UNIVERSIDAD AUTÓNOMA DE MADRID

DOCTORAL THESIS

Machine Learning with Functional Data: Methodological Advances and Computational Tools

Author:
Carlos RAMOS CARREÑO

Supervisors:
Dr. Alberto SUÁREZ GONZÁLEZ
Dr. José Luis TORRECILLA NOGUERALES

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Programa de Doctorado en Ingeniería Informática y de Telecomunicación
Department of Computer Science
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Friday 26th May, 2023

“There should be no obstacles to accessing knowledge.”

Alexandra Elbakyan

Abstract

Functional data consist of observations that depend on a continuous parameter, such as time or space. These types of data appear in many problems of practical interest in economics, biology, medicine, and environmental sciences, among others. They present characteristics that are markedly different from multivariate data, which are the most prevalent object of study in statistics and machine learning. For these reasons, it is important to have at one's disposal tools that take into account and exploit the functional nature of the observations.

The main goal of this thesis is to design statistical methods and computational tools for machine learning with functional data. A first set of methodological contributions have been made for dimensionality reduction, clustering, and classification. In particular, we derive optimal rules for the classification of Gaussian processes. Special attention is devoted to the singular case in which the processes are orthogonal and near-perfect classification (zero Bayes error) is obtained asymptotically. A second contribution consists in an exhaustive theoretical and empirical analysis of recursive maxima hunting (RMH), a filter method for variable selection that takes advantage of the functional nature of the data. In recursive maxima hunting, variables are selected iteratively. At each step one selects the variable whose dependence with the class label, measured using the distance covariance, is strongest. Then, the corresponding contribution is removed by subtracting from each functional observation the expectation of the underlying process conditioned to the value of the variable selected. Finally, the behavior of the clustering method fuzzy C-means has been analyzed when applied to functional data.

In the second part of the thesis, a suite of computational tools for retrieval, representation, exploratory analysis, preprocessing, and machine learning for functional data is introduced. Specifically, the Python libraries **scikit-datasets** and **rdata** have been developed to handle multivariate and functional datasets. These packages facilitate the retrieval of these data from a variety of sources, their conversion to a unified format, and the empirical evaluation of machine learning methods that utilize them. A prominent contribution of this thesis is the development of the library **scikit-fda**, a Python package for Functional Data Analysis (FDA). It provides a comprehensive set of tools for statistical analysis and machine learning with functional data. The library is built upon and integrated in the scientific Python ecosystem. In particular, it conforms to the **scikit-learn** application programming interface so as to take advantage of the functionality for machine learning provided by this package: pipelines, hyperparameter tuning, and model selection, among others. Finally, the **dcor** package is an additional contribution of this thesis. This package provides tools to compute dependency measures, such as the aforementioned distance covariance, as well as related tests of homogeneity and independence. The computational tools developed as part of this thesis have been released as free open-source software, and are open to contributions from the scientific community.

Resumen

Los datos funcionales son observaciones que dependen de un parámetro continuo, como el tiempo o el espacio. Estos tipos de datos aparecen en muchos problemas de interés práctico en economía, biología, medicina, y ciencias medioambientales, entre otros. Presentan características muy diferentes de los datos multivariantes, los cuales son el objeto de estudio predominante en estadística y aprendizaje automático. Por estas razones, es importante disponer de herramientas que tengan en cuenta y aprovechen la naturaleza funcional de los datos.

El objetivo principal de esta tesis es diseñar métodos estadísticos y herramientas computacionales para aprendizaje automático con datos funcionales. En primer lugar, se han realizado contribuciones metodológicas para reducción de dimensionalidad, agrupamiento y clasificación. Concretamente, se han derivado reglas óptimas para la clasificación de procesos gaussianos. Con especial atención se estudia el caso singular en el que los procesos son ortogonales y se obtiene asintóticamente clasificación casi perfecta (error de Bayes cero). Una segunda contribución consiste en un análisis exhaustivo, tanto teórico como empírico, de *recursive maxima hunting* (RMH), un método de filtro para selección de variables que aprovecha la naturaleza funcional de los datos. En *recursive maxima hunting* las variables se seleccionan de manera iterativa. En cada paso se selecciona la variable cuya dependencia con la etiqueta de clase, cuantificada en términos de la covarianza de la distancia, es mayor. A continuación, se elimina la contribución correspondiente, sustrayendo de cada observación funcional el valor esperado del proceso subyacente condicionado al valor de la variable seleccionada. Finalmente, se analiza el comportamiento del método de agrupamiento borroso *fuzzy C-means* aplicado a datos funcionales.

En la segunda parte de la tesis se presentan una serie de herramientas computacionales para el acceso, representación, análisis exploratorio, preprocesamiento, y aprendizaje automático para datos funcionales. En concreto, las bibliotecas de Python **scikit-datasets** y **rdata** se han desarrollado para el manejo de conjuntos de datos multivariantes y funcionales. Estos paquetes permiten el acceso a datos almacenados en diversos repositorios, su conversión a un formato unificado, y la evaluación empírica de métodos de aprendizaje automático que hacen uso de los mismos. Una contribución destacada de esta tesis es el desarrollo de la biblioteca **scikit-fda**, un paquete de Python para el análisis de datos funcionales (FDA, por sus siglas en inglés). Proporciona un amplio conjunto de herramientas para el análisis estadístico y aprendizaje automático con este tipo de datos. La librería ha sido construida sobre la base del ecosistema científico de Python y se integra en el mismo. En particular, se adecúa a la interfaz de programación de aplicaciones de **scikit-learn**, de forma que sea posible aprovechar la funcionalidad de aprendizaje automático proporcionada por este paquete: flujos de trabajo, ajuste de hiperparámetros, y selección de modelos, entre otros. Finalmente, el paquete **dcor** es una contribución adicional de esta tesis. Este paquete proporciona medidas de dependencia, como la ya mencionada covarianza de la distancia, así como contrastes de homogeneidad e independencia relacionados. Las herramientas desarrolladas a lo largo de esta tesis han sido publicadas como software libre, de código abierto, y están abiertas a contribuciones por parte de la comunidad científica.

Acknowledgements

First and foremost, I want to thank my advisors, Alberto Suárez and José Luis Torrecilla, for their continued support. This thesis has been elaborated through difficult times for all of us, in which health problems, changes in our lives, and even a pandemic ocurred. I appreciate that they did not abandon me at any moment, even when they themselves struggled with personal issues. I also want to thank Dr. Fabian Scheipl for inviting me to do my PhD stay at the LMU in Munich during the confusing COVID-19 pandemic times, assisting me in my research, and for his continuous guidance and encouragement. I would like to give special thanks to all the people who read this manuscript and offered advice for its improvement. These include not only my advisors, but also Dr. Scheipl, Dr. Alaíz and Dr. Nagy. I feel that their feedback greatly improved the quality of the document.

I want to thank all my friends and colleages, who helped me to keep my sanity during this time. I would like to thank my colleages from the University: Javi, Luis Diego, Juanmi, Álex, Rubén, and all the others, which brought me joy not only during my undergraduate years but also during my PhD. I want to also thank my friends Sara, Enrique, and Tatiana. We used to be closer when I started this journey, but the work, the pandemic and the distance have prevented us to hang out more. We have to celebrate together when I come back to Spain! I also want to thank all the new friends that I made during my PhD years: María, Gonzalo, Pablo, Rocío, Edu, David, Carlos, and many others. I know I am not very good at keeping in touch, but that does not mean that I do not think about you.

I want to give special thanks to my family, that encouraged me to continue the PhD in spite of the difficulties. None of this would have been possible without you. In particular, I want to thank my parents, who had to support me at my worst moments during these years, as well as my siblings, Cristina and David. More thanks go to my grandpa and all my uncles and cousins, who really know how to make me feel loved. I want also to use this space to remember my two grandmothers, both of which died during these years. I know they would have been very proud of me for achieving this. I miss you.

I finally want to thank my girlfriend, Sydney. Although she was not here since the beginning of this thesis, it would not have been the same without her. If you are reading this, I apologize for all the stress that you had to deal with because of this work, and I hope we could go on vacation when it is over, as promised.

Contents

List of Figures

List of Tables

Acronyms

AIC Akaike's Information Criterion.

ANOVA Analysis Of Variance.

API Application Programming Interface.

BD Band Depth.

CD Critical Distance.

DD Depth Vs Depth.

DD^G Generalized Depth-depth Classifier.

DTM Distance To Trimmed Means.

FCM Fuzzy C-means.

FDA Functional Data Analysis.

FPCA Functional Principal Components Analysis.

GCV Generalized Cross Validation.

GP Gaussian Process.

k-**NN** *K*-nearest Neighbors.

LDA Linear Discriminant Analysis.

MBD Modified Band Depth.

MD Maximum Depth.

MEI Modified Epigraph Index.

MH Maxima Hunting.

mRMR Minimum-redundancy-maximum-relevance.

MS-plot Magnitude-shape Plot.

NC Nearest Centroid Classifier.

PCA Principal Components Analysis.

PLS Partial Least Squares.

QDA Quadratic Discriminant Analysis.

RBF Radial Basis Function.

REGSSE Registration Sum Of Squared Errors.

RF Random Forest.

RKC Reproducing Kernel Classification.

RKHS Reproducing Kernel Hilbert Space.

RKVS Reproducing Kernel-based Variable Selection.

RMH Recursive Maxima Hunting.

SRSF Square Root Slope Function.

SVM Support Vector Machine.

Symbols

In this work we will use lowercase letters to refer to scalars or functions. Thus, x can be a scalar or a function, and it should be clear from context. Sometimes we will use uppercase letters for variables that are limits of a summation or integral, like N. We will use lowercase bold letters to represent vectors, like $\mathbf{x}$, and uppercase bold letters to represent matrices, like $\mathbf{X}$. Uppercase letters are also used for sets, random variables and stochastic processes (X) and for random vectors ($\mathbf{X}$). All ambiguities in the notation will be clarified in the text. As a shortcut we will use the notation $x(\mathbf{t})$ referring to the vector of all evaluations $(x(t_1), x(t_2), \ldots)$. Other symbols used in this work are shown below:

$\hat{\square}$ Sample estimate.

$\|\cdot\|_p$ Natural norm in an L^p space.

$\|\cdot\|$ A generic norm.

B Number of basis elements used in a basis expansion.

$\mathcal{C}$ Space of continuous functions.

C Number of classes or clusters in classification and clustering tasks.

$\mathcal{D}$ A particular dataset.

D Depth function.

d Distance function.

δ_{ij} Kronecker delta function.

δ Shift of the continuous parameter in shift registration.

$\mathcal{E}$ Energy distance.

$\mathbb{E}$ Expected value.

ϕ Element of a basis of functions.

γ Warping function.

h Smoothing parameter.

$\mathcal{H}_k$ Reproducing kernel Hilbert space (RKHS) with kernel k.

$\mathbb{I}_A$ Indicator function on set A.

id_A Identity function on set A.

J Inertia function minimized in clustering algorithms.

$\mathcal{K}$ Covariance operator.

K Smoothing kernel.

k Covariance function, also called a kernel function, or simply, kernel. Not to be confused with smoothing kernel.

L^∞ Space of (equivalence classes of) bounded functions.

L^p Space of (equivalence classes of) p-integrable functions.

M Number of observed points per sample, for functional data, or number of dimensions, for multivariate data.

μ Mean function.

N Number of observations in a sample.

$\mathbb{P}$ Law of a stochastic process.

$\mathcal{R}$ Distance correlation.

$\mathcal{R}^*$ Partial distance correlation.

$\mathbb{R}$ Set of real numbers.

$\mathbf{S}$ Smoothing matrix.

$\mathcal{T}$ Domain of the functional observations.

U Membership matrix in fuzzy clustering algorithms.

u Membership degree in fuzzy clustering algorithms.

$\mathcal{V}$ Distance covariance.

$\mathcal{V}^*$ Partial distance covariance.

ω Fuzzifier used in fuzzy clustering algorithms.

$\mathcal{X}$ Generic functional space.

Ξ Penalty function for smoothing.

$\mathcal{Y}$ Generic space of targets in supervised problems.

Chapter 1

Introduction

Numerous machine learning problems of practical interest involve data with a complex structure (Marron and Dryden, 2021). These include, for example, text (Sebastiani, 2002), pieces of music (Weihs et al., 2016), time series (Bagnall et al., 2017), shapes (Srivastava and Klassen, 2016), images (Litjens et al., 2017), and graphs (Guo, Srivastava, and Sarkar, 2021). For these types of data, it is necessary to develop methods that go beyond the classical multivariate paradigm, in which observations are encoded as vectors of real-valued or categorical attributes. A particularly interesting kind is functional data, in which the observations depend on a continuous parameter. This parameter can be a real number, such as time for trajectories, or frequency for the spectra of chemical substances, in which case the data are curves in one dimension. One can consider also observations that depend on a continuous parameter in two or more dimensions, such as surfaces, 2 and 3-D images, and videos.

Functional data arise naturally in a variety of fields. For example, in medicine, growth curves, electrocardiograms, electroencephalograms, and other measures of health indicators over time can be used for diagnosis and patients' follow-up. Images are also collected by means of X-ray radiography, magnetic resonance imaging (MRI) and other techniques. The analysis of these data is key to addressing important problems in public health, such as cancer studies (Crawford et al., 2020), detection of potential health risks (Maturo and Verde, 2022) or, in recent times, the evaluation of the effects of the COVID-19 pandemic (Boschi et al., 2021; Oshinubi et al., 2022). Meteorology is another discipline in which functional data commonly appears; for example, when analyzing temperature, precipitation, and wind speed as a function of time and location on the surface of the Earth. The study of these types of data has direct and profound impact in society, as decision-making tools. For instance, to determine the optimal placement of renewable energy facilities (Tapia et al., 2022) or to predict the effects of climate change (Ghumman et al., 2019), among others. Functional data are common also in other areas of application, such as biomechanics, finance, psychology, and linguistics (Ullah and Finch, 2013).

Functional observations present some characteristics that are very different from those of multivariate data (Cuevas, 2014; Wang, Chiou, and Müller, 2016). They are often assumed to be continuous and, in some cases, smooth as well. They have a high degree of redundancy because nearby points in the functions tend to be strongly correlated. In particular, the values of the functions cannot be reordered without altering the nature of the problem. This is in contrast to the multivariate case, in which the observations are not affected by permutations of the components of the attribute vectors. Additionally, due to the infinite-dimensional nature of functional data, there is no reference measure analog to the Lebesgue measure of finite-dimensional spaces (Cuevas, 2014). Thus, it is not possible to define probability densities, which are fundamental mathematical objects in multivariate statistics. New phenomena also appear, such as near-perfect classification (Delaigle and Hall, 2012).

The continuous and smooth nature of the data opens up the possibility of using additional mathematical tools. For instance, the information contained in the derivatives can be exploited to uncover patterns associated with the rate of variation of the functions. Local smoothing techniques can be used for noise reduction. The redundancy of the data can be leveraged to design variable selection procedures, in which the selected points embody the information of a whole range of points in their neighorhood. Because of their functional nature one could also utilize the tools of the theory of stochastic processes (Doob, 1990) and functional analysis (Rudin, 1991). In particular, in this thesis we make extensive use of the theory of reproducing kernel Hilbert spaces (RKHS), which bridges the two fields (Berlinet and Thomas-Agnan, 2004).

The distinct characteristics of functional data pose important challenges. Multivariate methods may have poor performance due to the high dimensionality of the data, or the collinearity of nearby points, among other reasons. In some cases, it is possible to sidestep these difficulties by applying dimensionality reduction techniques or approximating the data by a truncated basis expansion. However, in general, the functional nature of the data requires a novel framework, which is provided by functional data analysis (FDA). Functional data analysis is the branch of statistics that deals with mathematical objects that depend on a continuous parameter, such as curves or surfaces (Ramsay and Silverman, 2005; Kokoszka and Reimherr, 2017). In this framework, functional observations are assumed to be realizations of a random function (Ferraty and Vieu, 2006; Hsing and Eubank, 2015). These random functions are akin to random vectors in multivariate statistics. This correspondence makes it possible to formulate standard machine learning problems (e.g., classification, regression, and clustering) with functional data as direct extensions of their multivariate counterparts. FDA deals also with problems that are specific to functional data. One such problem is registration, in which one attempts to account for the potential misalignment of the data (Marron et al., 2015).

The objective of this thesis is to design machine learning methods and computational tools for functional data. In particular, optimal classification rules have been derived for the classification of Gaussian processes (Torrecilla et al., 2020). The approach taken is to consider the functional case as the limit of discretizations in a grid that becomes progressively finer. Using this approach, near-perfect classification arises when the quadratic discriminant, which is the optimal classification rule for the discretized data, presents terms that diverge in this limit. When these divergences cancel out, the two processes are equivalent and the optimal classification rule can be expressed in terms of the Radon-Nikodym derivative (Baíllo, Cuevas, and Cuesta-Albertos, 2011; Berrendero, Cuevas, and Torrecilla, 2018). Explicit formulas are derived and illustrated empirically for specific problems.

A second contribution is an exhaustive theoretical and empirical analysis of recursive maxima hunting (RMH), a filter variable selection method for functional data (Torrecilla and Suárez, 2016). Recursive maxima hunting is an iterative method. At each iteration one selects the variable that has the strongest dependence with the class label. This dependence is quantified using the distance covariance (Székely, Rizzo, and Bakirov, 2007). The contribution of each selected variable is then removed by subtracting the expectation of the underlying process conditioned to the value observed. In this analysis, different assumptions of the underlying noise process are considered. We prove that, under certain conditions, the algorithm selects precisely the points that appear in the optimal classification rule. An empirical comparison between RMH and other variable selection methods is also carried out.

A final methodological contribution is the study of the behavior of the clustering method fuzzy C-means (FCM) when it is applied to functional data in discretized form (Ramos-Carreño, 2023). This algorithm presents convergence problems in the high-dimensional multivariate case. We explored FCM considering simulated data from a Gaussian process in which the dependences between function values have a characteristic lengthscale. When the extent of the correlations is small compared to the resolution of the discretization grid, FCM has convergence problems as in the multivariate high-dimensional case. In contrast, the algorithm generally converges well when the lengthscale is increased and is comparable or larger to the spacing between grid points. This corresponds to a case in which nearby points are more correlated, as is common in functional datasets. In this manner we observe a case in which the behavior changes in a smooth way from the multivariate case to a more functional one.

Computational tools play a crucial role in both scientific research and applications. The design and implementation of open source software facilitates the reuse of previous work in research, thus reducing mistakes, fostering reproducibility, and the comparison between different methods. In addition, these tools lower the entry barrier for new practicioners interested in the practical application of these techniques, and thus broaden the appeal of the subject. When particular care is taken in maintaining high quality documentation and examples, these tools are also a didactical resource, which provides learning opportunities for people outside the field. Thus, a second group of contributions consists in the development of computational tools for handling datasets, statistical analysis, and machine learning methods to functional data:

- The package **scikit-datasets** is a Python library that can be used to retrieve multivariate and functional datasets from different repositories widely used by the scientific community. It also provides utilites to carry out empirical evaluation studies that employ these datasets and for the presentation of the results in the form of tables.

- The package **rdata** is a library to convert datasets in the rda and rds formats, typical of the R programming language, into Python objects. In combination with **scikit-datasets** this makes possible to load a wide variety of datasets only available in CRAN, the repository of R packages.

- The Python package **scikit-fda** (Ramos-Carreño et al., 2023) offers a variety of representation, preprocessing and machine learning tools for functional data. The package includes standard analysis tools, as well as all the methods implemented in this thesis. It is integrated in the scientific Python ecosystem, and can be used in combination with the tools of the widely-used machine learning package **scikit-learn**. The implementation work is complemented by detailed documentation including tutorials and examples of use (Ramos-Carreño et al., 2022).

- The **dcor** package (Ramos-Carreño and Torrecilla, 2023) provides fast implementations of the nonlinear dependence measures distance covariance and correlation. It provides also support for dependence and homogeneity tests based on related measures. This library has been used in several scientific articles and software tools. In particular, it is used in the RMH algorithm presented in this thesis to compute the dependence between the attributes that characterize an instance and the class label.

The computational tools developed are published as free and open-source software packages in the scientific Python ecosystem. They have been designed to be easily used along with other packages in that environment, and they are open to contributions from the scientific community.

1.1 Publications related to the thesis

The research presented in this thesis has given rise to several publications in international journals and contributions to conferences in the areas of computational statistics and machine learning. The methodological advances that are presented in Part I are described in the following documents:

- J. L. Torrecilla, C. Ramos-Carreño, M. Sánchez-Montañés, and A. Suárez (2020). "Optimal Classification of Gaussian Processes in Homo- and Heteroscedastic Settings". In: *Statistics and Computing* 30.4, pp. 1091–1111. ISSN: 1573-1375. DOI: 10.1007/s11222-020-09937-7
- J. L. Torrecilla, C. Ramos-Carreño, and A. Suárez (2023). "Recursive maxima hunting: Variable selection in functional data classification". (In preparation)
- C. Ramos-Carreño (2023). "Fuzzy C-means: Differences on Clustering Behavior between High Dimensional and Functional Data (Student Abstract)". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37

The computational tools described in Part II have been presented in the following journals and conferences:

- C. Ramos-Carreño, J. L. Torrecilla, M. Carbajo-Berrocal, P. Marcos, and A. Suárez (2023). "scikit-fda: A Python Package for Functional Data Analysis". In: *Journal of Statistical Software*. (Accepted for publication. Preprint available at http://arxiv.org/abs/2211.02566)
- C. Ramos-Carreño, J. L. Torrecilla, Y. Hong, and A. Suárez (2022). "scikit-fda: Computational Tools for Machine Learning with Functional Data". In: *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 213–218. DOI: 10.1109/ICTAI56018.2022.00038
- C. Ramos-Carreño, J. L. Torrecilla, and A. Suárez (2022). "Classification of Functional Data: A Comparative Study". In: *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 866–871. DOI: 10.1109/ICMLA55696.2022.00143
- C. Ramos-Carreño and J. L. Torrecilla (2023). "dcor: Distance Correlation and Energy Statistics in Python". In: *SoftwareX* 22, p. 101326. ISSN: 2352-7110. DOI: 10.1016/j.softx.2023.101326

As a result of the research carried out in this thesis the following free and open-source software packages have been released:

- D. Díaz-Vico and C. Ramos-Carreño (2022). *scikit-datasets: scikit-learn-compatible Datasets*. DOI: 10.5281/zenodo.6383047. URL: https://github.com/daviddiazvico/scikit-datasets (visited on 05/22/2023)

- C. Ramos-Carreño (2022). *rdata: Read R datasets from Python*. DOI: 10.5281/zenodo.6382237. URL: https://github.com/vnmabus/rdata (visited on 05/22/2023)
- C. Ramos-Carreño, A. Suárez, J. L. Torrecilla, M. Carbajo Berrocal, P. Marcos Manchón, P. Pérez Manso, A. Hernando Bernabé, D. García Fernández, Y. Hong, P. M. Rodríguez-Ponga Eyriès, Á. Sánchez Romero, and E. Petrunina (2022). *scikit-fda: Functional Data Analysis in Python*. DOI: 10.5281/zenodo.3468127. URL: https://github.com/GAA-UAM/scikit-fda (visited on 05/22/2023)
- C. Ramos-Carreño (2020). *dcor: Distance Correlation and Related E-Statistics in Python*. DOI: 10.5281/zenodo.3468124. URL: https://github.com/vnmabus/dcor (visited on 05/22/2023)

1.2 Structure of the thesis

The thesis is structured as follows: Chapter 2 presents an overview of concepts and methods in FDA, the branch of statistics that deals with random functions. A description of the functional datasets that are used in this thesis can be found in Chapter 3. After this chapter, the thesis is divided in two parts.

Part I comprises Chapters 4, 5 and 6, which focus on the methodological advances obtained during this thesis. In Chapter 4 we derive optimal rules for the classification of Gaussian processes as a limit of the quadratic discriminant rule for multivariate problems. In Chapter 5 we propose a general version of the variable selection algorithm RMH, and analyze it in the context of binary classification. The convergence properties of the clustering algorithm FCM when applied to multivariate and to functional data are studied in Chapter 6.

Part II consists of Chapters 7, 8, and 9. In these chapters the focus is on the computational tools developed for working with functional data. In Chapter 7 we present the software packages **scikit-datasets** and **rdata**. These can be used to handle multivariate and functional datasets. The package **scikit-fda**, which provides tools for statistical analysis and machine learning with functional data, is described in Chapter 8. This chapter includes also several examples of use in real-data problems. To conclude this chapter, a comparison between different classifiers for functional data is carried out using the tools in **scikit-fda**. The package **dcor**, which includes measures and tests for homogeneity and dependence of random vectors, is described in Chapter 9.

Finally, in Chapter 10 we summarize the conclusions of the thesis and propose avenues of exploration for future work.

Chapter 2

Learning from functional data

Functional data analysis (FDA) is the branch of statistics that deals with observations varying over a continuous parameter, such as curves, surfaces, and other types of functions (Cuevas, 2014; Wang, Chiou, and Müller, 2016). These types of data appear in many different fields of application, such as biology (Cremona et al., 2019), demographics (Hyndman and Shahid Ullah, 2007), economics (Frois Caldeira et al., 2020), energy (Gong, Wang, and Lin, 2021), genomics (Leng and Müller, 2006; Chen et al., 2020), medicine (Sørensen, Goldsmith, and Sangalli, 2013; Ferrando, Ventura-Campos, and Epifanio, 2020; Horsley et al., 2021), meteorology (Beyaztas and Yaseen, 2019), oceanography (Assunção et al., 2020), traffic control (Wagner-Muns et al., 2018; Hu et al., 2019), and other areas of application (Ullah and Finch, 2013).

The functional observations considered are of the form $x : \mathcal{T} \subseteq \mathbb{R}^p \to \mathbb{R}^q$, were $p, q \geq 1$. The parameter p is the dimension of the domain, which is 1 for curves, 2 for surfaces or images, etc. The dimension of the codomain q is the number of coordinates for vector valued functions. Thus, a grayscale two-dimensional image can be considered as a functional datum with $p = 2$, for the location of the pixels in the image, and $q = 1$, for the intensity at each pixel. A color image consisting of three channels (e.g., red, green, and blue) would have $p = 2$ and $q = 3$. The functions are assumed to belong to a *functional space* $\mathcal{X}$, such as L^2, the space of square integrable functions, or a particular reproducing kernel Hilbert space (RKHS) (Berlinet and Thomas-Agnan, 2004). In this work, we focus mainly on the case of real-valued univariate functions (that is, $p = q = 1$). We assume that the domain $\mathcal{T}$ of the functional observations is a compact interval. Typically, the continuous parameter on which the functions depend, $t \in \mathcal{T}$, is assumed to be time.

A functional dataset consists then of N observations $\{x_i(t), t \in \mathcal{T}\}_{i=1}^N$, where $x_i(t)$ is the i-th observation in the sample. Functional observations are usually measured at a grid of points $\mathbf{t} = (t_1, \ldots, t_M) \in \mathcal{T}^M$, which need not be regularly spaced. An observation x in the sample is represented by the vector $x(\mathbf{t})$, where $x(\mathbf{t}) = (x(t_1), \ldots, x(t_M))$. The discretization grid is assumed to be sufficiently fine so that the functional character of the data is apparent (Ramsay and Silverman, 2005; Ferraty and Vieu, 2006; Hsing and Eubank, 2015). As an illustration, three trajectories measured at a grid of irregularly-spaced points are displayed in Figure 2.1.

Alternatively, a functional observation can be represented as an expansion in a functional basis $\{\phi_b(t), t \in \mathcal{T}\}_{b \geq 1}$ of $\mathcal{X}$

$$x(t) = \sum_{b \geq 1} c_b \phi_b(t), \tag{2.1}$$

where $\{c_b\}_{b \geq 1}$ are the coefficients of the expansion.

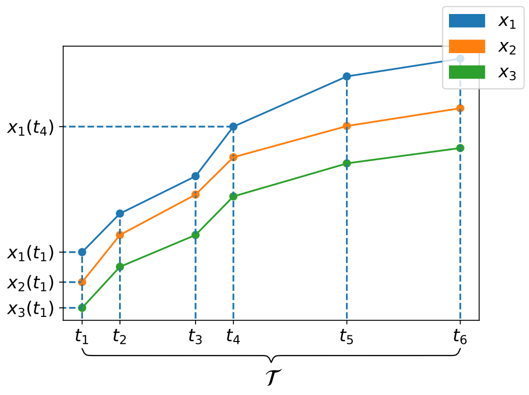


FIGURE 2.1: Functional observations in discretized form. The quantity $x_i(t_m)$ represents the value of the i-th trajectory at t_m.

It is commonly assumed that the data correspond to noisy observations of underlying functions that are continuous and smooth (with one or more derivatives). This means that the data, even when they are measured only at a finite number of points in a grid, present notable differences with their traditional multivariate counterparts (Cuevas, 2014; Wang, Chiou, and Müller, 2016). Their continuous structure means that the components of the observations $x(\mathbf{t}) = (x(t_1), \ldots, x(t_M))$, cannot be freely reordered, as was possible in the multivariate case. Another consequence is that each feature is strongly correlated with the nearby ones. This redundancy can be a source of difficulties in some cases. For instance, in variable selection methods it becomes necessary to control for it, as we will explain in Chapter 5. The infinite dimension of the functional data spaces of interest, also poses additional challenges. For example, there is no infinite-dimensional analog of the Lebesgue measure (Cuevas, 2014). As a result, it is not possible to define probability density functions for stochastic processes in general (Delaigle and Hall, 2010). Moreover, some useful multivariate constructs, such as the Mahalanobis distance and the linear and quadratic discriminant analyses (among others), cannot be naively extended to the functional context, and need to be adapted (Berrendero, Bueno-Larraz, and Cuevas, 2020). Both of these will be of special relevance in Chapter 4.

In the remaining part of the chapter we provide a more thorough description of the main theoretical concepts required for a complete understanding of this thesis. We first introduce formally the concept of stochastic processes, the random variables from which functional observations are drawn, in Section 2.1. A special emphasis is placed in Gaussian processes, the functional analog to the normal distribution, which are very important in this work. The discussion continues in Section 2.2 with the explanation of the different functional spaces in which the observations may live. A particular family of functional spaces are the RKHS, which are also very related with the theory of stochastic processes. In particular, each second order stochastic process has a related RKHS, associated with is covariance function. As explained in that section, studying the RKHS associated to a process we can derive some properties of interest of the process. Finally, Section 2.3 presents some of the machine learning problems with functional data which are of interest for this thesis. In particular, we start with registration, a new problem arising with functional data. In this problem, one tries to align the functional observations in order to improve the performance in further processing. We also explain the problem of dimensionality reduction, whose relevance is even higher in the functional case, as functional observations are in principle infinite-dimensional. Using dimensionality reduction techniques it is possible, in some cases, to summarize each observation as a vector in a low-dimensional space, and thus use the methods from multivariate statistics in further analysis. In addition, in this section the classical machine learning problems of regression, classification and clustering are also reframed in the functional data context.

2.1 Stochastic processes

Given a probability space $(\Omega, \mathcal{F}, P)$, a *stochastic process* is a collection $\{X(t, \omega) \mid \omega \in \Omega\}_{t \in \mathcal{T}}$ of random variables, indexed by the parameter $t \in \mathcal{T}$. When $\mathcal{T} \subseteq \mathbb{R}^p$, with $p > 1$, it is also common to use the name *random field* to refer to this collection. For a fixed value of t, $X(t, \omega)$ is a random variable. In particular, we will consider that the random values take values in a subset $S \subseteq \mathbb{R}^q$, and we will denote as $\mathcal{B}(S)$ the Borel σ-algebra in S. As it is common in probability theory, we will drop the ω in the notation, which allows us to refer to the process as X and to a particular random variable at index t with the notation $X(t)$.

When X is jointly measurable with respect to both t and ω, an alternative interpretation is possible (Hsing and Eubank, 2015). Let $\mathcal{X}$ be a space of functions from $\mathcal{T}$ to $S \subseteq \mathbb{R}^q$, so that for a fixed value of ω, $X(\,\cdot\,, \omega) \in \mathcal{X}$. Then, one can also see a stochastic process as a *random function* $X : \Omega \to \mathcal{X}$, and consider the dataset $\{x_i\}_{i=1}^N$ as a set of particular realizations of that random function.

Using this interpretation, we can define a measure $\mathbb{P}$ in $\mathcal{X}$, called the *law* of the stochastic process, as the pushforward measure

$$\mathbb{P}(B) = P(X^{-1}(B)) \quad B \in \mathcal{G}, \tag{2.2}$$

where $\mathcal{G}$ is a σ-algebra in $\mathcal{X}$. This measure is of special importance in FDA, as there is no analog of the Lebesgue measure for infinite-dimensional spaces, and thus there are no density functions. In the places where a density function would appear in the multivariate case, we will instead need to use alternative formulations using the laws of the stochastic processes involved.

In this work we will assume that the stochastic processes involved are of *second order*. That is, a random process X will have a finite mean

$$\mu(t) = \mathbb{E}[X(t)] \tag{2.3}$$

and a finite covariance function or *kernel*

$$k(s, t) = \mathbb{E}[(X(s) - \mu(s))(X(t) - \mu(t))]. \tag{2.4}$$

2.1.1 Gaussian processes

The normal or Gaussian distribution is of great importance in multivariate statistics, both because of its crucial role in the central limit theorem, and its analytical properties. For stochastic processes, the analog to the multivariate normal are *Gaussian processes (GPs)*. A stochastic process is Gaussian if and only if for every finite set of indices $t_1, \ldots, t_P \in \mathcal{T}$, the random vector $(X(t_1), \ldots, X(t_P))$ has a multivariate Gaussian distribution (Rasmussen and Williams, 2005). Analogously to the multivariate case, a Gaussian process is completely determined given its mean function μ and covariance function k. In this section we introduce several Gaussian processes, which are relevant for this thesis.

