the minimization procedure by providing smaller, but relevant, samples (see Sub-Section 2.1.3 and [Cochran, 1977]).

4.3.4 Registration with Parametric Functions

Consider the case where the theoretical surface is described by a parametric function $\psi : \mathbb{R}^2 \to \mathbb{R}^3$ and let the compact domain $\Omega \subset \mathbb{R}^2$ be such that the image $\psi(\Omega)$ corresponds to the theoretical surface. Furthermore let $\mathscr{P}$ be a set of $p \in \mathbb{N}^*$ nodes of a (possibly unstructured) grid of Ω and let $\mathscr{N}$ be the set of points in $\mathbb{R}^3$ that are the images of the elements of $\mathscr{P}$ by ψ. Since it is not possible to derive an implicit function ϕ from the parametric function ψ in the general case, it is necessary to derive a method for estimating the quantity Λ in Equation 4.15. The idea is to make an orthogonal projection of every point $\mathbf{z}_i$ onto the theoretical surface and to compute the distance that separates the points from their projections.

Given a point $\mathbf{z} \in \mathbb{R}^3$, we determine at first its nearest neighbor $\mathbf{n}_0 \in \mathscr{N}$. From $\mathbf{n}_0$ we obtain a vector of parameters $\mathbf{p}_0 \in \mathscr{P}$. It is recommend to perform the Nearest Neighbor Search (NNS) using a state partitioning method such as one involving a three dimensional *k*d-tree data structure [Bentley, 1975]. Using this approach, the building time complexity of the *k*d-tree (only one occurrence) is in $O(p \log p)$ and the query time complexity during the NNS is in $O(\log p)$. If the grid is coarse, then $\mathbf{n}_0$ is a poor estimate of the orthogonal projection of $\mathbf{x}$ onto the theoretical surface. Therefore, it is necessary to build a sequence of k points $\mathbf{n}_1, \ldots, \mathbf{n}_k$ onto the theoretical surface that converges toward the orthogonal projection.

Let $\mathbf{n}_i = \psi(\mathbf{p}_i)$, $i \in [\![1, k]\!]$, be an element of that sequence. First, we determine the tangent vectors $\mathbf{r}_1 = \partial_1 \psi(\mathbf{p}_i)$ and $\mathbf{r}_2 = \partial_2 \psi(\mathbf{p}_i)$ as well as the quantities $\nu_1 = \langle \mathbf{r}_1, (\mathbf{z} - \mathbf{n}_i) \rangle$ and $\nu_2 = \langle \mathbf{r}_2, (\mathbf{z} - \mathbf{n}_i) \rangle$ which are the coordinates of $\mathbf{z} - \mathbf{n}_i$ in the tangent plane at $\mathbf{n}_i$. Here, the parametric function ψ is assumed to be differentiable on Ω even though it might not be always the case. Thus it is safer to compute approximated vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ in the general

case using only ψ and not its partial derivatives. The metric tensor

$$(g_{ij}) = \begin{pmatrix} \langle \mathbf{r}_1, \mathbf{r}_1 \rangle & \langle \mathbf{r}_1, \mathbf{r}_2 \rangle \\ \langle \mathbf{r}_2, \mathbf{r}_1 \rangle & \langle \mathbf{r}_2, \mathbf{r}_2 \rangle \end{pmatrix} \tag{4.18}$$

defines the dot product on the tangent plane and provides a way to convert the quantities ν_1 and ν_2 into quantities π_1 and π_2 into the parametric space Ω. By computing $\Delta = g_{11}g_{22} - g_{12}^2$, we obtain

$$\pi_1 = \frac{g_{22}\nu_1 - g_{12}\nu_2}{\Delta}, \tag{4.19}$$

and

$$\pi_2 = \frac{g_{11}\nu_2 - g_{12}\nu_1}{\Delta}, \tag{4.20}$$

even though Δ is in practice replaced by $\Delta + \varepsilon$ where $\varepsilon > 0$ is a small numerical quantity avoiding possible divisions by zero. Then the next iteration $\mathbf{p}_{i+1} = \mathbf{p}_i + (\pi_1, \pi_2)$ is computed and the procedure is iterated until the number of iterations k is reached or until $\|(\pi_1, \pi_2)\|$ is below a certain threshold. Once $\mathbf{n}_k = \psi(\mathbf{p}_k)$ is computed, we derive its tangent vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ as well as its unit normal vector

$$\mathbf{u} = \frac{\mathbf{r}_1 \times \mathbf{r}_2}{\|\mathbf{r}_1 \times \mathbf{r}_2\|_2}. \tag{4.21}$$

Then the quantity $|\lambda| = |\langle \mathbf{u}, (\mathbf{z} - \mathbf{n}_k) \rangle|$, which is the absolute value of the projection of $\mathbf{z} - \mathbf{n}_k$ on the normal vector $\mathbf{u}$, replaces the quantity $|\phi(\mathbf{z})|$ which was to be approximated. Algorithm 4.1 implements the projection procedure.

4.3.5 Getting the Surface Parameters and Normal Deviations

If the theoretical surface is defined by a parametric function then both a set of parameters and algebraic distances from the theoretical surface are given by the registration procedure. These distances, when paired with parametric coordinates, may reveal a coherent

Algorithm 4.1: Projection onto a parametric surface

Function $[\mathbf{p}, \lambda]$ = `SurfaceProjection` $(\psi_{\mathbf{r}}, \mathbf{p}_0, \mathbf{z})$
Data: tolerance $\varepsilon > 0$.
Input: parametric function $\psi_{\mathbf{r}} : \mathbb{R}^2 \to \mathbb{R}^3$, initial parameter $\mathbf{p}_0 \in \mathbb{R}^2$, vector $\mathbf{z} \in \mathbb{R}^3$.
Output: parameter $\mathbf{p}$, altitude from surface λ.

$\mathbf{p} \leftarrow \mathbf{p}_0$
repeat
  $\mathbf{d} \leftarrow \mathbf{z} - \psi_{\mathbf{r}}(\mathbf{p})$, $\mathbf{r}_1 \leftarrow \partial_1 \psi_{\mathbf{r}}(\mathbf{p})$, $\mathbf{r}_2 \leftarrow \partial_2 \psi_{\mathbf{r}}(\mathbf{p})$
  $\nu \leftarrow \big(\langle \mathbf{r}_1, \mathbf{d} \rangle, \langle \mathbf{r}_2, \mathbf{d} \rangle \big)$, $\mathbf{g} \leftarrow \big(\|\mathbf{r}_1\|_2^2, \|\mathbf{r}_2\|_2^2, \langle \mathbf{r}_1, \mathbf{r}_2 \rangle \big)$
  $\delta \leftarrow 1/(g_1 g_2 - g_3^2 + \varepsilon)$, $\pi_1 \leftarrow \delta(g_2 \nu_1 - g_3 \nu_2)$, $\pi_2 \leftarrow \delta(g_1 \nu_2 - g_3 \nu_1)$
  $\mathbf{p} \leftarrow \mathbf{p} + (\pi_1, \pi_2)$
until $\|\pi\|_\infty \leqslant \varepsilon$
$\mathbf{r}_1 \leftarrow \partial_1 \psi_{\mathbf{r}}(\mathbf{p})$, $\mathbf{r}_2 \leftarrow \partial_2 \psi_{\mathbf{r}}(\mathbf{p})$, $\mathbf{n} \leftarrow \mathbf{r}_1 \times \mathbf{r}_2$
$\lambda \leftarrow \big\langle \mathbf{n}, \big(\mathbf{z} - \psi_{\mathbf{r}}(\mathbf{p})\big) \big\rangle / (\|\mathbf{n}\|_2 + \varepsilon)$
return $(\mathbf{p}, \lambda)$

pattern deformation due to the manufacturing process. Thus, predictive methods such as a nonparametric nonlinear regression analysis with ULMs can determine the underlying deformation rule (or function) and may generalize the shape of the deformations for similar surfaces and manufacturing processes.

In the case an implicit function is given, the gradients of ϕ at every transformed point $\mathbf{z}_i$ are computed (or approximated) and a line search is performed in order to compute the smallest $\lambda_i \in \mathbb{R}$ in absolute value such that the quantity $\big|\phi\big(\mathbf{z}_i + \lambda_i \nabla\phi(\mathbf{z}_i)\big)\big|$ is minimized. Appropriate derivative-free line searches for this task are the Golden Section Method [Kiefer, 1953] and the closely related Fibonacci Search [Avriel and Wilde, 1966]. Again, the function to be minimized might not be unimodal on $\mathbb{R}$ but it can be so on the search interval, and the starting solution might be close enough to the optimal solution so that one of the line searches will converge toward it. Once the optimal $\bar{\lambda}_i$'s are obtained, we compute the quantities $\bar{\lambda}_i \|\nabla\phi(\mathbf{z}_i)\|_2$ which are the algebraic distances from the theoretical surface. The points $\bar{\mathbf{z}}_i = \mathbf{z}_i + \bar{\lambda}_i \nabla\phi(\mathbf{z}_i)$ are all located on the level surface $\{\mathbf{z} \in \mathbb{R}^3 : \phi(\mathbf{z}) = 0\}$ which is, by definition, the nominal surface of the part.

The coordinate system for the points $\bar{\mathbf{z}}_i$ can also be changed in order to get rid of a

coordinate irrelevant for the description of the actual position of one point on the theoretical surface. If the surface has three axes of symmetry then spherical coordinates might be more appropriate. If the surface has only one axis of symmetry then cylindrical coordinates could be more suitable, etc. Many orthogonal coordinate systems in three dimensions can be tried in the attempt to reduce the dimensionality of the experimental points. If these basic transformation methods fail, then other more elaborate nonlinear dimensionality reduction methods can be successively tried to cancel one extra dimension. Among these nonlinear methods we find: the Kernel Principal Component Analysis (KPCA) [Diamantaras and Kung, 1996; Schölkopf et al., 1997; Mika et al., 1999]; Principal curves and manifolds [Hastie and Stuetzle, 1989]; Gaussian Process Latent Variable Models (GPLVM) [Tipping and Bishop, 1999]; Locally Linear Embedding (LLE) [Roweis and Saul, 2000]; Autoencoders [Hinton and Salakhutdinov, 2006]; and Self-Organizing Maps (SOM) [Kohonen, 1982, 2001; Kohonen and Mäkisara, 1986].

Alternatively, there exists a computationally intensive method that can associate a pair of parameters (p_1, p_2) in $\mathbb{R}^2$ to any point $\bar{\mathbf{z}}$ of the level surface $\{\mathbf{z} \in \mathbb{R}^3 : \phi(\mathbf{z}) = 0\}$. Assume that an arbitrary point $\bar{\mathbf{z}}_0$ is chosen as the origin. The unit vector

$$\mathbf{w} = \frac{\nabla\phi(\bar{\mathbf{z}}_0)}{\|\nabla\phi(\bar{\mathbf{z}}_0)\|_2} \tag{4.22}$$

is normal to the level surface and we may choose an arbitrary unit vector $\mathbf{u} \in \mathbb{R}^3$ such that $\langle \mathbf{u}, \mathbf{w} \rangle = 0$ and then define the unique unit vector $\mathbf{v} \in \mathbb{R}^3$ by $\mathbf{v} = \mathbf{w} \times \mathbf{u}$ so we have a unit basis $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ in $\mathbb{R}^3$. The parameter $\mathbf{p}_0 \in \mathbb{R}^2$ associated to $\bar{\mathbf{z}}_0$ is arbitrarily set equal to $\mathbf{0}$. For any point $\bar{z} \in \mathbb{R}^3$ such that $\phi(\bar{\mathbf{z}}) = 0$, $\langle \bar{\mathbf{z}}, \mathbf{u} \rangle \neq 0$ and $\langle \bar{\mathbf{z}}, \mathbf{v} \rangle \neq 0$, we define a sequence of $m \geqslant 1$ points $\mathbf{a}_1, \ldots, \mathbf{a}_m$ such that

$$\mathbf{a}_i = \bar{\mathbf{z}}_0 + \frac{i}{m+1}(\bar{\mathbf{z}} - \bar{\mathbf{z}}_0), \tag{4.23}$$

for all i in $[\![1,m]\!]$. The points $\mathbf{a}_1,\ldots,\mathbf{a}_m$ belong to the unique line in $\mathbb{R}^3$ that goes through $\bar{\mathbf{z}}_0$ and $\bar{\mathbf{z}}$. For each $\mathbf{a}_i$, we use a line search to find the smallest $\lambda_i \in \mathbb{R}$ in absolute value such that the quantity $|\phi(\mathbf{a}_i + \lambda_i \nabla\phi(\mathbf{a}_i))|$ is minimized. Once the optimal $\bar{\lambda}_i$'s are determined, we compute the points $\bar{\mathbf{a}}_1,\ldots,\bar{\mathbf{a}}_m$ such that

$$\bar{\mathbf{a}}_i = \mathbf{a}_i + \bar{\lambda}_i \nabla\phi(\mathbf{a}_i). \tag{4.24}$$

If we use the convention that $\bar{\mathbf{a}}_0 = \bar{\mathbf{z}}_0$ and $\bar{\mathbf{a}}_{m+1} = \bar{\mathbf{z}}$, then we define the parameter $p_1 \in \mathbb{R}$ associated to $\bar{\mathbf{z}}$ as the linear approximation of the shortest path on the level surface between $\bar{\mathbf{z}}_0$ and $\bar{\mathbf{z}}$, i.e.

$$p_1 = \frac{1}{\|\nabla\phi(\bar{\mathbf{z}}_0)\|_2} \sum_{i=1}^{m+1} \|\bar{\mathbf{a}}_i - \bar{\mathbf{a}}_{i-1}\|_2 . \tag{4.25}$$

The second parameter $p_2 \in \mathbb{R}$ is defined as the angle $\theta \in (-\pi,\pi]$ such that

$$\theta = \operatorname{atan2}\big(\langle \mathbf{v}, (\bar{\mathbf{a}}_1 - \bar{\mathbf{a}}_0)\rangle, \langle \mathbf{u}, (\bar{\mathbf{a}}_1 - \bar{\mathbf{a}}_0)\rangle\big), \tag{4.26}$$

where

$$\operatorname{atan2} : \mathbb{R}^2 \to \mathbb{R},\ (y,x) \mapsto \begin{cases} \arctan(y/x) & \text{if } x > 0 \\ \pi + \arctan(y/x) & \text{if } x < 0,\ y \geqslant 0 \\ -\pi + \arctan(y/x) & \text{if } x < 0,\ y < 0 \\ \pi/2 & \text{if } x = 0,\ y > 0 \\ -\pi/2 & \text{if } x = 0,\ y < 0 \\ \text{undefined} & \text{if } x = 0,\ y = 0. \end{cases} \tag{4.27}$$

4.3.6 Determination of the Features of Solids

Some particular surfaces can be defined with respect to predetermined parameters. For example, a torus with tube radius $r \in \mathbb{R}_+^*$ and distance from the center of the tube to the

origin $R \in (0,r)$ can be defined with a parametric function

$$\psi : (u,v) \in \Omega \mapsto \begin{pmatrix} (R + r\cos v)\cos u \\ (R + r\cos v)\sin u \\ r\sin v \end{pmatrix} \in \mathbb{R}^3, \tag{4.28}$$

where $\Omega = [0,2\pi] \times [0,2\pi] \subset \mathbb{R}^2$, or it can be defined with an implicit function

$$\phi : (x,y,z) \in \mathbb{R}^3 \mapsto \left(R - \sqrt{x^2+y^2}\right)^2 + z^2 - r^2 \in \mathbb{R}. \tag{4.29}$$

Here, the predetermined parameters of the torus are r and R and, even though they are presumed to be initially known, the manufactured solid may have slightly different values for r and R, on top of having unavoidable mechanical deformations on its surface. Thus, it is necessary to determined approximated values of these parameters before deriving the algebraic distances mentioned in the previous subsection. To do so the whole alignment methodology described in Sub-Sections 4.3.3 and 4.3.4 will now be considered as sub-step of an optimization procedure.