Brownian motion (Doob, 1942), which is also referred to as the Wiener process, is characterized by the covariance function

$$k_{BM}(s,t) = \sigma^2 \min(s,t) \quad s,t \geq 0, \tag{2.5}$$

with $\sigma \geq 0$. When $\sigma = 1$, and the process has zero mean, it is called standard Brownian motion (B), as depicted in the left plot of Figure 2.2. This σ parameter is common to all kernels and controls the dispersion. The definition of this process can be extended to $s,t < 0$ (Mishura and Shevchenko, 2017) as

$$k_(s,t) = \sigma^2 \frac{|s| + |t| - |s-t|}{2}. \tag{2.6}$$

This stochastic process is called two-sided Brownian motion, and it is shown in the right plot of Figure 2.2. Again, when $\sigma = 1$, and the mean is 0, it is called standard two-sided Brownian motion (B) As its name and plot indicates, this is equivalent to two independent Brownian motions spanning from 0 in opposite directions. Thus, given two independent standard Brownian motions B_1, B_2, a two-sided Brownian motion X can be expressed as

$$X(t) = \begin{cases} \sigma B_1(t) & \text{if } t \geq 0, \\ \sigma B_2(-t) & \text{if } t < 0. \end{cases} \tag{2.7}$$

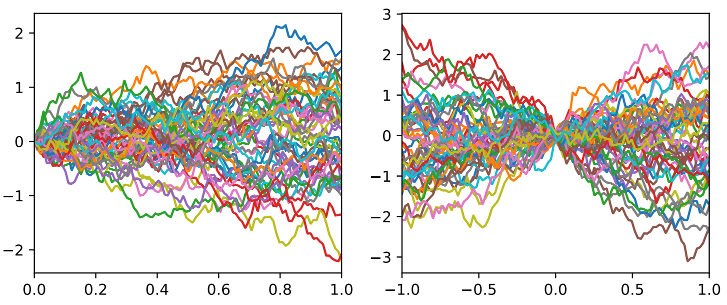


FIGURE 2.2: The left plot shows trajectories sampled from standard Brownian motion. In the right plot the trajectories are sampled from two-sided Brownian motion instead.

If one conditions Brownian motion to having the value 0 in an additional point T, apart from the origin, we obtain the so called Brownian bridge. This Gaussian process has the covariance

$$k_{BB}(s,t) = \sigma^2 \left(\min(s,t) - \frac{st}{T} \right). \tag{2.8}$$

In particular, the standard Brownian bridge BB has $\sigma = 1, T = 1$, as shown in Figure 2.3. A Brownian bridge X can also be expressed in terms of a standard Brownian motion B as

$$X(t) = \sigma \left(B(t) - \frac{t}{T} B(T) \right). \tag{2.9}$$

Some stochastic processes preserve their distribution when shifts are performed in the domain $\mathcal{T}$. These are called *stationary* processes. For Gaussian processes, in particular, this can be achieved when the mean μ is constant and the covariance

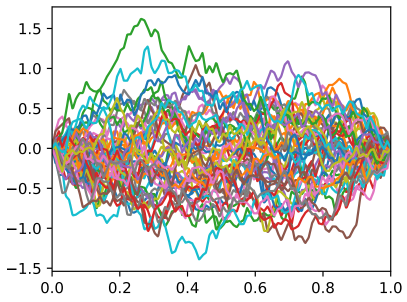


FIGURE 2.3: Plot of trajectories from a standard Brownian bridge.

depends only on the distance between the inputs, that is $k(s,t) = k(|s-t|,0) = k(|s-t|)$. Thus any covariance k of a stationary Gaussian process can be considered a *radial basis function (RBF)*, as it only depends on the distance $|s-t|$.

Of the stationary Gaussian processes, we can consider the one having as its covariance the *Gaussian RBF*

$$k_{RBF}(s,t) = \sigma^2 \exp\left(-\frac{|s-t|^2}{2l^2}\right), \tag{2.10}$$

with $l > 0$. This kernel is so popular in the literature of support vector machines (SVMs) and Gaussian process regression (Rasmussen and Williams, 2005), that it is usually referred simply as the RBF kernel, in spite of the existence of additional RBF covariances.

The parameter l, commonly called the *lengthscale*, characterizes the strength of the correlation between nearby points. Plots of the Gaussian RBF process with different lengthscales are shown in Figure 2.4.

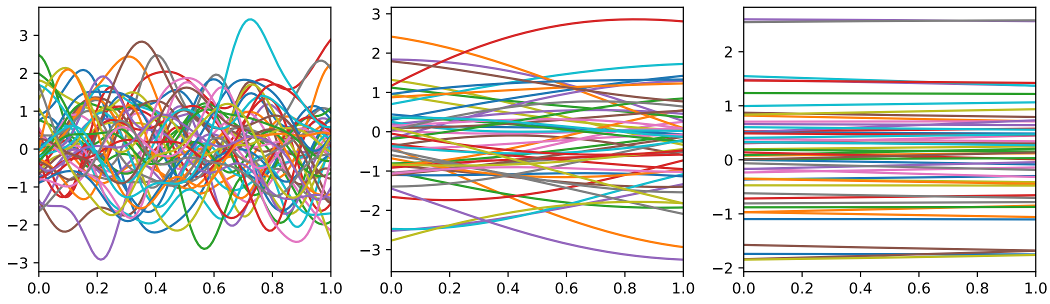


FIGURE 2.4: From left to right, plots of trajectories sampled from Gaussian RBF processes with lengthscales of 0.1, 1 and 10, respectively.

One apparent property of trajectories sampled from the Gaussian RBF process is their smoothness. In fact, these trajectories are infinitely differentiable everywhere. In some applications, however, the observed functions are not that smooth. For example, the Ornstein-Uhlenbeck process, that has the exponential covariance function

$$k_{OU}(s,t) = \sigma^2 \exp\left(-\frac{|s-t|}{l}\right), \tag{2.11}$$

has trajectories that are still continuous but nowhere differentiable, as shown in Figure 2.5.

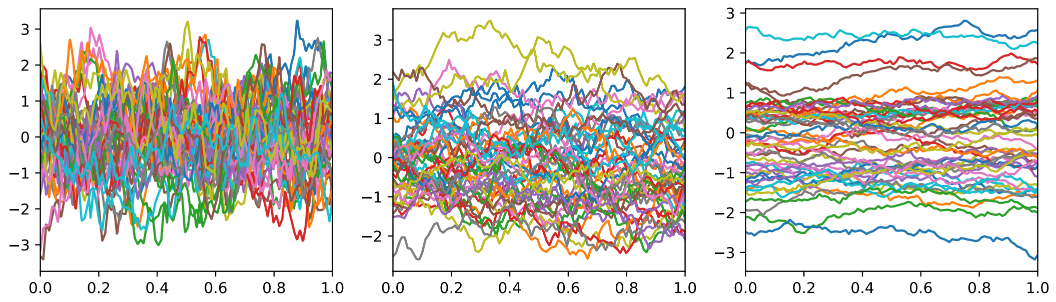


FIGURE 2.5: From left to right, plots of trajectories sampled from an Ornstein-Uhlenbeck process with lengthscales of 0.1, 1 and 10, respectively.

A generalization of both the Gaussian RBF process and the Ornstein-Uhlenbeck process is the family of Gaussian processes with Matérn kernels. A general Matérn kernel has the form

$$k_\nu(s,t) = \sigma^2 \cdot \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{|s-t|\sqrt{2\nu}}{l} \right)^\nu K_\nu \left(\frac{|s-t|\sqrt{2\nu}}{l} \right). \tag{2.12}$$

The parameter $\nu > 0$ controls the smoothness of the trajectories, as they are $\lceil \nu \rceil - 1$ times differentiable, while K_ν is a modified Bessel function of the second kind. If $\nu = p + \frac{1}{2}$, with p a non-negative integer, this formula further simplifies to

$$k(s,t) = \sigma^2 \exp\left(-\frac{|s-t|\sqrt{2\nu}}{l} \right) \frac{p!}{(2p)!} \sum_{i=0}^{p} \frac{(p+i)!}{i!(p-i)!} \left(\frac{|s-t|\sqrt{8\nu}}{l} \right)^{p-i}. \tag{2.13}$$

For $\nu = 1/2$ this gives the previously mentioned exponential kernel

$$k(s,t) = \sigma^2 \exp\left(-\frac{|s-t|}{l} \right), \tag{2.14}$$

which is thus an element of this family. For $\nu = 3/2$ we obtain

$$k(s,t) = \sigma^2 \left(1 + \frac{\sqrt{3}|s-t|}{l} \right) \exp\left(-\frac{\sqrt{3}|s-t|}{l} \right), \tag{2.15}$$

with one-time differentiable trajectories while for $\nu = 5/2$ we have

$$k(s,t) = \sigma^2 \left(1 + \frac{\sqrt{5}|s-t|}{l} + \frac{5|s-t|^2}{l^2} \right) \exp\left(-\frac{\sqrt{5}|s-t|}{l} \right), \tag{2.16}$$

whose trajectories are twice differentiable. These three processes are depicted in Figure 2.6. When $\nu \to \infty$, one obtains the Gaussian RBF process, with infinitely differentiable trajectories.

For the aforementioned stationary Gaussian processes, we could take the limit $l \to 0$. In this case we obtain the covariance function

$$k_{WN}(s,t) = \sigma^2 \delta_{st}, \tag{2.17}$$

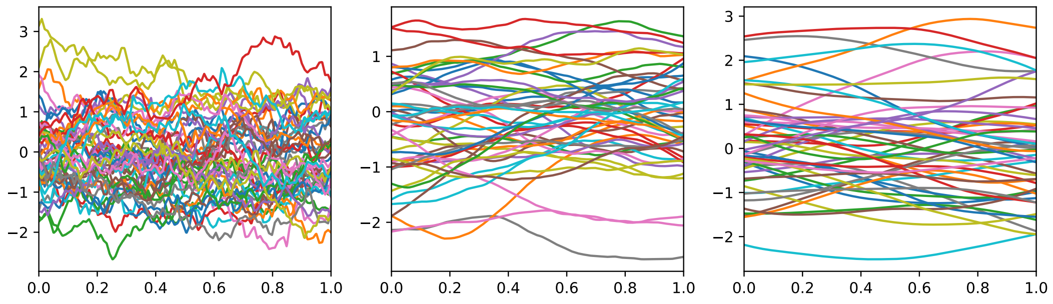


FIGURE 2.6: From left to right, plots of trajectories sampled from a Matérn process with $\nu = 1/2$ (Ornstein-Uhlenbeck), $\nu = 3/2$ and $\nu = 5/2$ respectively.

with nonzero values only in the diagonal. This corresponds to the so-called white noise, shown in the left plot of Figure 2.7.

We can also consider the limit $l \to \infty$. In this case, the covariance function becomes the constant covariance kernel

$$k_C(s,t) = \sigma^2, \tag{2.18}$$

as illustrated in the right plot of Figure 2.7.

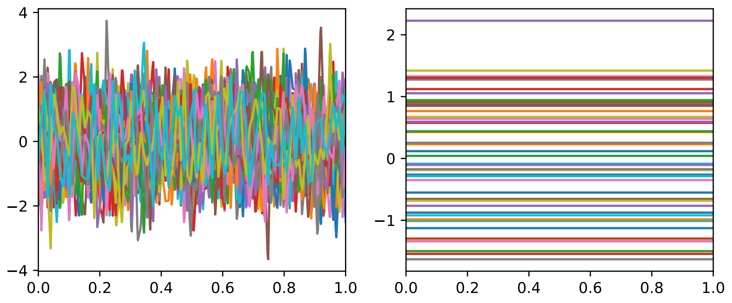


FIGURE 2.7: On the left plot, trajectories sampled from white noise are shown. The right plot corresponds to trajectories from a Gaussian process with constant covariance.

2.2 Functional spaces

As mentioned before, in FDA the functional observations belong to a common functional space $\mathcal{X}$. This space is assumed to be closed under addition and multiplication by an scalar, and is thus a vector space. For example, the space $\mathcal{C}(\mathcal{T})$ of continuous functions defined in $\mathcal{T}$, verifies this, as the result of adding two continuous function or the product of a continuous function by an scalar is still a continuous function.

Some functional spaces are in addition *metric spaces*, that is, they have a particular metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined for all possible pairs of elements of the space. A particular case of these are normed spaces, in which a norm $\|\cdot\| : \mathcal{X} \to \mathbb{R}$ can be used to measure the "length" of a particular element. A metric can be induced from the norm as $d(x_1, x_1) = \|x_1 - x_2\|$.

To define limits in a normed space it is necessary that the space is *complete* with respect to the induced metric, that is, that every Cauchy sequence converges to an element of the space. A normed space with this property is called a *Banach space*. A widely used family of functional Banach spaces is the collection of L^p spaces with $p > 0$,

$$L^p(\mathcal{T}) = \left\{ x : \mathcal{T} \to \mathbb{R} \middle| \int_{\mathcal{T}} |x(t)|^p dt < \infty \right\}, \tag{2.19}$$

where the (semi)norm in this space, or L^p-norm is

$$\|x\|_p = \sqrt[p]{\int_{\mathcal{T}} |x(t)|^p dt}. \tag{2.20}$$

In order for the $\|\cdot\|_p$ to be a proper norm, it needs to verify that $\|x\|_p = 0 \Leftrightarrow x = 0$. This is only the case if we consider that two functions in this space are equal if they agree almost everywhere. Therefore, properly speaking, L^p is not a space of functions, but a space of equivalence classes of functions. However, as our functions of interest do not have removable discontinuities, this distinction can be ignored in practice. We can also provide a definition for the limit case $p = \infty$ as

$$L^\infty(\mathcal{T}) = \{x : \mathcal{T} \to \mathbb{R} | |x(t)| < \infty \text{ almost everywhere}\}, \tag{2.21}$$

with the (semi)norm

$$\|x\|_\infty = \sup_{t \in \mathcal{T}} |x(t)|. \tag{2.22}$$

Finally, a *Hilbert space* is a Banach space whose norm is induced by an inner product $\langle \cdot , \cdot \rangle$ as

$$\|x\| = \sqrt{\langle x, x \rangle}. \tag{2.23}$$

This inner product is of crucial importance for several applications, such as projections, change of basis or the notion of orthogonality. Of the functional spaces mentioned so far, only L^2 is a Hilbert space, with

$$\langle x_1, x_2 \rangle = \int_{\mathcal{T}} x_1(t) x_2(t) dt. \tag{2.24}$$

Note that this definition is similar to the Euclidean inner product in finite dimension, with the sums turned into integrals. As L^2 is the most commonly used Hilbert space, we will use the notation $\langle \cdot , \cdot \rangle$ to refer to the inner product in L^2 in the rest of this work.

2.2.1 Reproducing Kernel Hilbert Spaces

A reproducing kernel Hilbert space (RKHS) is a space of real-valued functions on $\mathcal{T}$ endowed with an inner product that is a reproducing kernel. A kernel $k(s,t)$ is a symmetric positive-semidefinite function on $\mathcal{T} \times \mathcal{T}$, which throughout this work will also be assumed to be continuous and bounded. The reproducing property for the kernel k associated with the RKHS $\mathcal{H}_k$ implies that

$$x(t) = \langle x(\cdot), k(\cdot, t) \rangle_k, \forall x \in \mathcal{H}_k, \tag{2.25}$$

where $\langle \cdot , \cdot \rangle_k$ denotes the inner product in $\mathcal{H}_k$. The positivity condition implies that for any finite set of distinct values $\{t_i\}_{i=1}^D \in \mathcal{T}^D$ the $D \times D$ Gram matrix $\mathbf{K}$, whose elements are $K_{ij} = k(t_i, t_j)$, for $1 \leq i, j \leq D$, is positive-semidefinite; that is, all its eigenvalues are non-negative.

Let ${\mathcal{H}_k}^{(0)}$ be the space of functions that can be expressed as finite linear combinations of the form

$$x(\,\cdot\,) = \sum_{i=1}^{r} \alpha_i k(t_i,\,\cdot\,). \tag{2.26}$$

This space is endowed with the inner product

$$\langle x, z\rangle_k = \sum_{i=1}^{r}\sum_{j=1}^{s} \alpha_i \beta_j k(t_i, t_j), \tag{2.27}$$

where $z(\,\cdot\,) = \sum_{j=1}^{s} \beta_j k(t_j,\,\cdot\,)$. The RKHS associated to k, denoted by $\mathcal{H}_k$, is the set of functions $f : \mathcal{T} \to \mathbb{R}$ that is the completion in the corresponding norm of the space ${\mathcal{H}_k}^{(0)}$.

Associated to kernel k, it is possible to define a covariance operator $\mathcal{K} : L^2(\mathcal{T}) \to L^2(\mathcal{T})$, by the integral equation

$$\mathcal{K}x(t) = \int_{\mathcal{T}} k(t,s)x(s)ds. \tag{2.28}$$

This operator is linear, self-adjoint, and positive. The eigenvalues and eigenfunctions of this operator satisfy

$$\int_{\mathcal{T}} k(t,s)\phi_j(s)ds = \lambda_j \phi_j(t), \quad j \geq 1 \tag{2.29}$$

with $0 < \ldots \leq \lambda_2 \leq \lambda_1 < \infty$, and $\{\phi_j(s)\}_{j=1}^{\infty}$, an orthonormal basis in $L^2(\mathcal{T})$

$$\int_{\mathcal{T}} \phi_i(t)\phi_j(t)dt = \delta_{ij}, \quad i, j = 1, 2, \ldots. \tag{2.30}$$

If the kernel is only positive-semidefinite, some of the eigenvalues of $\mathcal{K}$ could be zero. In that case, there is no loss of generality in considering only the linear subspace spanned by the set of eigenvectors corresponding to positive (non-zero) eigenvalues of the covariance operator (see, e.g., *Remark* 3 of Section 3 in Cucker and Smale, 2002).

By Mercer's theorem (Parzen, 1959; Cucker and Smale, 2002), the spectral representation of the kernel is

$$k(s,t) = \sum_{i=1}^{\infty} \lambda_i \phi_i(s)\phi_i(t). \tag{2.31}$$

The convergence of this series is absolute for each $(s,t) \in \mathcal{T} \times \mathcal{T}$ and uniform on $\mathcal{T} \times \mathcal{T}$. In this representation, the RKHS associated to kernel k is the space of functions that fulfill

$$\mathcal{H}_k = \left\{ x \in L^2(\mathcal{T}) :\; x = \sum_{i=1}^{\infty} \alpha_i \phi_i,\; \sum_{i=1}^{\infty} \frac{\alpha_i^2}{\lambda_i} < \infty \right\}. \tag{2.32}$$

The corresponding inner product is

$$\langle x, z\rangle_k = \sum_{i=1}^{\infty} \frac{\alpha_i \beta_i}{\lambda_i}, \tag{2.33}$$

with $z(t) = \sum_{i=1}^{\infty} \beta_i \phi_i(t)$. Finally, the set of functions $\{\varphi_j(s) = \sqrt{\lambda_j}\phi_j(s)\}_{j=1}^{\infty}$ form an orthonormal basis in $\mathcal{H}_k$.

Note also that the map $\mathcal{K}^{1/2} : L^2(\mathcal{T}) \to \mathcal{H}_k$ given by

$$\sum_{i\geq 1} a_i \phi_i \in L^2(\mathcal{T}) \mapsto \sum_{i\geq 1} a_i \sqrt{\lambda_i} \phi_i \in \mathcal{H}_k, \tag{2.34}$$

makes it possible to build the RKHS from $L^2(\mathcal{T})$

$$\mathcal{H}_k = \left\{ \mathcal{K}^{1/2}(x), x \in L^2(\mathcal{T}) \right\}. \tag{2.35}$$

Using this square root of the kernel operator, it is also possible to write the inner product of $x, z \in \mathcal{H}_k$ as an inner product in $L^2(\mathcal{T})$

$$\langle x, z \rangle_k = \left\langle \mathcal{K}^{1/2}(x), \mathcal{K}^{1/2}(z) \right\rangle, \tag{2.36}$$

which is a convenient representation for some computations.

2.2.2 Stochastic processes and RKHSs

There is an alternative construction of the RKHS that takes as a starting point the zero-mean second-order stochastic process $Z(t), t \in \mathcal{T}$, whose covariance function is

$$k(s,t) = \mathbb{E}\left[Z(s)Z(t)\right], \quad s,t \in \mathcal{T}. \tag{2.37}$$

Consider $\mathcal{L}_0(Z)$, the linear span of the process Z, whose elements are of the form

$$\sum_{i=1}^{D} \alpha_i Z(t_i), \quad \{\alpha_i\}_{i=1}^D \in \mathbb{R}^D, \{t_i\}_{i=1}^D \in \mathcal{T}^D, \tag{2.38}$$

for some integer D. Let $\mathcal{L}(Z)$ be the closure of $\mathcal{L}_0(Z)$ in $L^2(\Omega)$, the space of zero-mean random variables with finite second moments. By Loève's representation theorem (Berlinet and Thomas-Agnan, 2004) it is possible to define an isomorphism $\psi : \mathcal{L}(Z) \to \mathcal{H}_k$ that maps each $\sum_{i\geq 1} \alpha_i Z(t_i) \in \mathcal{L}(Z)$ onto a unique element

$$\begin{aligned} \psi\left(\sum_{i\geq 1} \alpha_i Z(t_i)\right)(t) &= \mathbb{E}\left[\sum_{i\geq 1} \alpha_i Z(t_i) Z(t)\right] \\ &= \sum_{i\geq 1} \alpha_i k(t_i, t), \quad t \in \mathcal{T} \end{aligned} \tag{2.39}$$

in $\mathcal{H}_k$. Since $\sum_i \alpha_i k(t_i, t)$ converges in the norm sense to an element of the RKHS (because $\sum_i \alpha_i Z_i$ belongs to the closure of $\mathcal{L}_0(Z)$ in $L^2(\Omega)$), it also converges pointwise for all $t \in \mathcal{T}$ to the same limit (Corollary 1 of Berlinet and Thomas-Agnan, 2004). This isomorphism preserves the inner product; i.e., it is a congruence. This congruence is referred to as Loève's isometry (Lukić and Beder, 2001). It maps $Z(s)$ onto $k(s,t)$

$$\psi(Z(s))(t) = k(s,t), \quad s,t \in \mathcal{T}. \tag{2.40}$$

Conversely, the inverse congruence ψ_Z^{-1} maps the function $x(\,\cdot\,) = \sum_i \alpha_i k(t_i, \,\cdot\,) \in \mathcal{H}_k$ onto the random variable $\psi_Z^{-1}(x) = \sum_i \alpha_i Z(t_i) \in \mathcal{L}(Z)$. Therefore, the value of the random trajectory at $t \in \mathcal{T}$ can be expressed as

$$Z(t) = \psi_Z^{-1}\left(k(\,\cdot\,, t)\right). \tag{2.41}$$

In terms of this isometry, the inner product between the functions $x = \sum_{i=1}^{\infty} \alpha_i k(t_i, \cdot)$ and $z = \sum_{j=1}^{\infty} \beta_j k(t_j, \cdot)$, both in $\mathcal{H}_k$ is

$$\langle x, z \rangle_k = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \alpha_i \beta_j k(t_i, t_j) = \mathbb{E}\left[\psi_Z^{-1}(x) \psi_Z^{-1}(z)\right]. \tag{2.42}$$

Following Parzen, 1961a, this isometry can be used to define the mapping

$$\langle Z, x \rangle_k = \psi_Z^{-1}(x) = \sum_{i=1}^{\infty} \alpha_i Z(t_i) \in \mathcal{L}(Z), \tag{2.43}$$

for Z, a trajectory of the stochastic process, and $x = \sum_{i=1}^{\infty} \alpha_i k(t_i, \cdot) \in \mathcal{H}_k$. This mapping, which can be viewed as a formal extension of the definition of the inner product, is well-defined even though, except for trivial cases, $Z \notin \mathcal{H}_k$ with probability one (Kailath, 1971; Berlinet and Thomas-Agnan, 2004). The quantity $\langle Z, x \rangle_k$ is the unique linear square-integrable functional of Z that satisfies (Parzen, 1959; Kailath, 1971)

$$\mathbb{E}\left[Z \langle Z, x \rangle_k\right] = x, \ \forall x \in \mathcal{H}_k. \tag{2.44}$$

Finally, it is also possible to express this *congruence* inner product as

$$\langle Z, x \rangle_k = \sum_{i=1}^{\infty} \frac{\zeta_i x_i}{\lambda_i}, \tag{2.45}$$

in terms of $x(t) = \sum_{i=1}^{\infty} x_i \phi_i(t),\ t \in \mathcal{T}$, and of $Z(t) = \sum_{i=1}^{\infty} \zeta_i \phi_i(t),\ t \in \mathcal{T}$, the Karhunen-Loève expansion of the process (Berlinet and Thomas-Agnan, 2004). The coordinates of this expansion are computed by projecting Z onto the basis of eigenfunctions of $\mathcal{K}$

$$\zeta_i = \int_{t \in \mathcal{T}} Z(t) \phi_i(t) dt,\ i = 1, 2, \ldots. \tag{2.46}$$

They are zero-mean independent normal random variables

$$\begin{aligned} \mathbb{E}\left[\zeta_i\right] &= 0 \\ \mathbb{E}\left[\zeta_i \zeta_j\right] &= \lambda_i \delta_{ij}, \quad i, j = 1, 2, \ldots. \end{aligned} \tag{2.47}$$

In this thesis we will show how one can take advantage of the properties of RKHS to address problems in machine learning with functional data.

2.3 Machine learning with functional data

The main objective of this thesis is the development of new methodologies and tools for machine learning with functional data. To this end, we will first define the main problems in this functional context. For these problems we will consider that we have a functional dataset $\mathcal{D} = \{x_i\}_{i=1}^N$. In the so-called *unsupervised* problems, no additional information is given. However, in *supervised* problems, a *target* y_i associated with the observation x_i is provided for training data, but not present in the final evaluation. Thus, the annotated dataset in this case would be $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$. The y_i belong to a set $\mathcal{Y}$ of possible targets, whose elements could be scalar, categorical or even functional.

In this section, we present several problems in the intersection of machine learning and functional data. The problem of registration is first presented in Section 2.3.1.

This is a novel problem in functional data, in which the potential misalignment of the data is corrected. Although it could be considered a preprocessing technique, the complexity of the problem and the techniques employed are similar to those present in traditional machine learning tasks. In Section 2.3.2 we present the problem of dimensionality reduction with functional data. As this kind of data is infinite dimensional, this problem is of crucial importance to achieve performant and interpretable models. We present the problem of regression with functional data in Section 2.3.3. As functional data may appear as a covariate, a response, or both, this particular problem presents new challenges in the functional context. In Section 2.3.4 we introduce the problem of classification with functional data. This problem is of particular relevance in this thesis, and thus is explained in detail. In particular we explain how the optimal rule can be computed, and how the new phenomenon of near-perfect classification arises when the data is functional. Finally, Section 2.3.5 provides a cursory introduction to clustering problems with functional data, which are also present in this thesis.

2.3.1 Registration

For correct application of functional data techniques, it is usually required that the important features of the data, such as maxima, minima or zero-crossing points, that are common to every observation are properly aligned. These can be occasionally misaligned due to errors in the measurement process. For example, when measuring the heartbeats of a patient, we may measure two of them starting at different locations. If that happens, the peaks of the two heartbeats will not be aligned, and this could affect further processing. Even if the measurement is error-free, different functions may have inherent variations that cause those features to appear at different positions in each observation. This happens, for example, when data can be better described as a function of some internal and unobserved continuous parameter, unique for each observation. In that last case, the variations occasionated by the differences in the internal parameter could be useful in subsequent tasks, such as classification, clustering or regression. Therefore, it is often useful to preserve that information, in addition to the aligned observations.

The variability in the location of the features, which one attempts to remove, is usually called *phase variation*. In contrast, the remaining variability in the magnitude of the features is called *amplitude variation*. In the case of real valued functions of one variable, or curves, which is the most commonly studied, phase variation is related with variability in the horizontal axis of the graph of the function, while amplitude variation is related with variability in the vertical axis. The techniques employed to align the features, and separate both sources of variability, receive the name of *registration* methods. Phase and amplitude variation are often defined in an informal way. However, formal definitions have also been proposed (Vantini, 2012; Marron et al., 2015). There is some ambiguity in the definition of these quantities: in some cases some sources of variation can be equally attributed to phase or amplitude (Marron et al., 2015), referred to as the *identifiability* problem (Chakraborty and Panaretos, 2021).

Landmark registration

The simplest case of registration occurs when we have information about the location of a fixed number of points of interest $\{\tau_{i_p}\}_{p=1}^{P}$, or *landmarks*, for each curve

$x_i \quad i = 1, \dots, N$, such as maxima, minima or zero-crossing points. Given a particular observation $x_i \in \mathcal{X}$, the corresponding aligned function $\tilde{x}_i \in \mathcal{X}$ would be of the form $\tilde{x}_i = x_i \circ \gamma_i$, for some transformation of the domain, or *warping function* γ_i We want that in the registered functions the landmarks appear at some fixed locations $\{\tau_p^*\}_{p=1}^P$, and thus

$$\tilde{x}_i(\tau_p^*) = x_i(\tau_{i_p}), \quad p = 1, \dots, P. \tag{2.48}$$

When $P = 1$, this can be trivially achieved by considering only the warping functions that perform a *shift*, or displacement in the domain, that is, functions of the form $\gamma_i(t) = t + \delta_i$, for $\delta_i \in \mathcal{T}$. Then the shifts $\delta_i = \tau_{i_1} - \tau_1^*$ leave the landmark aligned.

Alternatively, when more than one landmark is given, or when shifting is undesirable, one may want to achieve an *elastic* transformation, that modifies the shape of the curve. Then, one would restrict the warping in order not to modify the values at the domain boundary. For example, if $\mathcal{T} = [a, b]$, we have that $\gamma_i(a) = a$ and $\gamma_i(b) = b$. The additional constraints $\gamma_i(\tau_p^*) = \tau_{i_p} \quad p = 1, \dots, P$, guarantee that the final curves have their landmarks aligned. In order for the warping function to be continuous and invertible, a monotonic interpolation between these fixed points, such as monotonically increasing splines (Ramsay and Silverman, 2005), can be applied to define it.

In most datasets, however, there is no given set of landmarks that we must align, but instead is required that we infer the appropriate alignment from the data itself. This requires a more formal and careful definition of phase and amplitude, that is given below.

Formal definition of phase and amplitude

The following definition of phase and amplitude is based on the work of Vantini, 2012. We consider that $\mathcal{X}$ is a metric space such as $L^2(\mathcal{T})$. We also must choose a family of allowed warping functions

$$\Gamma = \{\gamma : \mathcal{T} \to \mathcal{T}, \gamma \text{ continuous}\} \tag{2.49}$$

which is a subgroup with respect of the composition operator $\circ$ of the group of continuous automorphisms of $\mathcal{T}$. As mentioned before, we want that $\tilde{x} \in \mathcal{X}$. Thus we need to impose that $\forall x \in \mathcal{X} \quad \forall \gamma \in \Gamma \quad x \circ \gamma \in \mathcal{X}$. As an additional requirement to guarantee the soundness of the registration procedure, the family of warpings Γ should be compatible with the metric d in the sense that applying the same warping to two functions should leave their distance invariable, that is

$$\forall x_1, x_2 \in \mathcal{X} \quad \forall \gamma \in \Gamma \quad d(x_1, x_2) = d(x_1 \circ \gamma, x_2 \circ \gamma). \tag{2.50}$$

This property ensures that it is not possible to obtain a false reduction in similarity between two functions warping them in the same way.

In order to clarify the above definitions, it is useful to consider several possible choices of $\mathcal{X}$, d and Γ that verify the aforementioned properties. For example, one could consider the L^2 distance and the family of shifts of the form $\gamma(t) = t + \delta$, for $\delta \in \mathcal{T}$, mentioned before. Here is necessary that $\mathcal{T} = \mathbb{R}$, and in that case all the properties are verified (Sangalli, Secchi, and Vantini, 2014). Choosing this restricted family makes sense in case that one knows that each observation has the same internal “velocity”, and the only possible discrepancies in the alignment of the curves come from a different choice of origin for each observation during the measurement process. Other possible choices of the Γ family are the sets of scalings of the domain or linear or affine transformations (Marron et al., 2015), which are also compatible with the $L^2(\mathbb{R})$ distance.

After choosing an appropriate family Γ, we can define the semimetric

$$d_\Gamma(x_1, x_2) = \min_{\gamma_1, \gamma_2 \in \Gamma} d(x_1 \circ \gamma_1, x_2 \circ \gamma_2). \tag{2.51}$$

Intuitively, this semimetric measures the distance between the best possible registration of each pair of curves, using the warpings in Γ. Although the definition is clearly symmetrical, using the property in Equation (2.50) and the fact that the family of warping functions is a group, and thus every warping function γ has an inverse γ^{-1}, there exists an identity element, and is closed under composition, we can see that this is equivalent to aligning only one of the functions to the other, that is,

$$\begin{aligned} d_\Gamma(x_1, x_2) &= \min_{\gamma_1, \gamma_2 \in \Gamma} d(x_1 \circ \gamma_1, x_2 \circ \gamma_2) \\ &= \min_{\gamma_1, \gamma_2 \in \Gamma} d(x_1 \circ \gamma_1 \circ \gamma_2^{-1}, x_2 \circ \gamma_2 \circ \gamma_2^{-1}) \\ &= \min_{\gamma_1, \gamma_2 \in \Gamma} d(x_1 \circ \gamma, x_2) \quad \text{with } \gamma = \gamma_1 \circ \gamma_2^{-1} \\ &= \min_{\gamma \in \Gamma} d(x_1 \circ \gamma, x_2). \end{aligned} \tag{2.52}$$

We can now consider the equivalence relation $\sim$ given by the metric identification with d_Γ (that is, $x_1 \sim x_2 \iff d_\Gamma(x_1, x_2) = 0$). If we define as $[x]$ the equivalence class containing x as one of its elements, then we can define another metric in the quotient space $\mathcal{X}/\sim$ as

$$d_{\mathcal{X}/\sim}([x_1], [x_2]) = d_\Gamma(x_1, x_2). \tag{2.53}$$

Now we have the mathematical tools to provide a more rigorous definition of phase and amplitude variation. Functions belonging to the same equivalence class can be warped into each other composing them with an appropriate warping function $\gamma \in \Gamma$. Thus, we can define phase variation as the one that occurs between functions in the same equivalence class. As a consequence, the remaining source of variability, the amplitude variation, would be the variation between the equivalence classes themselves. If we have two functions x_1 and x_2 with $d_\Gamma(x_1, x_2) = d(x_1, x_2)$ then no warping function can bring them closer together, so all the variability is due to amplitude variation.

One important point to note about these definitions is that they depend on the choice of d and Γ. These choices in turn depend on the task at hand, as the nature of the particular problem can suggest a particular metric or warping family, which in turn defines a concept of amplitude and phase appropriate for that problem.

Registering a group of curves

Until now, we have considered the case where two observations are registered to each other. But in most registration problems we have a set of several observations $\{x_i\}_{i=1}^N$, and we want every one of them to be aligned with the others. A usual way to address that problem is to find a function, called a *template*, which is centered inside the set of observations in the phase space, and align each observation to it. Due to Equation (2.52), we know that registering the template and each observation to each other with two warping functions is equivalent to just register each observation towards the template, using only one warping function. It is natural to choose as the equivalence class $[x_0]$ for the template the analog of the mean using the available metric $d_{\mathcal{X}/\sim}$ in the quotient space, that is, the Fréchet mean

$$[x_0] = \operatorname*{argmin}_{[x]\in\mathcal{X}/\sim} \left(\sum_{i=1}^{N} d^2_{\mathcal{X}/\sim}([x_i],[x]) \right). \tag{2.54}$$

It is less clear, however, how to select the best template function x_0 inside this equivalence class. Some authors (Sangalli et al., 2010; Kneip and Ramsay, 2008; Srivastava et al., 2011) consider that the mean of the warping functions should be the identity, and thus use

$$\frac{1}{N}\sum_{i=1}^{N} \gamma_i = \mathrm{id}_{\mathcal{T}} \tag{2.55}$$

as a constraint. In Vantini, 2012 is argued that this constraint, although good enough in practice, is not the best approximation, and furthermore requires that the family of warping functions form a convex linear space. Instead, this work recommends using again a Fréchet mean inside the equivalence class of the template function as a constraint, that is,

$$x_0 = \operatorname*{argmin}_{x\in[x_0]} \left(\sum_{i=1}^{N} d^2(x_i, x) \right). \tag{2.56}$$

Finally, there remains the question of how these minimizations should be implemented in practice. A *Procrustes fitting criteria* seems to be a popular choice (Vantini, 2012), iteratively alternating a step in which the Fréchet mean of the functions is computed and another step in which the functions are registered towards that mean. This is indeed the usual procedure when one uses the warping families of shifts or affine functions (Ramsay and Li, 1998; Sangalli et al., 2009). For example, for the family of shifts mentioned before, a Procrustes method can iteratively compute the mean $\hat{\mu}(t)$ and register the data minimizing

$$REGSSE = \sum_{i=1}^{N} \int_{\mathcal{T}} [\tilde{x}_i(t) - \hat{\mu}(t)]^2 dt, \tag{2.57}$$

as proposed in Ramsay and Silverman, 2005.

Fisher-Rao elastic registration

The concepts and procedures previously explained have been magnificently applied to the warping family of *diffeomorphisms*, or differentiable and invertible automorphisms of an interval (Srivastava et al., 2011), for real valued functions of a single variable. In this setting, one usually consider that the domain of the functions is the interval $\mathcal{T} = [0,1]$. The family of warping functions is then

$$\Gamma = \{\gamma : \gamma(0) = 0, \gamma(1) = 1, \gamma \text{ is a diffeomorphism}\}. \tag{2.58}$$

An analogous definition could be made for a different interval, or even for the whole real line, in which case the family of affine transformations would be a strict subset of Γ (Marron et al., 2015).

Unfortunately this family of warpings is not compatible with the $L^2([0,1])$ norm in the sense of Equation (2.50). The alternative is to use as d the distance induced by an extension of the Fisher-Rao Riemannian metric. Recall that a Riemannian metric defines a smoothly varying inner product for the vectors in the tangent space of a manifold. That is, given the manifold $\mathcal{X}$ containing our observations, and if $T_x(\mathcal{X})$ is the tangent space to $\mathcal{X}$ at $x \in \mathcal{X}$, for every two vectors $v_1, v_2 \in T_x(\mathcal{X})$ this Riemannian metric will be defined as

$$\langle v_1, v_2 \rangle_x = \frac{1}{4} \int_0^1 v_1'(t) v_2'(t) \frac{1}{|x'(t)|} dt. \tag{2.59}$$

In order to compute a metric between functions using this inner product, one first defines the length L of a continuously differentiable curve $c : [a,b] \to \mathcal{X}$ in an analogous way as in the usual case, but using the aforementioned inner product

$$L(c) = \int_a^b \|c'(t)\|_{\mathcal{X}} \, dt = \int_a^b \sqrt{\langle c'(t), c'(t) \rangle_{\mathcal{X}}} dt. \tag{2.60}$$

Then, the distance between two functions is the infimum of the lengths of every continuously differentiable curve that connects them.

The aforementioned distance verifies Equation (2.50) for the family Γ of diffeomorphisms. Moreover, it can be shown that it is the only metric with this property (Srivastava et al., 2011). Although the above reasons provide a sufficient justification for choosing this metric, it is very difficult to compute it directly. Fortunately, it has been proven that a simple transformation, the square root slope function (SRSF), transforms the space of functions in such a way that the Fisher-Rao distance can be computed using the standard $L^2([0,1])$ metric. Given $x \in \mathcal{X}$ this transformation is defined as

$$SRSF(x)(t) = \begin{cases} x'(t)/\sqrt{|x'(t)|}, & \text{if } x'(t) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.61}$$

or equivalently

$$SRSF(x)(t) = \operatorname{sign}\left(x'(t)\right) \sqrt{|x'(t)|}. \tag{2.62}$$

Note that the composition of functions does not commute with the SRSF. Instead, we have the identity $SRSF(x \circ \gamma)(t) = (SRSF(x) \circ \gamma)(t)\sqrt{\gamma'(t)}$. Now the distance induced by the Fisher-Rao metric between two functions $x_1, x_2 \in \mathcal{X}$ can be expressed as

$$d(x_1, x_2) = \|SRSF(x_1) - SRSF(x_2)\|_2 = \sqrt{\int_0^1 |SRSF(x_1)(t) - SRSF(x_2)(t)|^2 dt}, \tag{2.63}$$

and we can apply the results of the previous section to define the registration problem in this case. The registration process that uses this warping family of diffeomorphisms and this particular distance is called *Fisher-Rao elastic registration*, and it is currently one of the best general purpose registration methods.

There are a few differences between the usual computations in elastic registration and the procedures mentioned before. First, in order to select a particular template function from its equivalence class, a constraint similar to Equation (2.55) is imposed, but using the Fréchet mean of the warping functions instead, that is

$$\underset{\gamma \in \Gamma}{\operatorname{argmin}} \left(\sum_{i=1}^{N} d^2(\gamma_i, \gamma) \right) = \mathrm{id}_{\mathcal{T}}. \tag{2.64}$$

Other difference is that instead of using a Procrustes method, the required minimizations in the algorithm are computed using a dynamic programming algorithm.

2.3.2 Dimensionality reduction

Dimensionality reduction methods play an essential role in FDA. Since functional observations can be evaluated at arbitrary points in a continuum, the nature of functional data is, in principle, infinite-dimensional. Even when the data has been discretized, the number of grid points can be fairly large. As a result, large amounts of storage and computation are required to handle functional data.

The continuous nature of functional data means that nearby points are strongly correlated. Due to these redundancies, it is often possible to reduce the dimensionality of functional data while retaining most of the relevant information. In particular, we are interested in methods by means of which functional observations are represented as a finite-dimensional vector while retaining most of the information.

A number of benefits can be obtained from reducing the dimensions of the data. First, it makes possible to take advantage of off-the-shelf multivariate models for regression, classification, and clustering (Vieu, 2018). In that case, the functional nature of the data needs to be taken into consideration only in the preprocessing steps, including dimensionality reduction itself. If the dimensionality reduction process is effective, the accuracy of the model trained in the lower-dimensional representation is comparable to the predictors induced from the original functional data. Second, dimensionality reduction can have a regularization effect, so that more accurate predictors that reduce overfitting can be built. Third, further processing can be made with lower computational costs. Finally, dimensionality reduction can improve the interpretability of the models, by uncovering relevant features of the data.

One common way to represent functional data in a lower dimensional space is to approximate it using a finite basis expansion. For example, suppose that an observation $x \in \mathcal{X}$ can be exactly represented as a series,

$$x = \sum_{b=1}^{\infty} c_b \phi_b(t), \tag{2.65}$$

where $\{\phi_b(t), t \in \mathcal{T}\}_{b=1}^{\infty}$ is an infinite-dimensional (Schauder) basis, and c_i are the particular coefficients in the basis expansion. Examples of these bases include the Fourier and monomial bases of $L^2(0,1)$. In this case, one could truncate the basis expansion after a finite number B of elements, to achieve a lower-dimensional approximation

$$x \approx \sum_{b=1}^{B} c_b \phi_b(t). \tag{2.66}$$

One could then represent the data in $\mathbb{R}^B$ as the vector of coefficients $\mathbf{c} = (c_1, \dots, c_B)$. The quality of this representation will depend not only on the number B of chosen basis elements, but also in the basis itself. Depending on the data and the nature of the problem, some bases would be better choices than others. Moreover, if the basis employed is orthonormal, the results obtained by applying functional procedures would be the same as just considering the vector $\mathbf{c}$ as if it was expressed in the standard basis of $\mathbb{R}^B$.

One widely used dimensionality reduction method that uses this approach is functional principal components analysis (FPCA) (Ramsay and Silverman, 2005). This method is based on the Karhunen-Loève expansion of a zero-mean stochastic process X which takes values in L^2. This process has a covariance function k and associated covariance operator $\mathcal{K}$. Let $\{\lambda_b\}_{b=1}^{\infty}$ and $\{\phi_b\}_{b=1}^{\infty}$ be the eigenvalues and eigenvectors of $\mathcal{K}$, respectively. Then, the Karhunen-Loève theorem, already mentioned in Section 2.2.2, states that such a process can be expressed as

$$X(t) = \sum_{b=1}^{\infty} \xi_b \phi_b(t), \tag{2.67}$$

where the basis functions are orthogonal in L^2 and the ξ_b are zero-mean uncorrelated random variables, defined as $\xi_b = \langle X, \phi_b \rangle$. Furthermore, the random variable ξ_b has variance λ_b. Thus, by truncating to a fixed number of elements in the expansion, we actually keep the directions ϕ_b which maximize the explained variance of the process.

Variable selection

Variable selection is a method of dimensionality reduction that is of particular interest. The goal of variable selection methods is to identify a set of impact points, denoted by $\mathbf{t} = \{t_1, \dots, t_D\}$, which capture the bulk of the information present in the original trajectories. Subsequently, the original functional data are replaced by their values at the selected impact points, i.e $\{X(t_1), \dots, X(t_D)\}$. Since the lower-dimensional representation corresponds to particular points of the original functional observations, models that make use of variable selection methods are often more interpretable than models using other dimensionality reduction methods. In this last case, the dimensions of the reduced space correspond to combinations and transformations of the original variables, which may make the models harder to explain.

There are three main categories of variable selection methods (Guyon and Elisseeff, 2003; Guyon et al., 2006). In *filter* methods, the variable selection is performed as a preprocessing step, independent of the particular procedures performed afterwards. For example, a method that rank the variables according to statistical tests (e.g. higher dependence with the target, high variance, etc) and then selects the best variables in the ranking would be in this category. On the contrary, in *wrapper* methods, a predictive procedure, such as a classifier or regressor is used as a black-box in order to select the points that have a greater predictive power. A possible method in this category is to apply forward selection using as a criterion the accuracy of the predictions. In that case, we proceed iteratively, adding at each step to the set of selected variables the one that best improves prediction, until the improvement is too small. Finally, in *embedded* procedures variable selection is integrated in the prediction process itself. That would be the case if, for example, we train a classification tree and discard all variables that end not being used by the model.

A possible approach to variable selection in the functional setting is to directly apply multivariate methods to the discretized trajectories (Jiménez-Cordero and Maldonado, 2021). The multivariate method can be enhanced by taking into consideration the continuity and underlying smoothness of the functional data. In particular, nearby values in the trajectories should carry similar information. This redundancy can be taken into account explicitly in the design of the variable selection method (Aneiros and Vieu, 2014). Alternatively, as in the mRMR (minimum Redundance Maximum Relevance) method Ding and Peng, 2005, redundant variables can be discarded (Berrendero, Cuevas, and Torrecilla, 2016a). In some cases, feature construction is performed to summarize the functional observations by vectors of attributes that can be subsequently passed to multivariate variable selection procedures (Gómez-Verdejo, Verleysen, and Fleury, 2009; Fraiman, Gimenez, and Svarc, 2016).

There is a sizeable literature on variable selection in the context of functional regression (Goia and Vieu, 2016; Vieu, 2018; Aneiros, Novo, and Vieu, 2022). Some of these methods employ linear regression models, wherein a penalty term is utilized to encourage sparsity (Kneip and Sarda, 2011; Lee and Park, 2012; Aguilera-Morillo et al., 2020; Feng, Zhang, and Tong, 2022; Belli, 2022). Extensions of LASSO to nonparametric cases have also been proposed (Ghosal and Maity, 2022). In contrast, variable selection can be performed using sliced inverse regression (Picheny, Servien, and Villa-Vialaneix, 2019) techniques or wrapper methods like sequential forward-backward selection (Ferraty, Hall, and Vieu, 2010), which do not require any specific assumptions about the regression model.

In the context of classification, variable selection methods have been proposed for predictors based on penalized linear regression (Grosenick, Greer, and Knutson, 2008), logistic regression (Lindquist and McKeague, 2009; McKeague and Sen, 2010; Matsui, 2014), Bayesian discriminant analysis (Yu et al., 2022), and support vector machines (Blanquero et al., 2019; Blanquero et al., 2020). A wrapper approach is introduced in Delaigle, Hall, and Bathia, 2012. In this work the continuous nature of the data is taken into account by searching in a coarse grid of points and refining the search in the most promising regions.

The problem of variable selection in functional classification is also addressed in Berrendero, Cuevas, and Torrecilla, 2018. This paper discusses classification problems in which Bayes classification rule (see Section 2.3.4) depends on a finite number of variables due to the structure of the mean functions. The authors propose a filter variable selection algorithm, reproducing kernel-based variable selection (RKVS). The name RKVS refers to the role of the reproducing kernel Hilbert spaces theory in deriving properties of the method, such as optimality results for some Gaussian models. RKVS aims at selecting the variables to maximize the (multivariate) Mahalanobis distance between the class means in the reduced space,

$$\phi(\mathbf{t}) = \mu(\mathbf{t})^T k(\mathbf{t}, \mathbf{t})^{-1} \mu(\mathbf{t}), \tag{2.68}$$

where $\mu(\mathbf{t}) = \mu_1(\mathbf{t}) - \mu_0(\mathbf{t})$ is the difference between the class means evaluated at the vector $\mathbf{t}$ of selected points and $k(\mathbf{t}, \mathbf{t})$ is the covariance matrix for these points. The method has good theoretical properties and performs well in practice.

The same authors introduced maxima hunting (MH) (Berrendero, Cuevas, and Torrecilla, 2016b), a non-parametric and data-driven variable selection method for functional data. MH relies on identifying impact points at which the dependence of the class label on the corresponding function values is maximal. Similar strategies of selecting the maxima of a relevance function have been followed in subsequent works (Ordóñez et al., 2018; Poß et al., 2020). A complete description of MH, as well as the analysis of recursive maxima hunting (RMH), a more advanced method inspired by it, are the main topics covered in Chapter 5.

2.3.3 Regression

Functional data regression is an example of a *supervised problem* in machine learning. In these kinds of problems, a initial set of previously labeled observations is available for training a model. The observations themselves belong to a set $\mathcal{X}$, which could be, for example, a multivariate space, such as $\mathbb{R}^M$, or a functional space like the ones described in Section 2.2. The labels, or targets, belong to the set $\mathcal{Y}$, which will depend on the particular problem considered. The trained model will then be used to predict the label of previously unseen data for which the correct label is not known.

In the case of regression, we will have a training dataset $\{x_i, y_i\}_{i=1}^N$ $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, in which $\mathcal{Y}$ is a continuous space, such as the set of real numbers $\mathbb{R}$. In this setting the label is often called the "dependent variable" or "response", while the attributes of the observations are called the "independent variables" or "covariates". Depending on the actual sets $\mathcal{X}$ and $\mathcal{Y}$, we can distinguish between different kinds of regression problems (Ramsay and Silverman, 2005; Morris, 2015).

When $\mathcal{X} = \mathbb{R}^M$ and $\mathcal{Y} = \mathbb{R}^P$, we are dealing with classical *multivariate regression*. When $P > 1$ sometimes the term *multi-output regression* is employed. It is common to restrict ourselves to the case $P = 1$ in this case, as the case $P > 1$ can often be treated as P different regression problems, each with $P = 1$.

If $\mathcal{X}$ is a functional space and $\mathcal{Y} = \mathbb{R}^P$, the problem is called functional regression with scalar response or *scalar-on-function* regression (Reiss et al., 2017). As in *multivariate regression*, it is common to consider just the case $P = 1$.

We can also have the opposite situation, when $\mathcal{X} = \mathbb{R}^M$ and $\mathcal{Y}$ is a functional space. This is the so-called *function-on-scalar* regression (Bauer et al., 2018). Note that in this case it is not desirable to consider this problem as unrelated regression models with scalar response. This is not only because there would be as many models as discretization points in the response, but also because these response values are not independent, but instead they are highly correlated.

It is also possible to consider the case were both $\mathcal{X}$ and $\mathcal{Y}$ are functional spaces. In this case the problem is *function-on-function* regression (Ivanescu et al., 2015). This case includes as sub-models two constrained cases of interest: the *historical* model (Malfait and Ramsay, 2003), in which the response at time t can only be predicted using the values of the covariates at times $s \leq t$, and the *concurrent* model (Ramsay and Silverman, 2005), in which only the values of the covariates at time t are considered.

One simple model that is often useful in multivariate and functional regression is the linear regression model (Ramsay and Silverman, 2005). In this model the response, either multivariate or functional, is considered to be a linear function of one or more covariates. The covariates themselves can be multivariate or functional, or even a combination of both. Consider a case with $p = p_1 + p_2$ covariates in which the first p_1 covariates are multivariate and the last p_2 covariates are functional. An observation x_i is then the set of covariates $\{\mathbf{x}_{i1}, \dots, \mathbf{x}_{ip_1}, x_{ip_1+1}, \dots, x_{ip}\}$. When the response is scalar, the functional linear regression model can be written as

$$y_i = \beta_0 + \sum_{j=1}^{p_1} \boldsymbol{\beta}_j \mathbf{x}_{ij} + \sum_{j=p_1+1}^{p} \int_{\mathcal{T}_j} \beta_j(t) x_{ij}(t) dt + \epsilon_i, \qquad i = 1, \dots, N, \tag{2.69}$$

were β_0 is the intercept term, β_j is the coefficient associated with the j-th covariate, $\mathcal{T}_j$ is the support of the j-th covariate and ϵ_i models some error or noise of the i-th observation. In order to prevent overfitting due to the presence of infinite-dimensional parameters β_j in the model, a regularization that penalizes the curvature of the functional β_j is often applied. In spite of the simplicity of this model, it is very useful not only for simple problems, but also because it provides an interpretation of how the response depends on the covariates, explaining the influence of each one.

In the functional response case, the linear regression model is now

$$y_i(t) = \beta_0(t) + \sum_{j=1}^{p_1} \boldsymbol{\beta}_j(t)\mathbf{x}_{ij} + \sum_{j=p_1+1}^{p} \int_{\mathcal{T}_j} \beta_j(t,s)x_{ij}(t,s)ds + \epsilon_i, \qquad i = 1,\dots,N. \tag{2.70}$$

Note that here each functional covariate x_{ij} can depend on the same parameter t as the response, on a different parameter s, and on both. This general model includes several particular interesting cases. For example, it is common that the functional covariates are univariate functions. In this case, each corresponding term in Equation (2.70) can be written as

$$\int_{\mathcal{T}_j} \beta_j(t,s)x_{ij}(s)ds. \tag{2.71}$$

If s and t are defined in the same domain, it is possible to obtain the historical linear model in Malfait and Ramsay, 2003 by restricting $\beta_j(t,s)$ such that $s > t \Rightarrow \beta_j(t,s) = 0$. Furthermore, a concurrent functional linear model can be written replacing the corresponding term in Equation (2.70) by

$$\beta_j(t)x_{ij}(t). \tag{2.72}$$

There have been additional developments in parametric functional regression, that generalize the linear model presented here. Generalized additive models, in particular, have been extended to the FDA context, for scalar (McLean et al., 2014) and functional (Scheipl, Gertheiss, and Greven, 2016) responses.

Nonparametric methods for regression can also be employed in the FDA context. The simplest of these methods are neighbor-based ones. In particular k-nearest neighbors (k-NN) regression can be applied to the functional case, both with scalar (Burba, Ferraty, and Vieu, 2009) and functional responses (Lian, 2011). In this model, a metric d in the original space $\mathcal{X}$ is required. In order to predict the target for a given $x \in \mathcal{X}$, one first finds the pairs corresponding to the nearest k observations in the train dataset $(x_{[1]}, y_{[1]}), \dots (x_{[k]}, y_{[k]})$. Then the predicted target is a weighted sum of these targets,

$$\hat{y} = \sum_{i=1}^{k} w_i y_{[i]}, \tag{2.73}$$

with $\sum_{i=1}^{k} w_i = 1$. The w_i are typically all $\frac{1}{k}$, to give equal importance to all neighbors, or inversely proportional to the distance between x and $x_{[i]}$.

Instead of considering only a fixed number of neighbors, it is possible to consider the weighted sum of all targets in the training data,

$$\hat{y} = \sum_{i=1}^{N} w_i y_i. \tag{2.74}$$

In order to achieve a good prediction, the weights should reflect the closeness of each observation in the training set to x. This is commonly done using a *kernel function* K, which weights the values in the real line according to their proximity to the origin (Wasserman, 2006). Given K a possible distribution of the weights is given by

$$w_i = \frac{K\left(\frac{d(x,x_i)}{h}\right)}{\sum_{j=1}^{N} K\left(\frac{d(x,x_j)}{h}\right)}, \quad i = 1,\dots,N, \tag{2.75}$$

where h is a positive real that depends on N and is called the bandwidth. This is the so-called Nadaraya–Watson kernel regression method, whose applicability to the functional data context has been studied in Ferraty and Vieu, 2006.

2.3.4 Classification

Functional data classification is another example of a supervised problem, in which the target, or *class label*, is categorical, that is, $\mathcal{Y} = \{0, \dots, C\}$. Given a training set $\{x_i, y_i\}_{i=1}^N$ $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, the objective is to construct a classifier, that is, a measurable function $g : \mathcal{X} \to \mathcal{Y}$, that given a previously unseen observation x obtains a correct prediction $\hat{y} = g(x)$ of its target.

Functional data classification is an active research field that has multiple applications in different areas, such as medicine (Zhu, Brown, and Morris, 2012; Epifanio and Ventura-Campos, 2014), genomics (Leng and Müller, 2006; Song et al., 2008; Rincón and Ruiz-Medina, 2012), spectrometry (Rossi and Villa, 2006), weather modelling and forecasting (Martin-Barragan, Lillo, and Romo, 2014), speech recognition (Rossi and Villa, 2006), and the analysis of handwriting (Hubert, Rousseeuw, and Segaert, 2017).

As mentioned before, there are substantial differences between the traditional multivariate classification problem and the functional one. The lack of densities of random functions, in particular, complicates the definition of some concepts, such as the optimal (Bayes) classification rule. In some cases, even if the individual class-conditional probability densities do not exist, such rule can still be given in terms of the Radon-Nikodym derivative, which plays the role of the likelihood ratio in these types of infinite-dimensional problems (Baíllo, Cuevas, and Cuesta-Albertos, 2011). The non-invertibility of the covariance operator in the functional case prevents a straightforward definition in the functional context of some of the classic classifiers, such as linear discriminant analysis (LDA) or quadratic discriminant analysis (QDA).

The theoretical analysis in functional data classification tends to focus on the binary case $C = 1$, as the notation becomes simpler, and the generalization to multiclass problems is often straightforward. In this case, the random function X taking values in $\mathcal{X}$, from which the observations have been obtained, can be defined as

$$\begin{cases} (X(t) \mid Y = 0) = \mu_0(t) + Z_0(t) & \text{with probability } 1 - p \\ (X(t) \mid Y = 1) = \mu_1(t) + Z_1(t) & \text{with probability } p, \end{cases} \tag{2.76}$$

where $0 \leq p \leq 1$, Y is the associated random variable for the target, μ_0, μ_1 are the deterministic functions corresponding with the class means and Z_0, Z_1 are class-dependent stochastic processes with mean 0, whose laws or induced measures are $\mathbb{P}_0$ and $\mathbb{P}_1$, respectively

Without loss of generality, the mean μ_0 can be subtracted from all trajectories to obtain the equivalent classification problem

$$\begin{cases} (X(t) \mid Y = 0) = Z_0(t) & \text{with probability } 1 - p \\ (X(t) \mid Y = 1) = \mu(t) + Z_1(t) & \text{with probability } p, \end{cases} \tag{2.77}$$

where $\mu(t) = \mu_1(t) - \mu_0(t)$.

A case of special interest from a theoretical point of view is where Z_0 and Z_1 are GPs, analogously to the discrimination between normally distributed populations in the multivariate setting. This model has been widely studied in the functional data literature (Delaigle and Hall, 2012; Delaigle and Hall, 2013; Dai, Müller, and Yao, 2017; Berrendero, Cuevas, and Torrecilla, 2018).

Another useful simplification is to consider the *homoscedastic* case, where Z_0 and Z_1 have the same covariance function k. In the particular case in which they are both Gaussian, their distribution is the same $Z_0 = Z_1 = Z$, as they are completely determined by the covariance, and thus the model becomes

$$\begin{cases} (X(t) \mid Y = 0) = Z(t) & \text{with probability } 1-p \\ (X(t) \mid Y = 1) = \mu(t) + Z(t) & \text{with probability } p. \end{cases} \tag{2.78}$$

The Bayes classification rule

The *Bayes classification rule* is the measurable function

$$g^*(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} P(Y = y \mid X = x). \tag{2.79}$$

This predictor achieves the minimum error for a classification problem characterized by the joint distribution $P(X, Y)$ (Devroye, Györfi, and Lugosi, 1996).

In some problems of interest it is possible to provide an explicit expression for the Bayes classifier. In order to do that we need to introduce some new concepts. Given two probability measures $\mathbb{P}_0$ and $\mathbb{P}_1$ in the same measurable space, we say that $\mathbb{P}_0$ is *absolutely continuous* with respect to $\mathbb{P}_1$ ($\mathbb{P}_0 \ll \mathbb{P}_1$) if for every measurable set A with $\mathbb{P}_1(A) = 0$ we also have $\mathbb{P}_0(A) = 0$. In this case is possible to define a *density function* $f = \frac{d\mathbb{P}_0}{d\mathbb{P}_1}$ or *Radon-Nikodym derivative* that relates these measures such that for every $\mathbb{P}_1$-measurable set A we have

$$\mathbb{P}_0(A) = \int_A f d\mathbb{P}_1. \tag{2.80}$$

For the binary classification problem in Equation (2.76), if $\mathbb{P}_0$ and $\mathbb{P}_1$ are both absolutely continuous with respect to a common measure λ, then we can express this optimal rule as

$$g^*(x) = \mathbb{I}_{\{pf_1(x) > (1-p)f_0(x)\}}, \tag{2.81}$$

where $f_i = \frac{d\mathbb{P}_i}{d\lambda}$ (Baíllo, Cuevas, and Cuesta-Albertos, 2011). This is precisely the case for multivariate data, where for λ we can choose the Lebesgue measure. However, this is not possible in the functional case, as there is no analog of the Lebesgue measure for infinite-dimensional spaces, and there is no other obvious dominant measure.

However, even in the absence of this common λ, in the case where $\mathbb{P}_0$ and $\mathbb{P}_1$ are mutually absolutely continuous with respect to each other, also called equivalent ($\mathbb{P}_0 \sim \mathbb{P}_1$), we can still write (Berrendero, Cuevas, and Torrecilla, 2018) the optimal rule as

$$g^*(x) = \mathbb{I}_{\left\{\frac{d\mathbb{P}_1}{d\mathbb{P}_0}(x) > \frac{1-p}{p}\right\}}. \tag{2.82}$$

In the Gaussian homoscedastic case of Equation (2.78), we know that $\mathbb{P}_0 \sim \mathbb{P}_1 \Leftrightarrow \mu \in \mathcal{H}_k$, and in this case we have an expression for the Radon-Nikodym derivative in terms of the theory of RKHS (Berrendero, Cuevas, and Torrecilla, 2018):

$$\frac{d\mathbb{P}_1}{d\mathbb{P}_0}(x) = \exp\left(\langle x, \mu \rangle_k - \frac{1}{2}\langle \mu, \mu \rangle_k\right). \tag{2.83}$$

Note that if the RKHS is infinite dimensional, the trajectories of the random process do not belong to $\mathcal{H}_k$ with probability one (Kailath, 1971; Lukić and Beder, 2001; Berlinet and Thomas-Agnan, 2004). In this case, we have to understand the notation $\langle x, \mu \rangle_k$ as the extension of the inner product given by Loève's isometry in Equation (2.43). The final expression for the Bayes classification rule in this case would be

$$g^*(x) = \mathbb{I}_{\{\eta^*(x)>0\}}, \tag{2.84}$$

with

$$\eta^*(x) = \langle x, \mu \rangle_k - \frac{1}{2} \langle \mu, \mu \rangle_k - \log\left(\frac{1-p}{p}\right). \tag{2.85}$$

The error associated with this optimal rule is

$$\begin{aligned} L^* =&(1-p)\operatorname{cdf}\left(-\frac{\|\mu\|_k}{2} - \frac{1}{\|\mu\|_k}\log\left(\frac{1-p}{p}\right)\right) + \\ &p\operatorname{cdf}\left(-\frac{\|\mu\|_k}{2} + \frac{1}{\|\mu\|_k}\log\left(\frac{1-p}{p}\right)\right), \end{aligned} \tag{2.86}$$

where cdf is the cumulative distribution function of a standard Gaussian variable. In the particular case where $p = \frac{1}{2}$, this formula simplifies to

$$L^* = 1 - \operatorname{cdf}\left(\frac{\|\mu\|_k}{2}\right). \tag{2.87}$$

Near-perfect classification

A surprising effect that appears in the context of functional classification is near-perfect classification (Delaigle and Hall, 2012). This is the observed behavior in some functional problems in which zero classification error can be achieved in the asymptotic (infinite sample) case. This happens for homoscedastic problems, even with non Gaussian processes, if

$$\sum_{i=1}^{\infty} \frac{\mu_i^2}{\lambda_i} = \infty, \tag{2.88}$$

where the λ_i are the eigenvalues of the common covariance operator $\mathcal{K}$, as defined in Equation (2.29). The $\{\mu_i\}_{i=1}^{\infty}$ are the coefficients of the mean in the base of eigenfunctions, that is, $\mu = \sum_{i=1}^{\infty} \mu_i \phi_i$.

In the general case, near-perfect classification will occur whenever the measures $\mathbb{P}_0$ and $\mathbb{P}_1$ are mutually singular. Recall that two measures $\mathbb{P}_0$ and $\mathbb{P}_1$ are mutually singular ($\mathbb{P}_0 \perp \mathbb{P}_1$) if there exist two measurable subsets $A, B \subseteq \mathcal{X}$, with $A \cap B = \emptyset$ and $A \cup B = \mathcal{X}$, and we have $\mathbb{P}_0(A) = 0$ and $\mathbb{P}_1(B) = 0$. In the case of classification, given a particular observation x, if $x \in A$ then $Y = 1$ with probability 1 and if $x \in B$ then $Y = 0$ with probability 1. In the particular case in which Z_0 and Z_1 are both Gaussian, the Feldman–Hájek dichotomy (Feldman, 1958) establishes that their respective laws are either mutually singular (so we have near-perfect classification, with an optimal error of 0) or equivalent (so in this case there is an expression for the Bayes classification rule in terms of the Radon-Nikodym derivatives).

In Berrendero, Cuevas, and Torrecilla, 2018, the near-perfect classification is defined in terms of the previous condition of mutually singular measures. The authors note that although this case rarely appears in multivariate data problems, it is nevertheless very common in the functional case. Moreover, they prove that for the particular homoscedastic problem in Equation (2.78), with Gaussian processes, and if we denote as $\mathcal{H}_k$ the RKHS associated with the common covariance function k, we have $\mathbb{P}_0 \perp \mathbb{P}_1 \Leftrightarrow \mu \notin \mathcal{H}_k$.

Functional classifiers

The differences between multivariate and functional data prevent some of the classifiers traditionally used in the former setting to be directly applicable to the latter. For example, the linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) classifiers, widely used in multivariate statistics, require the inversion of the covariance matrix of the data. As the covariance operator is not invertible in the functional setting in general, these methods are ill defined, and numerical problems may arise when applying them to discretized functional data. LDA and QDA are also interesting from a theoretical point of view, as for binary classification with multivariate Gaussian data (the multivariate analog of Equation (2.76)) they achieve the minimal (Bayes) error in the homoscedastic and heteroscedastic cases, respectively. In Chapter 4 we explore how this result can be generalized to functional data.

A rich family of classifiers in the multivariate setting are those based on the computation of distances between observations. These classifiers can be applied to functional data in a space with a suitable metric, such as those defined in Section 2.2. For example, nearest centroid classifiers are simple methods that assign to a given observation x the label of the class whose centroid (the mean) μ_c, $c = 0, \dots, C$ is closest to it:

$$\hat{y} = \operatorname*{argmin}_{c=0,\dots,C} d(x, \mu_c), \tag{2.89}$$

where $d(\cdot, \cdot)$ is a functional distance. In spite of their simplicity, they can achieve optimal results for some classification problems (Delaigle and Hall, 2012).

The k-nearest neighbors (k-NN) classifier can also be extended to the functional case by using a functional metric to find the neighbors. The main idea of this method is to choose a number k and a metric d. Given a test observation x, the method first finds the k nearest observations according to d in the train dataset, $x_{[1]}, \dots, x_{[k]}$, with respective class labels $y_{[1]}, \dots, y_{[k]}$. The class is predicted by majority voting:

$$\hat{y} = \arg\max_{y} \sum_{i=1}^{k} \mathbb{I}_{y_{[i]}=y}. \tag{2.90}$$

The k-nearest neighbors (k-NN) classifier has been shown to be consistent under mild assumptions (Cérou and Guyader, 2006). It can be considered a sort of reference method for supervised classification in FDA (Baíllo, Cuevas, and Fraiman, 2010). Some of the reasons for it are its simplicity, ease of motivation and the fact that it typically does not lead to gross classification errors.