Consider an implicit function $\phi(\mathbf{r},\cdot)$ parameterized by $\mathbf{r} \in \mathbb{R}^p$ and let F be a functional such that

$$F(\phi)(\mathbf{r}) = \min\left\{ \sum_{i=1}^{\ell} |\phi(\mathbf{r},\mathbf{z}_i)| : (\theta,\mathbf{t}) \in \mathbb{R}^6 \right\}. \tag{4.30}$$

Then the optimal vector of parameters $\bar{\mathbf{r}} \in \mathbb{R}^p$ is obtained by solving the following minimization problem

$$\min\{F(\phi)(\mathbf{r}) : \mathbf{r} \in \mathbb{R}^p\}. \tag{4.31}$$

If we assume that the objective function $F(\phi)$ is convex on the search domain, then traditional multidimensional convex minimization methods can be used. For example, the Simplex Search mentioned in Sub-Section 4.3.3 fits perfectly this task. If the theoretical surface is parameterized by a single parameter, then a simple line search like the Golden

Section Method or the Fibonacci Search can be performed instead. If the theoretical surface is defined by a parametric function $\psi(\mathbf{r},\cdot)$ parameterized by $\mathbf{r}\in\mathbb{R}^p$ then the objective function is modified according to the methodology introduced in Sub-Section 4.3.4.

4.4 Nonlinear Regression and Minimum Zone Estimation with ULMs

At the end of Section 4.3, all coordinate measurements $\mathbf{x}_1,\ldots,\mathbf{x}_\ell$ in $\mathbb{R}^3$ are assumed to be reduced into pairs $(\mathbf{p}_1,\lambda_1),\ldots,(\mathbf{p}_\ell,\lambda_\ell)$ in $\Omega\times\mathbb{R}$ where Ω is a compact subset of $\mathbb{R}^2$. The vectors $\mathbf{p}_1,\ldots,\mathbf{p}_\ell$ are parametric coordinates (observations) associated to altitudes $\lambda_1,\ldots,\lambda_\ell$ (targets).

The aim is to find a deformation pattern $f:\mathbb{R}^2\to\mathbb{R}$ that links observations to targets and this problem perfectly fits the task ULMs have been created for. By selecting an appropriate kernel for numerical data (see Sub-Section 2.2.4), we can repeatedly test and validate models for the deformation pattern (see Sub-Section 2.4.3). Once the best model has been validated, we have a pattern estimate $\hat{f}:\Omega\mapsto\mathbb{R}$ that fits the data with a tolerance $\varepsilon_\alpha>0$ at a confidence level $\alpha\in(0,1)$. In other words, we have

$$\left|\hat{f}(\mathbf{p})-\lambda\right|\leqslant\varepsilon_\alpha, \tag{4.32}$$

for a tolerance $\varepsilon_\alpha>0$ defined by an arbitrary $\alpha\in(0,1)$, and for all $\mathbf{p}\in\Omega$ and their associated deformation altitudes $\lambda\in\mathbb{R}$ (which are unknown *almost everywhere* since we only have pointwise measurements). Consequently, the size M_α of the minimum zone, with confidence level $\alpha\in(0,1)$, is defined as:

$$M_\alpha=\max\{\hat{f}(\mathbf{p}):\mathbf{p}\in\Omega\}-\min\{\hat{f}(\mathbf{p}):\mathbf{p}\in\Omega\}+2\varepsilon_\alpha. \tag{4.33}$$

The optimization of $\hat{f}$ has to be done, or exhaustively around the extrema, or by using an optimization meta-heuristic due to the fact that $\hat{f}$ is never convex in the general

case. Meta-heuristics can involve: genetic algorithms [Holland, 1975], simulated annealing [Kirkpatrick et al., 1983; Černý, 1985], tabu search [Glover, 1989, 1990], ant colony optimization [Dorigo, 1992], particle swarm optimization [Kennedy and Eberhart, 1995], etc.

4.5 Applications

The processing of contact point measurements was made with a DELL Precision Workstation 530 equipped with two 2.4 GHz Intel Xeon processors and 2 GiB of RAM. ULMs and registration codes were developed under MATLAB 7.4. Surfaces were inspected with a Brown & Sharpe MicroVal PFx™ 454 CMM equipped with a touch trigger probe head. This CMM has a linear displacement accuracy of 5.1 µm along each axis and a measurement repeatability of 3.8 µm. The touch trigger probe head is a Renishaw PH9/PH10 manual probe head capable of holding M2 and M3 styli. The Renishaw M2 stylus ref. A-5003-0577 was used throughout all the experiments. It has a ⌀ 0.7 mm ruby ball, a ⌀ 0.5 mm × 20 mm tungsten carbide stem and a mass of 0.32 g.

Different parts were manufactured: face-milled plates, an half cylinder, an half sphere, an half torus and a cone. The parts were probed by the CMM and the contact points were registered according to the method described in Section 4.3. During the registration procedure, the actual parameters of the nominal surfaces (sphere diameter, cylinder diameter, etc.) were successfully obtained. Once contact points were correctly registered, ULMs were used to recover deformation patterns on the surface of each part and the minimum zones were computed according to the approach discussed in Section 4.4.

4.5.1 Half Cylinder Deformations

One half cylinder was produced using an end milling operation that used the following cutting parameters:

- Workpiece: Aluminum 7075-T6, ⌀ 76.2 mm, 34.29 mm length.
- Tool: ⌀ 12.7 mm ball nose end mill with a High Speed Steel (HSS) cutter.
- Machining conditions: step over = 0.635 mm.

This half cylinder was inspected with a M2 stylus ref. A-5003-0577 using an uniform grid mesh of 630 contact points. Figure 4.4 shows the longitudinal deformations after re-alignment and determination of the actual diameter of the part.

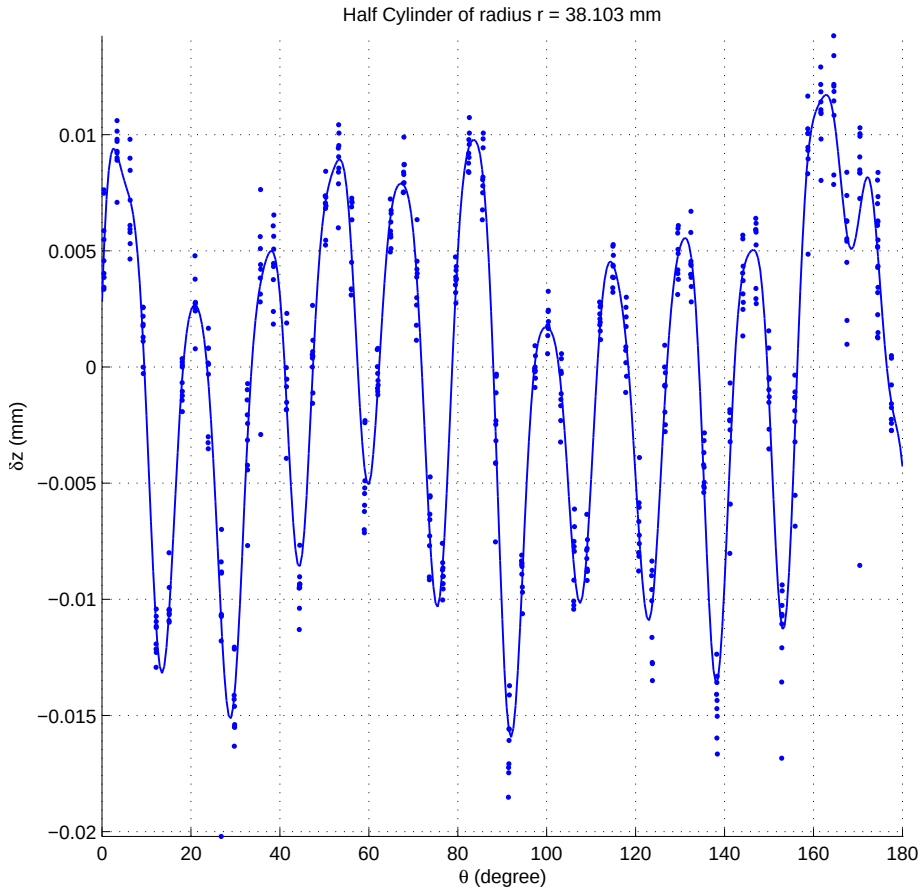


Figure 4.4: Longitudinal deformations of the half cylinder.

The final kernel used for the regression the deformation shape is a Gaussian RBF kernel with parameter σ set equal to 5.17. No significant vertical deformations were found, and the deformation form in Figure 4.4 is represented with respect to the coordinate angle $\theta \in [0, \pi]$. The actual diameter of the part was found to be slightly greater than desired one of 6

µm. The minimum zone is estimated to be 35 µm. The longitudinal wave can be interpreted by the movement of the cutter during the machining process where the cutter moves linearly between reference points that belong to the ideal shape. The real trajectory of the cutter is therefore a chain of small linear trajectories that closely matches the ideal cylindrical form (piecewise approximation) resulting in a wave pattern when being projected on Figure 4.4.

4.5.2 Half Sphere Deformations

One half ⌀ 63.5 mm sphere was manufactured and inspected with a sample of 256 contact points following a Hammersley distribution. A M2 stylus ref. A-5003-0577 was used for recording the measurements. After re-alignment the actual diameter of the half sphere was found to be 378 µm larger than the one desired and the minimum zone was estimated to be 70 µm. The interpolated deformation form is shown in Figure 4.5 and it displays the deformation according to the coordinate system represented on the left side on the figure.

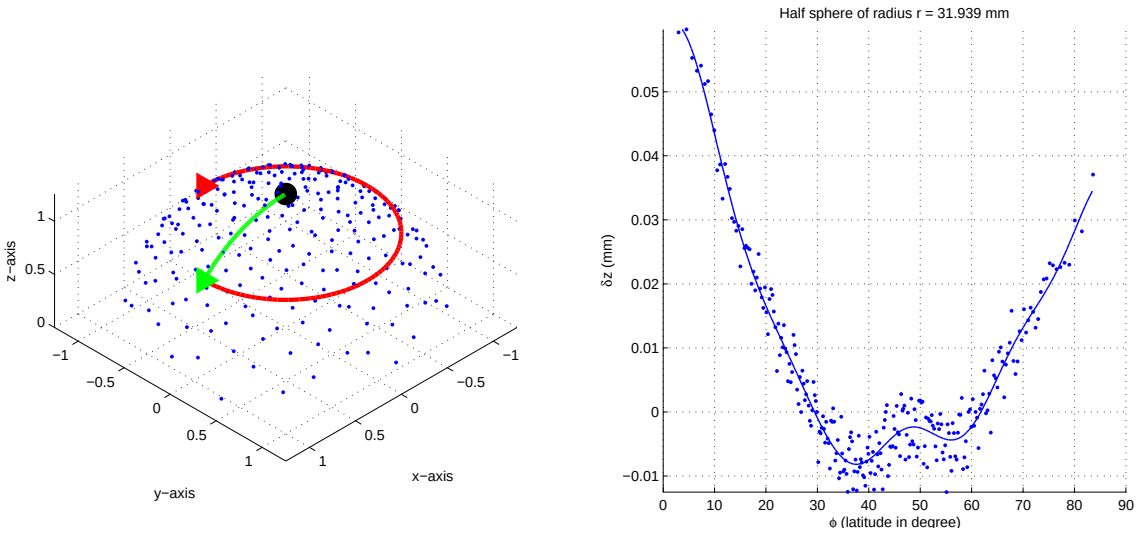


Figure 4.5: Deformations of the half sphere.

The final kernel used for the interpolation of the deformation shape is a Gaussian RBF kernel with parameter σ set equal to 6.30. Deformations were significant with respect to a single parameter, namely the latitude $\phi \in [0, \pi/2]$ of the contact point. Deviations from the nominal surface appear to occur at places where the angle between the cutter and the

contact surface is important like the top and the base of the half sphere.

4.5.3 Half Torus Deformations

One ⌀ 63.5 mm half torus of tube diameter 38.1 mm and was manufactured and inspected with a sample of 256 contact points following a Hammersley distribution. A M2 stylus ref. A-5003-0577 was used for recording the measurements. After registration, the actual diameter of the half torus was found to be 92 µm larger than the one desired and the tube diameter 68 µm smaller than expected. The minimum zone was estimated to be 150 µm with the largest deviations being inside the torus hole. The interpolated deformation form is shown in Figure 4.6 and it shows the deformations according to the coordinate system represented on the left side of the figure.

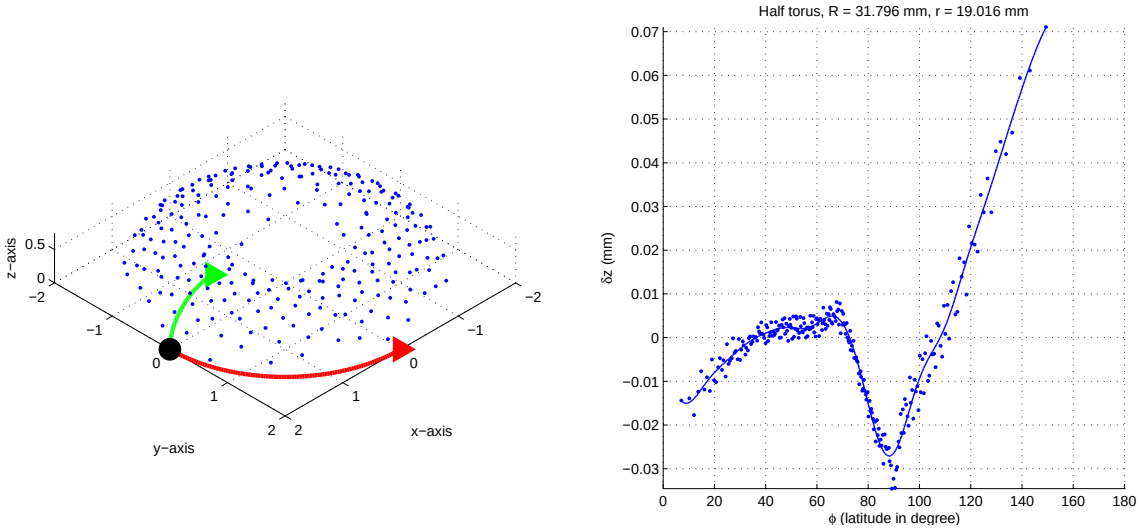


Figure 4.6: Deformations of the half torus.

The final kernel used for the interpolation of the deformation shape is a Gaussian RBF kernel with parameter σ set equal to 5.73. Deformations were significant with respect to a single parameter, namely the latitude $\phi \in [0, \pi]$ of the contact point. Deviations from the nominal surface, just like for the half sphere, appear to occur at places where the angle between the cutter and the contact surface is important like the inside hole of the torus.

4.5.4 Cone Deformations

One 63.5 mm high aluminum cone with a ⌀ 76.2 mm base was manufactured using a taper turning process and it was inspected with samples of 8, 64, and 256 data points. The surface contact point were sampled using a Hammersley distribution and recorded with a M2 stylus ref. A-5003-0577. After registration, the actual aperture of the cone was found to be 0.899°smaller than the one desired. The minimum zone was estimated to be 230 µm with the largest deviations being a the tip of the cone. The interpolated deformation form is shown in Figure 4.7 and it shows the deformations according to the coordinate system represented on the left side of the figure.

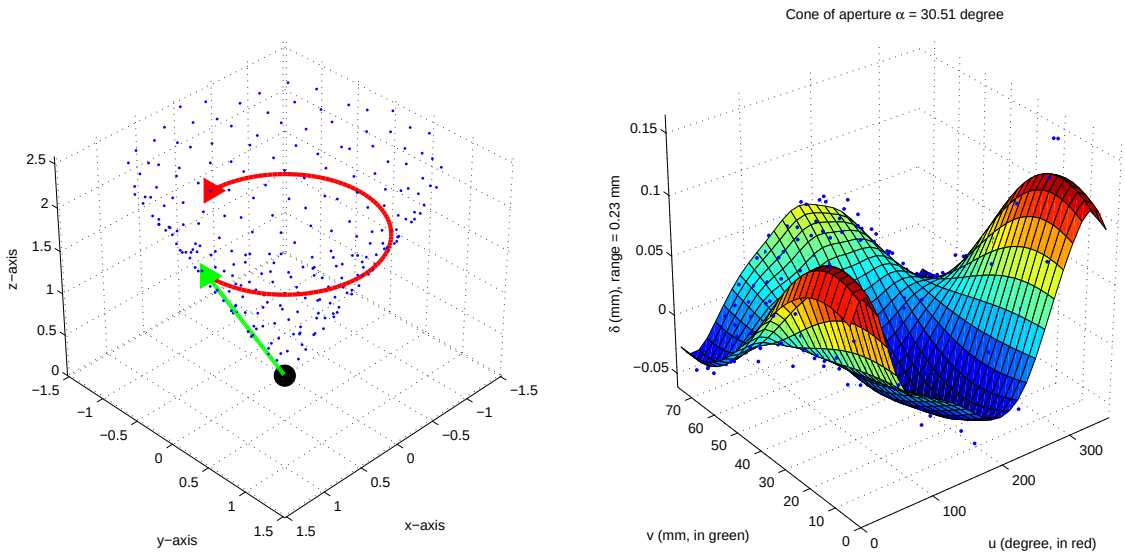


Figure 4.7: Deformations of the cone.