Logistic regression, a widely used method in the multivariate setting, has been also extended to the functional domain. A recent contribution in this area is detailed in Berrendero, Bueno-Larraz, and Cuevas, 2022. In that work an extended model of logistic regression for binary classification is constructed using the theory of RKHS, as

$$P(Y = 1 \mid X) = \frac{1}{1 + \exp(-\beta_0 - \langle \beta, X \rangle_k)} \quad \beta_0 \in \mathbb{R}, \beta \in \mathcal{H}_k, \tag{2.91}$$

were again the inner product $\langle \cdot, \cdot \rangle_k$ has to be interpreted using Loève's isometry in Equation (2.43).

Another classification strategy is labelling observations according with their centrality in the populations: the more central an observation is in a sample, the more probable is that it belongs to that sample. The notion of centrality is quite natural in one-dimensional data, but it becomes fuzzy in higher dimensions. This effect is particularly critical in the infinite dimensional spaces of function where nothing similar to a natural order exists. Fortunately, there is a bunch of statistical depth functions that allow us to quantify this centrality in order to define the median, detect outliers or classify.

In this context, the simpler proposals for classifying are the so called maximum depth methods (Cuevas, Febrero, and Fraiman, 2007). They assign a new observation into the class in which it is deeper. That is, the class label assigned to x is

$$\hat{y} = \operatorname*{argmax}_{c=0,\dots,C} D_c(x), \tag{2.92}$$

where D_c denotes the depth measure in the c-th class.

A different approach to depth-based classification in the binary case ($C = 1$), the depth vs depth (DD) classifier, is described in Li, Cuesta-Albertos, and Liu, 2012. They propose to first transform each observation to a 2-dimensional vector whose coordinates are the depth of the datum inside each class. In that space, they try to find the polynomial of a particular degree which best separates the data points. Maximum depth can then be seen as a particular case of this method in which the polynomial used is the line $y = x$. This approach is thus more flexible, but computationally more costly, and restricted to the binary classification case.

A generalized version of the DD classifier is followed by the generalized depth-depth classifier (DD^G) (Cuesta-Albertos, Febrero-Bande, and Oviedo de la Fuente, 2017). This methodology consists in projecting the functional data into a lower-dimensional *space of depths* and then, use any multivariate classification rule. The features in this reduced space are the depth values of the trajectories in each class, for one or more depth measures. In other words, given a functional observation x and a set of depth functions $D^1, \dots, D^G$, the DD^G classifier performs the transformation $x \to (D_c^g(x)), \quad g = 1, \dots, G, \quad c = 0, \dots, C$, before classifying.

Finally, more advanced multivariate methods have been adapted to the functional setting. These include SVM classifiers (Rossi and Villa, 2006) and neural networks (Rossi and Conan-Guez, 2005). In SVMs, one can use kernels that only need the Hilbert space structure of the functional data, by computing inner products or norms. For neural networks, it is possible to construct a first functional layer in the network using the inner product, and keep the hidden layers multivariate.

2.3.5 Clustering

Clustering consists in identifying groups of observations (*clusters*) that exhibit some sort of similarity. This is an example of an *unsupervised problem*, with categorical prediction (the cluster label). As opposed to supervised problems, such as classification, in clustering we receive a dataset $\{x_i\}_{i=1}^N$, with every $x_i \in \mathcal{X}$, without any predefined label. The objective is to assign to each observation x_i a particular cluster label $\hat{y}_i \in \mathcal{Y} = \{0, \dots, C\}$. However in this case the assignment is done purely looking at the structure of the observations, as no prior labels are known. This also means that a particular cluster label has no predefined meaning: any relabeling of the clusters that produces the same partition of the data is essentially the same clustering. Finally, note that altough some clustering methods are capable of finding the "right" number of clusters that should be used, most classical clustering methods require that the number of clusters is known a priori.

The k-means algorithm (Hartigan and Wong, 1979) is perhaps the best-known clustering procedure (Jacques and Preda, 2014). In this algorithm the number of clusters C should be known in advance. Starting from C cluster centroids, which are selected at random from the observations, the algorithm proceeds by iterating until convergence the following two steps:

1. Assign each datum to the cluster whose centroid is the nearest one, given a distance measure.

2. Recompute the cluster centroids as the means of the observations belonging to each cluster.

This procedure can be used both for multivariate and functional data (Tokushige, Yadohisa, and Inada, 2007).

Fuzzy C-means (FCM) is a variant of the previous algorithm in which each observation is assigned to all C clusters with different degrees of membership (Dunn, 1973; Bezdek, 1981). In this setting, the degree of membership u_{ij} of the datum x_i to the cluster c_j is in the range $[0, 1]$, with the restriction $\sum_{j=1}^{C} u_{ij} = 1$. A new parameter $\omega \geq 1$, called the *fuzzifier*, controls the level of fuzziness of the result. A value $\omega = 1$ corresponds to crisp results, similar of those achieved by k-means. Higher values of ω lead to increased fuzziness in the solution. The algorithm works in the same fashion as k-means, with modified computations of the cluster assignment and centroid recalculation steps to take into account the fuzzy membership. In Chapter 6, the algorithm is described in detail, and its application to functional data is studied.

Other family of clustering methods for functional data includes the so-called hierarchical clustering methods (Ferreira and Hitchcock, 2009). In these method, a measure of similarity between clusters, based on a distance d is defined. Then, the data is clustered using one of two possible approaches. It is possible to start with individual clusters for each of the observations and successively merge the two clusters which are more similar, until the desired number of clusters is achieved (agglomerative clustering). Alternatively, it is possible to start with one big cluster containing all observations and sucessively perform the split that creates the most dissimilar clusters (divisive clustering).

Chapter 3

Functional datasets

This chapter describes some functional datasets that are used in the current work. They arise in applications in a variety of fields such as biology, meteorology, chemometrics or medicine. In this work, they are used to illustrate the methods and tools developed and for benchmarking.

3.1 Berkeley Growth Study

In this dataset, shown in Figure 3.1, the heights of 54 girls and 38 boys have been recorded at 31 non equidistant ages between 1 and 18 (Tuddenham and Snyder, 1954; Ramsay and Silverman, 2005). This dataset has been widely used as an example in binary classification, where the goal is to predict the sex of each observation from the height trajectory (Ramsay and Silverman, 2005; López Pintado and Romo, 2005; Cuevas, Febrero, and Fraiman, 2007; Baíllo and Cuevas, 2008; Cuesta-Albertos and Nieto-Reyes, 2010; Baíllo, Cuevas, and Fraiman, 2010; Sguera, Galeano, and Lillo, 2014; Berrendero, Cuevas, and Torrecilla, 2016b; Cuesta-Albertos, Febrero-Bande, and Oviedo de la Fuente, 2017; Mosler and Mozharovskyi, 2017). Additionally, this dataset is a clear example where misalignment can occur in a natural way. As the growth pattern is different for each kid, we observe that natural landmarks, such as the puberty peak in growth speed, happen at distinct points in time for each child. One can consider that instead the growth is a function of a "biological time" inherent to each kid. Representing the data as a function of this intrinsical time would be the objective of an elastic registration procedure (Srivastava et al., 2011). Note that in this case the warping funtions obtained from registration would be useful for classification, as puberty tends to occur earlier for girls than for boys.

3.2 Medflies dataset

In this dataset the trajectories correspond to daily egg-laying patterns of flies (Carey et al., 1998; Müller et al., 2001). They are 512 30-day curves (beginning at day 5) of flies which live at most 34 days and 266 curves of long-lived flies (that reach the day 44). The curves correspond to the daily count of eggs produced, and the objective is to predict if the fly would be long-lived or not. The original hypothesis was that flies that lay more eggs on the first days would be short-lived, which would justify treating this dataset as a binary classification problem. However, this correlation was found to be very weak, which makes this problem notoriously difficult (Müller and Stadtmüller, 2005; Baíllo and Cuevas, 2008; Baíllo, Cuevas, and Fraiman, 2010; Mosler and Mozharovskyi, 2017). The trajectories are also not very smooth, as they come from a counting process. Nevertheless, displaying the aggregate eggs layed

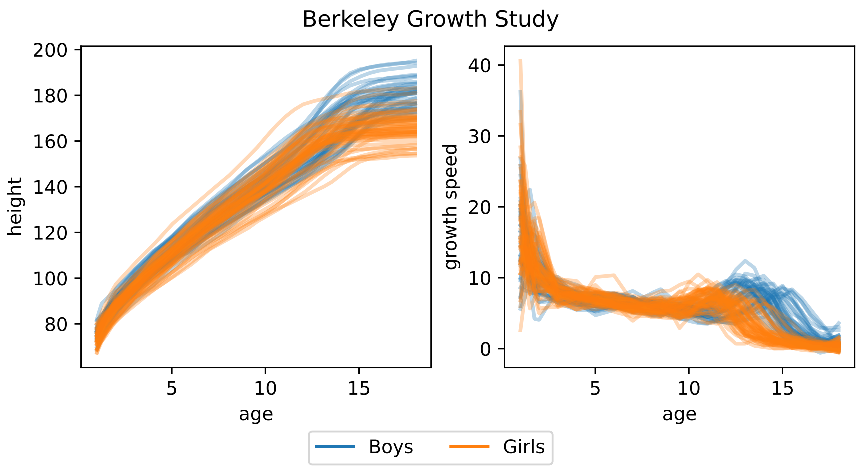


FIGURE 3.1: Berkeley Growth dataset. On the left, the original dataset, with the trajectories of the boys in blue and the girls in orange. On the right, the first derivative, corresponding to growth speed. Here the puberty peak is clearly visible, as well as the fact that in general girls reach puberty earlier than boys.

(which would be analog to taking the integral of the data) shows a more natural-looking functional pattern. We can observe 80 of these curves on Figure 3.2.

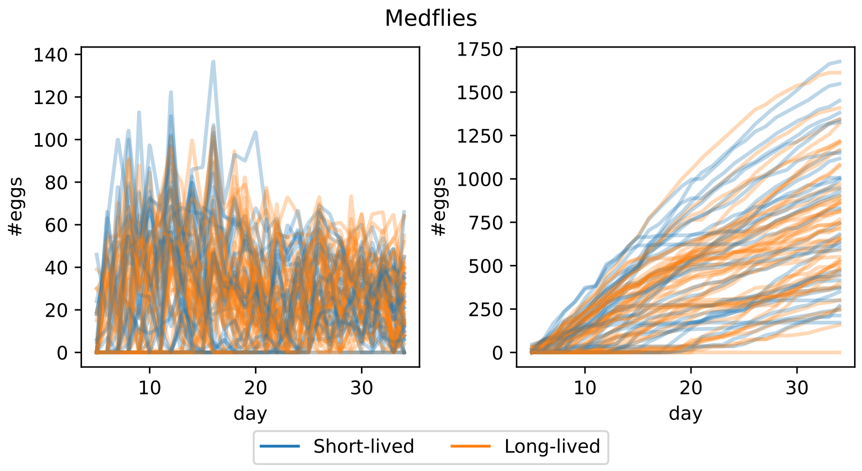


FIGURE 3.2: Medflies dataset. On the left, 80 of the original curves, with the trajectories of short lived flies in blue and the long-lived ones in orange. On the right, the aggregated version of them (with the total eggs laid), in which its functional nature is more apparent.

3.3 Phoneme dataset

The trajectories in this dataset are 1717 log-periodograms constructed from 32 millisecond long recordings of males pronouncing several phonemes: “sh” as in “she” (872 curves), “dcl” as in “dark” (757 curves), “iy” as the vowel in “she” (1163 curves), “aa” as the vowel in “dark” (695 curves), and “ao” as the first vowel in “water” (1022 curves) (Hastie, Tibshirani, and Friedman, 2009). Those trajectories are discretized at 256 equidistant points. This data serves to illustrate the potential role of functional data analysis (FDA) in voice recognition, in a multiclass classification setting. A binary classification version of this dataset is often obtained by keeping the most difficult phonemes to discriminate between: “aa” and “ao”. It is well known that in this dataset the relevant information is on the first variables, corresponding with lower frequencies. Thus, some authors truncate it to the first 150 grid points (Baíllo and Cuevas, 2008; Sguera, Galeano, and Lillo, 2014; Galeano, Joseph, and Lillo, 2015), while others keep only the first 50 (Delaigle and Hall, 2012; Cuesta-Albertos, Febrero-Bande, and Oviedo de la Fuente, 2017; Berrendero, Cuevas, and Torrecilla, 2018). Figure 3.3 illustrates all these different versions of this dataset.

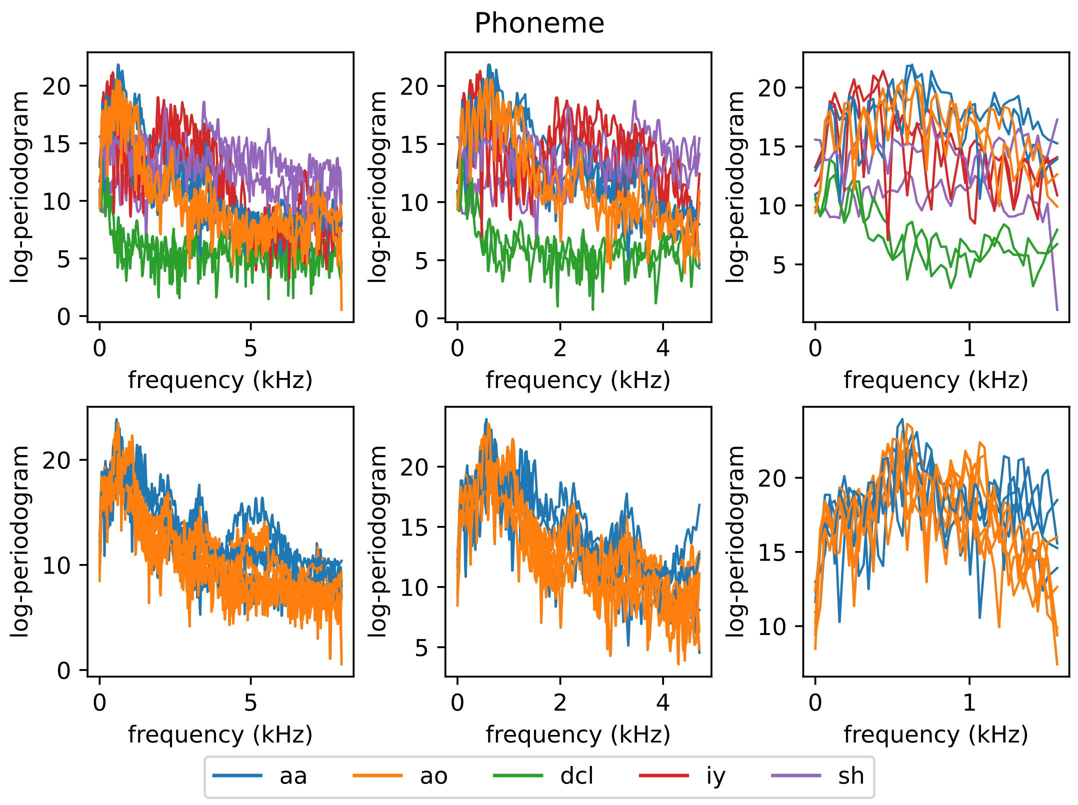


FIGURE 3.3: Phoneme dataset. In the top row the multiclass classification problem is plotted, showing only the first 10 curves. In the bottom row we show the binary classification version of this dataset, again showing only 10 curves. In both cases, from left to right we represent the original data, the data truncated to the first 150 variables and the data truncated to the first 50 variables, respectively.

3.4 Tecator dataset

A regression dataset that consists of 215 near-infrared absorbance spectra of different pieces of finely chopped meat (Borggaard and Thodberg, 1992; Ferraty and Vieu, 2006), as well as their moisture, fat and protein contents. The trajectories are sampled at 100 equally spaced points. The objective is to predict the fat percent of the meat from the spectra. A binary classification version of this dataset can be obtained by discriminating between high fat concentration (above 20%) and low concentration, as proposed in Ferraty and Vieu, 2006. For this dataset the magnitude of the original curves presents very low correlation with the target. Thus, it is often a good choice to discard this information and work instead with the first or the second derivatives (Ferraty and Vieu, 2006; Berrendero, Cuevas, and Torrecilla, 2016b). Both the regression and classification versions of this dataset, including the derivatives, can be seen in Figure 3.4.

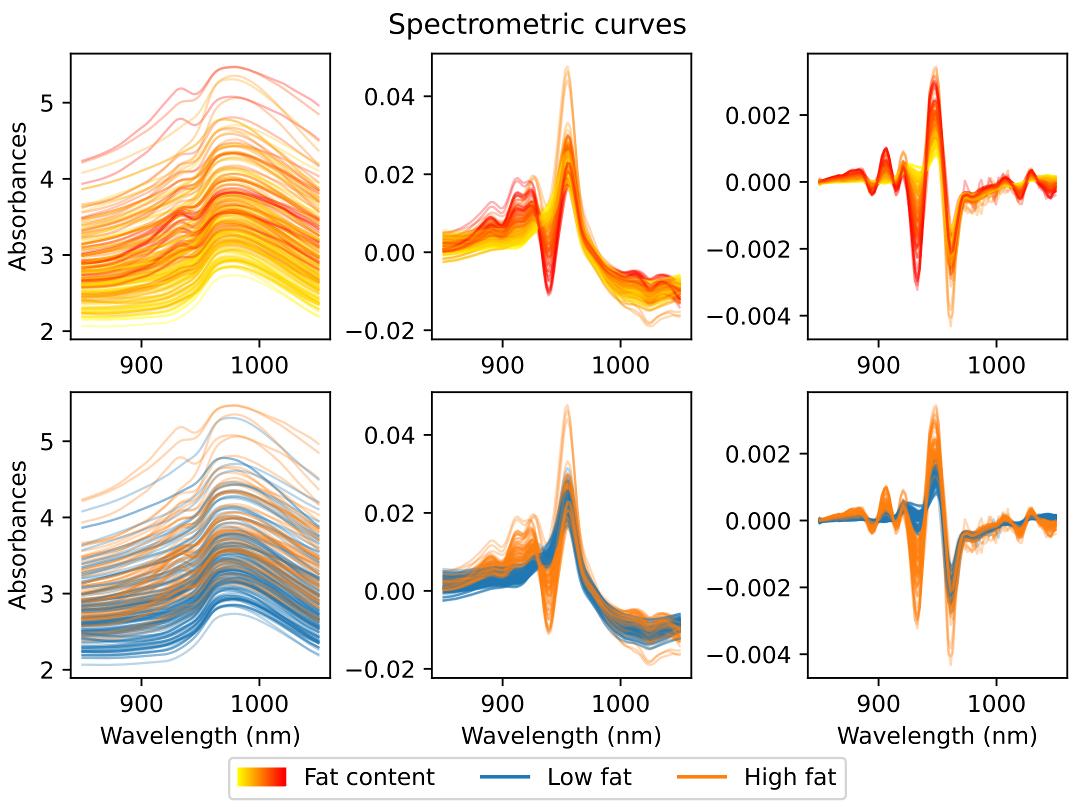


FIGURE 3.4: Tecator dataset. In the top row we represent the regression problem, using a gradient for the target, with more red curves indicating a greater percent of fat. The second row is the binary classification version of this dataset. In both cases, from left to right we represent the original data, the first derivative and the second derivative, respectively.

3.5 Wheat dataset

This dataset contains near-infrarred spectra of 100 samples of wheat and is thus analogous to the previously mentioned Tecator dataset (Kalivas, 1997). These curves have been measured from 1100 nm to 2500 nm in 2 nm intervals, obtaining a total of 701 different wavelengths. The objective of the original regression problem is to predict the measured protein content of the samples from the curve information. A binary classification version of this dataset was created in Delaigle and Hall, 2012, by splitting the data in the observations whose protein content is less than 15 (41 curves) and those with higher content (59 curves). In the same article, the authors recommend working with the first derivative of the data, as in the Tecator dataset. Moreover, they show that in this case the dataset can be considered a real-world problem in which near-perfect classification (see Section 2.3.4) arises. All versions of the dataset are illustrated in Figure 3.5. The binary problem, using the first derivatives, has been employed afterwards in several other classification studies

(Delaigle and Hall, 2013; Cuesta-Albertos, Febrero-Bande, and Oviedo de la Fuente, 2017; Berrendero, Cuevas, and Torrecilla, 2018) and is the one used in this work.

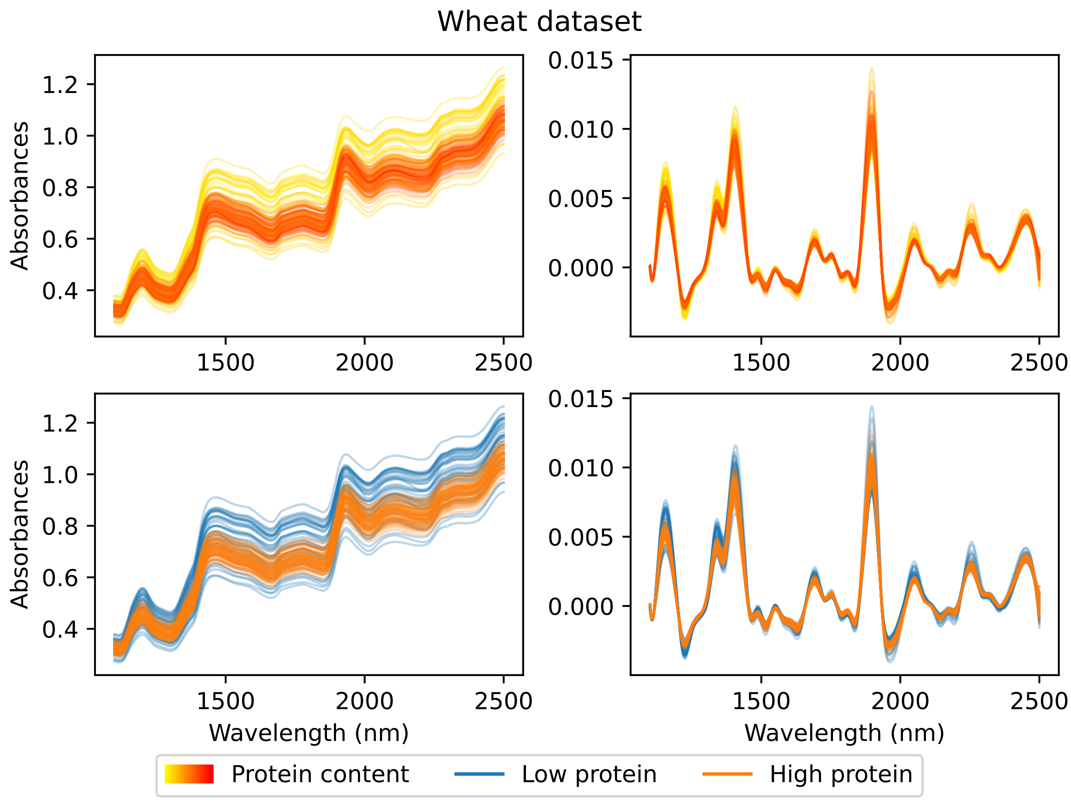


FIGURE 3.5: Wheat dataset. The top row shows the original problem, with the percent of protein depicted as a color gradient, as in Figure 3.4. The second row shows the binary classification version of the problem. In both cases the left plot shows the original curves, and the right plot shows the first derivatives.

3.6 Canadian Weather dataset

In this dataset the anual temperatures at 35 different locations in Canada are averaged over 1960 to 1994 (Ramsay and Silverman, 2005). The measurements are performed daily, and thus we have 365 equally spaced points. This dataset contains additional information about the location and climates of the different stations, which is useful for illustrating clustering with functional data. The dataset also includes precipitation data that we do not use in this work. The temperature curves are shown in the left part of Figure 3.6. Note that the curves exhibit the typical pattern of temperatures in the northern hemisphere, with the lowest temperatures in winter, at the beginning and end of the year, and higher summer temperatures in the middle.

3.7 AEMET dataset

This dataset is very similar to the previously mentioned Canadian weather dataset. The data are taken from the State Meteorological Agency of Spain (AEMET) and contain several meteorological observations (temperature, precipitation and wind

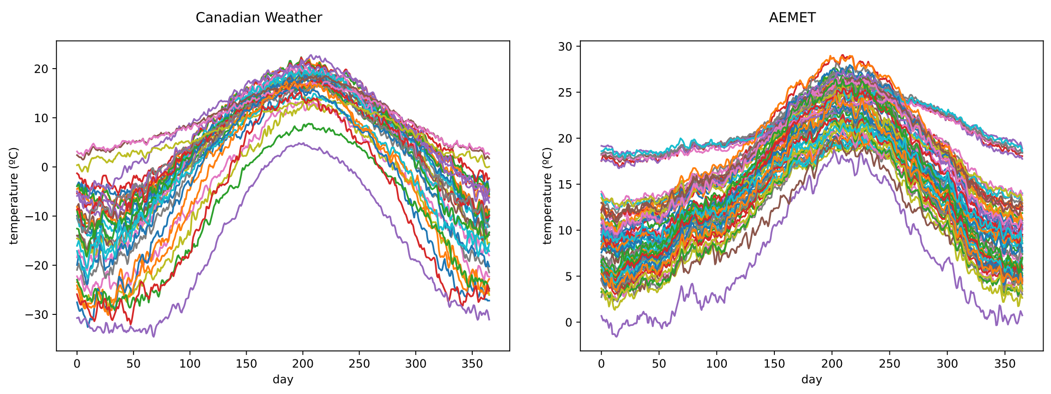


FIGURE 3.6: On the left, the temperarure curves of the Canadian Weather dataset are shown. We can compare them with the temperatures in the AEMET dataset, on the right. Note that both datasets present similar annual variations in temperature, corresponding to the seasonal pattern of the north hemisphere.

speed) at 73 weather stations throughout Spain. These indicators are measured daily ($M = 365$) between 1980 and 2009 and averaged over years. Only the temperature data, shown in the right plot of Figure 3.6, as well as the station locations are used in this work. As opposed to Canadian weather, this dataset presents a very marked cluster with curves that have different shape than the others, with winters not as pronunciated. These are the stations in the Canary Islands, with a very distinctive subtropical climate.

3.8 Australian Weather dataset

This binary classification dataset consists again in averages of yearly measurements from different Australian weather stations. However, the measured quantity is now the precipitation instead of temperature, and the average is taken for all the years that each particular station has been operating (Bureau of Meteorology, 1992). From the original 191 stations, one has been removed because it is considered an outlier, according to Delaigle and Hall, 2010. Using these data, a classification problem has been made, consisting on predicting if the weather station is on the north (43 curves) or the south (147 curves) given a particular precipitation pattern (Delaigle and Hall, 2012). From the precipitation plots shown in Figure 3.7, it is clear that usually stations in the north present a more "tropical" pattern, in which the largest precipitations appear in the last summer months, while the stations on the south have the majority of the rain taking places at the cooler months[1].

3.9 Electrocardiogram (ECG) dataset

This dataset, shown in Figure 3.8, was extracted from the data in the MIT-BIH Arrhythmia Database (Moody and Mark, 2001). This database, compiled originally in 1980, and distributed ever since in several media, is an important resource for medical analysis, and illustrates the potential applications of FDA to this field. The

[1]Note that, as Australia is in the southern hemisphere, summer and winter are reversed with respect to stations in the northern hemisphere, such as in the Canadian weather and AEMET datasets previously mentioned.

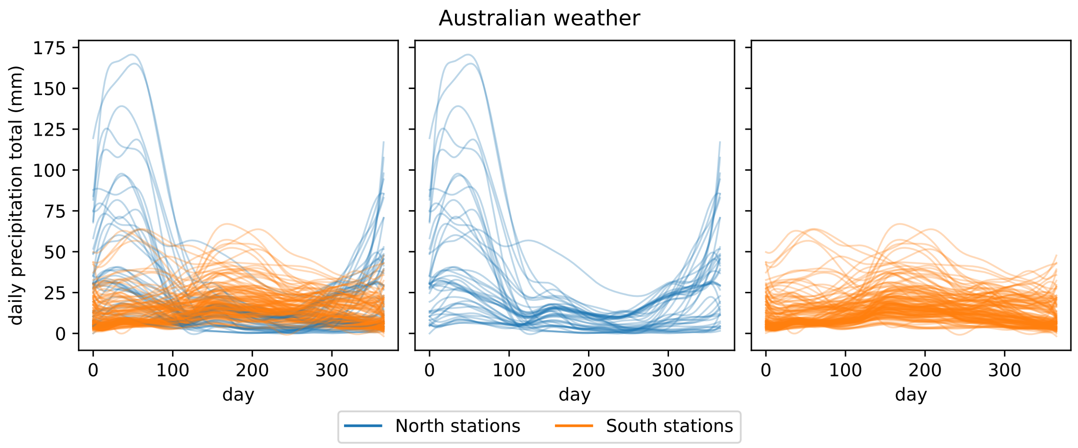


FIGURE 3.7: Precipitation curves of the australian weather dataset. On the left, all curves are displayed. On the middle and right plots the north and south stations are depicted, respectively.

ECG dataset itself contains the measurements recorded by one electrode during one heartbeat. There are 2026 different records, each containing 85 points. It presents a binary classification problem in which we have 520 curves labeled as abnormal and 1506 labeled as normal by a group of cardiologists (Wei and Keogh, 2006). This dataset has been used by several authors in the context of functional classification (Baíllo and Cuevas, 2008; Baíllo, Cuevas, and Fraiman, 2010) and outlier detection (Dai and Genton, 2019).

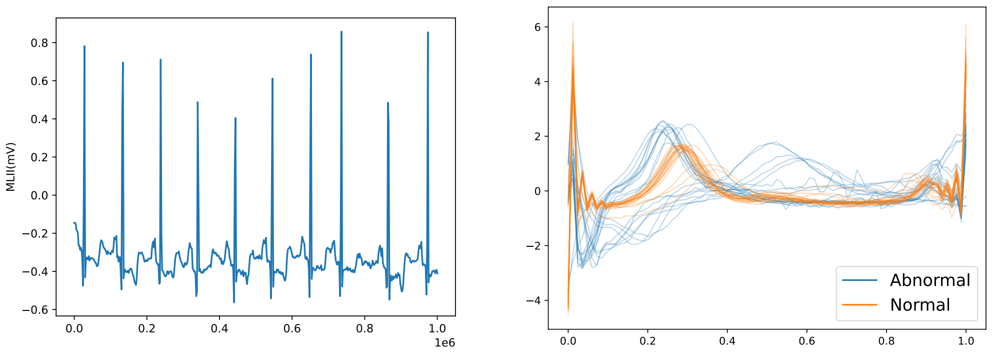


FIGURE 3.8: ECG dataset. On the left, the first ten heartbeats of a particular record in the MIT-BIH Arrhythmia Database are shown. The right plot depicts some of the trajectories from the ECG dataset, containing only one heartbeat, labeled as either abnormal (blue) or normal (orange).

3.10 Mitochondrial calcium overload (MCO) dataset

This dataset contains periodic measurements of mitochondrial calcium overload (MCO) for 89 isolated mouse cardiac cells (Ruiz-Meana et al., 2003). High levels of MCO indicated a higher protection against the ischemia process. The measurements have been taken at 10 second intervals during an hour of simulated ischemia. The objective of the original study was to test the effectiveness of a drug (Cariporide), that selectively blocks ion exchange and was conjectured to increase the

MCO, and thus offer protection to ischemia. To that end, 44 cells have been treated with the drug while the remaining 45 ones were kept as a control group. The data has been analized with analysis of variance (ANOVA) to check that there are differences between the two groups, obtaining a positive result (Cuevas, Febrero, and Fraiman, 2004). In the graph of the curves, shown in Figure 3.9 it is also possible to detect visually a general increase in MCO magnitude in the treatment group, as well as a difference in the shape of the curves between both groups. Posterior works have used this dataset in the context of binary classification, to differentiate between these groups (Baíllo and Cuevas, 2008; Baíllo, Cuevas, and Fraiman, 2010; Cuesta-Albertos, Febrero-Bande, and Oviedo de la Fuente, 2017), as well as a prototypical example of functional data (Cuevas, 2014).

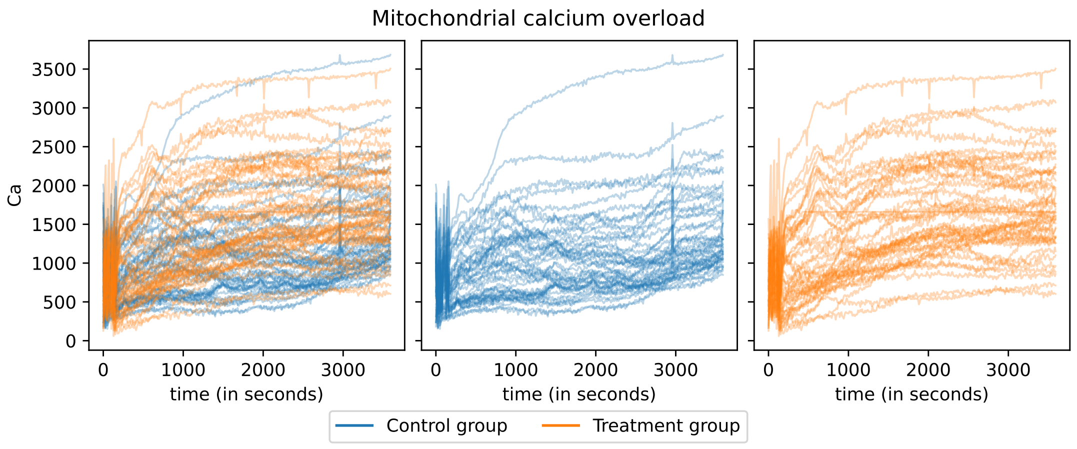


FIGURE 3.9: MCO dataset. On the left, all the measured curves have been plotted together. On the middle and right, the curves in the control and treatment groups, respectively.