The final kernel used for the regression the deformation shape is a Gaussian RBF kernel with parameter σ set equal to 25.15. Deformations were significant with respect to the two parameters describing the cone surface, namely the latitude $\phi \in [0, 2\pi]$ and the distance from the tip. Extreme deviations from the nominal surface appear where the amount of matter pressing against the cutting tool is minimal. It makes the tip of the cone appear like

a chunk of matter is missing on one side or has been pushed on the other side.

4.5.5 Face-Milled Plates Deformations

Two batches of four and five face-milled plates were produced for this experimentation. The cutting parameters for the plates are as follows:

- Workpiece: Aluminum 6061-T6, 101.6 mm × 101.6 mm × 12.7 mm.
- Tool: ⌀ 76.2 mm cutter, 7 inserts with carbide coating.
- Machining conditions: coolant, cutter speed = 750 rpm, step depth = 0.254 mm, cutting feed = 25.4 mm/s (first batch) and 29.6 mm/s (second batch).

The plates were labeled from 1-1 to 1-4 for the first batch, and from 2-1 to 2-5 for the second batch. The plates were visually inspected before being measured and plates 1-1 and 1-2 were discarded due to faulty machining. The probe-type CMM did not produce significant quantities of measurements in a short period of time, thus the data sets rarely exceed 300 points and the computational times are extremely small. The contact points were chosen such as they form a uniform mesh on the surface of the plates to maximize the surface information on the whole plate. Hence, the CMM traced a zig-zag pattern at a fast pace when taking measurements. This relatively speedy measurements allowed for different mesh densities to be tested for the same plates. The results on different meshes were compared to verify the integrity of the registration of surface model validation procedures.

A test run was made on plate 2-2 with a total of 293 measurements. The parameter σ of the Gaussian RBF kernel, as well as the parameters of surface model were tuned until the residuals were small enough and deemed to be random and uncorrelated. Results of the ULM-based nonlinear regressions can be seen in Figure 4.8 where the deviation surface has the shape of a saddle.

All the plates that were inspected have similar deformation surfaces. These surfaces are saddle-shaped with the “valley” part oriented along the direction of the cutter pass. Addi-

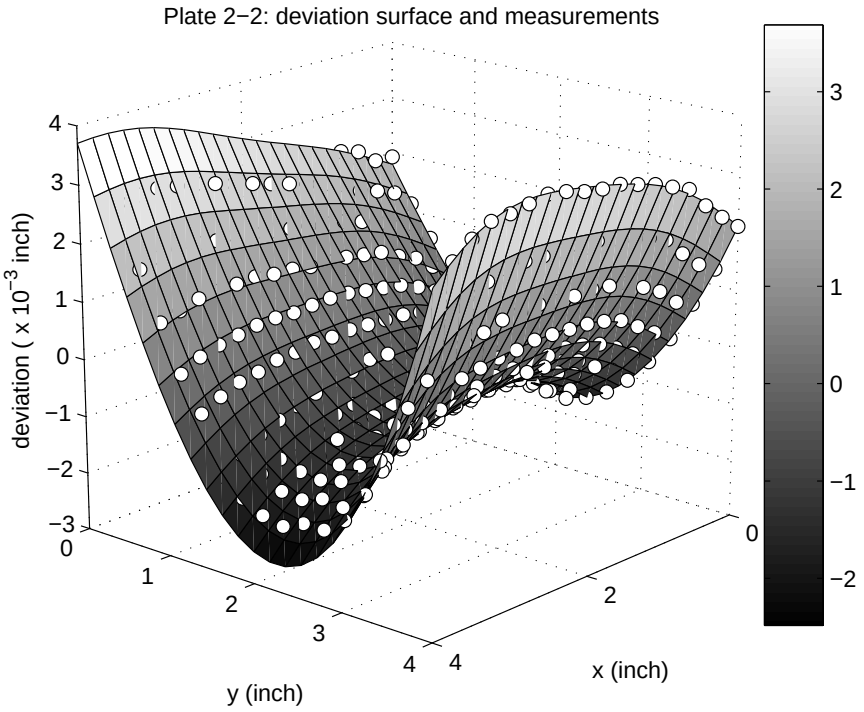


Figure 4.8: Deformations of plate 2-2.

tional tests on plates manufactured with different processes will confirm if it is a general behavior or if it is an artifact produced by the particular machining conditions. If the shape is found to be general, then it will be a precious hint for inspecting the plates with optimized meshes since, on a saddle, the extrema are located on the sides.

On seven plates, four were inspected with different meshes. Using the minimum zone formulation of Section 4.4, the estimates M_α, with $\alpha = 0.99$, of the minimum zones of the tested plates are shown in Table 4.1.

Table 4.1: Minimum zones of four face-milled plates.

Plate reference	Number of points	Minimum zone
1-3	348	57.9 µm
1-4	81	18.3 µm
2-2	293	166.9 µm
2-3	64	19.3 µm

The minimum zones for plates 1-4 and 2-3 were consistent with their visual aspect. Their surfaces appeared to be smooth and better finished, and their minimum zones were estimated to be 2.4 times bigger than the accuracy of the CMM probe. At this scale, the impact of artifacts are not negligible with regard to the deformations of the surface as it is illustrated in Figure 4.9.

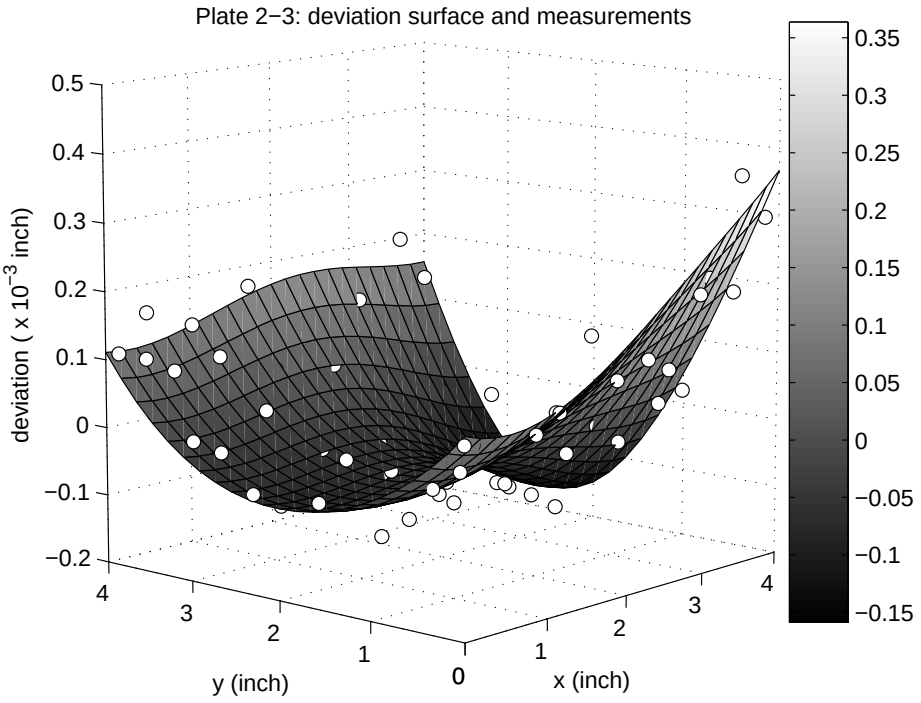


Figure 4.9: Deformations of plate 2-3.

The artifacts of plate 2-3 were generated by random perturbations (vibrations, etc..) and they do not represent actual deformations of the surface. The choice of a Gaussian RBF kernel was an important factor to the robustness of the surface model and it gave consistent results during the validation procedure. This show that carefully selected models can handle noisy observations at scales which are few times larger than the probe accuracy.

Different meshes were also tested on the same plates to assess the consistency of the chosen models. Tests performed on plate 1-3 are reproduced in Table 4.2. They indicate the estimated minimum zones (second column) with respect to the number of nodes of the

mesh (first column).

Table 4.2: Minimum zones against the number of nodes of a mesh.

Number of points	Minimum zone
36	58.7 µm
64	61.2 µm
81	56.4 µm
138	61.0 µm
174	60.2 µm
202	61.7 µm
219	58.4 µm
283	58.4 µm
348	57.9 µm

No significant effects were found between the density of the mesh and the estimated minimum zones for the four tested plates. All the meshes had similar estimated minimum zones regardless of the number of nodes of the mesh (all experiments were made with meshes of at least 36 nodes). Thus, if this behavior is found to be general, then relatively small uniform meshes could be used for the flatness inspection of face-milled plates since such meshes reduce time-induced errors during measurements.

Finally, in order to test the robustness to perturbations of the Gaussian RBF kernel, measurements were made on a 76.2 mm × 76.2 mm portion of an optical flat (⌀ 127 mm Lapmaster optical flat) for which the accuracy is certified to $\frac{1}{10}$-th light band. This accuracy is much higher than the probe accuracy of the CMM (which is ± 9 µm on the position of a particular point along each axis), hence deviations from the nominal plane obtained by ULMs are only artifacts and not actual deviations. If the approach is robust to artifacts, then the interpolated deformation surface should be a plane passing by the coordinate origin and parallel to the canonical *xOy* plane.

The deformation surface shown in Figure 4.10 appears to be completely flat and contained within a cloud of artifacts. Additional numerical tests showed it to be true. This test seems to confirm the immunity of the surface model against random variations.

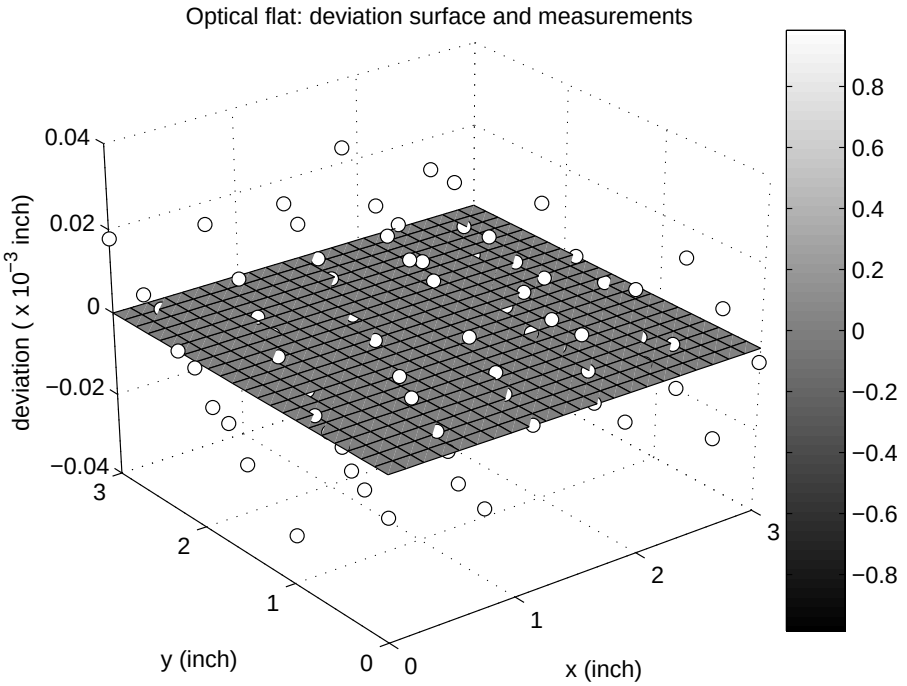


Figure 4.10: Deformations of the Lapmaster optical flat.

Chapter 5

State Forecasts of a Weather System

5.1 Introduction, Context and Aims

Kalman Filters (KF) have been traditional approaches to data assimilation in the geosciences for the past few decades [Kalman and Bucy, 1961]. KFs have been constantly improved to better fit their application requirements and circumvent their implementation issues. Nevertheless, these approaches were never able to overcome their unrealistic assumptions (e.g. linearity, multivariate normality, prior knowledge of the model, error covariances properties, etc.), and their scalability issues when handling large sets of data. The inability to efficiently process large amounts of data is a significant drawback of KFs since, at any instant, huge sets of data are always available in geosciences. Furthermore, the size of the data to be processed is growing at an ever-increasing rate. Therefore, the data processing technology should be rendered capable of analyzing the huge amounts of information to come. Fortunately, the rapid discovery of patterns within data can be done with a clever use of state-of-the-science supervised machine learning techniques such as ULMs.

The analysis of weather systems require a framework capable to assimilate temporal patterns in vector fields outputs and make accurate predictions regarding the future states of these fields. The assimilation part of the analysis of vector fields also requires the ability to remove noise within observations and to properly guess state trajectories, even with a partial lack of knowledge on the way data were generated. This lead to consider several objectives for ULM-based techniques for weather system analysis:

- develop robust near real-time procedures based on ULMs to assimilate state trajectories of the weather systems;
- provide fast "zero-knowledge" prediction schemes as helping tools for traditional, but computationally intensive, techniques such as KFs.
- exploit geometric characteristics of observation sets to thin data and make assimilation procedures scalable (see Sub-Section 2.1.3);
- automate model validation procedures in a weather system context.

While some objectives will be the subject of future work, the first two points of the list have been investigated in this dissertation. The ULM-based robust assimilation procedure is described in Sub-Section 5.2.1 and the "zero-knowledge" predictive analysis in Sub-Section 5.2.2. Additionally, simulations were run on simplified weather models: the Lorenz 96 model (Section 5.3), and the Quasi-Geostrophic model (Section 5.4).

5.2 Assimilation and State Predictions

5.2.1 Assimilation using ULMs

Consider an FCD system represented by the 5-tuple (T, U, x, f, g) (see Chapter C). The domain T of the system is (t_0, t_∞) and the input space U is the set of all constant functions over T with values in Ω, where Ω is a compact domain of $\mathbb{R}^m$, with $1 \leqslant m \leqslant 3$. In other words, for all $t \in (t_0, t_\infty)$, we have $u(t) = \mathbf{u} \in \Omega \subset \mathbb{R}^m$. The state function $x : T \to \mathbb{R}^n$ is such that

$$\mathrm{d}x(t) = f\big(x(t), \mathbf{u}, t\big), \tag{5.1}$$

where f is continuously differentiable. The output function g is defined over $\mathbb{R}^n \times \Omega \times T$ with values in $\mathbb{R}^p$.

Given a location $\mathbf{u} \in \Omega$, we assume that there exists ℓ increasing *observation* instants $t_1 < t_2 < \ldots < t_\ell$ in T such that we have ℓ *targets* outputs $\mathbf{y}_1, \ldots, \mathbf{y}_\ell$ in $\mathbb{R}^p$ defined by

$$\mathbf{y}_i = g\big(x(t_i), \mathbf{u}, t_i\big), \tag{5.2}$$

for all $i \in [\![1, \ell]\!]$. Without loss of generality, the output function g is deemed to be *known* such that it is possible to identify its outputs with the system states. Namely, we have $p = n$ and

$$\mathbf{y}_i = x(t_{i-1}) + \int_{t_{i-1}}^{t_i} f\big(x(t), \mathbf{u}, t\big)dt + \varepsilon_i = x(t_i) + \varepsilon_i, \tag{5.3}$$

where $\varepsilon_i \in \mathbb{R}^n$ is an *unknown* perturbation. The state-transition function f is deemed *unknown* (unlike in the KF framework) and it is not to be recovered by ULM-based approaches. However, ULM-based robust regression approaches are used in this context to estimate n functions $\hat{F}_1, \ldots, \hat{F}_n$ such that

$$\big|(\mathbf{y}_i)_j - \hat{F}_j(t_i)\big| \leqslant (\varepsilon_i)_j, \tag{5.4}$$

for all $(i, j) \in [\![1, \ell]\!] \times [\![1, n]\!]$. In fact, ULM-based approaches are used to recover the functions $\hat{F}_1, \ldots, \hat{F}_n$ at a given location $\mathbf{u} \in \Omega$ such that

$$\big|(\mathbf{y})_j - \hat{F}_j(t)\big| \leqslant (\varepsilon)_j, \tag{5.5}$$

for all $t \in [t_1, t_\ell]$ and $j \in [\![1, n]\!]$ where ε is an arbitrary perturbation dependent of a certain confidence level. Naturally, this approach is valid if we consider weather system to be non-Markovian since ULMs are searching for dependencies between states taken at the same location but at different times. This assimilation step is then repeated for all discrete locations $\mathbf{u}_1, \ldots, \mathbf{u}_q$ in Ω. The results of this analysis by ULM-based regression techniques are smooth interpolated state trajectories at each discrete location of a compact domain in

$\mathbb{R}^m$, with $1 \leqslant m \leqslant 3$.

5.2.2 Predictive Analysis

The predictive analysis attempts to give correct estimates of the system states $x(t)$ for $t \in (t_\ell, t_\infty)$ and for all possible locations in Ω. Unfortunately, this step cannot be done with great accuracy if the state-transition function f is unknown. Furthermore, attempts to estimate f would require to recover state trajectories at a fairly large number of different locations in the domain Ω, and assume that perturbations were all successfully removed during the assimilation step. Additionally, the upper bounds on the generalization errors of the estimates of the state trajectories $\hat{F}_1, \ldots, \hat{F}_n$ increase drastically as the time t moves away from t_ℓ (see Sub-Section 2.3.4). Hence, $\hat{F}_1(t), \ldots, \hat{F}_n(t)$, with $t \in (t_\ell, t_\infty)$, are not necessarily good estimates of the future system states, excepted when t is "close" to t_ℓ.

Hence, there are two possible ways to extrapolate the state trajectories of a weather system at a location $\mathbf{u} \in \Omega$ when the time t is in the neighborhood of t_ℓ:

- Compute the estimates $\hat{F}_1(t), \ldots, \hat{F}_n(t)$ for $t > t_\ell$.
- Compute a polynomial extrapolation.

The polynomial extrapolation is an extrapolation method that attempts to alleviate the generalization error of the state trajectories estimates outside the time domain $[t_1, t_\ell]$. To perform such an extrapolation, we choose a time window $[t_\ell - \delta t, t_\ell]$ of size $\delta t > 0$ in which $q \geqslant 2$ Chebyshev nodes[1] $\tilde{t}_1, \ldots, \tilde{t}_q$ are computed. Then, for all $(i, j) \in [\![1, q]\!] \times [\![1, n]\!]$, we compute the outputs

$$(\tilde{\mathbf{y}}_i)_j = \hat{F}_j(\tilde{t}_i). \tag{5.6}$$

[1] The $q \geqslant 2$ Chebyshev nodes on the interval $[t_\ell - \delta t, t_\ell]$ are defined by

$$\tilde{t}_i = t_\ell + \delta t \big(\cos\big((2i-1)\pi/(2q)\big) - 1\big)/2,$$

for all $i \in [\![1, q]\!]$. Interpolation polynomials built upon those nodes are minimizing Runge's phenomenon.

Then, for all $j \in [\![1,n]\!]$, the pairs $\left(\tilde{t}_1,(\tilde{\mathbf{y}}_1)_j\right),\ldots,\left(\tilde{t}_q,(\tilde{\mathbf{y}}_q)_j\right)$ are used to compute the j-th Lagrange polynomial L_j defined by

$$L_j(t) = \sum_{i=1}^{q}\left((\tilde{\mathbf{y}}_i)_j \prod_{\substack{k=1\\k\neq i}}^{q}\frac{t-t_k}{t_i-t_k}\right). \tag{5.7}$$

The state estimates for $t > t_\ell$ are then given by $L_1(t),\ldots,L_n(t)$.

5.2.3 Summary

The assimilation and prediction of the states of a weather system is accomplished as follows (see Figure 5.1):

1. One location $\mathbf{u}$ in the domain Ω is selected. This location is usually the position of a node on a grid in $\mathbb{R}^2$ or $\mathbb{R}^3$.

2. There are n feature for the state of the system at the location $\mathbf{u}$. Hence, the j-th feature, with $j \in [\![1,n]\!]$, is selected.

3. For the specific feature that was selected, we collect q observations $\left(\tilde{t}_1,(\tilde{\mathbf{y}}_1)_j\right),\ldots,\left(\tilde{t}_q,(\tilde{\mathbf{y}}_q)_j\right)$ which will be used during the assimilation step.

4. An ULM-based robust regression approach (see Sub-Section 3.3.3) assimilate the past q observations of the j-th feature of the system state at location $\mathbf{u}$.

5. A polynomial extrapolation estimate the future values $L_1(t)$ to $L_n(t)$, with $t > t_\ell$, of the j-th feature of the system state at location $\mathbf{u}$ (see Sub-Section 5.2.2).

6. The next state feature is selected and the procedure loops back to point 2. If no feature is left then the procedure continues to point 6.

7. The next location on the grid is selected and the procedure loops back to point 1. If no nodes are left on the grid then the procedure stops.

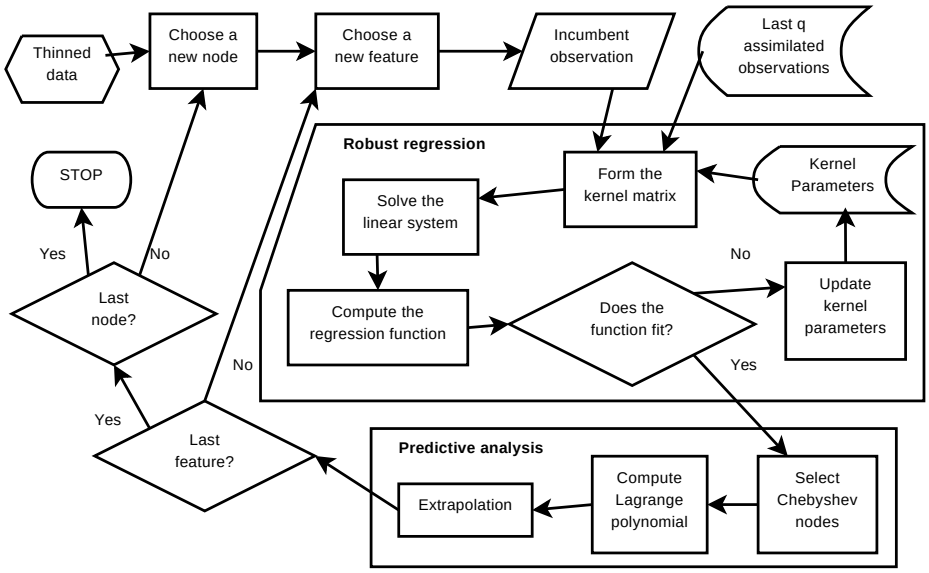


Figure 5.1: Simplified outline of the ULM-based assimilation scheme combined with a polynomial predictive analysis.

5.3 Lorenz 96 Model

The Lorenz 96 model is representing the values of atmospheric quantities at discrete locations spaced equally on a latitude circle (i.e. it is a 1-D problem). The system state at the i-th location on the latitude circle is noted by $x_i \in \mathbb{R}$, with $i \in [\![1, l]\!]$. The state transition model at a location i on the latitude circle is

$$\mathrm{d}x_i = \underbrace{(x_{i+1} - x_{i-2})x_{i-1}}_{\text{advection}} \underbrace{-x_i}_{\text{dissipation}} + \underbrace{F}_{\text{external forcing}}, \tag{5.8}$$

for all $i \in [\![1, l]\!]$ with the convention that $x_0 = x_l$ and $x_{l+1} = x_1$. The states represent an unspecified scalar meteorological quantity, e.g. "vorticity or temperature" [Lorenz and Emanuel, 1998]. This model was introduced in order to select which locations on a latitude circle are the most effective in improving weather assimilation and forecasts.

The observations for this example where generated with a Gaussian perturbation with a variance of one (of the order of 10% noise) and the external forcing was set to the strongly supercritical value of eight. The results were obtained on a model with 40 locations on

the latitude circle and the time step for integrating Equation 5.8 was set to $dt = 0.01$. The assimilation results obtained with a Gaussian RBF kernel are shown in Figure 5.2.

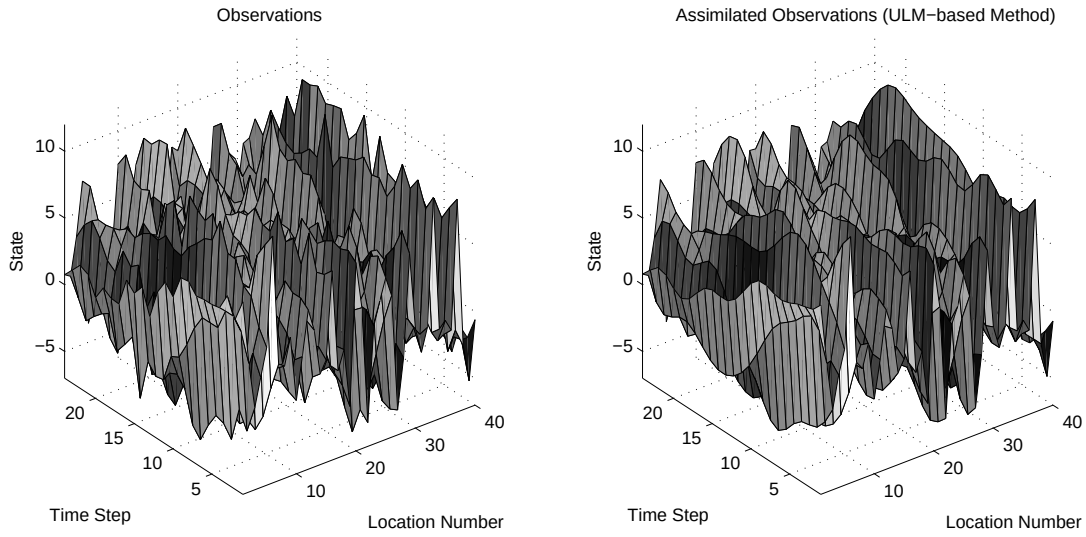


Figure 5.2: Assimilation results for the Lorenz 96 model. The left plot shows the observations before assimilation (the noise component has a variance equal to 1). The right plot shows the assimilated states. The analysis of the residuals between assimilated states and the true field have an average RMSD equal to $1/2$.

The average RMSD between assimilated states and the true field illustrates the problem of using an ULM-based approach with a model that is *Markovian by design*. The RMSD that was obtained is just half the value of the variance of the Gaussian perturbation, which is still a good result when it is considered that the model is *chaotic by nature* and that the state transition function of Equation 5.8 was first ignored and then guessed from scratch by the ULM-based approach. Nevertheless, actual weather systems are typically non-Markovian which is a factor mitigating such an high RMSD on the assimilated states.

In our framework, the ULM-based approach used for assimilation is only the step that pre-process observations for making predictions on the future system states. While this approach alone is not particularly remarkable on Markovian processes, it actually shows more potential when combined with the polynomial extrapolation discussed in Sub-Section 5.2.2. The results of the chosen approach for state forecasting is shown in Figure 5.3 where

the RMSD of the forecasts of the polynomial extrapolation are compared to the RMSDs of the forecasts obtained by the Ensemble Kalman Filter (EnKF).

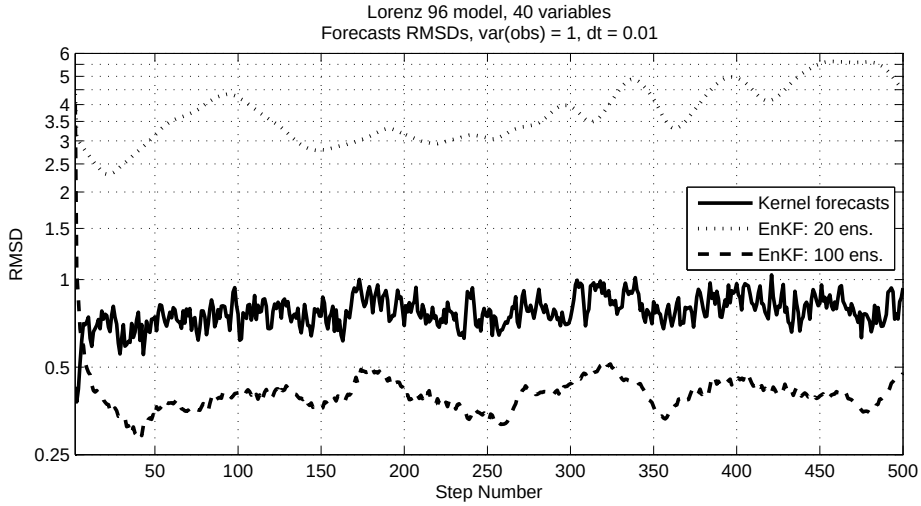


Figure 5.3: Evolution of the Root Mean Square Deviations (RMSDs) between the forecasts and the truth for the Lorenz 96 model. Comparisons are made between the EnKF with a different number of ensembles and the forecasts obtained after assimilation with the kernel approach.

The average RMSD of the forecasts for the machine learning approach is 0.8 while this number is 0.4 for the EnKF with 100 ensembles [Evensen, 1994]. The polynomial extrapolation is twice less accurate than the EnKF with 100 ensembles but it requires far less computational resources (100 ensembles are never used to model real-life weather systems since it would require more computational power than what is currently available. The "usual" number of ensemble is *at most* 20). The RMSDs of the ENKF with 20 ensembles keep growing from 2.5 to 6 after 500 steps which shows that the forecasts obtained with such a number of ensembles are completely irrelevant (an average error of 6 units on the forecasts represents a deviation of 50% from the true field).

Forecasts results show that, despite the chaotic behavior of the Lorenz 96 model and the relatively high observational error, the ULM-based assimilation scheme was able to correctly approximate the true states and, hence, provide good assimilation for viable forecasts. It also demonstrates that the EnKF needs a high number of ensembles, much higher

than commonly used in meteorological applications, to provide reliable forecasts. Such a large number of ensembles increases computational time that could be detrimental for an online system.