3.11 Cell dataset

This dataset, obtained in Spellman et al., 1998, contains 90 curves corresponding to the expression of different genes involved in the cell cycle of the yeast “Saccharomyces cerevisiae”. The expression levels have been measured every 7 minutes during 119 minutes, obtaining a total of 18 time measurements per functional observation. Thus, this is another example of a functional dataset not very densely measured. The genes observed are related with the different phases and phase transitions of the cell cycle: the first gap (G1) phase, in which the cell grows larger; the synthesis (S) phase, in which a copy of the DNA is synthetized in the nucleus; the second gap (G2), involving additional cell growth; and the mitosis (M) phase, in which the cell divides in two. In particular, the dataset contains 44 genes involved in phase G1, 8 involved in phase S, 6 involved in the S/G2 transition, 14 involved in the G2/M transition and 18 involved in the M/G1 transition. The differences between these genes can be observed in Figure 3.10. A balanced binary classification problem can be constructed by discriminating the 44 G1-related genes from the 46 remaining ones, as shown in the top left plot of the figure. This classification problem has been studied in the functional data literature (Leng and Müller, 2006; Rincón Hidalgo and Ruiz-Medina, 2012; Cuesta-Albertos, Febrero-Bande, and Oviedo de la Fuente, 2017).

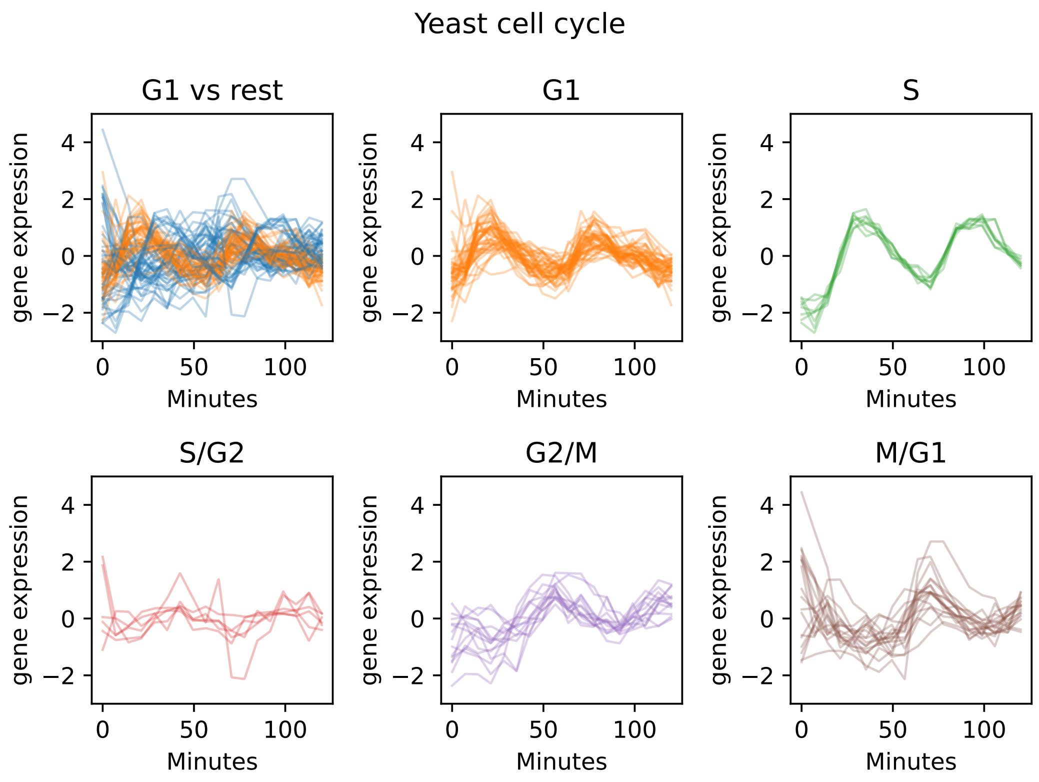


FIGURE 3.10: Cell dataset. The top left plot shows the binary classification problem consisting on differentiating the genes involved in the G1 phase from the rest. The remaining plots depict separately the genes involved in each phase or transition, in the order they take place.

3.12 NO$_x$ dataset

The curves in this dataset are hourly measurements of nitrogen oxides (NO$_x$) during different days in Poblenou, a neighborhood in Barcelona in the surroundings of an industrial area. This is thus an example of functional data which is not very densely observed, as there are only 24 points per observation. From the original 127 daily curves, only 115 observations were complete and kept in the dataset. Of these, 39 are non-working days, including Saturdays, Sundays and festive days, and the remaining 76 correspond to working days. These curves are displayed in Figure 3.11. The levels of NO$_x$ increase in the morning, peaking at 8:00, decrease until 14:00 and increase again afterwards. This suggest a strong dependence on the traffic patterns. This is further evidenced when comparing the curves on the working and non-working days, as the decrease on traffic on the latter can be observed as lower NO$_x$ levels. This dataset has been used in functional outlier detection (Febrero, Galeano, and González-Manteiga, 2008; Sguera, Galeano, and Lillo, 2016). As the two groups are not very unbalanced, this dataset is also useful for binary classification.

3.13 UCR datasets

The UCR/UEA time series classification archive (Dau et al., 2019; Bagnall et al., 2018) is one of the most well known repositories for time series and functional datasets.

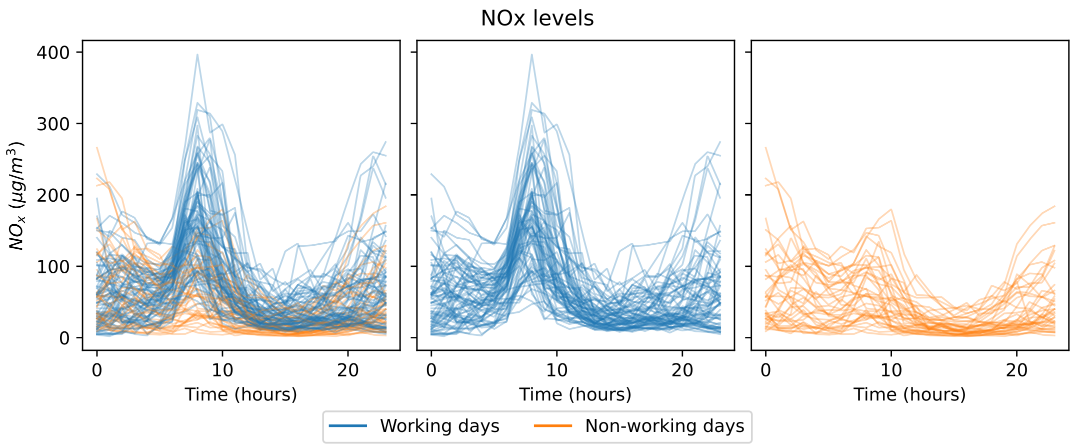


FIGURE 3.11: NO_x dataset. On the left, all the measured curves have been plotted together. On the middle and left, the curves corresponding to working and non-working days, respectively.

Created originally by the University of California Riverside, the repository contained univariate functional classification problems. This was later expanded to include also multivariate functional datasets, with the addition of new datasets from the University of East Anglia. It is now hosted at www.timeseriesclassification.com. In this work we use several of the datasets located in this repository. Some of them are depicted in Figure 3.12.

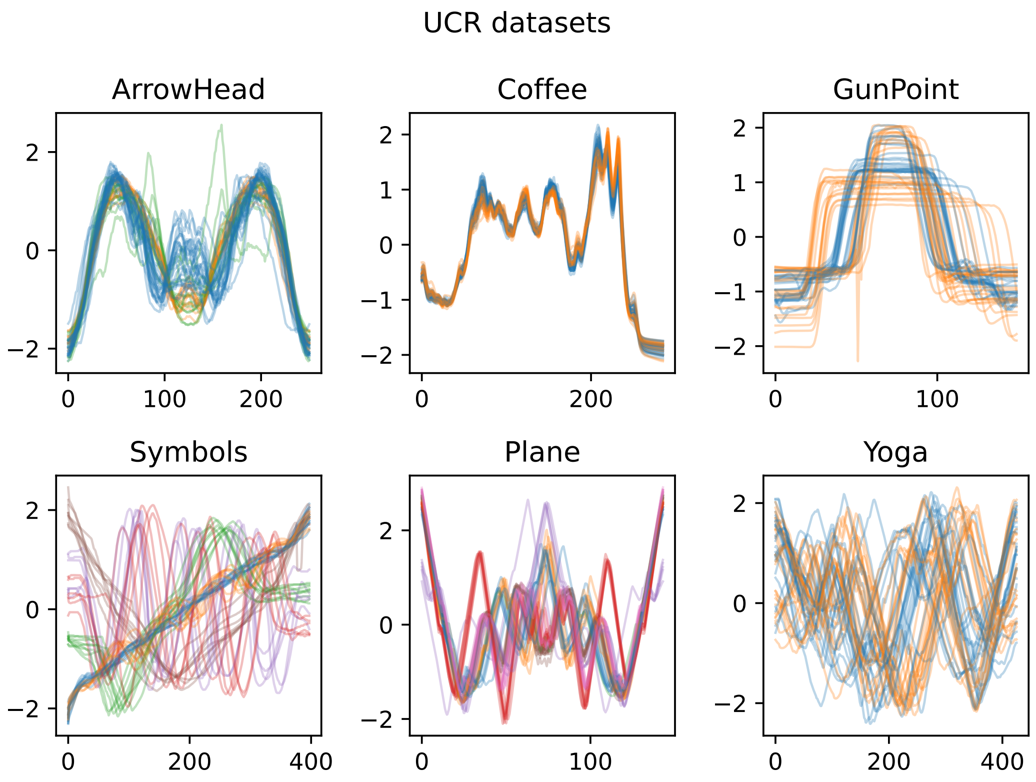


FIGURE 3.12: UCR datasets. Several of the UCR datasets used in this work. Only the first 50 curves of each are shown.

Part I

Methodological advances

Chapter 4

Classification of Gaussian processes

In this chapter we introduce a procedure to derive optimal rules for the discrimination of trajectories sampled from two Gaussian processes (Torrecilla et al., 2020). To derive these rules we first consider the problem of classifying the discrete-time process that results from monitoring the Gaussian process at a finite set of times. Since the joint distribution of the values of the discretely monitored process is a multivariate Gaussian random variable, the Bayes classification rule is the quadratic discriminant (Hastie, Tibshirani, and Friedman, 2009). By taking the limit of this quadratic discriminant as the set of monitoring points becomes dense, one obtains an optimal rule for the classification of the continuous-time Gaussian processes. In the general case, this is a singular limit because some of the terms in the discriminant rule diverge. Carrying out a detailed analysis of these optimal classification rules and their singularities in the dense monitoring limit, we provide novel derivations of some known results and gain insight into the mechanisms by which near-perfect classification occurs (Delaigle and Hall, 2012). This singular limit is a consequence of the orthogonality of the probability measures associated to the stochastic processes from which the trajectories are sampled. As a further novel result of this analysis, we formulate rules to determine whether two Gaussian processes are equivalent or mutually singular (orthogonal).

The structure of this chapter is as follows: the functional classification problem and the discretization methodology are presented in Section 4.1. The procedure for deriving optimal rules for the classification of Gaussian processes based on discrete monitoring is introduced in Section 4.2. The type of classification problem that is obtained depends on whether the Gaussian processes are equivalent (non-singular classification) or orthogonal (singular, near-perfect classification). For this reason, the conditions for the equivalence of Gaussian processes are discussed in Section 4.3. Sections 4.4 and 4.5 are devoted to homoscedastic and heteroscedastic classification problems, respectively. An experimental evaluation of the limit rules derived in this work and a comparison with other functional classification methods is presented in Section 4.6 for both simulated and real-world problems. Finally, Section 4.7 provides a summary of the conclusions of this work.

4.1 Statement of the problem

Consider the binary classification problem in which the data instances are characterized by trajectories x, sampled from one of two different Gaussian processes (GPs) defined on the interval $\mathcal{T} = [0, T]$ in the real line

$$\begin{cases} (X(t) \mid Y = 0) = Z_0(t) & \text{with probability } 1 - p \\ (X(t) \mid Y = 1) = \mu(t) + Z_1(t) & \text{with probability } p, \end{cases} \tag{4.1}$$

where $Z_0(t)$ and $Z_1(t)$ are zero-mean Gaussian processes, whose covariance functions are k_0 and k_1.

If $\mathbb{P}_1$ is absolutely continuous with respect to $\mathbb{P}_0$, the optimal classification rules can be expressed in terms of the Radon-Nikodym derivative of the measures:

$$g^*(x) = \mathbb{I}_{\left\{\frac{d\mathbb{P}_1}{d\mathbb{P}_0}(x) > \frac{1-p}{p}\right\}}. \tag{4.2}$$

In other cases, an expression of the optimal classification rule that is valid in all cases is not known. Nevertheless, one can consider classification rules that are of the same form as Equation (4.2), in the hope that they approximate the optimal classification rule in some limit. In particular Delaigle and Hall, 2013; Galeano, Joseph, and Lillo, 2015; Dai, Müller, and Yao, 2017 propose to use density ratios of finite dimensional projections. The derivation of explicit forms for the optimal rule for a limited class of functional classification problems of this type has also been considered earlier in the literature mainly in the homoscedastic ($k_0 = k_1 = k$) setting (Kailath, 1966; Kailath, 1971). However, the derivation of optimal classification rules in the heteroscedastic setting ($k_0 \neq k_1$) and for singular cases in which near-perfect classification is obtained remains elusive (Delaigle and Hall, 2012; Delaigle and Hall, 2013; Dai, Müller, and Yao, 2017; Berrendero, Cuevas, and Torrecilla, 2018; Cuesta-Albertos and Dutta, 2022).

4.2 Optimal rules for Gaussian process classification

Consider a stochastic process X defined by Equation (4.1) with $Z_0(t)$ and $Z_1(t)$ zero-mean Gaussian processes whose covariance functions are k_0 and k_1, respectively. Assume that the trajectories of this process are monitored at a set of appropriately chosen distinct discrete times $\mathbf{t}_M = \{t_m\}_{m=1}^{M} \in \mathcal{T}^M$. Let $\mathbf{X}$ represent the M-dimensional random column vector whose components are the discretely monitored values of the trajectories

$$\mathbf{X}^T = \left(X(t_1), X(t_2), \ldots, X(t_M)\right), \tag{4.3}$$

where the superscript T indicates the standard transposition of matrices. By the properties of Gaussian processes, the class-conditioned distribution of $\mathbf{X}$ is a multivariate Gaussian

$$\begin{aligned} \mathbf{X} \mid Y = 0 &\sim N\left(\mathbf{0}, \mathbf{K}_0\right) \quad \text{w. p. } 1 - p \\ \mathbf{X} \mid Y = 1 &\sim N\left(\boldsymbol{\mu}, \mathbf{K}_1\right) \quad \text{w. p. } p \end{aligned} \tag{4.4}$$

where

$$\boldsymbol{\mu}^T = \left(\mu(t_1), \mu(t_2), \ldots, \mu(t_M)\right) \tag{4.5}$$

is a row vector whose components are the values of mean of the class 1 trajectories at the monitoring times. The corresponding column vector is denoted by $\boldsymbol{\mu}$.

The quantities $\mathbf{K}_0$ and $\mathbf{K}_1$ are the corresponding $M \times M$ Gram matrices. The elements of these matrices are the autocovariances of the discretely monitored processes

$$\left(\mathbf{K}_i\right)_{mn} = k_i(t_n, t_m) = \mathbb{E}\left[Z_i(t_n) Z_i(t_m)\right], \tag{4.6}$$

for $i = 0, 1$, and $n, m = 1, 2, \dots M$. Since they characterize the structure of autocovariances, Gram matrices are positive-semidefinite (i.e., their eigenvalues are non-negative). If they have zero eigenvalues, in what follows, the derivations apply to the space spanned by the eigenvectors corresponding to the positive (non-zero) eigenvalues.

In the general heteroscedastic case, the Bayes rule for this multivariate Gaussian binary classification problem is the quadratic discriminant (see, e.g., Hastie, Tibshirani, and Friedman, 2009)

$$g^*(\mathbf{x}) = \mathbb{I}_{\left\{-\frac{1}{2}\log\frac{|\mathbf{K}_1|}{|\mathbf{K}_0|}-\frac{1}{2}\mathbf{x}^T\left(\mathbf{K}_1^{-1}-\mathbf{K}_0^{-1}\right)\mathbf{x}+\mathbf{x}^T\mathbf{K}_1^{-1}\mu-\frac{1}{2}\mu^T\mathbf{K}_1^{-1}\mu>\log\frac{1-p}{p}\right\}}, \tag{4.7}$$

where $\mathbb{I}$ is the indicator function and $|\mathbf{K}_0|, |\mathbf{K}_1|$ are the determinants of the corresponding covariance matrices.

We conjecture that this rule tends to the optimal rule for the functional case when $M \to \infty$ and the set of monitoring points $\mathbf{t}_M = \{t_m\}_{m=1}^M$ becomes dense in $\mathcal{T}$. In that case, the limit of this rule can be formally written as

$$g^*(x) = \mathbb{I}_{\left\{-\frac{1}{2}\log\frac{|\mathcal{K}_1|}{|\mathcal{K}_0|}-\frac{1}{2}\left(\langle x,x\rangle_{k_1}-\langle x,x\rangle_{k_0}\right)+\langle x,\mu\rangle_{k_1}-\frac{1}{2}\langle \mu,\mu\rangle_{k_1}>\log\frac{1-p}{p}\right\}}. \tag{4.8}$$

The angular brackets $\langle \cdot, \cdot \rangle_{k_i}$ denote the inner product in $\mathcal{H}_{k_i}$, the reproducing kernel Hilbert space (RKHS) associated to the kernel k_i, or, if such quantity is ill-defined, the extension mapping defined in Equation (2.43). The quantity $\frac{|\mathcal{K}_1|}{|\mathcal{K}_0|}$ represents the asymptotic form of the ratio of determinants of the Gram matrices $\frac{|\mathbf{K}_1|}{|\mathbf{K}_0|}$ in the limit of dense monitoring (see Appendix A).

In general cases, this limit is singular. A first type of singularity occurs if $\mu \notin \mathcal{H}_{k_1}$. In such case, the terms $\langle x, \mu\rangle_{k_1}$ and $\langle \mu, \mu\rangle_{k_1}$ in Equation (4.8) diverge. A second type of singularity appears in the quadratic terms of the discriminant when the Hilbert space $\mathcal{H}_{k_i}$ is infinite-dimensional. In that case, the trajectories of the process X do not belong to $\mathcal{H}_{k_i}$ with probability one (Kailath, 1971; Berlinet and Thomas-Agnan, 2004). Therefore, in the dense monitoring limit, the quantities $\langle x, x\rangle_{k_i}$ also diverge. Finally, also for infinite-dimensional $\mathcal{H}_{k_i}$, the determinant of the corresponding covariance operator, $|\mathcal{K}_i|$, vanishes and its logarithm diverges.

As illustrated in this work, these singularities are in fact at the origin of the near-perfect classification phenomenon (Delaigle and Hall, 2012). Specifically, if the singularities present in the classification rule Equation (4.8) cancel out, the measures of the two underlying Gaussian processes are equivalent. Otherwise, they are mutually singular (orthogonal) and near-perfect classification is obtained.

From these observations, we note that Equation (4.8), which can be seen as the functional generalization of the quadratic discriminant for multivariate data, should be viewed only as a mnemonic for Equation (4.7) in the limit of dense monitoring. In subsequent sections, the singular limit of this rule is analyzed in detail for different classification problems in both the homo- and heteroscedastic settings. In particular, the conditions for the equivalence between the two Gaussian processes are discussed in Section 4.3. In Section 4.4 we analyze homoscedastic classification problems, for which $k_0 = k_1 = k$. In this case, if $\mu = \mu_1 - \mu_0 \in \mathcal{H}_k$, the laws $\mathbb{P}_0$ and $\mathbb{P}_1$ are equivalent. In consequence, the classification problem is not singular and Equation (4.8) is the Bayes rule (Berrendero, Cuevas, and Torrecilla, 2018). Near-perfect classification is obtained when μ does not belong to $\mathcal{H}_k$. In such case the singularities in the terms that involve μ dominate in Equation (4.8). Nonetheless, it is still possible to derive optimal classification rules by carrying out a careful analysis of the behavior of those divergent terms in this singular limit. The general heteroscedastic classification problem $k_0 \neq k_1$ is analyzed in Section 4.5. Near-perfect classification can be obtained by an alternative mechanism that involves the quadratic terms of the discriminant, which are singular. As in the homoscedastic case, if the singularities in Equation (4.8) cancel out, $\mathbb{P}_0$ and $\mathbb{P}_1$ are equivalent, and the classification problem is not singular.

4.3 Equivalence of Gaussian processes

From the form of the optimal classification rule introduced in the previous section it is possible to derive conditions for the equivalence of the probability measures $\mathbb{P}_0$ and $\mathbb{P}_1$ associated to the Gaussian processes $GP(\mu_0, k_0)$ and $GP(\mu_1, k_1)$, respectively.

The derivation starts from the observation that $\mathbb{P}_0$ and $\mathbb{P}_1$ are equivalent if the corresponding classification problem is not singular (Baíllo, Cuevas, and Cuesta-Albertos, 2011; Berrendero, Cuevas, and Torrecilla, 2018). In that case, the optimal classification rule is the one in Equation (4.2).

If $\mu_0 \neq 0$ it is possible to determine whether the trajectory $x \in \mathcal{X}$ has been sampled either from $GP(\mu_0, k_0)$ or from $GP(\mu_1, k_1)$ using Equation (4.8) with the replacements $x - \mu_0$ for x and $\mu = \mu_1 - \mu_0$. The classification rule becomes

$$g^*(x) = \mathbb{I}_{\left\{-\frac{1}{2}\log\frac{|\mathcal{K}_1|}{|\mathcal{K}_0|} - \frac{1}{2}\left(\|x-\mu_1\|_{k_1}^2 - \|x-\mu_0\|_{k_0}^2\right) > \log\frac{1-p}{p}\right\}}, \tag{4.9}$$

where $\|x - \mu_i\|_{k_i}^2 = \langle x - \mu_i, x - \mu_i\rangle_{k_i}$, with $i = 0, 1$, which are the functional analogues of the Mahalanobis distance (Galeano, Joseph, and Lillo, 2015; Berrendero, Bueno-Larraz, and Cuevas, 2020). Again, the quantities that appear in this expression exhibit singularities and should therefore be interpreted as the corresponding discrete approximations in the limit of dense monitoring.

A first type of singularity in this classification rule arises when $\mu = \mu_1 - \mu_0 \notin \mathcal{H}_{k_0} \cap \mathcal{H}_{k_1}$. Consider the case in which $\mu \notin \mathcal{H}_{k_0}$. If $X \sim GP(\mu_1, k_1)$, then it can be written as $X = \mu_1 + Z_1$, where Z_1 is the zero-mean Gaussian process $GP(0, k_1)$. In the expression of $\|x - \mu_0\|_{k_0}^2$ one would get, among others, the term $\|\mu_1 - \mu_0\|_{k_0}^2$, which diverges because $\mu = \mu_1 - \mu_0 \notin \mathcal{H}_{k_0}$. Note that this singularity cannot be cancelled by any other term in Equation (4.9). A parallel argument can be used when $\mu \notin \mathcal{H}_{k_1}$, interchanging the subindices 0 and 1.

Even if one assumes that $\mu = \mu_1 - \mu_0 \in \mathcal{H}_{k_0} \cap \mathcal{H}_{k_1}$, which implies that the divergences described in the previous paragraph are not obtained, a second type of singularity can appear in the quadratic terms of Equation (4.9). Specifically, the terms $\|x - \mu_i\|_{k_i}^2$, $i = 0, 1$ diverge when $\mathcal{H}_{k_i}$ is infinite dimensional (Berrendero, Bueno-Larraz, and Cuevas, 2020). However, if the Gaussian processes are equivalent, the singularities in the term $\left(\|x - \mu_1\|_{k_1}^2 - \|x - \mu_0\|_{k_0}^2\right)$ cancel out, so that the classification rule given by Equation (4.9) is well defined.

A related singularity affects also the term that involves the ratio of the determinants of the covariance operators. If the Hilbert space $\mathcal{H}_{k_i}$ is infinite dimensional, zero is an accumulation point of the spectrum of the covariance operator k_i (Spence, 1975). Therefore, the individual determinants of the covariance operators vanish

$$|\mathcal{K}_i| \equiv \lim_{D\to\infty} \prod_{j=1}^{D} \lambda_{ij} = 0, \quad i = 0, 1, \tag{4.10}$$

where $\{\lambda_{0j}\}_{j=1}^{\infty}$ and $\{\lambda_{1j}\}_{j=1}^{\infty}$ are the eigenvalues of $\mathcal{K}_0$ and $\mathcal{K}_1$, respectively. Therefore, the condition for equivalence between $\mathbb{P}_0$ and $\mathbb{P}_1$ is that the ratio of the determinants

$$\frac{|\mathcal{K}_0|}{|\mathcal{K}_1|} = \lim_{M\to\infty} \frac{|\mathbf{K}_0|}{|\mathbf{K}_1|} = \lim_{D\to\infty} \prod_{j=1}^{D} \frac{\lambda_{0j}}{\lambda_{1j}}, \tag{4.11}$$

be finite and different from zero. The last equality in Equation (4.11), which involves the limit of dense monitoring, was derived in Appendix A.

In summary, if the processes are equivalent, the classification problem is not singular. Therefore,

$$\mu = \mu_1 - \mu_0 \in \mathcal{H}_{k_0} \cap \mathcal{H}_{k_1}, \tag{4.12}$$

so that the terms that depend solely on the means in Equation (4.9) are well-defined, and the singularities of the quadratic terms cancel out

$$\|x - \mu_1\|_{k_1}^2 - \|x - \mu_0\|_{k_0}^2 < \infty, \tag{4.13}$$

$$0 < \frac{|\mathcal{K}_0|}{|\mathcal{K}_1|} = \lim_{D\to\infty} \prod_{j=1}^{D} \frac{\lambda_{0j}}{\lambda_{1j}} < \infty. \tag{4.14}$$

If these conditions do not hold, the classification problem is singular and the measures are not equivalent. In consequence, according to the Hájek-Feldman dichotomy (Hájek, 1958; Feldman, 1958), they are mutually singular (orthogonal).

For processes that are equivalent, combining Equations (4.2) and (4.9), the Radon-Nikodym derivative of $\mathbb{P}_1$ with respect to $\mathbb{P}_0$ is

$$\frac{d\mathbb{P}_1}{d\mathbb{P}_0}(x) = \left(\frac{|\mathcal{K}_0|}{|\mathcal{K}_1|}\right)^{1/2} \exp\left(-\frac{1}{2}\left(\|x - \mu_1\|_{k_1}^2 - \|x - \mu_0\|_{k_0}^2\right)\right). \tag{4.15}$$

Furthermore, the inner products in $\mathcal{H}_{k_0}$ and $\mathcal{H}_{k_1}$, the RKHS corresponding to the kernels k_0 and k_1, respectively, are related by the expression

$$\langle f, g\rangle_{k_1} = \langle f, g\rangle_{k_0} - \langle f, \langle \delta k, g\rangle_{k_1}\rangle_{k_0}, \quad f, g \in \mathcal{H}_{k_0} \cap \mathcal{H}_{k_1}, \tag{4.16}$$

where $\delta k = k_1 - k_0$. This relation can be proven making use of the reproducing property of k_0 in $\mathcal{H}_{k_0}$, and of k_1 in $\mathcal{H}_{k_1}$:

Let $f \in \mathcal{H}_{k_0} \cap \mathcal{H}_{k_1}$. Using $g(\cdot) = k_1(x, \cdot)$ in Equation (4.16)

$$\begin{aligned}
\langle f(\cdot), k_1(x,\cdot)\rangle_{k_1} &= \langle f(\cdot), k_1(x,\cdot)\rangle_{k_0} - \langle f(\cdot), \langle \delta k(\cdot,\cdot), k_1(x,\cdot)\rangle_{k_1}\rangle_{k_0} \\
&= \langle f(\cdot), k_1(x,\cdot)\rangle_{k_0} - \langle f(\cdot), \delta k(x,\cdot)\rangle_{k_0} \\
&= \langle f(\cdot), k_0(x,\cdot)\rangle_{k_0} \\
&= f(x).
\end{aligned}$$

Using these results for equivalence, we now proceed to analyze the classification of Gaussian processes in both the homo- and heteroscedastic settings.

4.4 Homoscedastic classification problems

In homoscedastic classification problems, the kernels of the Gaussian processes for the two classes are equal $k_0 = k_1 = k$. Therefore, the quadratic terms of the functional discriminant function (i.e., the first two terms on the left-hand side of the expression inside the indicator function in Equation (4.7)) cancel out. The optimal rule for the discretely monitored process is Fisher's linear discriminant

$$g^*(\mathbf{x}) = \mathbb{I}_{\left\{\mathbf{x}^T\mathbf{K}^{-1}\boldsymbol{\mu}-\frac{1}{2}\boldsymbol{\mu}^T\mathbf{K}^{-1}\boldsymbol{\mu}>\log\frac{1-p}{p}\right\}}. \tag{4.17}$$

In the limit of dense monitoring the decision rule can be formally written as

$$g^*(x) = \mathbb{I}_{\left\{\langle x,\mu\rangle_k-\frac{1}{2}\langle \mu,\mu\rangle_k>\log\frac{1-p}{p}\right\}}. \tag{4.18}$$

The angular brackets, $\langle\cdot,\cdot\rangle_k$ denote the inner product in $\mathcal{H}_k$, the RKHS associated to the kernel k, or, if such quantity is ill-defined, the extension mapping defined in Equation (2.43). Note that this is the same rule as in Equation (B.28), obtained from prior work.

In what follows, the Bayes rule will be derived for general non-singular homoscedastic GP classification problems. Then we will illustrate how to derive optimal rules in specific singular instances of such problems.

4.4.1 Non-singular homoscedastic classification

In a homoscedastic setting, the classification problem is not singular if $\mu = \mu_1 - \mu_0 \in \mathcal{H}_k$ (Lemma 5d of Parzen, 1961b). In that case, $\mathbb{P}_0$ and $\mathbb{P}_1$ are equivalent. Provided that an appropriate interpretation is given to its constituents, Equation (4.18) is the Bayes rule for this functional classification problem. The error of this optimal rule, which is the infinite-dimensional analogue of Fisher's linear discriminant, was given in Equation (2.86).

As mentioned earlier, the terms in Equation (4.18) need to be given an appropriate interpretation in the limit of dense monitoring. Let's consider first the term that involves the inner product of μ

$$\lim_{M\to\infty} \boldsymbol{\mu}^T\mathbf{K}^{-1}\boldsymbol{\mu} = \langle\mu,\mu\rangle_k = \|\mu\|_k^2. \tag{4.19}$$

The convergence of the discretized approximation to the square norm of $\mu \in \mathcal{H}_k$ can be proven for monotone increasing (nested) sets of monitoring times using Lemma 5c of Parzen, 1961b.

If the RKHS is infinite dimensional, the trajectories of the random process do not belong to $\mathcal{H}_k$ with probability one (Kailath, 1971; Lukić and Beder, 2001; Berlinet and Thomas-Agnan, 2004). In such case, $\langle x,\mu\rangle_k$ cannot represent an inner product in $\mathcal{H}_k$. Nonetheless, since $\mu \in \mathcal{H}_k$, we have

$$\begin{aligned} \mathbb{E}\left[Z(t_m)\left(\mathbf{x}^T\mathbf{K}^{-1}\boldsymbol{\mu}\right)\right] &= \mu(t_m),\ t_m \in \mathbf{t}_M \\ \mathbb{E}\left[Z(t)\left(\mathbf{x}^T\mathbf{K}^{-1}\boldsymbol{\mu}\right)\right] &= \hat{\mu}(t),\ t \notin \mathbf{t}_M, \end{aligned} \tag{4.20}$$

where $Z(t)$ is a random function sampled from a zero-mean Gaussian process with a kernel function k. The quantity $\hat{\mu}(t)$ is the optimal prediction for $\mu(t)$ with $t \in \mathcal{T}$, assuming that $\{\mu(t_m)\}_{m=1}^M$, the values of the mean at the monitoring times, are known (Rasmussen and Williams, 2005). In the limit of dense monitoring

$$\lim_{M\to\infty} \mathbb{E}\left[Z(t)\left(\mathbf{x}^T\mathbf{K}^{-1}\boldsymbol{\mu}\right)\right] = \mu(t), \quad \forall t \in \mathcal{T}, \tag{4.21}$$

for any $\mu \in \mathcal{H}_k$. Extending by continuity this relation to all $t \in \mathcal{T}$, and using Equation (2.44), the dense-monitoring limit of this linear functional defines Loève's isometry

$$\langle x, \mu \rangle_k = \lim_{M \to \infty} \mathbf{x}^T \mathbf{K}^{-1} \boldsymbol{\mu} = \psi_x^{-1}(\mu). \tag{4.22}$$

The spectral form of this *congruence* inner product is (Berlinet and Thomas-Agnan, 2004)

$$\langle x, \mu \rangle_k = \sum_{j=1}^{\infty} \frac{\mu_j \xi_j}{\lambda_j}, \tag{4.23}$$

where $\{\lambda_j\}_{j=1}^{\infty}$ are the eigenvalues of $\mathcal{K}$, and $\{\mu_j\}_{j=1}^{\infty}$, $\{\xi_j\}_{j=1}^{\infty}$ are the coefficients of the Karhunen-Loève expansions

$$\mu(t) = \sum_{j=1}^{\infty} \mu_j \phi_j(t), \tag{4.24}$$

$$x(t) = \sum_{j=1}^{\infty} \xi_j \phi_j(t), \tag{4.25}$$

respectively.