5.4 Quasi-Geostrophic Model

The Quasi-Geostrophic (QG) model is a 2-D atmospheric dynamical model involving an approximation of actual winds $\mathbf{v} \in \mathbb{R}^2$ which is used in the analysis of large scale extratropical weather systems. System states are scalar quantities representing the air flow, namely the geopotential field ϕ. Horizontal winds $\mathbf{v} = \mathbf{v}_g + \mathbf{v}_a$ are replaced by their *geostrophic*[2] values $\mathbf{v}_g$ in the horizontal acceleration terms of the momentum equations (the term $\mathbf{v}_a$ represents the *ageostrophic* winds), and the horizontal advection in the thermodynamic equation is approximated by geostrophic advection. However, for practical forecasting purposes, the horizontal momentum equation is typically replaced by the vorticity equation in the quasi-geostrophic model where the geostrophic vorticity ζ_g is equal to $\Delta\phi / f_0$ (the quantity f_0 is the *Coriolis force*). Furthermore, vertical advection of momentum is neglected. The quasi-geostrophic equations are:

$$\mathbf{v}_g = \frac{\mathbf{k} \times \nabla\phi}{f_0} \tag{5.9}$$

$$\partial_t \zeta_g = -\mathbf{v}_g \cdot \nabla(\zeta_g + f_0 + \beta y) + f_0 \partial_p \omega \tag{5.10}$$

$$\text{div}\, \mathbf{v}_a + \partial_p \omega = 0 \tag{5.11}$$

$$(\partial_t + \mathbf{v}_g \cdot \nabla)(-\partial_p \phi) - \sigma\omega = \frac{\kappa J}{p} \tag{5.12}$$

The system variables are the geopotential field ϕ, the geostrophic wind $\mathbf{v}_g$, the ageostrophic wind $\mathbf{v}_a$ and the *pressure change following the motion* ω. These variables are all dependent and the system states can be expressed solely on the geopotential field ϕ. All the other

[2]Geostrophic winds are theoretical winds which result from the balance between the Coriolis force and the pressure gradient force. They are directed parallel to all isobars.

terms in Equations 5.9 to 5.12 are constants in which we have:

- The vector normal to the surface **k**;
- The time t;
- The pressure p;
- The rate of heating per unit of mass $J = 0.01\ \mathrm{J \cdot kg^{-1} \cdot s^{-1}}$;
- The advection term $\sigma = 4 \cdot 10^{-5}\ \mathrm{m^2 \cdot Pa^{-2} \cdot s^{-2}}$;
- The coefficient $\beta = 2 \cdot 10^{-7}\ \mathrm{m^{-1}}$;
- The ratio $\kappa = R/c_p$ where R is the specific gas constant of the air ($287.058\ \mathrm{J \cdot kg^{-1} \cdot K^{-1}}$) and c_p is the specific heat capacity of the air ($1003.5\ \mathrm{J \cdot kg^{-1} \cdot K^{-1}}$). It follows that κ is a dimensionless number equal to 0.286;

Equation 5.9 is the definition of the geostrophic wind $\mathbf{v}_g$; Equation 5.10 is the *vorticity equation*; Equation 5.11 is the *continuity equation*; and Equation 5.12 is the *thermodynamic energy equation*. This system of equations is the quasi-geostrophic model that is used in the numerical experiments of this section where geopotential fields are assimilated then predicted with machine learning based approaches.

For the experiments, the atmosphere has a single level in the vertical and was represented by a 33×33 square grid where each system state is located on a node of that grid. Observations were generated with a Gaussian noise of variance equal to 1 and the time step for integration was set to 5.

Experiments with the ULM-based assimilation technique shows that the average RMSD between assimilated states and the true field was 2.25 which is an excellent value overall. The left plot on Figure 5.4 illustrates the performances of the assimilation step at one specific location on the 2-D grid. The assimilated state curve closely matched the true state

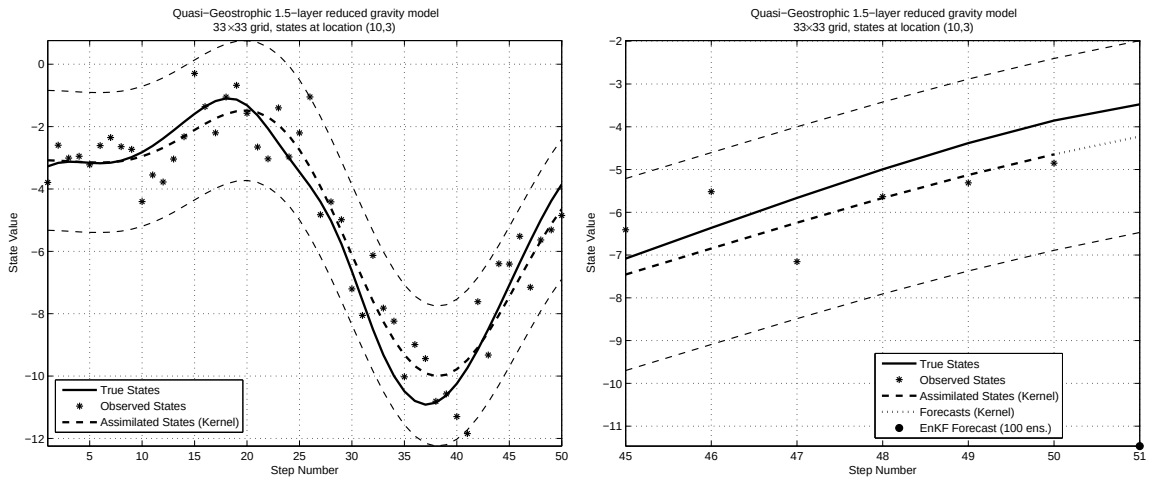


Figure 5.4: Illustration of the robustness of the ULM-based assimilation approach (left plot) and accuracy of the polynomial extrapolation (right plot). The stream of observed states for the QG model was located on the point (10,3) of a 33 by 33 grid was captured.

curve despite the high noise and the absence of knowledge about the state transition function of this system. The computational time required to interpolate the state trajectory of Figure 5.4 was only a fraction of the time needed by the EnKF method during the assimilation step.

The right plot on Figure 5.4 shows how the polynomial extrapolation technique behaves with respect to a classic EnKF approach. The smoothing capabilities of the assimilation step paved the way to the extrapolation approach by giving the guarantee that the assimilated state curve was following the trend of the true field and was not significantly influenced by the noise. On the other hand, the EnKF has no memory of the previously assimilated states and was not able to properly follow the trend of the true field. This lack of memory in a non-Markovian process was detrimental to the forecast, and the EnKF prediction was completely off the real value of the geopotential field.

Figure 5.5 illustrate the evolution of the forecast RMSDs between the polynomial extrapolation technique and the EnKF with various number of ensembles. While the RMSD for the polynomial extrapolation technique stays around 2.8 over the first 500 steps, the forecast RMSDs for all EnKF approaches increase. The results of the 100 ensembles

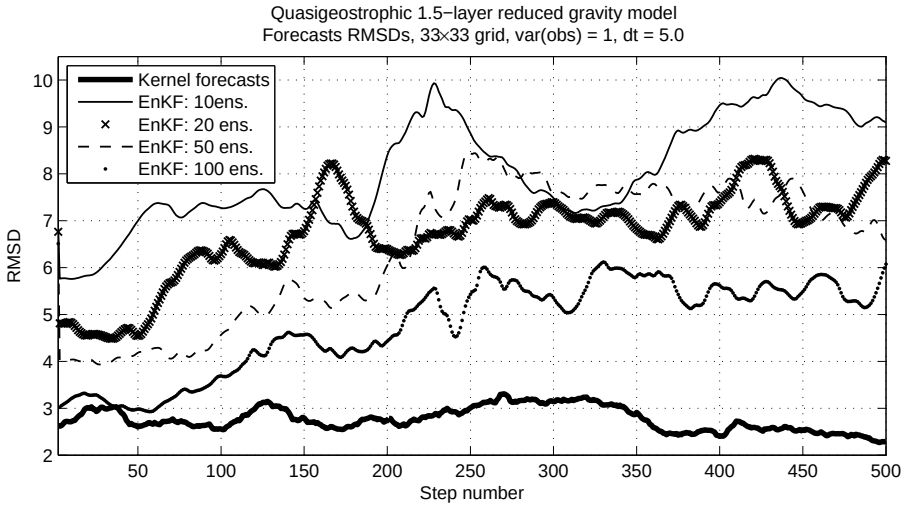


Figure 5.5: Evolution of the forecast root mean square deviations on the QG model. Comparisons are made between the EnKF with a different number of ensembles and the forecasts obtained after assimilation with the ULM-based approach.

EnKF become unreliable after 250 steps, and the performances with a lower number of ensembles are much worse than that. These poor results are due to some structural inadequacies of Kalman filters for this specific example, namely the lack of memory and the non-robustness of the approach. If the noise amplitude and the time step are decreased then the EnKF results get a bit better for the variant with 100 ensembles.

Results demonstrated that the ULM-based assimilation scheme can mitigate the effect of noise on observations, without specific knowledge of the underlying mathematical model. The uncovered state trajectories and their trends in the state space were successfully used for short-term weather forecast which is an aspect of the method that may prove useful in retrieving knowledge from unknown parts of the mathematical weather model.

Conclusions and Recommendations

This dissertation introduced a family of supervised learning algorithms called *Unconstrained Learning Machines* (ULMs). Elementary notions of supervised learning, data processing, kernel methods and statistical learning theory were discussed in order to properly construct ULMs. Very little statistical assumptions on the data are required. Only the existence of a *pattern function* between *observations* and *targets* is needed. The statistical distribution of the features of the data or the structure of the pattern function can remain unknown. ULMs rely heavily on kernel methods which allow the processing of exotic forms of data through the use of particular measures of similarity between observations. Furthermore, the capabilities of ULMs to recover complex nonlinear patterns from data is demonstrated by the combination of the fundamentals of statistical learning theory and the elementary properties of kernels.

The design and implementation of ULMs aim to provide scalable, robust and accurate methods for solving supervised learning tasks such as classification and regression. ULMs require the storage of a Gramian matrix which grows quadratically with the number of observations. A couple of general data thinning schemes were discussed to counter this memory requirement, including a pipe-lining technique that can be used to emulate online schemes. The mathematical programming formulations that form the core of ULMs are intrinsically error-tolerant which allows observations to have outliers and/or inaccurate measurements. The numerical stability of the implementation of ULMs is guaranteed under certain conditions that were properly established. However, it is the impact of the implementation of ULMs on the actual values of the optimal solutions that still need to be analyzed further.

The research work left a few theoretical and practical questions opened. Regarding the

background of ULMs, the connection between the ULM performances and the condition number of the linear systems to be solved is a novel kind of theoretical consideration that needs further investigation. This question of numerical stability is rather important since that solutions are always guaranteed in theory, but the case may arise where no implementation can compute viable solutions for a specific problem. Interestingly, that point was never investigated for closely related learning algorithms. For instance, the implementations of SVMs rely heavily on third party software (e.g. quadratic programming solvers) with very little focus, if any, on numerical stability. This aspect becomes critical as the use of ML techniques for many supervised learning tasks increases in very sensitive applications (e.g. aerospace engineering).

The investigations in manufacturing engineering opened the door to new applications. For example, the registration technique that was tailored for ULMs can be used to solve offshoot problems such as the parallelism or the concentricity of many surfaces with an unprecedented confidence level. The technique also overcomes the systemic bias introduced by many minimum zone computation techniques due to an inherent faulty registration. The ULM-based nonlinear nonparametric regression approach for the determination of minimum zones is a general approach that can be applied to any smooth bi-dimensional surface embedded in $\mathbb{R}^3$. Hence, it is no longer needed to change the numerical method for the computation of the minimum zone when switching from plates to cylinders for example. Furthermore, the ULM approach allows the definition of the minimum zone on surfaces for which it was not defined before (e.g. catenoid, pseudosphere, unduloid).

The applications of ULMs for the analysis of numerical weather models need additional investigations. The assimilation and prediction of system states was successfully tested on a limited number of simple weather models, but these techniques still need to be adapted to more realistic models with much larger sets of observational inputs. Future improvements have to include computationally efficient ways to recover estimates of the state-transition function of the dynamical systems. Specialized data thinning methods for weather data and

reliable ULM-based pipe-lining assimilation methods must be implemented to guarantee the viability of the approach in actual meteorological applications. Additionally, future research will investigate the automation of this assimilation/forecast approach in the context of meteorology.

Bibliography

I. Adrianto, T. M. Smith, K. Scharfenberg, and T. B. Trafalis. Evaluation of various algorithms and display concepts for weather forecasting. In *85th AMS Annual Meeting, American Meteorological Society - Combined Preprints*, pages 351–356, Boston, MA, USA, January 2005. American Meteorological Society.

M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundations of potential function method in pattern recognition learning. *Automation and Remote Control*, 25(6): 821–837, 1964.

A. Alenezi, S. A. Moses, and T. B. Trafalis. Flowtime Estimation for Multi-Resource Production Systems Using Support Vector Regression. In C. H. Dagli et al., editors, *Smart engineering system design: neural networks, evolutionary programming, data mining, and artificial life*, volume 15 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 699–702, New York, NY, USA, November 2005. ASME Press.

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

D. F. Andrews. Plots of High-Dimensional Data. *Biometrics*, 28(1):125–136, March 1972. Special Multivariate Issue.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, May 1950.

ASME. *Mathematical definition of dimensioning and tolerancing principles*. Y14.5.1M-1994. ASME, New York, NY, USA, 1995a.

ASME. *Dimensioning and tolerancing*. Y14.5M-1994. ASME, New York, NY, USA, 1995b.

M. Athans and P. L. Falb. *Optimal Control: An Introduction to the Theory and Its Applications*. McGraw-Hill, New York, NY, USA, 1966.

M. Avriel and D. J. Wilde. Optimality proof for the symmetric Fibonacci search technique. *Fibonacci Quarterly*, 4(3):265–269, 1966.

M. A. Badar, S. Raman, and P. S. Pulat. Intelligent search-based selection of sample points for straightness and flatness estimation. *Journal of Manufacturing Science and Engineering*, 125(2):263–271, 2003.

M. A. Badar, S. Raman, and P. S. Pulat. Experimental verification of manufacturing error pattern and its utilization in form tolerance sampling. *International Journal of Machine Tools and Manufacture*, 45(1):63–73, 2005a.

M. A. Badar, S. Raman, P. S. Pulat, and R. L. Shehab. Experimental analysis of search-based selection of sample points for straightness and flatness estimation. *Journal of Manufacturing Science and Engineering*, 127(1):96–103, 2005b.

Z.-Z. Bai and S.-L. Zhang. A Regularized Conjugate Gradient Method For Symmetric Positive Definite System Of Linear Equations. *Journal of computational mathematics*, 20(4):437–448, 2002.

P. Balakrishna, S. Raman, T. B. Trafalis, and B. Santosa. Support vector regression for determining the minimum zone sphericity. *International Journal of Advanced Manufacturing Technology*, 35(9-10):916–923, January 2008.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. In D. Helmbold and B. Williamson, editors, *Computational Learning Theory: 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001*, volume 2111 of *Lecture Notes in Computer Science*, pages 224–240, Berlin, Germany, July 2001. Springer.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research*, 3:463–482, 2002.

M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley-Interscience, Hoboken, NJ, USA, 3rd edition, May 2006.

E. A. Beardslee and T. B. Trafalis. Data Mining Methods in a Metrics-Deprived Inventory Transactions Environment. In A. Zanasi, C. A. Brebbia, and N. F. F. Ebecken, editors, *Data mining VI: data mining, text mining and their business applications*, volume 35 of *WIT transactions on information and communication technologies*, pages 513–522, Boston, MA, USA, 2005. WIT Press.

J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discover*, 2(2):121–167, June 1998.

K. Carr and P. Ferreira. Verification of form tolerances part I: Basic issues, flatness, and straightness. *Precision Engineering*, 17(2):131–143, 1995a.

K. Carr and P. Ferreira. Verification of form tolerances part II: Cylindricity and straightness of a median line. *Precision Engineering*, 17(2):144–156, 1995b.

G. J. Chaitin. On the Length of Programs for Computing Finite Binary Sequences. *Journal of the ACM*, 13(4):547–569, October 1966.

J. Chen and Z. Shen. Regularized conjugate gradient method for skew-symmetric indefinite system of linear equations and applications. *Applied Mathematics and Computation*, 187 (2):1484–1494, April 2007.

J.-H. Chen. M-estimator based robust kernels for support vector machines. In J. Kittler, M. Petrou, M. S. Nixon, and E. R. Hancock, editors, *Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004)*, volume 1, pages 168–171, Los Alamitos, CA, USA, August 2004. IEEE Computer Society Press.

W. S. Chung, L. Gruenwald, and T. B. Trafalis. Support vector clustering for web usage mining. In C. H. Dagli, A. L. Buczak, J. Ghosh, M. J. Embrechts, O. Ersoy, and S. W. Kercel, editors, *Smart engineering system design: neural networks, fuzzy logic, evolutionary programming, complex systems and artificial life*, volume 12 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 385–390, New York, NY, USA, November 2002. ASME Press.

W. G. Cochran. *Sampling techniques*. Probability and mathematical statistics. Wiley, New York, NY, USA, 3rd edition, 1977.

D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, March 1990.

R. Courant and D. Hilbert. *Methods of mathematical physics*. Interscience Publishers, New York, NY, USA, 1953.

J. Couto. Kernel K-Means for Categorical Data. In *Advances in Intelligent Data Analysis VI: Proceedings of the 6th International Symposium on Intelligent Data Analysis, IDA 2005*, volume 3646 of *Lecture Notes in Computer Science*, pages 46–56, Berlin, Germany, September 2005. Springer.

M. d'Ocagne. *Coordonnées parallèles et axiales : méthode de transformation géométrique et procédé nouveau de calcul graphique déduits de la considération des coordonnées parallèles*. Gauthier-Villars, Paris, France, 1885.

K. I. Diamantaras and S. Y. Kung. *Principal Component Neural Networks*. Wiley, New York, NY, USA, 1996.

C. H. Q. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.

M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Milan, Italy, 1992.

G. Evensen. Sequential data assimilation with nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99 (C5):143–162, 1994.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7(2):179–188, 1936.

C. F. Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. S. F. Perthes and I. H. Besser, Hamburg, 1809.

R. C. Gilbert and T. B. Trafalis. Quadratic Programming Formulations for Classification and Regression. *Optimization Methods and Software*, 24(2):175–185, April 2009.

R. C. Gilbert, S. Raman, T. B. Trafalis, S. M. Obeidat, and J. A. Aguirre-Cruz. Mathematical Foundations for Form Inspection and Adaptive Sampling. *Journal of Manufacturing Science and Engineering*, 131(4):041001, August 2009a.

R. C. Gilbert, T. B. Trafalis, M. B. Richman, and S. Lakshmivarahan. Real-Time Prediction Using Kernel Methods and Data Assimilation. In C. H. Dagli, K. M. Bryden, S. M. Corns, M. Gen, K. Tumer, and G. Süer, editors, *Computational Intelligence in Architecturing Complex Engineering Systems*, volume 19 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 35–42, New York, NY, USA, November 2009b. ASME Press.

R. C. Gilbert, S. Raman, and T. B. Trafalis. Form Inspection Using Kernel Methods. *Journal of Engineering Manufacture*, 2010. doi: 10.1243/09544054JEM1433. Published online.

F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

F. Glover. Tabu Search – Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

F. Glover. Tabu Search – Part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.

R. W. Hamming. Error detecting and error correcting codes. *Bell Labs Technical Journal*, 26(2):147–160, 1950.

T. Hastie and W. Stuetzle. Principal Curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

H. He and E. A. Garcia. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, September 2009.

M. R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, December 1952.

G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006.

R. J. Hocken, J. Raja, and U. Babu. Sampling Issues in Coordinate Metrology. *Manufacturing Review*, 6(4):282–294, 1993.

J. H. Holland. *Adaptation in natural and artificial systems*. Complex adaptive systems. MIT Press, Cambridge, MA, USA, 1975.

R. Hooke and T. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the ACM*, 8(2):212–229, 1961.

T. H. Hopp. Computational Metrology. *Manufacturing Review*, 6(4):295–304, 1993.

F. L. Hulting. Methods for the Analysis of Coordinate Measurement Data. In H. J. Newton, editor, *Proceedings of the 24th Symposium on the Interface. Computing science and statistics: graphics and visualization*, volume 24, pages 160–169, Fairfax Station, VA, USA, 1992. Interface Foundation of North America.

J. J. Hurt and L.-V. Colwell. A Comparison of Several Plane Fit Algorithms. *CIRP Annals - Manufacturing Technology*, 29(1):381–384, 1980.

H. Ince and T. B. Trafalis. Short term forecasting with support vector machines and application to stock price prediction. In C. H. Dagli et al., editors, *Smart engineering system design: neural networks, fuzzy logic, evolutionary programming, complex systems and artificial life*, volume 13 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 737–742, New York, NY, USA, November 2003. ASME Press.

H. Ince and T. B. Trafalis. Kernel methods for short-term portfolio management. *Expert Systems with Applications*, 30(3):535–542, April 2006a.

H. Ince and T. B. Trafalis. A hybrid model for exchange rate prediction. *Decision Support Systems*, 42(2):1054–1062, November 2006b.

H. Ince and T. B. Trafalis. Kernel principal component analysis and support vector machines for stock price prediction. *IIE Transactions*, 39(6):629–637, June 2007.

R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Transactions of the ASME. Series D, Journal of Basic Engineering*, 83:95–107, 1961.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, New York, NY, USA, November 1995. IEEE Computer Society Press.

J. Kepler. *Harmonices mundi*. J. Plancus, Linz, Austria, 1619.

J. Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4(3):502–506, 1953.

W.-S. Kim and S. Raman. On the selection of flatness measurement points in coordinate measuring machine inspection. *International Journal of Machine Tools and Manufacture*, 40(3):427–443, 2000.

S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by Simulated Annealing . *Science*, 220(4598):671–680, May 1983.

T. Kohonen. Self-organized formation of topologically correct feature maps . *Biological Cybernetics*, 43(1):59–69, January 1982.

T. Kohonen. *Self-organizing maps*. Information Sciences. Springer, New York, NY, USA, 3rd edition, 2001.

T. Kohonen and K. Mäkisara. Representation of sensory information in self-organizing feature maps. In J. S. Denker, editor, *AIP Conference Proceedings*, volume 151, pages 271–276, New York, NY, USA, 1986. American Institute of Physics.

A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

V. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. In E. Giné, D. M. Mason, and J. A. Wellner, editors, *High Dimensional Probability II*, volume 47 of *Progress in probability*, pages 443–459, Boston, MA, USA, 2000. Birkhäuser.

R. I. Kondor and J. Lafferty. Diffusion Kernels on Graphs and Other Discrete Structures. In C. Sammut and A. G. Hoffmann, editors, *Machine learning : proceedings of the Nineteenth International Conference (ICML 2002)*, pages 315–322, San Francisco, CA, USA, July 2002. Morgan Kaufmann.

T. R. Kurfess and D. L. Banks. Statistical verification of conformance to geometric tolerance. *Computer-Aided Design*, 27(5):353–361, 1995.

Y. LeCun. Learning processes in an asymmetric threshold network. In E. Bienenstock, F. Fogelman-Soulié, and G. Weisbuch, editors, *Disordered systems and biological organizations*, pages 233–240. Springer, New York, NY, USA, 1986.

Y.-J. Lee, O. L. Mangasarian, and W. H. Wolberg. Survival-Time Classification of Breast Cancer Patients. *Computational Optimization and Applications*, 25(1-3):151–166, 2003.

A.-M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, Paris, France, 1805.

E. N. Lorenz and K. A. Emanuel. Optimal sites for supplementary weather observations: simulation with a small model. *Journal of the Atmospheric Sciences*, 55(3):399–414, 1998.

A. M. Malyscheff and T. B. Trafalis. Support vector machines and the electoral college. In *Proceedings of the 2003 IEEE International Joint Conference on Neural Networks*, volume 3, pages 2344–2348, Piscataway, NJ, USA, July 2003. IEEE.

A. M. Malyscheff, T. B. Trafalis, and S. Raman. From support vector machine learning to the determination of the minimum enclosing zone. *Computers and Industrial Engineering*, 42(1):59–74, April 2002.

H. Mansouri, R. C. Gilbert, T. B. Trafalis, L. M. Leslie, and M. B. Richman. Ocean Surface Wind Vector Forecasting Using Support Vector Regression. In C. H. Dagli, A. L. Buczak, D. L. Enke, M. J. Embrechts, and O. Ersoy, editors, *Smart systems engineering:*

computational intelligence in architecting complex engineering systems, volume 17 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 333–338, New York, NY, USA, November 2007. ASME Press.

J. Mercer. Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909.

S. Mika, B. Schölkopf, S. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and De-Noising in Feature Spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 536–542, Cambridge, MA, USA, 1999. MIT Press.

R. E. A. Moustafa. QGPCP: Quantized Generalized Parallel Coordinate Plots for Large Multivariate Data Visualization. *Journal of Computational and Graphical Statistics*, 18 (1):32–51, March 2009.

T. S. R. Murthy and S. Z. Abdin. Minimum zone evaluation of surfaces. *International Journal of Machine Tool Design and Research*, 20(2):123–136, 1980.

J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, 1964.

J. Nelder and R. Mead. A simplex method for function minimization – Errata. *Computer Journal*, 8(1):27, 1965.

I. Newton. *Philosophiae naturalis principia mathematica*. Joseph Streater for the Royal Society, London, UK, 1687.

A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, Brooklyn, NY, USA, 1963. Polytechnic Press.

G. S. Ohm. *Die galvanische Kette: mathematisch bearbeitet*. T. H. Riemann, Berlin, Prussia, 1827.

O. Oladunni and T. B. Trafalis. Single Phase Laminar and Turbulent Flow Classification in Annulus via a Knowledge-Based Linear Model. In C. H. Dagli et al., editors, *Smart systems engineering: infra-structure systems engineering, bio-informatics and computational biology and evolutionary computation*, volume 16 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 599–604, New York, NY, USA, November 2006. ASME Press.

O. Oladunni, T. B. Trafalis, and D. V. Papavassiliou. Knowledge-based multiclass support vector machines applied to vertical two-phase flow. In V. N. Alexandrov, G. D. van Albada, P. M. Sloot, and J. Dongarra, editors, *Computational science – ICCS 2006*, volume 3991 of *Lecture notes in computer science*, pages 188–195, Berlin, Germany, May 2006. Springer.

K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, 2(6):559–572, 1901.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in kernel methods: support vector learning*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

J. C. Platt, N. Cristianini, and J. Shawe-taylor. Large margin DAGs for multiclass classification. In S. A. Solla, K.-R. Müller, and T. K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 61–73, Cambridge, MA, USA, 2000. MIT Press.

C. Prakasvudhisarn, T. B. Trafalis, and S. Raman. Determination of the minimum enclosing zone for straightness and flatness using support vector regression. In C. H. Dagli, A. L. Buczak, J. Ghosh, M. J. Embrechts, O. Ersoy, and S. W. Kercel, editors, *Smart engineering system design: neural networks, fuzzy logic, evolutionary programming, complex systems and artificial life*, volume 12 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 937–944, New York, NY, USA, November 2002. ASME Press.

C. Prakasvudhisarn, T. B. Trafalis, and S. Raman. Support vector regression for determination of minimum zone. *Journal of Manufacturing Science and Engineering*, 125(4): 736–739, November 2003.

W. H. Press. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 3rd edition, 2007.

S. Raman, T. B. Trafalis, and R. C. Gilbert. Adaptive Methods in Coordinate Metrology. In B. Gopalakrishnan, editor, *Intelligent Systems in Design and Manufacturing VI*, volume 5999 of *Proceedings of SPIE, the International Society for Optical Engineering*, page 59990A, Bellingham, WA, USA, November 2005. SPIE.

A. A. G. Requicha. Mathematical Definition of Tolerance Specifications. *Manufacturing Review*, 6(4):269–274, 1993.

F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6):386–408, November 1958.

H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *Computer Journal*, 3(3):175–184, 1960.

S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

U. Roy and X. Zhang. Establishment of a pair of concentric circles with the minimum radial separation for assessing roundness error. *Computer-Aided Design*, 24(3):161–168, 1992.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. McClelland, editors, *Parallel distributed processing: explorations in the microstructure of cognition*, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

Y. Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, PA, USA, 2nd edition, 2003.

S. Saitoh. *Theory of reproducing kernels and its applications*. Wiley, New York, NY, USA, 1988.

G. L. Samuel and M. S. Shunmugam. Evaluation of straightness and flatness error using computational geometric techniques. *Computer-Aided Design*, 31(13):829–843, 1999.

B. Santosa, T. Conway, and T. B. Trafalis. Knowledge-based clustering and application of multi-class SVM for genes expression analysis. In C. H. Dagli, A. L. Buczak, J. Ghosh, M. J. Embrechts, O. Ersoy, and S. W. Kercel, editors, *Smart engineering system design: neural networks, fuzzy logic, evolutionary programming, complex systems and artificial life*, volume 12 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 391–396, New York, NY, USA, November 2002. ASME Press.

B. Santosa, T. Conway, and T. B. Trafalis. A Hybrid Knowledge Based-Clustering Multi-Class SVM Approach for Genes Expression Analysis . In P. M. Pardalos, V. L. Boginski, and A. Vazacopoulos, editors, *Data Mining in Biomedicine*, volume 7 of *Springer Optimization and Its Applications*, pages 261–274. Springer, New York, NY, USA, 2007.

B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Artificial Neural Networks – ICANN'97*, volume 1327 of *Lecture Notes in Computer Science*, pages 583–588, New York, NY, USA, 1997. Springer.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.

M. S. Shunmugam. New approach for evaluating form errors of engineering surfaces. *Computer-Aided Design*, 19(7):368–374, 1987.

A. J. Smola and B. Schölkopf. Bayesian Kernel Methods . In S. Mendelson and A. J. Smola, editors, *Advanced Lectures on Machine Learning*, volume 2600 of *Lecture Notes in Computer Science*, pages 65–117, Berlin, Germany, 2003. Springer.

W. Spendley, G. Hext, and F. Himsworth. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4(4):441–461, 1962.

K. Steinbuch. Adaptive networks using learning matrices . *Biological Cybernetics*, 2(4): 148–152, February 1965.

V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, August 1969.

J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300, 1999.

M. E. Tipping and C. M. Bishop. Mixtures of Probabilistic Principal Component Analyzers. *Neural Computation*, 11(2):443–482, 1999.

T. B. Trafalis and R. C. Gilbert. Maximum Margin Classifiers with Noisy Data: A Robust Optimization Approach. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN) 2005*, volume 5, pages 2826–2830, Piscataway, NJ, USA, July 2005. IEEE Operations Center.

T. B. Trafalis and R. C. Gilbert. Robust Classification and Regression Using Support Vector Machines. *European Journal of Operational Research*, 173(3):893–909, September 2006.

T. B. Trafalis and R. C. Gilbert. Robust Support Vector Machines for Classification and Computational Issues. *Optimization Methods and Software*, 22(1):187–198, February 2007.

T. B. Trafalis, O. Oladunni, and D. V. Papavassiliou. Two-phase flow regime identification with a multiclassification support vector machine (SVM) model. *Industrial and Engineering Chemistry Research*, 44(12):4414–4426, June 2005.

T. B. Trafalis, I. Adrianto, and M. B. Richman. Active learning with support vector machines for tornado prediction. In Y. Shi, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, editors, *Computational Science – ICCS 2007*, volume 4487 of *Lecture Notes in Computer Science*, pages 1130–1137, Berlin, Germany, May 2007. Springer.

A. N. Tychonoff. Solution of incorrectly formulated problems and the regularization method. *Proceedings of the USSR Academy of Sciences*, 151(3):501–504, 1963.

V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer series in statistics. Springer, New York, NY, USA, 1982.

V. N. Vapnik. *The nature of statistical learning theory*. Springer, New York, NY, USA, 1995.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, USA, 1998.

V. N. Vapnik and A. Y. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability and its Applications*, 16(2): 264–280, January 1971.

V. N. Vapnik and A. Y. Chervonenkis. The necessary and sufficient conditions for consistency of the method of empirical risk minimization. *Pattern Recognition and Image Analysis*, 1(3):284–305, 1991.

V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, January 1985.

G. F. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik*, 133:97–178, 1907.

P. Wegner. A technique for counting ones in a binary computer. *Communications of the ACM*, 3(5):322, May 1960.

B. Widrow. Generalization and Information Storage in Networks of Adaline Neurons. In M. C. Yovits, G. T. Jacobi, and G. D. Goldstein, editors, *Self-Organizing Systems*, pages 435–461. Spartan Books, Washington, DC, USA, 1962.

T. C. Woo and R. Liang. Dimensional measurement of surfaces and their sampling. *Computer-Aided Design*, 25(4):233–239, 1993.

M. A. Woodbury. Inverting modified matrices. Memorandum Report 42, Statistical Research Group, Princeton University, Princeton, NJ, USA, 1950.

H. Yin. Learning nonlinear principal manifolds by self-organising maps . In A. N. Gorban, B. Kégl, D. C. Wunsch, and A. Y. Zinovyev, editors, *Principal Manifolds for Data Visualization and Dimension Reduction*, volume 58 of *Lecture Notes in Computational Science and Engineering*, pages 68–95. Springer, Berlin, Germany, 2007.