4.4.2 Singular (near-perfect) homoscedastic classification

When $\mu \notin \mathcal{H}_k$, the measures $\mathbb{P}_0$ and $\mathbb{P}_1$ are mutually singular (orthogonal). In this case, near-perfect classification is obtained (Parzen, 1961a; Kailath, 1966; Kailath, 1971; Berrendero, Cuevas, and Torrecilla, 2018). The terms $\langle x, \mu \rangle_k$ and $\langle \mu, \mu \rangle_k$ diverge. These divergences, which are of the same type, dominate in Equation (4.18). Therefore the term that depends on the class priors, which is non-singular for $0 < p < 1$, can be dropped out. With an appropriate interpretation of the limit, the decision rule is

$$\begin{aligned} g^*(x) &= \mathbb{I}_{\{\eta^*(x) > 0\}} \\ \eta^*(x) &= \lim_{\hat{\mu}_{\mathcal{H}} \to \mu} \left(\langle x, \hat{\mu}_{\mathcal{H}} \rangle_k - \frac{1}{2} \langle \hat{\mu}_{\mathcal{H}}, \hat{\mu}_{\mathcal{H}} \rangle_k \right), \end{aligned} \tag{4.26}$$

where $\hat{\mu}_{\mathcal{H}} \in \mathcal{H}_k$ is an approximation to the mean $\mu \notin \mathcal{H}_k$, whose squared norm is $\langle \hat{\mu}_{\mathcal{H}}, \hat{\mu}_{\mathcal{H}} \rangle_k$. The quantity $\langle x, \hat{\mu}_{\mathcal{H}} \rangle_k$ is defined through Loève's isometry.

The limit in Equation (4.26) needs to be understood as follows: since the elements of $\mathcal{H}_k$ are dense in $L^2(\mathcal{T})$ when k has no zero eigenvalues (Cucker and Zhou, 2007), it is possible to build a sequence of approximating classification problems with $\hat{\mu}_{\mathcal{H}} \in \mathcal{H}_k$ that converges to $\mu \in L^2(\mathcal{T})$. The singular homoscedastic classification problem with $\mu \notin \mathcal{H}_k$ can be seen as the limit of a sequence of classification problems involving functions $\hat{\mu}_{\mathcal{H}} \in \mathcal{H}_k$ in the approximating sequence (Theorem 6 of Berrendero, Cuevas, and Torrecilla, 2018). An optimal classification rule for these related problems is given by Equation (4.18). Since $\langle \mu, \mu \rangle_k$ diverges, the corresponding classification errors, which are given by limit of Equation (2.86), tend to zero.

Brownian processes with different means

This singular limit can be illustrated in the discrimination of trajectories sampled from one of two Brownian motions with the same variance but different means. Let us consider a homoscedastic classification problem in which the class 0 trajectories are realizations of a zero-mean Brownian motion in $t \in [0, T]$ and the class 1 trajectories are sampled from a Brownian motion with a piecewise linear mean

$$\mu(t) = \begin{cases} 0 & 0 \leq t < t_1 \\ \mu_T \frac{t-t_1}{t_2-t_1} & t_1 \leq t < t_2 \\ \mu_T & t_2 \leq t < T \end{cases}, \tag{4.27}$$

with $\mu_T = \mu(T)$, a constant, and $0 < t_1 \leq t_2 < T$. Its derivative is

$$\mu'(t) = \begin{cases} 0 & 0 \leq t < t_1 \\ \mu_T \frac{1}{t_2-t_1} & t_1 \leq t < t_2 \\ 0 & t_2 \leq t < T \end{cases}. \tag{4.28}$$

Recall from Section 2.1.1 that the Brownian motion kernel is

$$k_{BM}(s, t) = \sigma^2 \min(s, t), \quad \sigma > 0. \tag{4.29}$$

The RKHS associated with this kernel is $\mathcal{H}_{BM}$, the Sobolev space of absolutely continuous functions x with $t \in [0, T]$, such that $x(0) = 0$, and whose derivatives, defined in the weak sense, are square integrable in that time interval (i.e. $x' \in L^2[0, T]$). The corresponding inner product between $x_1, x_2 \in \mathcal{H}_{BM}$ is

$$\langle x_1, x_2 \rangle_{BM} = \frac{1}{\sigma^2} \int_0^T x_1'(t) x_2'(t) dt. \tag{4.30}$$

The mean μ given by Equation (4.27) is in $\mathcal{H}_{BM}$ provided that $t_1 < t_2$. In such case, its squared norm is

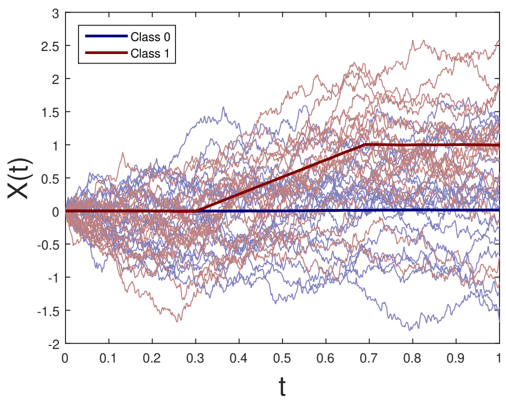
$$\langle \mu, \mu \rangle_{BM} = \frac{1}{\sigma^2} \int_0^T \left|\mu'(t)\right|^2 dt = \frac{\mu_T^2}{\sigma^2} \frac{1}{t_2 - t_1}. \tag{4.31}$$

Recall that since $\mathcal{H}_{BM}$ is an infinite-dimensional RKHS, the sample trajectories x do not belong to that space with probability one (Lukić and Beder, 2001). In the case of Brownian motion it is clear that the sample trajectories, which are continuous but non-differentiable, are not in $\mathcal{H}_{BM}$. In consequence, we need to understand the operation $\langle x, \mu \rangle_{BM}$, using Loève's isometry, as explained in Section 2.3.4. In this particular case, using Equation (4.30), the inner product can be written as

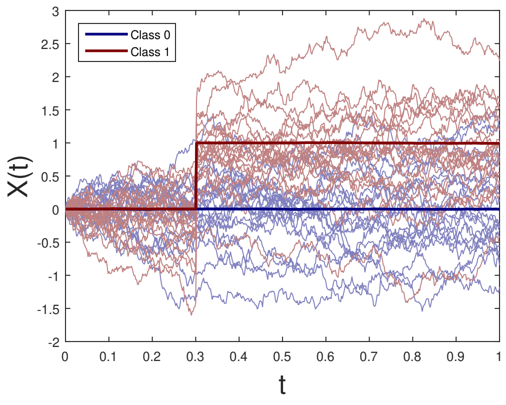
$$\langle x, \mu \rangle_{BM} = \frac{1}{\sigma^2} \int_0^T \mu'(t) x'(t) dt. \tag{4.32}$$

Note that this is only a formal expression because $x(t)$ is not differentiable. Nevertheless, identifying $x'(t)dt$ with $dx(t)$, one can interpret Equation (4.32) as

$$\begin{aligned} \langle x, \mu \rangle_{BM} &= \frac{1}{\sigma^2} \int_0^T \mu'(t) dx(t) = \frac{\mu_T}{\sigma^2} \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} dx(t) \\ &= \frac{\mu_T}{\sigma^2} \frac{x(t_2) - x(t_1)}{t_2 - t_1}, \end{aligned} \tag{4.33}$$



(A) Homoscedastic classification: zero-mean Brownian motion vs. Brownian motion with a piecewise linear mean.



(B) Homoscedastic near-perfect classification: zero-mean Brownian motion vs. Brownian motion with a step function mean.

FIGURE 4.1: Examples of homoscedastic classification with Brownian motion and different means.

where $\mu'(t)$ is given by Equation (4.28). Using Equations (4.18), (4.31), and (4.33), the Bayes classification rule becomes

$$g^*(x) = \mathbb{I}_{\left\{\frac{\mu_T}{\sigma^2}\left((x(t_2)-x(t_1))-\frac{\mu_T}{2}\right)>(t_2-t_1)\log\frac{1-p}{p}\right\}}.$$

A non-singular classification problem of this type is depicted in Figure 4.1a, where $T = 1$, $\mu_T = 1$, $t_1 = 0.3$, $t_2 = 0.7$, and $\sigma = 1$.

In the limit $t_2 \to t_1^+$, the mean exhibits a finite discontinuity at t_1 and therefore, is not in $\mathcal{H}_{BM}$. In this case, the decision rule is

$$g^*(x) = \mathbb{I}_{\left\{\left(x(t_1^+)-x(t_1)\right)>\frac{\mu_T}{2}\right\}}, \tag{4.34}$$

and near-perfect classification (zero asymptotic error) is obtained. As is apparent from Figure 4.1b, this rule has an obvious interpretation: one needs to compare the values of trajectory immediately before and after the jump of $\mu(t)$ at $t = t_1$. Class 0 trajectories should be continuous. Class 1 trajectories should exhibit the same discontinuity as the mean. This rule guarantees perfect classification provided that the values of the trajectories can be monitored with arbitrarily high resolution in t.

4.5 Heteroscedastic classification problems

In contrast with the homoscedastic case, which has received wide attention in the literature (Parzen, 1961a; Kailath, 1966; Kailath, 1971; Delaigle and Hall, 2012; Berrendero, Cuevas, and Torrecilla, 2018), most work on the heteroscedastic case is fairly recent and limited to specific examples (Delaigle and Hall, 2012; Delaigle and Hall, 2013; Dai, Müller, and Yao, 2017; Berrendero, Cuevas, and Torrecilla, 2018). In fact, no general rule has been proposed in the literature for this setting, even in the non-singular case. The difficulty lies in the interpretation of the singular terms that appear in the optimal rule (Equation (4.8)). As discussed in Section 4.3, when the divergences cancel out, we have a non-singular classification problem. By contrast, if the divergences do not cancel out, an optimal decision rule can be derived by balancing the singular terms. In this singular case near-perfect classification is obtained. We shall now proceed to study these cases separately and in detail.

4.5.1 Non-singular heteroscedastic classification

A heteroscedastic classification problem is non-singular if the Gaussian process laws $\mathbb{P}_0$ and $\mathbb{P}_1$ are equivalent. For this to be the case, $\mu \in \mathcal{H}_{k_0} \cap \mathcal{H}_{k_1}$, so that the terms that involve μ in Equation (4.8) must be well-defined. Furthermore, the singularities of the quadratic terms in the optimal rule need to cancel out: on the one hand, the limit

$$\lim_{M\to\infty} \mathbf{x}^T \left(\mathbf{K}_1^{-1} - \mathbf{K}_0^{-1}\right) \mathbf{x} \equiv \langle x, x\rangle_{k_1} - \langle x, x\rangle_{k_0} \tag{4.35}$$

should be finite when the set of monitoring points becomes dense in $\mathcal{T}$. On the other hand, the limit

$$\lim_{M\to\infty} \prod_{j=1}^{M} \frac{\lambda_{1j}}{\lambda_{0j}} \equiv \frac{|\mathcal{K}_1|}{|\mathcal{K}_0|} \tag{4.36}$$

should exist and be different from zero.

If these conditions are obtained, the divergences in the individual terms of Equation (4.8) cancel out and the resulting classification rule is well defined. Formally, it can be written as

$$g^*(x) = \mathbb{I}_{\left\{-\frac{1}{2}\log\frac{|\mathcal{K}_1|}{|\mathcal{K}_0|} - \frac{1}{2}\left(\langle x-\mu, x-\mu\rangle_{k_1} - \langle x,x\rangle_{k_0}\right) > \log\frac{1-p}{p}\right\}}. \tag{4.37}$$

In the following subsection, this non-singular limit will be illustrated using the standard Brownian motion and the standard Brownian bridge processes in the interval $[0, T]$, which are known to be equivalent when $0 \le T < 1$ (Varberg, 1961; Shepp, 1966). Therefore, for this range of values of T, the problem is heteroscedastic, but not singular. It becomes singular at $T = 1$.

Standard Brownian vs. Brownian bridge processes

Recall from Section 2.1.1 that the standard Brownian bridge in $[0, 1]$ is a zero-mean Gaussian process whose kernel is

$$k_{BB}(s,t) = \min(s,t) - st, \quad s,t \in [0,1]. \tag{4.38}$$

The corresponding RKHS is

$$\mathcal{H}_{BB} = \left\{ x : x(t) = \int_0^t x'(s)ds; x(1) = 0; x' \in L^2[0,1] \right\}.$$

This process, if considered in the interval $[0, T]$, with $T < 1$, has the inner product

$$\langle x_1, x_2 \rangle_{BB} = \int_0^T x_1'(t)x_2'(t)dt + \frac{x_1(T)x_2(T)}{1-T}. \tag{4.39}$$

In this interval, the Brownian bridge is equivalent to the standard Brownian motion, whose kernel is

$$k_{BM}(s,t) = \min(s,t), \tag{4.40}$$

and whose associated RKHS is

$$\mathcal{H}_{BM} = \left\{ x : x(t) = \int_0^t x'(s)ds; x' \in L^2[0,1] \right\}. \tag{4.41}$$

When restricted to the interval $[0, T]$, its inner product is

$$\langle x_1, x_2 \rangle_{BM} = \int_0^T x_1'(t)x_2'(t)dt. \tag{4.42}$$

Let x be a trajectory in the $[0, T]$ interval that is either a sample from a Brownian motion, with probability $1-p$ (class 0), or from a Brownian bridge process, with probability p (class 1). Trajectories from either of these processes are continuous but not differentiable. Therefore, they do not belong to the corresponding Hilbert spaces. In consequence, the individual inner products $\langle x, x \rangle_{k_i}$, for $i \in \{0,1\}$ are singular. However, the difference

$$\langle x, x \rangle_{BB} - \langle x, x \rangle_{BM} = \frac{(x(T))^2}{1-T}, \tag{4.43}$$

is well defined for $0 \leq T < 1$ because the singular terms in Equations (4.39) and (4.42), which involve the derivatives x_1' and x_2', are identical, and therefore cancel out.

To derive the expression for the ratio of determinants in Equation (4.37) we consider the discretely monitored process in the interval $[\frac{1}{M}, T]$ at regularly spaced times $\{t_m = m\Delta T\}_{m=1}^{T}$, with $\Delta T = \frac{T}{M}$ for some integer M, which will eventually be made to approach ∞. The point $t_0 = 0$ is excluded because both processes take the same deterministic value (i.e., $x(t=0) = 0$).

The Gram (autocovariance) matrix of such a discretely monitored standard Brownian motion is

$$(\mathbf{K}_{BM})_{mn} = \Delta T \min(m,n), \quad m,n = 1,\ldots,M. \tag{4.44}$$

The determinant of this matrix is

$$|\mathbf{K}_{BM}| = (\Delta T)^M. \tag{4.45}$$

The corresponding Gram matrix for the discretely monitored standard Brownian bridge is

$$(\mathbf{K}_{BB})_{mn} = \frac{T}{M}\left(\min(m,n) - mn\frac{T}{M}\right), \tag{4.46}$$

for $m, n = 1, \dots, M$. The determinant of this matrix is

$$|\mathbf{K}_{BB}| = (1 - T)\,(\Delta T)^{M}. \tag{4.47}$$

Thus, the ratio of the determinants of the covariance matrices for the discretely monitored standard Brownian motion and the standard Brownian bridge is

$$\frac{|\mathbf{K}_{BB}|}{|\mathbf{K}_{BM}|} = 1 - T, \quad \forall M > 0. \tag{4.48}$$

Therefore,

$$\frac{|\mathcal{K}_{BB}|}{|\mathcal{K}_{BM}|} = \lim_{M\to\infty} \frac{|\mathbf{K}_{BB}|}{|\mathbf{K}_{BM}|} = 1 - T. \tag{4.49}$$

Using this result in Equation (4.8), we get the optimal classification rule for this problem (see, e.g., Berrendero, Cuevas, and Torrecilla, 2018)

$$g^*(x) = \mathbb{I}_{\left\{-\frac{1}{2}\log(1-T) - \frac{1}{2}\frac{(x(T))^2}{1-T} > \log\frac{1-p}{p}\right\}}. \tag{4.50}$$

The error of this rule is

$$\begin{aligned} L^* =& (1-p)\left(1 - 2\,\mathrm{cdf}\left(\frac{-D}{\sqrt{T}}\right)\right) \\ &+ p\left(2\,\mathrm{cdf}\left(\frac{-D}{\sqrt{T(1-T)}}\right)\right), \end{aligned} \tag{4.51}$$

where cdf is the cumulative distribution function of a standard normal distribution, and

$$D = \sqrt{-2(1-T)\left(\log\left(\frac{1-p}{p}\right) + \frac{1}{2}\log(1-T)\right)}.$$

Note that in the limit $T \to 1^-$, the two terms on the left of the expression within the indicator function diverge, and dominate the classification rule. These divergences signal that near-perfect classification is obtained in this limit. Dropping the term that involves the priors, which is not singular, the optimal rule in the limit $T \to 1^-$ becomes

$$g^*(x) = \mathbb{I}_{\left\{(x(T))^2 < (1-T)\log\frac{1}{1-T}\right\}}. \tag{4.52}$$

Since $(x(T))^2 \geq 0$ and the term on the right hand side approaches zero, the optimal rule for $T = 1$ is

$$g^*(x) = \mathbb{I}_{\{x(1)=0\}}. \tag{4.53}$$

That is, one needs to inspect the value of $x(1)$. This quantity is 0 for Brownian bridge trajectories, and different from 0 with probability 1 for Brownian trajectories.

Finally, using the result given by Equation (4.15), the Radon-Nikodym derivative of the Brownian bridge measure with respect to the Brownian motion measure is

$$\frac{d\mathbb{P}_{BB}}{d\mathbb{P}_{BM}}(x) = \left(\frac{1}{1-T}\right)^{\frac{1}{2}} \exp\left\{-\frac{1}{2}\frac{(x(T))^2}{1-T}\right\}. \tag{4.54}$$

It is straightforward to also verify that the inner-products of the two processes are related by Equation (4.16) with $\delta k(s,t) = -st$

$$\begin{aligned}
-\langle x_1, \langle x_2, \delta k \rangle_{BB} \rangle_{BM} &= \int_0^T x_1'(t) \frac{d}{dt} \left[\int_0^T x_2'(s)\, t\, ds + \frac{x_2(T)\, Tt}{1-T} \right] dt \\
&= x_1(T) \left[x_2(T) + \frac{x_2(T) T}{1-T} \right] = \frac{x_1(T) x_2(T)}{1-T} \\
&= \langle x_1, x_2 \rangle_{BB} - \langle x_1, x_2 \rangle_{BM}.
\end{aligned}$$

4.5.2 Singular (near-perfect) heteroscedastic classification

In the heteroscedastic setting a first type of singular classification problem arises when $\mu = \mu_1 - \mu_0 \notin \mathcal{H}_{k_0} \cap \mathcal{H}_{k_1}$ (Delaigle and Hall, 2012). In this case, the analysis made in Section 4.4.2 remains valid and near-perfect classification is obtained. For this case, an optimal classification rule is (4.26) with $k = k_1$.

A second mechanism for near-perfect classification is obtained if the singularities in the terms $\log \frac{|\mathcal{K}_1|}{|\mathcal{K}_0|}$ and $(\langle x, x \rangle_{k_0} - \langle x, x \rangle_{k_1})$, do not separately cancel out. In this case, the measures induced by $GP(0, k_0)$ and $GP(\mu, k_1)$ are mutually singular. The decision rule Equation (4.8) is dominated by the divergent terms. One can therefore drop the terms that involve μ and the class priors, which are non-singular, and obtain the near-perfect classification rule

$$g^*(x) = \mathbb{I}_{\left\{ \frac{\langle x,x \rangle_{k_0} - \langle x,x \rangle_{k_1}}{\log |\mathcal{K}_1| - \log |\mathcal{K}_0|} > 1 \right\}}. \tag{4.55}$$

In this rule, the ratio of divergent terms needs to be understood as

$$g^*(\mathbf{x}) = \mathbb{I}_{\left\{ \lim_{M\to\infty} \frac{\mathbf{x}^T \left(\mathbf{K}_0^{-1} - \mathbf{K}_1^{-1} \right) \mathbf{x}}{\log |\mathbf{K}_1| - \log |\mathbf{K}_0|} > 1 \right\}}, \tag{4.56}$$

in the limit of dense monitoring.

In what follows, the validity of Equation (4.56) is illustrated in the classification of two Brownian motions with equal mean and different variances, which are known to be mutually singular.

Classification of Brownian motions with different variances

Consider the heteroscedastic functional classification problem

$$X(t) = \begin{cases} Z_0(t) & \text{w. p.} \quad 1-p, \\ Z_1(t) & \text{w. p.} \quad p, \end{cases} \tag{4.57}$$

for $t \in [0, T]$, where $Z_0(t)$ and $Z_1(t)$ are zero-mean Brownian motions of variances σ_0^2 and σ_1^2, respectively. Since all trajectories start at the same level $X(0) = 0$, they need to be monitored only at times $\{t_m = m\Delta T\}_{m=1}^M$ with $\Delta T = T/M$. The autocovariance matrices of the discretely monitored processes are

$$(\mathbf{K}_i)_{mn} = \sigma_i^2 \Delta T \min(m, n), \quad m, n = 1, \dots, M, \tag{4.58}$$

for $i = 0, 1$. The determinant of this matrix is

$$|\mathbf{K}_i| = \left(\sigma_i^2 \Delta T \right)^M. \tag{4.59}$$

The corresponding inverses are symmetric tridiagonal matrices

$$\mathbf{K}_i^{-1} = \frac{1}{\sigma_i^2 \Delta T} \begin{pmatrix} 2 & -1 & 0 & \dots & 0 & 0 \\ -1 & 2 & -1 & \dots & 0 & 0 \\ 0 & -1 & 2 & \dots & 0 & 0 \\ & & & \dots & & \\ 0 & 0 & 0 & \dots & 2 & -1 \\ 0 & 0 & 0 & \dots & -1 & 1 \end{pmatrix}. \tag{4.60}$$

The term in the denominator of Equation (4.56) is

$$\log |\mathbf{K}_1| - \log |\mathbf{K}_0| = \sum_{j=1}^{M} \log \frac{\sigma_1^2}{\sigma_0^2} = \frac{T}{\Delta T} \log \frac{\sigma_1^2}{\sigma_0^2}, \tag{4.61}$$

where we have used that $M = \frac{T}{\Delta T}$. Similarly,

$$\mathbf{x}^T \mathbf{K}_i^{-1} \mathbf{x} = \frac{T}{\Delta T} \frac{\sigma_x^2(M)}{\sigma_i^2}, \tag{4.62}$$

where

$$\sigma_x^2(M) = \frac{1}{M} \sum_{m=1}^{M} \frac{(x(t_m) - x(t_{m-1}))^2}{\Delta T}. \tag{4.63}$$

For this problem, if the non-singular terms are dropped, Equation (4.7) becomes

$$g^*(x) = \mathbb{I}_{\{\sigma_x^2(M) > \theta\}}, \tag{4.64}$$

with $\theta = \left(\frac{1}{\sigma_0^2} - \frac{1}{\sigma_1^2}\right)^{-1} \log \frac{\sigma_1^2}{\sigma_0^2}$. In the limit $M \to \infty$ (therefore, $\Delta T \to 0^+$), the optimal classification rule is

$$g^*(x) = \mathbb{I}_{\{\sigma_x^2 > \theta\}}, \tag{4.65}$$

where $\sigma_x^2 = \lim_{M\to\infty} \sigma_x^2(M)$. This near-perfect classification rule can be written also in terms of Kullback-Leibler divergences between normal distributions with the same mean and different variances

$$g^*(x) = \mathbb{I}_{\left\{KL\left(N(0,\sigma_x^2), N(0,\sigma_1^2)\right) < KL\left(N(0,\sigma_x^2), N(0,\sigma_0^2)\right)\right\}}. \tag{4.66}$$

As in the homoscedastic case, this rule guarantees perfect classification only if the values of the trajectories can be measured with arbitrarily high resolution in time.

We will now analyze the convergence of the singular decision rule for the discretely monitored Brownian motions (Equation (4.64)) to its asymptotic limit (Equation (4.65)). Without loss of generality, we will assume $\sigma_1^2 > \sigma_0^2$. The accuracy of the prediction rule given by Equation (4.64) as a function of M, the number of monitoring points, is

$$\begin{aligned} Acc(M) &= pP\left(\sigma_x^2(M) > \theta \mid Y = 1\right) \\ &\quad + (1-p)\,P\left(\sigma_x^2(M) < \theta \mid Y = 0\right). \end{aligned} \tag{4.67}$$

Since $(X(t_m) - X(t_{m-1})) \sim N\left(0, \sigma_X^2 \Delta T\right)$, then $M\frac{\sigma_X^2}{\sigma_i^2}$ follows a chi-square distribution with M degrees of freedom. In consequence,

$$
\begin{aligned}
Acc(M) = p\left[1 - \mathrm{cdf}_{\chi^2}\left(M\frac{\theta}{\sigma_1^2}, M\right)\right] \\
+ (1-p)\,\mathrm{cdf}_{\chi^2}\left(M\frac{\theta}{\sigma_0^2}, M\right),
\end{aligned}
\tag{4.68}
$$

where $\mathrm{cdf}_{\chi^2}(z, \nu)$ is the cumulative distribution function of a χ^2 distribution with ν degrees of freedom evaluated at z. In the asymptotic limit of a densely monitored process $\lim_{M\to\infty} Acc(M) = 1$, which means that near-perfect classification is obtained.

To illustrate this convergence, we have performed a set of experiments with simulated trajectories of zero-mean Brownian motions of variances $\sigma_0^2 = 1$ and $\sigma_1^2 \in \{1.05, 1.5, 5\}$ in the time interval $[0, 1]$, starting from 0 at $t = 0$. Each experiment consists in generating $N = 50$ trajectories from each of these classes. The trajectories are sampled at M regularly spaced times, including the origin, with $M = 2^b + 1$, $0 \leq b \leq 10$. Then, the decision rule given by Equation (4.64) is used to classify the $2N$ trajectories generated. The whole process was repeated 1000 times so that, for each M, the expected accuracy and its standard deviation of the accuracy on the sample of $2N = 100$ trajectories can be computed. In a sample of $2N$ trajectories the number of correctly classified cases follows a binomial distribution with a success probability equal to $Acc(M)$. Since the variance of the number of successes in the binomial distribution is given by $2NAcc(M)\,(1 - Acc(M))$, the standard deviation of the observed accuracy of the decision rule given by Equation (4.64) in a sample of size $2N$ is $\sqrt{\frac{Acc(M)\cdot(1-Acc(M))}{2N}}$ with $Acc(M)$ given by Equation (4.68).

Figure 4.2 displays the dependence of the accuracy of the optimal decision rule as a function of M, for the different values of the ratio σ_1^2/σ_0^2 considered. From the analysis of these plots one concludes that sample estimates are in good agreement with the theoretical values of the expected accuracy and their standard deviations. Furthermore, it is apparent that the larger the differences between σ_1^2 and σ_0^2 are, the faster the approach to the asymptotic regime in which near-perfect classification is obtained.

4.6 Empirical evaluation

In this section we compare the performance of the limit rules derived in this work with functional classifiers proposed in the literature. As test-bed for comparison we use simulated data and a real-world problem from quantitative finance. To make it possible to reproduce the results, the code for the experiments is available at `https://github.com/GAA-UAM/GP-Bayes-Rules-Experiments`. The simulated data correspond to the classification problems considered in Subsections 4.4.2 (homoscedastic and singular), 4.5.1 (heteroscedastic and equivalent), and 4.5.2 (heteroscedastic and singular). Assuming that the classes are balanced ($p = 1 - p = 1/2$), we generate N_{train} trajectories in the interval $[0, T]$ and their corresponding class labels, $\{(x_i, y_i)\}_{i=1}^{N_{train}}$, according to Equation (4.1). To investigate how the accuracies of the different methods depend on the amount of data available for induction, experiments with different training set sizes ($N_{train} \in \{50, 200, 1000\}$) have been carried

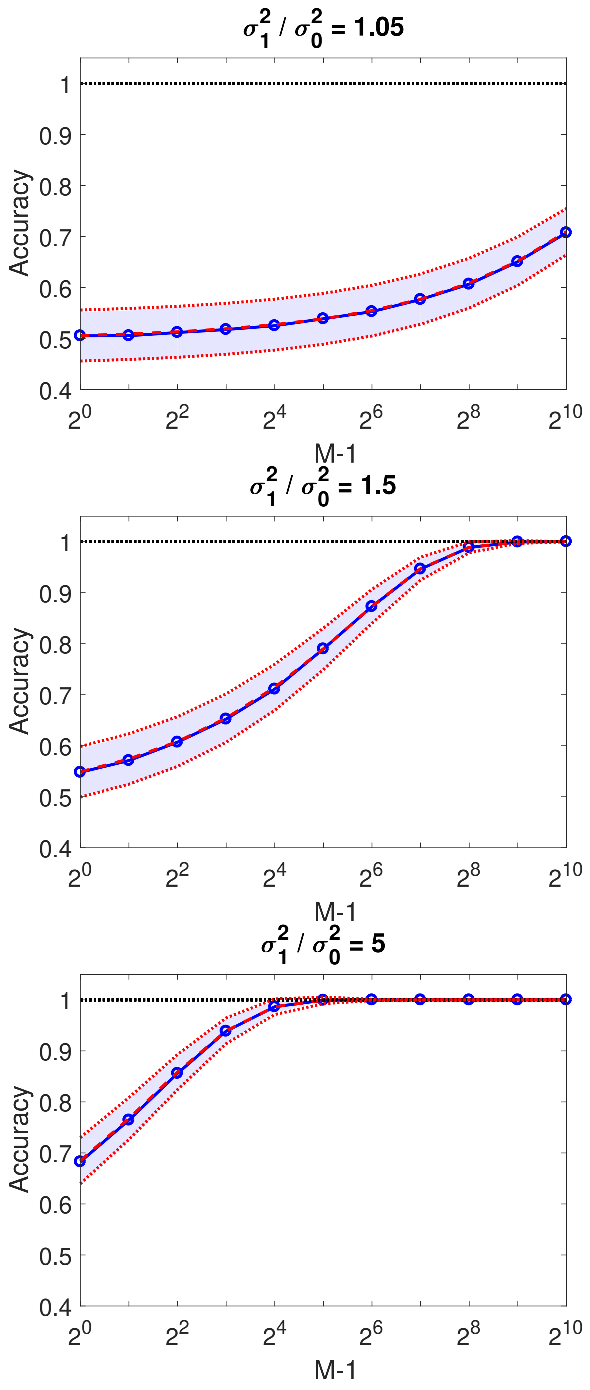


FIGURE 4.2: Classification accuracy for the discrimination of Brownian motions with equal means and different variances (Equation (4.57)). The solid lines trace the dependence of the accuracy, averaged over 1000 replications of the problem, as a function of $M-1$, the number of monitoring intervals. The shaded band corresponds to one standard deviation above and below this average. The dashed lines correspond to the theoretical expected accuracy. Finally, the red dotted lines correspond to the expected accuracy plus/minus one standard deviation.

out. The trajectories $\{x(t),\, t \in [0,T]\}$ are monitored discretely on a regular grid

$$x(t_m), \quad t_m = \Delta T_b, \quad m = 0, 1, \ldots, N_b, \tag{4.69}$$

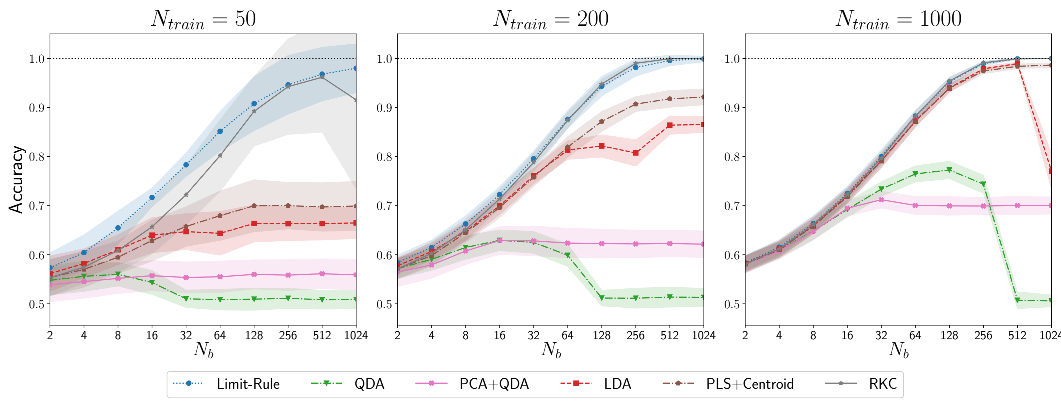


FIGURE 4.3: Classification accuracy for the discrimination of Brownian motions with equal variance and different means (zero mean, and step function mean). The solid lines trace the dependence of the accuracy, averaged over 100 replications of the problem, as a function of N_b, the number of monitoring intervals, for different values of N_{train}, the size of the training data. The shaded bands correspond to one standard deviation above and below the corresponding averages.

where $\Delta T_b = \frac{T}{N_b}$. The dependence on the size of the monitoring grid is analyzed by considering different numbers of discretization intervals; namely $N_b = 2^b$, $b = 1, \ldots, 10$. Unbiased estimates of the generalization accuracy are made in test sets of size $N_{test} = 1000$, which are generated independently of the training data. The values reported are averages over 100 independent replications, with the corresponding standard deviations.

The financial classification problem consists in the discrimination between the stocks of different car manufacturing companies (*BMW*, *GM*, and *Tesla*) on the basis of the time series of their market prices. According to expert knowledge, in this real-world example, the log-differences of the asset prices are expected to approximately follow a Brownian motion (Osborne, 1959; Fama, 1965). The standard deviations of these processes, or, in financial terminology, their volatilities, should be different. Therefore, an appropriate model for their classification is given by Equation (4.57); i.e., two Brownian motions with different variances. In this case, besides the comparison of methods, the experiments serve also as an empirical validation of this Brownian hypothesis.

The classifiers that are compared in this section are the following:

- Linear discriminant analysis (LDA): The standard multivariate linear discriminant applied to the discretely monitored trajectories.
- Quadratic discriminant analysis (QDA): The standard multivariate quadratic discriminant applied to the discretely monitored trajectories.
- Partial least squares (PLS)+Centroid: This classifier consists in applying a centroid rule to the output of a partial least squares regression model. It is one of the most accurate methods among those considered in the seminal paper by Delaigle and Hall, 2012.
- Principal components analysis (PCA)+QDA: This classifier is based on a proposal by Galeano, Joseph, and Lillo, 2015 to compute the functional analogue of the Mahalanobis distance. In that paper, the authors argue that this method

is equivalent to applying a quadratic discriminant to the first few principal components of the trajectories to be classified.

- Reproducing kernel classification (RKC): The Reproducing Kernel classification rule is based on first performing variable selection by reproducing kernel-based variable selection (RKVS), a criterion that involves the Mahalanobis distance (see Section 2.3.2 for details), and then applying a linear discriminant analysis (Berrendero, Cuevas, and Torrecilla, 2018). The name reflects the fact that it has a natural interpretation in the corresponding RKHS. This rule has been proven to be optimal if the functional classification problem is homoscedastic and the probability measures are equivalent.

- Limit-Rule: Classification rule derived from the analysis of the quadratic discriminant for the discretized process (Equation (4.7)) in the limit of dense monitoring.

The different methods have been implemented in Python. LDA, QDA, PCA and PLS regression make extensive use of objects and functions in the *scikit-learn* package (Pedregosa et al., 2011). The RKC method has been freshly implemented following Berrendero, Cuevas, and Torrecilla, 2018. Functional data objects have been manipulated with the tools provided by the *scikit-fda* package (Ramos-Carreño et al., 2023). The number of components of the dimensionality reduction methods (PCA, PLS and RKC) is determined by 10-fold cross-validation in the range 1 to 20.

We now proceed to present a summary of the results of this empirical evaluation for the different cases analyzed in this work.

4.6.1 Brownian motions with different means

We first study a homoscedastic problem of the form given by Equation (2.78), in which $Z_0(t) = Z_1(t) = Z(t)$ is a standard Brownian Motion process in $[0,1]$ and $\mu(t)$ a step function

$$\mu(t) = \begin{cases} 0, & 0 \leq t \leq t_* \\ \mu_T, & t_* < t \leq 1 \end{cases} \tag{4.70}$$

with μ_T a constant level. This corresponds to the problem analyzed in Subsection 4.4.2, with $t_2 \to t_1^+$, $t_1 = t_*$ in Equation (4.27), and $T = 1$. In such limit, the probability measures of the two Gaussian processes are mutually singular and near-perfect classification is obtained. In the experiments carried out, the step is located at $t_* = 0.5$ and has a height of $\mu_T = 0.3$.

The limit-rule for this problem is given by Equation (4.34). It depends only on the location and the size of the discontinuity. For this rule, t_*, the time instant at which the discontinuity occurs, is estimated from the training data. Specifically, a two sided t-test is used to determine whether the difference of the sample means in class 1 trajectories between consecutive monitoring points are significantly different from zero. The discontinous jump is assumed to be within the interval $\left[\hat{t}_*, \hat{t}_* + \Delta T_b\right]$, where $\left\{\hat{t}_*, \hat{t}_* + \Delta T_b\right\}$ is the pair of consecutive points for which this test yields the lowest p-value. The height of the step μ_T is estimated as the empirical mean of the values of the class 1 trajectories right after the step

$$\hat{\mu}_T = \frac{1}{N_{train}^{[1]}} \sum_{i=1}^{N_{train}} x_i(\hat{t}_* + \Delta T_b) \mathbb{I}_{\{y_i=1\}}, \tag{4.71}$$

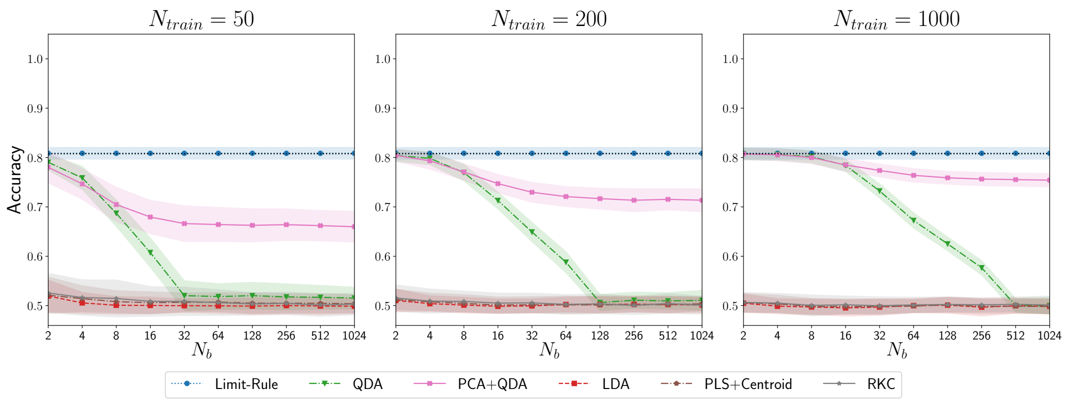


FIGURE 4.4: Classification accuracy for the discrimination of standard Brownian and Brownian bridge processes in $[0, T]$ with $T = 0.95$. The solid lines trace the dependence of the accuracy, averaged over 100 replications of the problem, as a function of N_b, the number of monitoring intervals, for different values of N_{train}, the size of the training data. The shaded bands correspond to one standard deviation above and below the corresponding averages.

where $N_{train}^{[1]} = \sum_{i=1}^{N_{train}} \mathbb{I}_{\{y_i=1\}}$ is the number of class 1 trajectories in the training set.

The curves plotted in Figure 4.3 display the dependence of the average accuracies of the different classifiers as a function of the number of discretization intervals for different training set sizes. The shaded bands correspond to deviations of one standard deviation above and below the average accuracies. The black horizontal dotted line marks the optimal accuracy, which in this case is 1.0. From the results obtained we observe that the limit rule approaches this value asymptotically, for sufficiently dense monitoring. The RKC method performs remarkably well in this problem and also approaches perfect accuracy for large training samples and dense monitoring. For Brownian motions, the RKC method can be shown to be optimal for a difference of means that is a continuous piecewise linear function starting at 0 (Berrendero, Cuevas, and Torrecilla, 2018). The problem considered in our simulation is not of this form, because the step function exhibits a discontinuity, albeit finite. Nevertheless, as discussed in Subsection 4.4.2, the discontinuous jump can be obtained as the limit of a sequence of such continuous functions. Therefore, it is reasonable that RKC performs as well as the limit rule, which is optimal.

The differences between the accuracies of these two methods and the remaining ones, which are small for coarse monitoring, become larger as N_b increases. The superior performance of RKC and the limit rule also at small training sizes resides in the fact that they require estimating fewer parameters. In this problem, all the information needed for discrimination is in the difference of means between the two Brownian motions. For this reason, the classifiers that require the estimation of the covariance matrix, especially QDA, and PCA+QDA, which furthermore do not assume homoscedasticity, obtain very poor results. This is consistent with previous observations in the literature on the limited accuracy of quadratic discriminant functions when the dimensions are large and the sample sizes for the estimation of the covariances are small (Marks and Dunn, 1974; Wahl and Kronmal, 1977; Berrendero and Cárcamo, 2019). For larger values of N_b, PCA+QDA, which involves a dimensionality reduction step before the quadratic determinant function is computed, becomes more accurate than standard QDA. This behavior is consistently observed for all the classification problems analyzed.

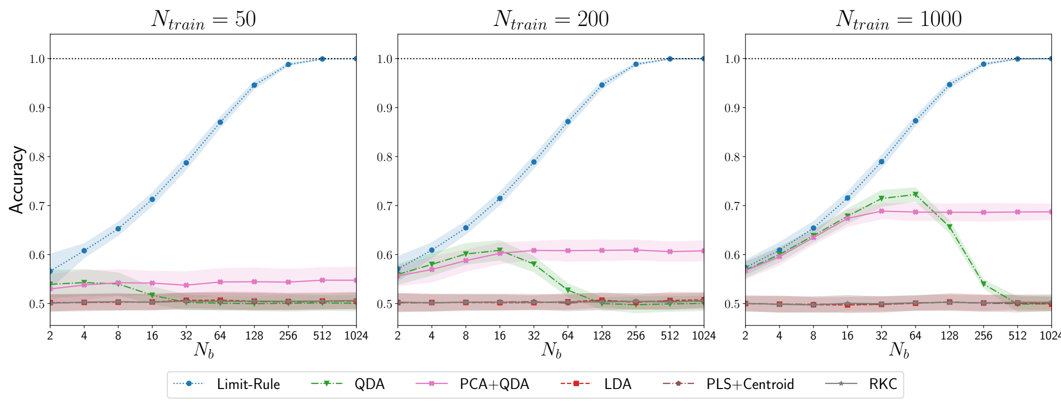


FIGURE 4.5: Classification accuracy for the discrimination of Brownian motions of different variances. The solid lines trace the dependence of the accuracy, averaged over 100 replications of the problem, as a function of N_b, the number of monitoring intervals, for different values of N_{train}, the size of the training data. The shaded bands correspond to one standard deviation above and below the corresponding averages.

The accuracies of PLS+Centroid and LDA are also low when the training sets are small. Both are global methods, which are not well adapted to problems in which the discriminant information is concentrated at a single point. Nonetheless, their accuracy markedly improves (especially that of PLS+Centroid) as the size of the training data becomes larger.

Finally, note that even though QDA and LDA are optimal for the multivariate version of this problem, the collinearity inherent to functional data has a marked negative impact in their predictive performance for finite training sample sizes. The reason is that these methods require the inversion of the empirical covariance matrix. This inversion is numerically unstable when the number of variables (monitoring times) increases for a fixed size of the training sample. By contrast, the accuracies of PLS+Centroid, PCA+QDA, and RKC (which makes use of LDA) do not deteriorate with increasing N_b, because they involve a dimensionality reduction in a previous step.

4.6.2 Brownian motion vs. Brownian bridge

The problem addressed in this second batch of experiments is the discrimination of trajectories sampled from a standard Brownian motion and from a Brownian bridge process in the interval $[0, T]$. As described in Section 4.5.1, these processes are equivalent for $T < 1$. In the experiments carried out the value selected is $T = 0.95$. Therefore, this is a standard classification problem with a non-zero Bayes error. Specifically, for the current simulation Equation (4.51) yields $L^* = 0.193$. The limit rule is Equation (4.50). It depends only on the class priors and on the value of the trajectory to be classified at T.

In Figure 4.4 we present the comparison between the classifiers described in the introduction of the current section. As expected, the average accuracy obtained with the limit rule classifier is close to the optimal value of $1 - L^* = 0.807$ for all values of N_{train} and N_b. In this case, since both processes have the same mean, the information that is useful for discrimination is in the covariance structure. Therefore linear classifiers, such as LDA, PLS+Centroid, and RKC are unable to predict better than random guessing. Both QDA and PCA+QDA obtain good results when the number of monitoring intervals is small and the size of the training data is large. Their predictive performance deteriorates as N_b becomes larger. As in the previous case, the reason can be traced to the estimation of the covariance matrix from the sample, which becomes unreliable at higher dimensions. PCA+QDA is more robust than QDA because of the dimensionality reduction step.

4.6.3 Brownian motions with different variances

We now address the classification of trajectories sampled from two zero-mean Brownian motions of different variances. The problem, which has been analyzed in detail in Section 4.5.2, is singular and exhibits near-perfect classification. The limit rule is given by Equation (4.65). It requires the estimation of σ_0^2, σ_1^2 and σ_x^2 from the training data. The variance of each trajectory is estimated using Equation (4.63). The variances σ_0^2 and σ_1^2 are estimated as the averages of the variances in the class 0 and class 1 trajectories, respectively. In the experiments performed the class 0 trajectories are realizations of a standard Brownian motion with $\sigma_0^2 = 1$. The class 1 trajectories are sampled from a Brownian motion with $\sigma_1^2 = 1.5$.

The overall comparison of the different classifiers considered in this study for this problem is presented in Figure 4.5. As in the previous set of experiments, since the two processes have the same mean, this is a purely heteroscedastic classification problem. In consequence, the linear methods, such as LDA, PLS+Centroid, and RKC, which are based solely on the differences between means, are equivalent to random guessing.

The predictions of QDA and PCA+QDA are better than random and improve with the size of the training data. Nonetheless, both methods are suboptimal. In particular, the accuracy of QDA severely deteriorates with the number of monitoring points, because of the increased dimension of the problem and the high collinearity of the functional data.

In this singular case, the limit rule approaches perfect accuracy when the number of monitoring intervals is sufficiently large even for small training samples. The reason is that the estimation given by Equation (4.63) approaches the exact value of the variance in the limit $M \to \infty$ for a single trajectory. Therefore, the classification rule achieves perfect accuracy asymptotically, in the limit of dense monitoring, independently of the size of the training data.

4.6.4 Near-perfect classification of financial time series

We now provide an illustration of near-perfect classification with real-world data. The goal is to discriminate between time series of market prices of financial assets. In our experiments, the daily closing prices of General Motors (GM) from the New York Stock Exchange (NYSE), Tesla from NASDAQ, and BMW from Xetra, between January 1, 2014 and January 31, 2018, are used. The data have been retrieved via the *Google Finance* API (https://finance.google.com). Days in which not all three asset price quotations were available have been discarded. A more sophisticated treatment of missing values (e.g., linear or Brownian bridge interpolation) does not lead to significant changes of the results.

For each asset, the sample consists of $N = 30$ time series of market prices during non-overlapping periods of $N_B = 2^5 = 32$ days. These series are displayed in the top plots of Figure 4.6. They are computed as follows: let $\{S_i(t_0), S_i(t_1), \ldots, S_i(t_L)\}$ be

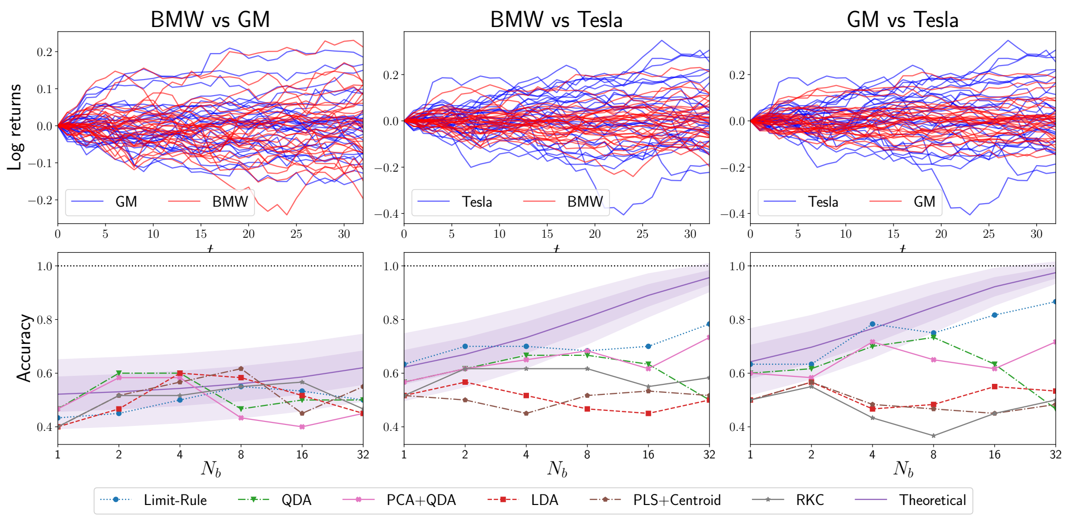


FIGURE 4.6: Discrimination of financial assets on the basis of the time series of their market prices (top plots). The curves that trace the dependence of the accuracies of different classifiers as a function of N_b, the number of monitoring intervals, are displayed in the bottom plots.

the complete time series of asset market prices for stock i monitored at the equally-spaced instants

$$t_m = t_0 + m\Delta T;\ m = 0, 1, \dots, L,$$

where $L = N(N_B + 1) - 1$. In the data analyzed ΔT is one day. Therefore, the quantity $S_i(t_m)$ is the closing price of the corresponding stock on the m-th day of the period considered.

The time series is broken up into N segments of length $N_B + 1$, with $N_B = 2^B$ for some integer B

$$\left\{ S_i(t_0^{[j]}), S_i(t_1^{[j]}), \dots, S_i(t_{N_B}^{[j]}) \right\}_{j=1}^{N},$$

where $t_m^{[j]} = t_{m+(j-1)N_B}$, with $m = 0, 1, \dots, N_B$, and $j = 1, 2, \dots, N$. These N time series of $N_B + 1$ prices are then transformed into the corresponding time series of log-returns

$$\left\{ X_i(t_0^{[j]}), X_i(t_1^{[j]}), \dots, X_i(t_{N_B}^{[j]}) \right\}_{j=1}^{N}, \tag{4.72}$$

where

$$X_i(t_m^{[j]}) = \log \frac{S_i(t_m^{[j]})}{S_i(t_0^{[j]})}, \quad m = 0, 1, \dots N_B.$$

The goal is to discriminate between different stocks on the basis of the corresponding time series of log-returns. In particular, we will analyze how the accuracy of the predictions depends on the monitoring frequency. For this reason, discrimination is made on the basis of $N_b + 1$ subsampled values within each segment

$$\left\{ X_i(t_0^{[j]}), X_i(t_{n_b}^{[j]}), X_i(t_{2n_b}^{[j]}), \dots, X_i(t_{N_b n_b}^{[j]}) \right\},$$

where $N_b = 2^b$, and $n_b = 2^{B-b}$ with $b = 0, 1, \dots, B$. As an illustration, for $b = 0$, only two inputs in each time series are used for discrimination

$$\left\{ X_i(t_0^{[j]}), X_i(t_{N_B}^{[j]}) \right\}.$$

For $b = B$ ($n_B = 1$) the complete time series given by Equation (4.72) is used as input to the different classifiers. The higher monitoring the frequency is, the closer the problem is to a functional paradigm.

There is ample empirical evidence that the time series of stock prices approximately follow a geometric Brownian motion (Osborne, 1959; Fama, 1965). Consequently, their log-differences (i.e. the log-returns) follow an arithmetic Brownian motion. According to standard financial wisdom, financial assets are characterized mainly by their *volatility*, which is the financial term used for the standard deviation of these log-returns. By contrast, the expected returns (i.e., the drift of the Brownian motion) are much less reliable for discrimination. Therefore, Equation (4.57), which corresponds to Brownian motions with different standard deviations (volatilities), should provide a suitable model for the classification problem. We can test the validity of these observations by comparing the accuracies of the classification methods described in the introduction to the current section and the limit rule given by Equation (4.64). In this limit rule, the information on the means (expected returns) is discarded. Classification is made solely in terms of the sample estimates of the asset volatilities.

The results of the empirical evaluation are summarized in the bottom plots of Figure 4.6. In each of the columns in this figure a different binary classification problem is considered. From left to right: BMW vs. GM, BMW vs. Tesla, and GM vs. Tesla. The inputs for classification are the discretely monitored trajectories of asset log-returns, computed as described before. The accuracy of different classifiers is estimated using 10-fold stratified cross validation. The plots display the curves that trace the dependence of the accuracy of the different classifiers as a function of $N_b \in \{1, 2, 4, 8, 16, 32\}$, the number of monitoring intervals. The value $N_b = 32$ corresponds to daily intervals, which is the highest monitoring resolution that can be employed with the available data. For reference, we provide the theoretical accuracy curves for the corresponding Brownian motions with the same mean and volatility as each of the financial asset returns. Uncertainty intervals of one and two standard deviations above and below the theoretical accuracy curves are given as shaded bands.

In the first classification problem considered, *BMW vs. GM*, all classifiers perform poorly, close to random guessing. The reason is that these two assets have similar volatilities ($\sigma^2_{BMW} = 2.545 \cdot 10^{-4}$ and $\sigma^2_{GM} = 2.183 \cdot 10^{-4}$, respectively) and, in consequence, are difficult to distinguish. This should be expected because both companies are car manufactures that have comparable profiles and are exposed to the same risk factors. Therefore, the prices of their stock should exhibit similar characteristics. By contrast, Tesla is a highly specialized manufacturer of electric cars, whose main asset is technological innovation. Correspondingly, it exhibits higher volatility than the other two ($\sigma^2_{Tesla} = 6.147 \cdot 10^{-4}$). The characteristics of the *BMW vs. Tesla* and the *GM vs. Tesla* classification tasks are similar. In these two problems, the limit rule given by Equation (4.64) has the best overall results. By contrast, the methods that rely on the difference of means (LDA, PLS+Centroid, and RKC) for discrimination have poor accuracies, at the level of random guessing. This means that the sample means (expected log-returns) are not useful to discriminate between these assets. The quadratic discriminant with a covariance matrix estimated from

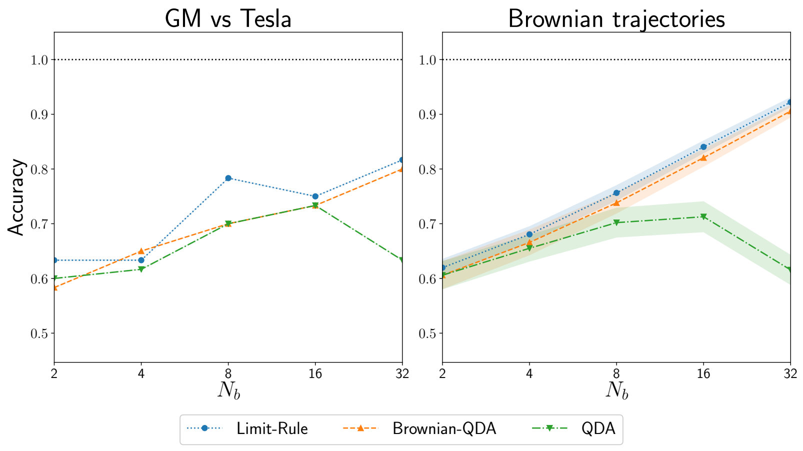


FIGURE 4.7: Comparison of QDA, Brownian QDA and the limit rule in the *GM vs. Tesla* classification problem (left plot). The right plot corresponds to the same comparison for simulated data of the same characteristics as the real-world problem.

the sample (QDA) has slightly better accuracy than random guessing for intermediate values of N_b. However, for larger values of N_b the results deteriorate. As in the synthetic data examples, this is a consequence of the poor quality of the sample estimates of the covariance matrices in higher dimensions and the high collinearity of functional data (Marks and Dunn, 1974; Wahl and Kronmal, 1977; Berrendero and Cárcamo, 2019). The PCA+QDA method does not exhibit this degradation thanks to the fact that the dimension of the problem is reduced by selecting a few principal components before the quadratic discriminant rule is applied.

One way to avoid this limitation of the quadratic discriminant rule is to use expert knowledge and assume that the covariance matrix has a Brownian structure. Taking advantage of this structure, the elements of the covariance matrix need not be estimated separately. They can be computed in terms of the volatilities of the assets, which are the only parameters that are actually estimated from the training sample. To illustrate this point, the accuracy of this method (Brownian-QDA) is compared with the standard QDA, in which the individual elements of the covariance matrix are estimated separately, and the limit rule given by Equation (4.64) in the GM vs. Tesla classification problem. The results of this comparison are displayed in the plot on the left-hand side of Figure 4.7. By contrast with the behavior of the standard QDA, the accuracy of Brownian-QDA improves with the monitoring frequency. Nevertheless, comparable or better accuracies are achieved if we use the limit rule, in which only the singular terms in the quadratic discriminant are retained.

The time series of log-returns analyzed do not necessarily follow a Brownian motion. Therefore, one may wonder whether the conclusions obtained with the real-world data are reliable. To clarify this point, we carried out simulations of the classification problem using trajectories from two Brownian motions with the same volatilities as the GM and Tesla assets. The results of these experiments are presented in the plot on the right-hand side of Figure 4.7. To obtain these results the different classifiers (QDA, Brownian QDA, and the limit rule) are trained under the same conditions as in the experiments with the real-world data. Their accuracies are then computed on a test set of size 1000. The values reported are averages over 100 replications of the classification problem. Uncertainty intervals of one standard deviation above and below the averages are plotted as shaded bands. From these results one concludes that the behavior observed in the experiments with real-world data is not spurious: The predictive accuracy of the standard quadratic discriminant rule eventually deteriorates as N_b increases. By contrast, the accuracies of Brownian-QDA and the limit rule given by Equation (4.64) improve with denser monitoring. Note, however, that even for the largest values of N_b considered, these classifiers do not achieve perfect classification. There are several reasons for this shortfall: First, the number of trajectories available for induction is very small (30 instances per class). Unfortunately, it is not possible to use much longer periods for which the hypothesis of constant volatility holds even in an approximate manner. Second, the daily monitoring is insufficiently dense. However, higher frequency data cannot be used because the intra-day series of prices exhibits discontinuities and large deviations from the log-normal model. Finally, systematic deviations from the Brownian model are observed in the data. In the period considered, the Brownian assumption holds only in an approximate manner. Empirically, one observes that the time series exhibit heteroscedasticity in time and the log-returns are leptokurtic. Therefore, a more accurate model should account for the stochastic dynamics of the volatility (Bollerslev, Chou, and Kroner, 1992) and the heavy-tailedness of the log-returns (Cont, 2001). In spite of these limitations, when the volatilities are sufficiently different, the limit rule given by Equation (4.64), performs quite well in practice.

4.7 Conclusions

In this chapter we have addressed the problem of learning by induction from data that are characterized by functions of a continuous parameter. In particular, we have derived optimal classification rules for binary classification problems in which the instances are trajectories sampled from different Gaussian Processes, depending on the class label. The problem has been addressed earlier in the literature in both the homo- and heteroscedastic settings (see, e.g., Delaigle and Hall, 2012; Delaigle and Hall, 2013; Dai, Müller, and Yao, 2017; Berrendero, Cuevas, and Torrecilla, 2018). However, the procedure proposed in this work, which is based on the asymptotic analysis of the optimal rules for the discretely monitored trajectories in the limit of dense monitoring, is new. Furthermore, this procedure has been used to gain insight into the emergence of near-perfect classification, which was first analyzed in Delaigle and Hall, 2012 for differences in means. The current research expands on that work by analyzing cases in which near-perfect classification arises from the covariance (quadratic) components as well. Specifically, a detailed analysis of the dense monitoring limit reveals that some of the terms that appear in such rules diverge. If the Gaussian processes are equivalent these divergences cancel out and non-singular optimal classification rules are obtained. By contrast, if the Gaussian processes are orthogonal the divergences do not cancel out. As a matter of fact, the singular terms dominate and near-perfect classification is obtained. In this latter context, optimal rules that achieve zero prediction error asymptotically (i.e. for sufficiently large sample sizes) have been derived by considering only the terms that diverge in the limit of dense monitoring.

To illustrate the validity of the analysis, explicit rules are given for some classification problems involving Brownian and Brownian bridge processes. In the cases that such optimal rules were known, the limit of dense monitoring provides a novel procedure for their derivation. We also provide explicit rules for cases in which near-perfect classification is obtained. The accuracy of such limit rules has been evaluated in extensive simulations and in the classification of time series of financial asset prices, which are modeled as geometric Brownian motion.

Even though the asymptotic analysis of the classification rules for the discretely monitored trajectories in the limit of dense monitoring has been introduced in the context of Gaussian processes, the procedure may be applicable to more general stochastic processes, which are not necessarily Gaussian. This is a promising line of research that will be addressed in future work.

Chapter 5

Recursive maxima hunting: Variable selection for functional data classification

Recursive maxima hunting (RMH) is a filter variable selection method for supervised learning with functional data (Torrecilla and Suárez, 2016). RMH is based on the same ideas as maxima hunting (MH) (Berrendero, Cuevas, and Torrecilla, 2016b). In both methods the dependence of the class label on the values of the functional observations at a specific point is utilized to guide the selection process. The functional nature of the data is exploited by excluding points in the neighborhood of the selected ones. Because of continuity, these points generally provide redundant information. The main difference between MH and RMH is the manner in which the impact points are identified. In MH, this process is carried out in parallel: the points selected correspond to the local maxima of a suitable defined relevance function. By contrast, RMH is a sequential process whose first step consists in identifying the *global* maximum of the relevance function. This is a much simpler problem than finding a local one. Before selecting additional variables, the dependencies of the class label on the previously selected ones are removed by correcting the trajectories that characterize the observations. By removing such dependencies, previously hidden dependencies can be uncovered. The subsequent point selected is the global maximum of the relevance function for the corrected process. The process is repeated until no significant dependencies of the class label and the values of the corrected trajectories remain. In this chapter we analyze the properties of RMH for binary classification problems. Furthermore, we show that, for some specific cases, RMH selects the variables that appear in the optimal classification rule. Finally, an empirical comparison between RMH and other variable selection methods is also performed using both synthetic and real-data examples.

The structure of this chapter is as follows: the problem of variable selection in supervised classification problems is introduced in Section 5.1. In Section 5.2 we review maxima hunting (MH). Recursive maxima hunting (RMH) is introduced in Section 5.3 with special attention to the computational details of the algorithm. The theoretical properties of RMH are analyzed in Section 5.4. The proofs of the propositions stated in this section can be found in Appendix B. Section 5.6 presents the results of an empirical evaluation of the method in a series of experiments with both simulated and real-world data. Finally, the conclusions of this chapter are presented in Section 5.7.

5.1 Statement of the problem

Consider a classification problem in which the observations are random continuous trajectories $X \in \mathcal{X}$, drawn from the model

$$\begin{cases} \mathbb{P}_0 : (X(t)|\, Y = 0) = \mu_0(t) + Z(t) & \text{with probability } 1 - p, \\ \mathbb{P}_1 : (X(t)|\, Y = 1) = \mu_1(t) + Z(t) & \text{with probability } p, \end{cases} \tag{5.1}$$

where μ_0 and μ_1 are deterministic functions, and Z is a zero-mean noise process with a continuous covariance function $k(s,t) = \mathbb{E}(Z(s)Z(t))$. The domain of the random trajectories is $\mathcal{T} = [0, T]$. Classification consists in determining to which population an unlabeled trajectory X belongs. This determination is made by measurable functions $g : \mathcal{X} \to \{0, 1\}$, called classifiers. The goal is to induce a classifier that minimizes the classification error rate, which is defined as the probability of misclassification:

$$g^* = \operatorname*{argmin}_g P\left[g(X(t)) \neq Y\right]. \tag{5.2}$$

Assuming that μ_0 is known, one can consider, without loss of generality, the equivalent classification problem

$$\begin{cases} \mathbb{P}_0 : (X(t)|\, Y = 0) = Z(t) & \text{with probability } 1 - p, \\ \mathbb{P}_1 : (X(t)|\, Y = 1) = \mu(t) + Z(t) & \text{with probability } p, \end{cases} \tag{5.3}$$

where $\mu(t) = \mu_1(t) - \mu_0(t)$. If the class 0 mean, μ_0, is not known, it can be estimated from the sample.

The goal of variable selection is to identify a set of impact points, $\mathbf{t} = \{t_1, \ldots, t_D\}$, which capture the bulk of the information for discrimination in the original trajectories. Subsequently, the original functional datum X is replaced by the attribute vector $X(\mathbf{t}) = (X(t_1), \ldots, X(t_D))$, whose components are the values of the trajectory at the selected impact points. This vector of attributes is then used as input of a multivariate classifier for prediction. Several benefits of this dimensionality reduction are obtained. First, it makes it possible to take advantage of off-the-shelf multivariate models for classification and regression, such as those implemented in **caret** (Kuhn, 2008), and **mlr** (Bischl et al., 2016), in R, or **scikit-learn**, in Python (Pedregosa et al., 2011). Second, training and prediction can be made with lower computational costs. In addition, the predictive model is typically more interpretable. Finally, the accuracy of the model trained in the lower-dimensional representation can be comparable to the predictors induced from the original functional data. In some problems, variable selection has a regularization effect, so overfitting is reduced and more accurate predictors can be built.

A literature review of state-of-the-art methods for variable selection in the functional setting can be found in Section 2.3.2. The next section introduces maxima hunting (MH), the algorithm that serves as inspiration for the RMH algorithm proposed in this chapter.

5.2 Maxima hunting

Maxima hunting (MH) is a variable selection method for functional data that can be applied in supervised learning settings (Berrendero, Cuevas, and Torrecilla, 2016b).

It is a filter method (i.e., independent of the classifier), non-parametric (it only requires the piecewise continuity of the trajectories), and takes into account the functional structure of the data. Furthermore, it has an intuitive interpretation and good empirical performance, which is backed by some appealing statistical properties.

MH is based on identifying the points at which the dependence of the class label on the corresponding function values presents a local maximum (Berrendero, Cuevas, and Torrecilla, 2016b). The relevance of $X(t)$, the value of the process at time $t \in [0, T]$, is quantified in terms of $\operatorname{Rel}(X(t), Y)$, a suitable non-negative measure of statistical dependence with the class label. Then, MH selects the local maxima of this measure

$$\{t_1, \ldots, t_D\} = \left\{ t : \exists (a, b) \subset [0, T], t = \underset{s \in (a,b)}{\operatorname{argmax}} \operatorname{Rel}(X(s), Y) \right\}. \tag{5.4}$$

In Berrendero, Cuevas, and Torrecilla, 2016b, the distance covariance $\mathcal{V}^2(X(t), Y)$, or the distance correlation $\mathcal{R}^2(X(t), Y)$, introduced by Székely, Rizzo, and Bakirov (2007) were employed for measuring this relevance. The rationale behind selecting local maxima is that these summarize the features of the functions in the neighborhood of the selected points. Relevant variables close to the selected ones are excluded because they provide redundant information. MH was originally formulated in the context of binary classification. Nevertheless, the method can be readily extended to regression with functional predictors and scalar response (Kneip, Poss, and Sarda, 2016; Berrendero, Bueno-Larraz, and Cuevas, 2019).

To illustrate this method, consider a set of functional observations drawn from two different stochastic processes, as in Equation (5.3). In this example, we assume that Z is a standard Brownian motion, and the class 1 mean is the piecewise linear function

$$\mu(t) = \begin{cases} \frac{5}{4}t - \frac{1}{4} & , \frac{1}{5} \leq t < \frac{2}{5}, \\ -10t + \frac{17}{4} & , \frac{2}{5} \leq t < \frac{1}{2}, \\ \frac{15}{4}t - \frac{21}{8} & , \frac{1}{2} \leq t < \frac{7}{10}, \\ 0 & , \text{otherwise}. \end{cases} \tag{5.5}$$

Some trajectories of class 0 (blue lines) and class 1 (orange lines) are displayed in the left plot of Figure 5.1. Superimposed are the corresponding means (thick lines). In this example, the measure of dependence between $X(t)$, the functional covariate, and Y, the class label, is $\operatorname{Rel}(X(t), Y) = \mathcal{V}^2(X(t), Y)$, the square-distance covariance (Székely, Rizzo, and Bakirov, 2007). The plot of $\mathcal{V}^2(X(t), Y)$ as a function of t is displayed at the right of this figure. The locations of the selected variables, which correspond to local maxima of $\mathcal{V}^2(X(t), Y)$, are marked by vertical dashed lines.