L. A. Zadeh and C. A. Desoer. *Linear System Theory: The State Space Approach*. McGraw-Hill, New York, NY, USA, 1963.

W. Zangwill. Minimizing a function without calculating derivatives. *Computer Journal*, 10(3):293–296, 1967.

Appendix A

Glossary

A.1 Sets

The empty set is $\emptyset$. The set of natural integers is written $\mathbb{N}$ and the group of relative integers is written $\mathbb{Z}$. The set of consecutive integers between a and b is written $[\![a,b]\!]$. The field of rational numbers is represented by $\mathbb{Q}$. The real algebras of dimensions 1 and 2 are denoted respectively by $\mathbb{R}$ and $\mathbb{C}$. The interval of $\mathbb{R}$ between a and b is $[a,b]$ when closed, (a,b) when opened, and $(a,b]$ or $[a,b)$ when semi-closed or semi-opened. If a set does not include the element 0, then a super-scripted star is added. For example, the set of strictly positive natural integers is written $\mathbb{N}^*$. For the space $\mathbb{R}$, the set of non-negative real numbers is $\mathbb{R}_+$ and the set of non-positive real numbers is $\mathbb{R}_-$.

Any other set is represented by an uppercase Latin letter in standard or calligraphic shape (e.g., sets E and $\mathscr{H}$). The use of uppercase Gothic letters for sets is not excluded. For example, the set of the subsets of E will be written $\mathfrak{P}(E)$. The Cartesian product of two spaces E and F is written $E \times F$, and the exponential notation is used in the case of multiple Cartesian products (e.g., $\mathbb{R}^n$). The dual of a vector space E is represented by E^* and the cardinal of a finite discrete set F is written $|F|$.

A.2 Scalars, Vectors and Matrices

The notation allows the distinction between scalars, vectors and matrices by using different typefaces. Scalars and functions are all written with lowercase characters (e.g., scalar a, function f). The Latin letters f, g, h and the Greek letter ϕ, φ and θ commonly refer to

functions. In the same fashion, Latin letters k, l, i, j, n, m often represent integers while Latin letters t, u, v, w, x, y often represent real numbers. The imaginary unit is written $\imath$.

Vectors are written with boldface lowercase characters (e.g., vector $\mathbf{v}$) and are assumed to be column vectors unless specified otherwise. Non-boldface Greek letters can also be used to write vectors. The transpose of a vector $\mathbf{v}$ is represented by $\mathbf{v}^{\mathrm{t}}$ and the vector of dimension n for which all components are equal to one is written $\mathbf{1}_n$. The canonical unit vectors which form the basis of $\mathbb{R}^n$ are written $\mathbf{e}_1, \ldots, \mathbf{e}_n$. The i-th element of a vector $\mathbf{v}$ is a scalar written v_i, and the sub-vector generated from a set of indices I is $\mathbf{v}_I$.

Boldface uppercase Latin letters (e.g., $\mathbf{M}$) or uppercase Greek letters represent matrices. The (i, j)-th element of a matrix $\mathbf{M}$ is a scalar and is represented by M_{ij}. The i-th row of a matrix $\mathbf{M}$ is written $\mathbf{M}_{i\cdot}$ and its j-th column is written $\mathbf{M}_{\cdot j}$. If two sets I and J of row and columns indices of a matrix $\mathbf{M}$ is given, then the sub-matrix generated by the rows in I and the columns in J is written $\mathbf{M}_{IJ}$. The identity matrix of dimension n is written $\mathbf{I}_n$ and the transpose of a matrix $\mathbf{M}$ is represented by $\mathbf{M}^{\mathrm{t}}$.

A.3 Dot Products, Norms and Other Operators

The Euclidean norm of a vector $\mathbf{v} \in \mathbb{R}^n$ is written $\|\mathbf{v}\|$ and the Euclidean dot product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is written $\langle \mathbf{u}, \mathbf{v} \rangle$. The p-norm of a vector $\mathbf{v}$ is represented by $\|\mathbf{v}\|_p$ (note that $\|\mathbf{v}\| = \|\mathbf{v}\|_2$). When using a dot product in a Hilbert space $\mathscr{H}$ or its associated norm, the dot product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is written $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathscr{H}}$ and the associated norm of a vector $\mathbf{v}$ is written $\|\mathbf{v}\|_{\mathscr{H}}$. The operator $\cdot$ between two matrices or vectors of equal dimensions is the Hadamard product (i.e. the entry-wise multiplication).

Appendix B

Additional Algorithms

B.1 Computation of Solutions of Linear Systems

B.1.1 Cholesky Decomposition

The Cholesky decomposition algorithm B.1 is a method that decomposes a symmetric and positive definite matrix **A** into the product of a lower triangular matrix **L** and its transpose, i.e. $\mathbf{A} = \mathbf{L}\mathbf{L}^{\mathrm{t}}$ [Press, 2007]. If n is the size of the matrix, then the method requires $n^3/3 + n^2/2 + n/6$ FLOPS to decompose the matrix.

Algorithm B.1: Cholesky Decomposition

Function [**L**] = CHOLDC (**A**)
Input: $n \times n$ symmetric positive definite matrix **A**.
Output: $n \times n$ lower triangular matrix **L**.

1 $\mathbf{L} \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n}$
2 **for** $i \in [\![1, n]\!]$ **do**
3 **for** $j \in [\![i, n]\!]$ **do**
4 $s \leftarrow A_{ij} -$ DOT $(\mathbf{L}_{i,[\![1,i-1]\!]}, \mathbf{L}_{j,[\![1,i-1]\!]})$ // $\mathbf{L}_{i,\emptyset} = \emptyset$
5 **if** $i = j$ **then**
6 $L_{ii} \leftarrow \sqrt{s}$
7 **else**
8 $L_{ji} \leftarrow s / L_{ii}$
9 **end**
10 **end**
11 **end**
12 **return L**

B.1.2 Computation of the Solution of a Triangular System

Algorithm B.2 uses the lower triangular matrix $\mathbf{L}$ obtained from the Cholesky decomposition of a symmetric and positive definite matrix $\mathbf{A}$ to compute the solution of the linear system $\mathbf{Ax} = \mathbf{b}$ [Press, 2007]. If n is the size of the matrix, then the method requires $2n^2$ FLOPS to compute a solution.

Algorithm B.2: Triangular System Solution

Function [$\mathbf{x}$] = CHOLSL ($\mathbf{L}, \mathbf{b}$)
Input: $n \times n$ lower triangular matrix $\mathbf{L}$, vector $\mathbf{b} \in \mathbb{R}^n$.
Output: vector $\mathbf{x} \in \mathbb{R}^n$.

$\mathbf{x} \leftarrow \mathbf{0} \in \mathbb{R}^n$
for $i \in [\![1, n]\!]$ **do**
  $x_i \leftarrow \big(b_i - \texttt{DOT}\,(\mathbf{L}_{i,[\![1,i-1]\!]}, \mathbf{x}_{[\![1,i-1]\!]})\big)/L_{ii}$ // $\mathbf{L}_{i,\emptyset} = \emptyset$, $\mathbf{x}_\emptyset = \emptyset$
end
for $i \in [\![n, 1]\!]$ **do**
  $x_i \leftarrow \big(x_i - \texttt{DOT}\,(\mathbf{L}_{[\![i+1,n]\!],i}, \mathbf{x}_{[\![i+1,n]\!]})\big)/L_{ii}$
end
return x

B.1.3 Conjugate Gradient Method

The conjugate gradient algorithm B.3 is an iterative method for solving linear systems of equations whose matrix is symmetric and positive definite. This method is derived from the method of Arnoldi which is based on orthogonal projection processes onto Krylov subspaces [Saad, 2003]. If n is the size of the system and if k is the number of iterations of the algorithm, then this method requires $2kn^2 + (9k+2)n + k - 1$ FLOPS to compute a solution. This method provides a computational advantage over a Cholesky decomposition of the system matrix only if the linear system is large and sparse. This situation may arise for function evaluation problems with a large sample of observations and a Gaussian RBF kernel tuned appropriately. In this case, the computation of the solution of the linear systems in Algorithms 3.1, 3.2 and 3.3 can be done by a Conjugate Gradient method rather

than an approach based on a Cholesky decomposition.

Algorithm B.3: Conjugate Gradient Method

Function [**x**] = CG (**A**, **b**)
Data: tolerance $\varepsilon > 0$, iteration limit $k_{\max} > 0$.
Input: $n \times n$ symmetric and positive definite matrix **A**, non-null vector $\mathbf{b} \in \mathbb{R}^n$.
Output: vector $\mathbf{x} \in \mathbb{R}^n$.

$k \leftarrow 0$
$\mathbf{x} \leftarrow \mathbf{0} \in \mathbb{R}^n$
$\mathbf{r} \leftarrow \mathbf{b}$
$\mathbf{p} \leftarrow \mathbf{r}$
$\rho \leftarrow \mathtt{DOT}\ (\mathbf{r}, \mathbf{r})$
while $\rho > \varepsilon$ *and* $k \leqslant k_{\max}$ **do**
    $\mathbf{q} \leftarrow \mathtt{GEMV}\ (\mathbf{A}, \mathbf{p})$
    $\alpha \leftarrow \rho / \mathtt{DOT}\ (\mathbf{q}, \mathbf{p})$
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
    $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{q}$
    $\alpha \leftarrow \mathtt{DOT}\ (\mathbf{r}, \mathbf{r})$
    $\mathbf{p} \leftarrow \mathbf{r} + (\alpha/\rho)\mathbf{p}$
    $\rho \leftarrow \alpha$
    $k \leftarrow k + 1$
end
return x

B.2 Algorithms for Elementary Linear Algebra

B.2.1 Symmetric Matrix Rank-2k Update

Algorithm B.4 performs a rank-2k update $\mathbf{C} = \mathbf{A}\mathbf{B}^{\mathrm{t}}$ where **A** and **B** are $n \times m$ matrices and **C** is a $n \times n$ symmetric matrix. Despite that this algorithm is twice faster than the general naive matrix-matrix multiplication, the Strassen algorithm is asymptotically faster although less stable numerically [Strassen, 1969]. An asymptotically faster algorithm also exists, however it only brings an advantage for matrices that are too large to be multiplied on modern hardware [Coppersmith and Winograd, 1990]. If n is the size of the matrices, then the following algorithm requires $n(n+1)(m-½)$ FLOPS to multiply the matrices.

Algorithm B.4: Symmetric Matrix Rank-2k Update

Function [**C**] = SYR2K (**A**, **B**)
Input: $n \times m$ matrices **A** and **B**.
Output: $n \times n$ matrix **C**.

```
1  C ← 0 ∈ ℝ^{n×n}
2  for i ∈ ⟦1, n⟧ do
3  |   for j ∈ ⟦i, n⟧ do
4  |   |   C_ij ← DOT (A_i·, B_j·)
5  |   |   if i ≠ j then
6  |   |   |   C_ji ← C_ij
7  |   |   end
8  |   end
9  end
10 return C
```

B.2.2 General Matrix-Vector Multiplication

Algorithm B.5 is the implementation of the general matrix-vector multiplication **Ax** between a $n \times m$ matrix **A** and a vector $\mathbf{x} \in \mathbb{R}^m$. This method requires $n(2m-1)$ FLOPS to perform the multiplication.

Algorithm B.5: General Matrix-Vector Multiplication

Function [**z**] = GEMV (**A**, **x**)
Input: $n \times m$ square matrix **A**, vector $\mathbf{x} \in \mathbb{R}^m$.
Output: vector $\mathbf{z} \in \mathbb{R}^n$.

```
1  z ← 0 ∈ ℝ^n
2  for i ∈ ⟦1, n⟧ do
3  |   z_i ← DOT (A_i·, x)
4  end
5  return z
```

B.2.3 Dot Product

Algorithm B.6 is the implementation of the dot product $\langle \mathbf{x}, \mathbf{y} \rangle$ between two vectors **x** and **y** in $\mathbb{R}^n$. This method implies the use of a numerically stable summation algorithm named SUM and requires $2n-1$ FLOPS to compute the result.

Algorithm B.6: Dot Product

Function $[z]$ = DOT $(\mathbf{x}, \mathbf{y})$
Input: vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$.
Output: scalar $z \in \mathbb{R}$.

if $n = 0$ **then** // Case for empty sets
  $z \leftarrow 0$
else
  if $\mathbf{x} \in \mathbb{R}^{1\times n}$ **then**
    if $\mathbf{y} \in \mathbb{R}^{1\times n}$ **then**
      $z \leftarrow \mathsf{SUM}\left(\mathbf{x}^{\mathrm{t}} \cdot \mathbf{y}^{\mathrm{t}}\right)$ // Symbol $\cdot$ stands for Hadamard product
    else
      $z \leftarrow \mathsf{SUM}\left(\mathbf{x}^{\mathrm{t}} \cdot \mathbf{y}\right)$
    end
  else
    if $\mathbf{y} \in \mathbb{R}^{1\times n}$ **then**
      $z \leftarrow \mathsf{SUM}\left(\mathbf{x} \cdot \mathbf{y}^{\mathrm{t}}\right)$
    else
      $z \leftarrow \mathsf{SUM}\left(\mathbf{x} \cdot \mathbf{y}\right)$
    end
  end
end
return z

B.2.4 Matrix Row Summation

Algorithm B.7 computes the sum of the rows of a $n \times m$ matrix $\mathbf{A}$. The aim of this algorithm is to use a numerically stable way to compute the resulting vector. This method requires $n(m-1)$ FLOPS to compute the result.

Algorithm B.7: Matrix Row Summation

Function $[\mathbf{x}]$ = ROWSUM $(\mathbf{A})$
Input: $n \times m$ matrix $\mathbf{A}$.
Output: vector $\mathbf{x} \in \mathbb{R}^n$.

$\mathbf{x} \leftarrow \mathbf{0} \in \mathbb{R}^n$
for $i \in [\![1, n]\!]$ **do**
  $x_i \leftarrow \mathsf{SUM}\left(\mathbf{A}^{\mathrm{t}}_{i\cdot}\right)$
end
return $\mathbf{x}$

B.2.5 Diagonal-Matrix Multiplication

Algorithm B.8 computes the product $\Delta\mathbf{A}$ where Δ is a $n \times n$ diagonal matrix with diagonal $\delta \in \mathbb{R}^n$ and $\mathbf{A}$ is a $n \times m$ matrix. This method requires nm FLOPS to compute the result.

Algorithm B.8: Diagonal-Matrix Multiplication

Function [$\mathbf{B}$] = `DIMM` ($\delta, \mathbf{A}$)
Input: vector $\delta \in \mathbb{R}^n$, $n \times m$ matrix $\mathbf{A}$.
Output: $n \times m$ matrix $\mathbf{B}$.

$\mathbf{B} \leftarrow \mathbf{0} \in \mathbb{R}^{n \times m}$
for $i \in [\![1, n]\!]$ **do**
  $\mathbf{B}_{i\cdot} \leftarrow \delta_i \mathbf{A}_{i\cdot}$
end
return B

B.2.6 Matrix-Diagonal Multiplication

Algorithm B.9 computes the product $\mathbf{A}\Delta$ where $\mathbf{A}$ is a $n \times m$ matrix and Δ is a $m \times m$ diagonal matrix with diagonal $\delta \in \mathbb{R}^m$. This method requires nm FLOPS to compute the result.

Algorithm B.9: Matrix-Diagonal Multiplication

Function [$\mathbf{B}$] = `MDIM` ($\mathbf{A}, \delta$)
Input: $n \times m$ matrix $\mathbf{A}$, vector $\delta \in \mathbb{R}^m$.
Output: $n \times m$ matrix $\mathbf{B}$.

$\mathbf{B} \leftarrow \mathbf{0} \in \mathbb{R}^{n \times m}$
for $i \in [\![1, m]\!]$ **do**
  $\mathbf{B}_{\cdot i} \leftarrow \delta_i \mathbf{A}_{\cdot i}$
end
return B

B.2.7 Basic Tikhonov Regularization

Algorithm B.10 computes the sum $\mathbf{A} + \mu\mathbf{I}_n$ where $\mathbf{A}$ is a $n \times n$ matrix and $\mu > 0$ is a scalar. This method requires n FLOPS to compute the result. This algorithm is referred as "basic"

since the general Tikhonov regularization is a procedure of the form $\mathbf{A}+\Gamma$ where the matrix Γ is positive definite. The general procedure then requires n^2 operations which is an order greater than the basic approach.

Algorithm B.10: Basic Tikhonov Regularization

Function [**B**] = `TIKREG` ($\mathbf{A}, \mu$)
Input: $n \times n$ matrix **A**, scalar $\mu \in \mathbb{R}_+^*$.
Output: $n \times n$ matrix **B**.

```
1 B ← 0 ∈ ℝ^{n×n}
2 for i ∈ ⟦1, n⟧ do
3 |   B_ii ← μ + A_ii
4 end
5 return B
```

Appendix C

Elementary Notions of Optimal Control

C.1 Dynamical Systems

Hereby we define the notion of dynamical systems based on a series of axioms taken from Athans and Falb [1966]. These axioms aim to abstract the behavior of physical systems in order to characterize them with adequate mathematical tools. Before introducing the axioms, it is necessary to define several variables we should work with. These variables are:

- The set $T \subset \mathbb{R}$ is a subset of the real numbers called the *domain* of the system.
- The ordered pair (Ω, ω) is a metric space where Ω is a set and ω is a distance on Ω.
- The set U is a set of bounded piecewise continuous functions over T with values in Ω. The set U is called the *input space* of the system. The restriction of a function $u \in U$ over the semi-open interval $(t_0, t] \subset T$, denoted by $u_{(t_0,t]}$, is called an *input over the observation interval* $(t_0, t]$ to the system.
- The ordered pair (Σ, σ) is a metric space where Σ is a set and σ is a distance on Σ. The set Σ is called *state space* of the system.
- The values of the function $x : T \to \Sigma$ for any $t \in T$ are called *states* of the system.
- The function $g : \Sigma \times \Omega \times T \to \mathbb{R}^p$ is called *output function* of the system. The identity $y_{(t_0,t]} = g\big(x(t_0), u_{(t_0,t]}\big)$ is called *output equation* of the system with the function $y_{(t_0,t]}$ being an *output over the observation interval* $(t_0, t]$ of the system.

The following axioms that will define the notion of dynamical system.

Axiom C.1 (Uniquely defined output)**.** *For all* $(t_0,t] \subset T$*, all* $x(t_0) \in \Sigma$ *and all* $u \in U$ *the function* $y_{(t_0,t]}$ *is uniquely defined by* $x(t_0)$ *and* $u_{(t_0,t]}$*.*

Axiom C.2 (Existence of states for input-output pairs)**.** *For all* $\tau \in (t_0,t) \subset T$ *and all* $u \in U$*, if* $\Sigma(\tau,u) = \left\{x(\tau) \in \Sigma : y_{(\tau,t]} = g\left(x(\tau), u_{(\tau,t]}\right)\right\}$ *then for any* $v \in U$*,*

$$\bigcap_{u_{(\tau,t]}=v_{(\tau,t]}} \Sigma(\tau,u) \neq \emptyset.$$

Zadeh and Desoer [1963] showed that axioms C.1 and C.2 imply the existence of a so-called *transition function* ϕ such that the *state equation* of the system,

$$x(t) = \phi\left(x(t_0), u_{(t_0,t]}, t\right), \tag{C.1}$$

holds for any given semi-open interval $(t_0,t] \subset T$ and any input $u \in U$. For an interval $I \subset T$, the set $X(I) = \left\{x(\tau) : \tau \in I\right\}$ is called the *trajectory over the observation interval I* of the system.

Axiom C.3 (Smoothness condition)**.** *The functions g and* ϕ *are continuous.*

Axiom C.4 (Initial state is the starting point of trajectory)**.** *For all* $(t_0,t] \subset T$*, all* $x(t_0) \in \Sigma$ *and all* $u \in U$ *we have* $\limsup_{t \to t_0} \phi\left(x(t_0), u_{(t_0,t]}, t\right) = x(t_0)$*.*

Axiom C.5 (Transition condition)**.** *For all* $\tau \in (t_0,t) \subset T$*, all* $x(t_0) \in \Sigma$ *and all* $u \in U$ *we have*

$$\phi\left(x(t_0), u_{(t_0,t]}, t\right) = \phi\left(\phi\left(x(t_0), u_{(t_0,\tau]}, \tau\right), u_{(\tau,t]}, t\right).$$

Axiom C.6 (Nonanticipatory condition)**.** *For all* $\tau \in (t_0, t] \subset T$*, all* $x(t_0) \in \Sigma$ *and all* $u \in U$ *we have*

$$\phi\big(x(t_0), u_{(t_0,t]}, \tau\big) = \phi\big(x(t_0), u_{(t_0,\tau]}, \tau\big).$$

These axioms can now used to define a dynamical system.

Definition C.1 (Dynamical system)**.** *The 6-tuple denoted by* $(T, \Omega, \Sigma, U, x, g)$ *which satisfies axioms C.1 to C.6 is called a* dynamical system.

Definition C.2 (Finite dimensional dynamical system)**.** *A dynamical system is a* finite-dimensional *system of order n if* $\Sigma = \mathbb{R}^n$ *and* $\Omega = \mathbb{R}^m$ *with* $m \leqslant n$*.*

Definition C.3 (Continuous-time dynamical system)**.** *A dynamical system is a* continuous time *system if the domain* T *of the system is an open interval.*

Definition C.4 (Differential dynamical system)**.** *A dynamical system is a* differential *system if the transition function* ϕ *is the solution of the differential equation* $\mathrm{d}x(t) = f\big(x(t), u(t), t\big)$ *where* $x(t_0)$ *is the initial point and where* f *is continuously differentiable.*

In this dissertation, a finite-dimensional continuous-time differential dynamical system will be referred as a *FCD system* and denoted by a 5-tuple (T, U, x, f, g) where $T = (t_0, t_\infty)$.

C.2 Optimal-Control Problems

Let (T, U, x, f, g) be a FCD system with domain $T = (t_0, t_\infty)$ and let $\mathscr{S} \subset \mathbb{R}^n \times T$ be a set called *target set*. Suppose $t \in T$ then the functional $J : U \to \mathbb{R} \cup \{\infty\}$, called *performance functional*, is defined by

$$J(u) = K\big(x(t)\big) + \int_{t_0}^{t} L\big(x(t), u(t), t\big)dt, \tag{C.2}$$

where K is a real-valued function on $\mathbb{R}^n$ and L is a continuous real-valued function on $\mathbb{R}^{n+m+1}$. If there is no element in T such that the set $\mathscr{X}(T) = \Big\{ \big(x(\tau), \tau\big) : \tau \in T \Big\}$ intersects $\mathscr{S}$ then, conventionally, $J(u) = \infty$ for all $t \in T$. If this element exists then the smallest element $t_f \in T$ such that $\mathscr{X}(T)$ intersects $\mathscr{S}$ is called *terminal time*. The state $x(t_f)$ is called *terminal state* and $K\big(x(t_f)\big)$ is called *terminal cost*.

Definition C.5 (Optimal-control problem)**.** *Let (T, U, x, f, g) be an FCD system with domain $T = (t_0, t_\infty)$ and a target set $\mathscr{S}$. The* optimal-control problem *is defined by*

$$\min_{u \in U} \big\{ J(u) : \mathscr{X}(T) \cap \mathscr{S} \neq \emptyset \big\}.$$

Definition C.6 (Bounded-state optimal-control problem)**.** *Let (T, U, x, f, g) be an FCD system with domain $T = (t_0, t_\infty)$ and a target set $\mathscr{S}$. If $S \subset \mathbb{R}^n$ is a closed set such that $\mathscr{S} \subseteq S \times T$, then the bounded-state optimal-control problem is defined by*

$$\min_{u \in U} \Big\{ J(u) : \mathscr{X}(T) \cap \mathscr{S} \neq \emptyset \text{ and } X\big((t_0, t_f]\big) \subseteq S \Big\}.$$

The optimal-control problem is called a *free-time* problem if the target set $\mathscr{S}$ is of the form $\bigcup_{t \in T} \big(S(t) \times \{t\}\big)$ with $\emptyset \neq S(t) \subseteq \mathbb{R}^n$. Furthermore if $S(t) = z(t)$ where $z : T \to \mathbb{R}^n$ then the optimal-control problem is called a *pursuit* problem. On the other hand if we suppose that the target set is of the form $S \times \{t\}$ where $S \subseteq \mathbb{R}^n$ and $t \in T$ are fixed elements then the optimal-control problem is called a *fixed-time* problem. If the target set is of the form $\{\mathbf{x}\} \times T$ where $\mathbf{x} \in \mathbb{R}^n$ then the optimal-control problem is called a *fixed-end-point, free-time* problem. Furthermore if for all $t \in T$ we have $f(\mathbf{x}, \mathbf{0}, t) = \mathbf{0}$ then the optimal-control problem is called a *regulator* problem.

Index

ASME [1995a], 81, 125
ASME [1995b], 81, 125
Adrianto et al. [2005], 6, 125
Aizerman et al. [1964], 5, 20, 125
Alenezi et al. [2005], 6, 125
Allwein et al. [2000], 44, 48, 125
Andrews [1972], 16, 125
Aronszajn [1950], 5, 55, 125
Athans and Falb [1966], 125, 145
Avriel and Wilde [1966], 95, 125
Badar et al. [2003], 80, 125
Badar et al. [2005a], 80, 125
Badar et al. [2005b], 80, 125
Bai and Zhang [2002], 67, 126
Balakrishna et al. [2008], 80, 126
Bartlett and Mendelson [2001], 38, 40, 41, 57, 126
Bartlett and Mendelson [2002], 38, 40, 41, 57, 126
Bazaraa et al. [2006], 62, 126
Beardslee and Trafalis [2005], 6, 126
Bentley [1975], 93, 126
Besl and McKay [1992], 80, 89, 126
Burges [1998], 32, 126
Carr and Ferreira [1995a], 81, 126
Carr and Ferreira [1995b], 81, 126
Chaitin [1966], 3, 126
Chen and Shen [2007], 67, 126
Chen [2004], 32, 127
Chung et al. [2002], 6, 127
Cochran [1977], 17, 92, 127
Coppersmith and Winograd [1990], 70, 127, 140
Courant and Hilbert [1953], 20, 127
Couto [2005], 25, 26, 127
Diamantaras and Kung [1996], 14, 96, 127
Ding and Dubchak [2001], 5, 127
Dorigo [1992], 100, 127
Evensen [1994], 117, 127
Fisher [1936], 2, 3, 14–16, 127
Gauss [1809], 2, 127
Gilbert and Trafalis [2009], 6, 60, 128
Gilbert et al. [2009a], 5, 128
Gilbert et al. [2009b], 6, 128
Gilbert et al. [2010], 5, 128
Girosi et al. [1995], 5, 128
Glover [1989], 100, 128
Glover [1990], 100, 128
Hamming [1950], 24, 128
Hastie and Stuetzle [1989], 96, 128
He and Garcia [2009], 17, 128
Hestenes and Stiefel [1952], 4, 67, 128
Hinton and Salakhutdinov [2006], 96, 128
Hocken et al. [1993], 81, 128
Holland [1975], 100, 128
Hooke and Jeeves [1961], 92, 128
Hopp [1993], 80, 129
Hulting [1992], 80, 129
Hurt and Colwell [1980], 81, 129
Ince and Trafalis [2003], 6, 129
Ince and Trafalis [2006a], 6, 129
Ince and Trafalis [2006b], 6, 129
Ince and Trafalis [2007], 6, 129
Kalman and Bucy [1961], 110, 129
Kennedy and Eberhart [1995], 100, 129
Kepler [1619], 2, 129
Kiefer [1953], 95, 129
Kim and Raman [2000], 80, 129
Kirkpatrick et al. [1983], 100, 129
Kohonen and Mäkisara [1986], 14, 96, 130
Kohonen [1982], 14, 96, 129
Kohonen [2001], 14, 96, 129
Kolmogorov [1965], 3, 130
Koltchinskii and Panchenko [2000], 5, 41, 57, 130
Kondor and Lafferty [2002], 26, 130
Kurfess and Banks [1995], 81, 130
LeCun [1986], 4, 130
Lee et al. [2003], 5, 130
Legendre [1805], 2, 130

Lorenz and Emanuel [1998], 115, 130
Malyscheff and Trafalis [2003], 6, 130
Malyscheff et al. [2002], 5, 80, 130
Mansouri et al. [2007], 6, 130
Mercer [1909], 4, 21, 22, 131
Mika et al. [1999], 14, 96, 131
Moustafa [2009], 16, 131
Murthy and Abdin [1980], 81, 131
Nelder and Mead [1964], 92, 131
Nelder and Mead [1965], 92, 131
Newton [1687], 2, 131
Novikoff [1963], 3, 131
Ohm [1827], 2, 131
Oladunni and Trafalis [2006], 5, 131
Oladunni et al. [2006], 5, 131
Pearson [1901], 3, 14, 90, 131
Platt et al. [2000], 44, 46, 132
Platt [1999], 5, 132
Prakasvudhisarn et al. [2002], 80, 132
Prakasvudhisarn et al. [2003], 6, 80, 132
Press [2007], 132, 138, 139
Raman et al. [2005], 6, 132
Requicha [1993], 81, 132
Rosenblatt [1958], 3, 132
Rosenbrock [1960], 92, 132
Roweis and Saul [2000], 96, 132
Roy and Zhang [1992], 81, 132
Rumelhart et al. [1986], 4, 132
Saad [2003], 133, 139
Saitoh [1988], 5, 21, 133
Samuel and Shunmugam [1999], 81, 133
Santosa et al. [2002], 5, 133
Santosa et al. [2007], 5, 133
Schölkopf et al. [1997], 14, 96, 133
Shawe-Taylor and Cristianini [2004], 11, 29, 30, 42, 133
Shunmugam [1987], 81, 133
Smola and Schölkopf [2003], 5, 133
Spendley et al. [1962], 92, 133
Steinbuch [1965], 3, 133
Strassen [1969], 70, 133, 140
Suykens and Vandewalle [1999], 5, 133
Tipping and Bishop [1999], 96, 134
Trafalis and Gilbert [2005], 6, 134
Trafalis and Gilbert [2006], 6, 134
Trafalis and Gilbert [2007], 6, 134
Trafalis et al. [2005], 5, 134
Trafalis et al. [2007], 6, 134
Tychonoff [1963], 3, 66, 134
Vapnik and Chervonenkis [1971], 4, 38, 39, 134
Vapnik and Chervonenkis [1991], 4, 134
Vapnik [1982], 4, 7, 37, 134
Vapnik [1995], 5, 7, 9, 44, 45, 134
Vapnik [1998], 5, 7, 134
Voronoi [1907], 83, 135
Wegner [1960], 25, 135
Widrow [1962], 3, 135
Woo and Liang [1993], 81, 135
Woodbury [1950], 66, 135
Yin [2007], 15, 135
Zadeh and Desoer [1963], 135, 146
Zangwill [1967], 92, 135
Černý [1985], 100, 134
d'Ocagne [1885], 2, 16, 127

Bootstrapping, 51

Classification, 37, 38, 44, 122
Computational complexity, 25, 67, 70, 73, 93
Condition number, 64, 66, 69
Coordinate Measuring Machine, 81, 89
Cross-validation, 49

Data thinning, 12, 17, 122, 124

Grid, 88, 93

Implicit function, 89, 91, 95

Kernel, 4, 7, 14, 20, 37, 52

Lorenz 96 model, 115

Mathematical programming, 6, 30, 48, 54, 69, 73
Mesh, 81, 82

Normalization, 16

Parametric function, 89, 93, 97

Principal Component Analysis, 2, 14, 90, 96

Quasi-Geostrophic model, 118

Rademacher complexity, 5, 38, 42, 57
Radial Basis Function, 30, 33
Registration, 89
Regression, 7, 36, 37, 43, 81, 84, 95, 99
Reproducing Kernel Hilbert Space, 4, 29, 32, 55
Risk, 4, 38, 40
Robustness, 31, 60, 76

State, 93, 110, 111, 145
Support Vector Machine, 5, 7, 37, 44, 61
Symmetric, 21, 24, 29, 30, 32, 54, 64, 68, 71

Tikhonov regularization, 3, 66, 143

VC dimension, 4, 38
Voronoi tessellation, 17