In spite of its intuitive quality, appealing theoretical properties, and good empirical performance, MH presents some drawbacks. In practice, the empirical estimates of the relevance function can be rather irregular because of noisy measurements or sampling fluctuations. This makes it difficult to identify the true local maxima. In Berrendero, Cuevas, and Torrecilla, 2016b, one attempts to exclude spurious maxima by selecting only points $t_{\max}$ that are also maxima in the interval $(t_{\max} - h, t_{\max} + h)$, where h is a smoothing parameter. As an alternative, in Ordóñez et al., 2018 the relevance function is smoothed by using a truncated expansion in the basis of splines. Then, the local maxima of this smoothed relevance curve are calculated using the STEM (Smoothing and TEsting of Maxima) algorithm (Schwartzman, Gavrilov, and Adler, 2011).

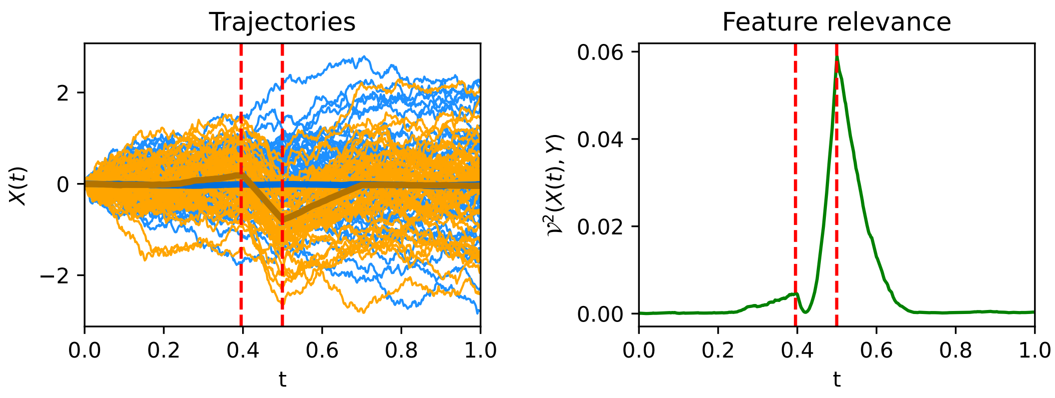


FIGURE 5.1: Application of MH to the classification problem defined in Equation (5.3), with Brownian motion noise and class 1 mean given by Equation (5.5). On the left plot, class 0 and class 1 trajectories and the corresponding means (thicker lines) are displayed in blue and orange, respectively. The plot on the right-hand side displays the graph of $\mathcal{V}^2(X(t), Y)$ as a function of t. MH selects the values of t that are local maxima of this function. These are marked in both graphs with vertical dashed lines.

A more fundamental limitation is that MH cannot identify variables that are not relevant by themselves, but become relevant when considered jointly. This limitation is illustrated in Figure 5.1. In this example, MH selects the two most important variables for prediction, $X(2/5)$ and $X(1/2)$. However, optimal classification can be achieved only by considering all the non-differentiable points of $\mu_1(t)$ (Berrendero, Cuevas, and Torrecilla, 2018), which include $X(1/5)$ and $X(7/10)$. These are not selected in MH because, when considered in isolation, they are statistically independent of the class; that is, $\mathcal{V}^2(X(1/5), Y) = \mathcal{V}^2(X(7/10), Y) = 0$. However, they become relevant when taken in combination with $X(2/5)$ and $X(1/2)$.

In recursive maxima hunting, the method that is described in the following section, this limitation is addressed by first removing the contribution of the selected variable before proceeding to select new ones. This reveals variables whose relevance becomes apparent only when considered in combination with other variables that have been selected previously.

5.3 Recursive maxima hunting

Recursive maxima hunting (RMH) is a filter variable selection method that takes maxima hunting as its starting-point. As in MH, the first variable selected is the one that maximizes a measure of dependence of the class label and $X(t)$ globally. Then, before searching for a new variable, one removes from the trajectories the information on the class provided by the variable that has been selected. This is achieved by subtracting from the original trajectories the expectation of the process conditioned on the values observed at the selected point. The next variable selected is the one that maximizes the dependence between the class label and the value of the modified process at that point. The selection process proceeds until the desired number of variables has been selected or some stopping criterion is met.

To formalize the algorithm, it is useful to introduce some notation: Let $X^{[0]} = X$ and $Z^{[0]} = Z$ denote the original stochastic processes defined in (5.3). Assume that t_i is the variable selected at the i-th iteration of the algorithm. The quantities $X^{[i]}$ and $Z^{[i]}$ represent the processes that result from applying the sequence of corrections made to the original ones up to the i-th step of the algorithm. In terms of these quantities, the pseudocode for RMH is

Algorithm 1 Recursive maxima hunting

```
  **Input:**
    $X$: The random trajectory.
    $Y$: The class labels.
    $Z$: The noise process.
    Rel: A univariate variable relevance function.
  **Output:**
    $\{t_1, \dots, t_D\}$: List of impact points, in the order of selection.
 1: $i \leftarrow 0$
 2: $X^{[0]}(t) \leftarrow X(t)$
 3: $Z^{[0]}(t) \leftarrow Z(t)$
 4: **repeat**
 5:     $i \leftarrow i + 1$
 6:     $t_i \leftarrow \operatorname{argmax}_{t \in [0,T]} \operatorname{Rel}(X^{[i-1]}(t), Y)$
 7:     Update the trajectories:
            $$X^{[i]}(t) \leftarrow X^{[i-1]}(t) - \mathbb{E}\left[Z^{[i-1]}(t) \mid Z^{[i-1]}(t_i) = X^{[i-1]}(t_i)\right] \qquad (5.6)$$
 8:     Update the noise process
            $$Z^{[i]}(t) \leftarrow \left[Z^{[i-1]}(t) \mid Z^{[i-1]}(t_i) = 0\right] \qquad (5.7)$$
 9: **until** the stopping condition is met.
10: $D \leftarrow i$
11: **return** $\{t_1, \dots, t_D\}$
```

If, as assumed in (5.3), Z is a zero-mean Gaussian process whose covariance function is $k(s,t)$, the correction in Equation (5.6) at the i-th step of the algorithm is

$$\mathbb{E}\left[Z^{[i-1]}(t) \mid Z^{[i-1]}(t_i) = X^{[i-1]}(t_i)\right] = \frac{k^{[i-1]}(t, t_i)}{k^{[i-1]}(t_i, t_i)} X^{[i-1]}(t_i). \qquad (5.8)$$

In this expression, $k^{[i-1]}(s,t)$ is the covariance function associated with $Z^{[i-1]}$, which is also a zero-mean Gaussian process. For $i = 1$, $Z^{[0]}(t) = Z(t)$, and $k^{[0]}(s,t) = k(s,t)$. Note that, after applying this correction to $X^{[i-1]}(t)$, $X^{[i]}(t_i) = 0$.

There are two key differences between this algorithm and MH. The first one is the maximization step: in contrast to MH, where local maxima of $\operatorname{Rel}(X(t), Y)$ are sought, the goal of RMH is to identify the global maximum of $\operatorname{Rel}(X^{[i-1]}(t), Y)$. If the sample fluctuations are large or the data are noisy, there can be local maxima that are spurious. Furthermore, nearby local maxima are likely to be redundant and should therefore not be selected simultaneously. To alleviate these difficulties several regularization strategies have been proposed. These strategies require the determination of either a lengthscale to define the local maxima (Berrendero, Cuevas, and Torrecilla, 2016b), or a smoothing parameter (Ordóñez et al., 2018). In RMH, one seeks a global optimum, which is a much simpler problem and does not require to specify any hyperparameter. The second difference is that, since $X^{[i]}(t_i) = 0$, the correction made at each step removes the information on the class provided by the values of the trajectories at t_i. In this manner, it is possible to identify variables that are relevant not only by themselves, but also when considered jointly with previously selected ones.

A simple example

Before delving into the details of the method, we illustrate the workings of RMH in a simple example. As in Figure 5.1, consider the problem of discriminating between standard Brownian motion trajectories ($\mathbb{P}_0$) and Brownian motion trajectories with the piecewise linear mean given in Eq. (5.5) ($\mathbb{P}_1$). Following Berrendero, Cuevas, and Torrecilla (2018), the optimal classification rule for this problem is

$$\begin{aligned} g^*(X) &= \langle X, \mu \rangle_k - \frac{1}{2}\|\mu(t)\|_k^2 - \log\left(\frac{1-p}{p}\right) \\ &= -\frac{5}{4}X\left(\frac{1}{5}\right) + \frac{45}{4}X\left(\frac{2}{5}\right) - \frac{55}{4}X\left(\frac{1}{2}\right) + \frac{15}{4}X\left(\frac{7}{10}\right) \\ &\quad - \frac{105}{16} - \log\left(\frac{1-p}{p}\right). \end{aligned} \tag{5.9}$$

This optimal rule depends only on the variables $X\left(\frac{1}{5}\right)$, $X\left(\frac{2}{5}\right)$, $X\left(\frac{1}{2}\right)$ and $X\left(\frac{7}{10}\right)$.

The stepwise application of RMH is displayed in the sequence of plots in Figure 5.2. Each row corresponds to an iteration of the algorithm. The plots in the left column display the trajectories of the process, which are modified at each iteration by applying the correction given by Equation (5.6). The original trajectories are shown in the first row of this column. Subsequent plots in the left column correspond to corrected trajectories at each step of the algorithm. In this case, Brownian bridges appear with each conditioning of the original Brownian motion trajectories. In each of these plots, class 0 and class 1 trajectories are depicted in blue and orange, respectively. The thicker lines correspond to the empirical mean functions of the trajectories in each class.

The plots in the right column display the dependence between the class label and the values of $X^{[i]}(t)$ as a function of t. The squared distance covariance $\mathcal{V}^2\left(X^{[i]}(t), Y\right)$ is used as a measure of dependence. At each iteration, the value of t_i that maximizes this quantity is located. This value is marked in the plots with a dashed vertical line.

In this example, RMH finds all the variables involved in the optimal classification rule given in Eq. (5.9) and only those. Note that the importance of the variables $X(7/10)$ and $X(1/5)$, as quantified by an univariate measure of dependence, is revealed only after conditioning to $X(1/2)$ and $X(2/5)$, respectively. This is an example of the types of relevant variables that cannot be identified using MH. Once the features that appear in the optimal classification rule have been selected, the relevance function is zero up to sample fluctuations. Therefore, RMH stops once all relevant variables have been identified.

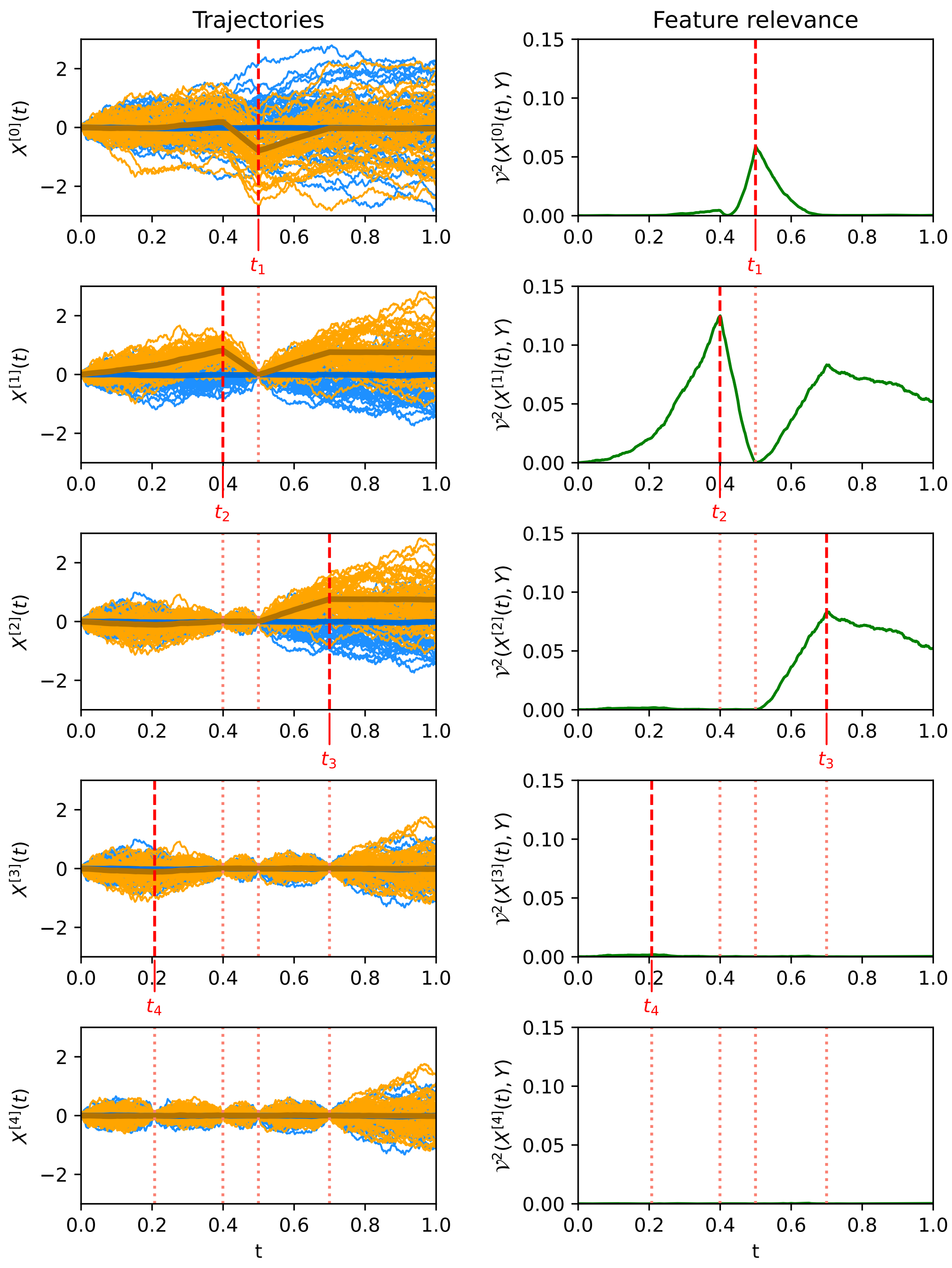


FIGURE 5.2: Example of the execution of RMH with Brownian motion trajectories corresponding to two classes with different means.

5.4 Properties of RMH

In this section we derive some important properties of the RMH corrections with the assumptions that the classification problem is of the form given by (5.3), and Z is a zero-mean Gaussian process. Note that the Gaussianity assumption for Z is not overly restrictive. In many cases of interest, it is possible to approximate the underlying noise in functional classification problems by a Gaussian process with the covariance function $k(s,t) = \mathbb{E}\left[Z(s)Z(t)\right]$. Furthermore, RMH can be applied even if Z is not Gaussian or the model is different from (5.3). However, in that case there are no guarantees that properties derived in this section hold. The proofs of all propositions and theorems stated are in Appendix B.

Recursivity

An important property of RMH under the assumption that the classification problem is of the form given by (5.3) with Z a zero-mean Gaussian process, is its recursive nature. This is formally stated in the following theorem:

Theorem 1. *Consider the process*

$$X(t) = \begin{cases} Z(t) & \text{if } Y = 0, \\ \mu(t) + Z(t) & \text{if } Y = 1, \quad i \geq 1, \end{cases} \tag{5.10}$$

with μ a deterministic function, and Z a zero-mean Gaussian process whose covariance function is $k(s,t)$. The modified process

$$X^{[i]}(t) = X^{[i-1]}(t) - \mathbb{E}\left[Z^{[i-1]}(t) \mid Z^{[i-1]}(t_i) = X^{[i-1]}(t_i)\right], \quad i \geq 1 \tag{5.11}$$

with $X^{[0]} = X$ and $Z^{[0]} = Z$ is of the same form as the original one:

$$X^{[i]}(t) = \begin{cases} Z^{[i]}(t) & \text{if } Y = 0, \\ \mu^{[i]}(t) + Z^{[i]}(t) & \text{if } Y = 1, \quad i \geq 1, \end{cases} \tag{5.12}$$

with $\mu^{[i]}$ a deterministic function, and $Z^{[i]}$ a zero-mean Gaussian process.

In this expression, the noise term $Z^{[i]}$, is the zero-mean Gaussian process that results from conditioning to $Z^{[i-1]}$ being 0 at t_i:

$$\begin{aligned} Z^{[i]}(t) &= Z^{[i-1]}(t) - \mathbb{E}\left[Z^{[i-1]}(t) \mid Z^{[i-1]}(t_i)\right] \\ &= \left[Z^{[i-1]}(t) \mid Z^{[i-1]}(t_i) = 0\right] = Z^{[i-1]}(t) - \frac{k^{[i-1]}(t,t_i)}{k^{[i-1]}(t_i,t_i)} Z^{[i-1]}(t_i), \end{aligned} \tag{5.13}$$

with $Z^{[0]} = Z$. The kernel function of this modified noise process is

$$\begin{aligned} k^{[i]}(s,t) &= \mathbb{E}\left[Z^{[i]}(s)Z^{[i]}(t)\right] \\ &= k^{[i-1]}(s,t) - \frac{k^{[i-1]}(s,t_i)k^{[i-1]}(t_i,t)}{k^{[i-1]}(t_i,t_i)}, \end{aligned} \tag{5.14}$$

with $k^{[0]}(s,t) = k(s,t)$. The function $\mu^{[i]}$ is the mean of the modified class 1 trajectories that results from applying the i-th correction in RMH:

$$\mu^{[i]}(t) = \mathbb{E}\left[X^{[i]}(t) \mid Y = 1\right] = \mu^{[i-1]}(t) - \frac{k^{[i-1]}(t,t_i)}{k^{[i-1]}(t_i,t_i)}\mu^{[i-1]}(t_i), \quad i \geq 1, \tag{5.15}$$

with $\mu^{[0]} = \mu$. Note that, as a result of the successive corrections

$$X^{[i]}(t_j) = 0, \quad j = 1,\dots,i, \tag{5.16}$$

for both $Y = 0$ and $Y = 1$. Therefore, the values $\left\{X^{[i]}(t_j)\right\}_{j=1}^{i}$ cannot be used any longer to discriminate between the two classes.

Simultaneous application of the corrections

A naïve approach to computing the modified process at the i-th step of RMH is to use recursion (5.14) from $Z^{[0]} = Z$. However, assuming that the values of the impact points $\{t_1,\dots,t_i\}$ are known, the corrections at the i-th step of RMH can be computed directly from Z:

$$\begin{aligned} Z^{[i]}(t) &= Z(t) - [Z(t) \mid Z(t_1),\dots Z(t_i)] \\ &= [Z(t) \mid Z(t_1) = 0,\dots Z(t_i) = 0]. \end{aligned} \tag{5.17}$$

The correction applied to Z to obtain $Z^{[i]}$ can be expressed as the sum of the subsequent corrections

$$\mathbb{E}\left[Z(t) \mid Z(t_1),\dots,Z(t_i)\right] = \sum_{k=0}^{i-1} \mathbb{E}\left[Z^{[k]}(t) \mid Z^{[k]}(t_{k+1})\right]. \tag{5.18}$$

It is also possible to derive the form of the mean at step i from Equation (5.17), as

$$\mu^{[i]}(t) = \mu(t) - \mathbb{E}\left[Z(t) \mid Z(t_1) = \mu(t_1),\dots,Z(t_i) = \mu(t_i)\right]. \tag{5.19}$$

In a similar way as in Equation (5.18), the total correction over the mean at step i is

$$\mathbb{E}\left[Z(t) \mid Z(t_1) = \mu(t_1),\dots Z(t_i) = \mu(t_i)\right] = \sum_{k=0}^{i-1} \mathbb{E}\left[Z^{[k]}(t) \mid Z^{[k]}(t_{k+1}) = \mu^{[k]}(t_{k+1})\right]. \tag{5.20}$$

This property is illustrated in Figure 5.3.

According to this equation, at the i-th iteration of the algorithm, the expression

$$\hat{\mu}_i(t) = \mathbb{E}\Big[Z(t) \mid Z(t_1) = \mu(t_1),\dots,Z(t_i) = \mu(t_i)\Big] \tag{5.21}$$

provides an approximation of μ by interpolation from $\{(t_1,\mu(t_1)),\dots,(t_i,\mu(t_i))\}$. The form of the approximation depends on the type of Gaussian process considered. This is illustrated in Figure 5.4, which displays interpolations of the sine function assuming different types of Gaussian Processes.

If, after D iterations of RMH, $\mu(t) \approx \hat{\mu}_D(t)$, then $\mu^{[D]}(t) = \mu(t) - \hat{\mu}_D(t) \approx 0$. That is, the corrected trajectories of both classes have approximately the same mean. In such case, the dependencies between $X^{[D]}(t)$ and Y are small and RMH halts. This

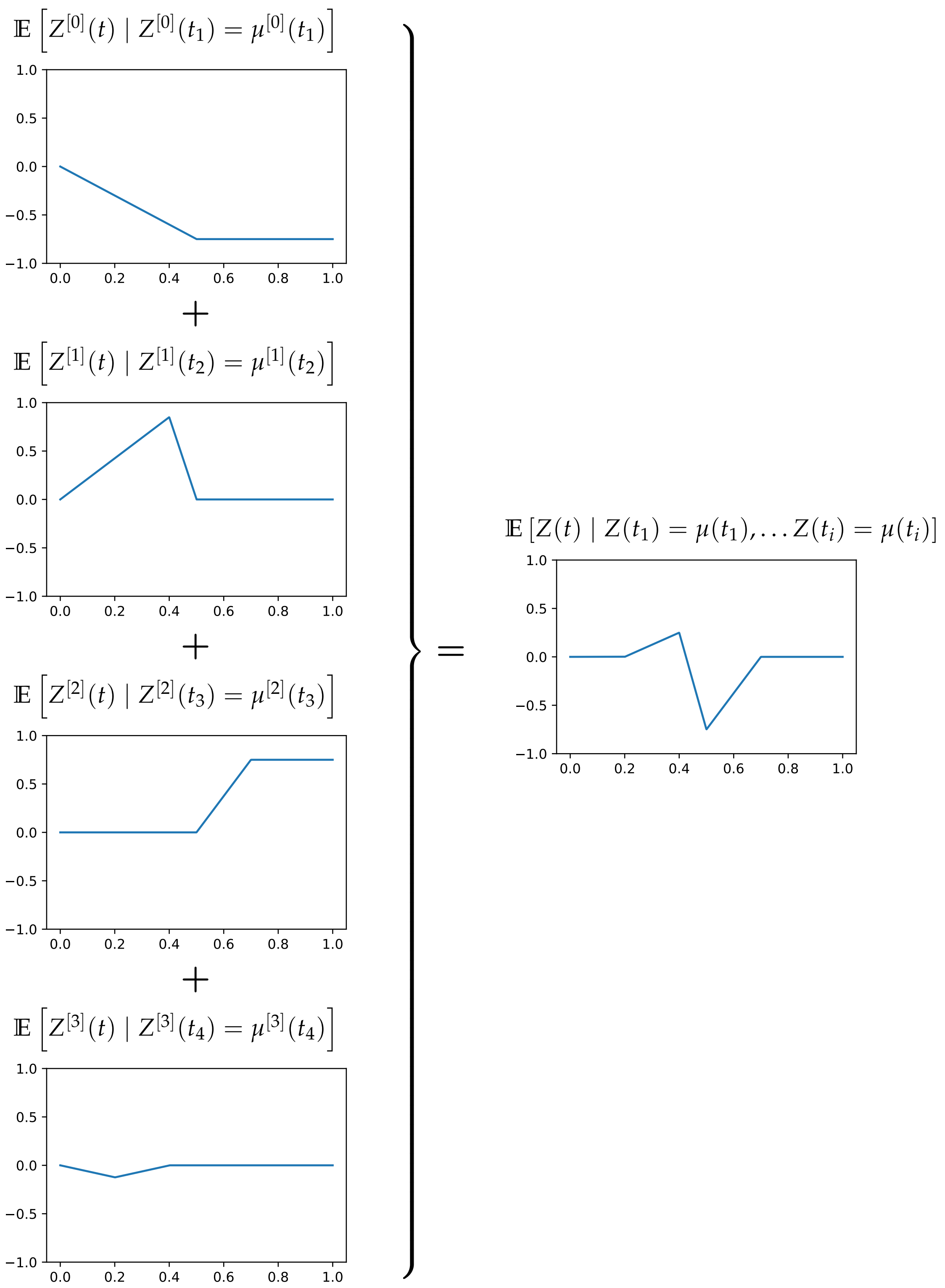


FIGURE 5.3: Simultaneous application of the RMH corrections. This figure illustrates Equation (5.20). On the left column, the corrections for the mean of class 1 at each step of the algorithm are shown, for the example in Figure 5.2. The total correction performed after the four steps is depicted on the right. As it can be seen on top, it is possible to compute this correction directly from the original data when the selected points are known, without computing the correction at each step.

observation offers a novel perspective on RMH as providing an approximation of $\mu(t)$ from $\{(t_1, \mu(t_1)), \ldots, (t_D, \mu(t_D))\}$.

Of special interest is the case in which μ is in the Reproducing Kernel Hilbert Space (RKHS) whose kernel is the covariance function of Z. In this case, it is possible to derive the following theorem.

Theorem 2. *Under the conditions specified in Theorem 1, if*

$$\mu(t) = \sum_{d \geq 1} m_d k(t_d, t), \tag{5.22}$$

with $m_d > 0$ and $t_d \neq t_{d'}$ for $d \neq d'$, and one applies the RMH corrections for $\{t_1, \ldots, t_i\}$, then

$$\mu^{[i]}(t) = \sum_{d \geq i+1} m_d k^{[i]}(t_d, t). \tag{5.23}$$

Furthermore, if the linear combination in Equation (5.22) is finite

$$\mu(t) = \sum_{d=1}^{D} m_d k(\tau_d, t), \tag{5.24}$$

and one applies the RMH corrections for $\{\tau_d\}_{d=1}^{D}$, then $\mu^{[D]}(t) = 0$. This means that, if the impact points $\{\tau_1, \ldots, \tau_D\}$ are selected in the first D iterations, not necessarily in that order, then RMH halts. This is an important result because the variables $\{X(\tau_1), \ldots, X(\tau_D)\}$ are precisely those involved in the Bayes classification rule (Berrendero, Cuevas, and Torrecilla, 2018, Theorem 7a). As shown in the following section, this property is obtained when $Z(t)$ is a Brownian motion, or an Ornstein-Uhlenbeck Gaussian process.

5.5 RMH with different types of Gaussian Processes

In what follows, we explore the properties of RMH with different assumptions for the Gaussian process noise Z. First, we analyze the properties of RMH with Markov-Gaussian processes. As a particular case, we consider Brownian motion noise. Then, we analyze RMH with Ornstein-Uhlenbeck noise. This case is of particular interest because it is the only stationary Gaussian process with a continuous covariance function that has the Markov property (Doob, 1942).

5.5.1 RMH with Markovian noise

Consider the process X defined in Equation (5.3) with Z a zero-mean Gauss-Markov process. The corrected process at the i-th step of RMH, $Z^{[i]}$, is also Markovian. As a consequence of this property, after the i-th correction, $X^{[i]}(s)$ with $s \in [0, t_i]$ and $X^{[i]}(t)$ with $s \in [t_i, T]$ are independent. A proof of this result is given in Appendix B. To prove this independence of these quantities, it is sufficient to prove the following property for Z:

Proposition 1. *Let Z be a zero-mean Gaussian process with the Markov property. The process $\tilde{Z}(t) = Z(t) - \mathbb{E}[Z(t) \mid Z(\tau)]$, with $s < \tau < t$ verifies that $\tilde{Z}(s)$ and $\tilde{Z}(t)$ are independent.*

In consequence, the next steps in RHM can be carried out separately to the right and to the left of t_i. This can be exploited to parallelize the algorithm by processing each of these intervals separately (Torrecilla and Suárez, 2016).

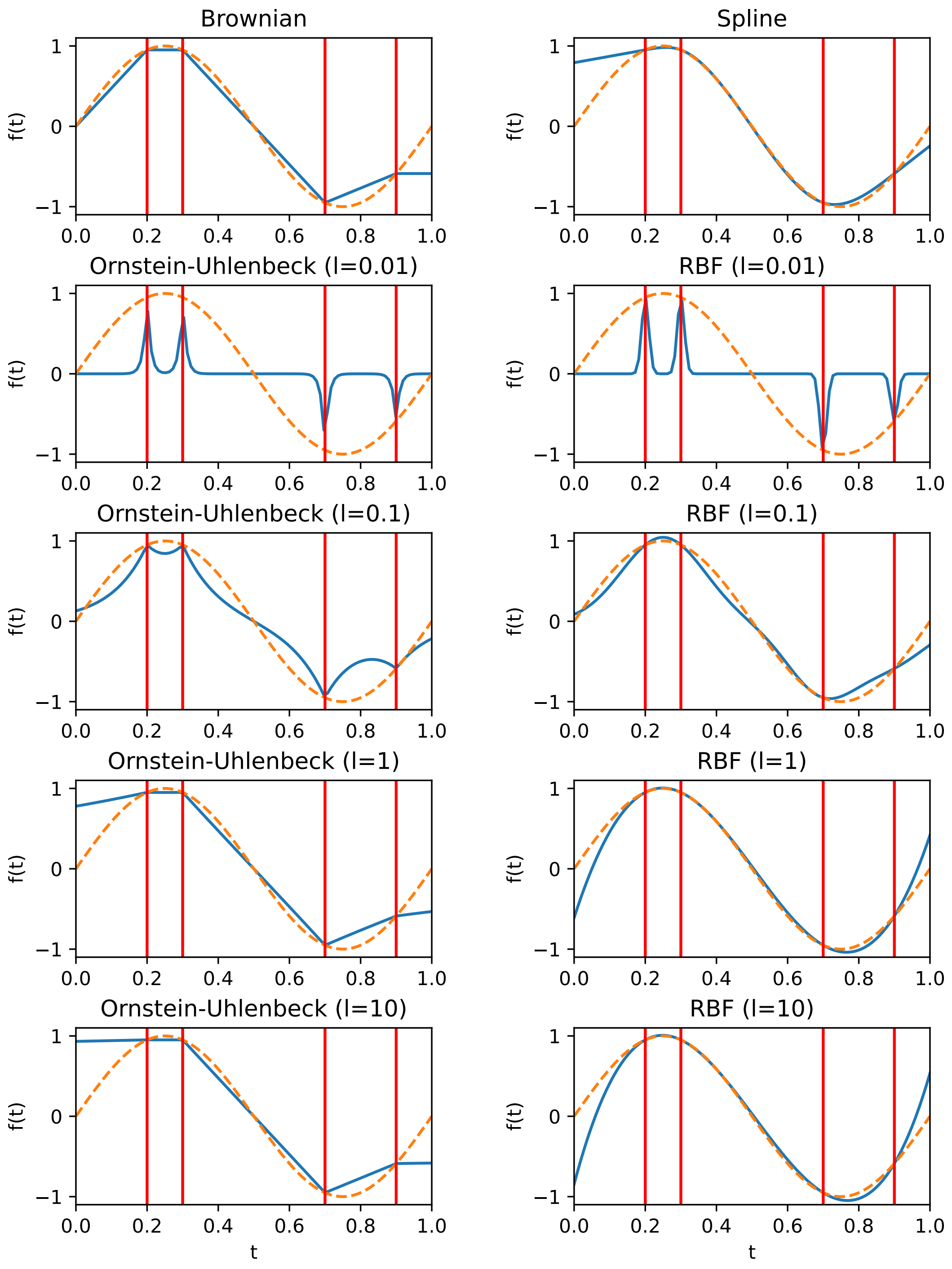


FIGURE 5.4: Interpolation of $f(t) = \sin(2\pi t)$ in $[0, 1]$ (shown as a dashed line) based on its values at $t_1 = 0.2, t_2 = 0.3, t_3 = 0.7, t_4 = 0.9$ (marked with red vertical lines) assuming different Gaussian processes. From top to bottom and from left to right: Brownian motion, spline, Ornstein-Uhlenbeck and RBF, the last two with lengthscales 0.01, 0.1, 1.0 and 10.0.

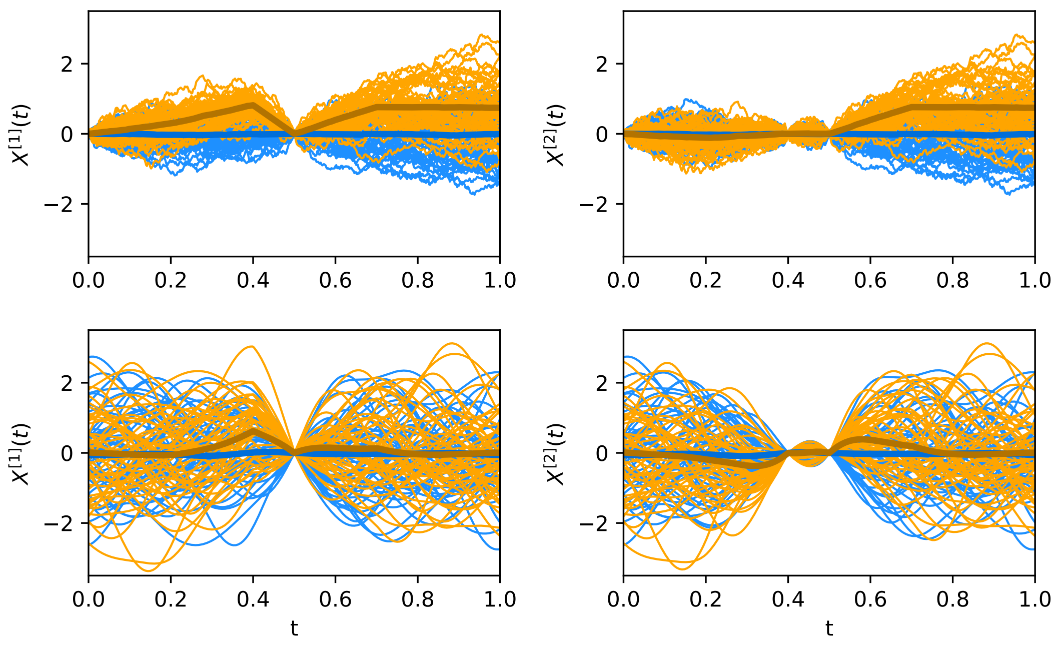


FIGURE 5.5: The first row shows an iteration of RMH using the Brownian correction, which has the Markov property. It is apparent that the correction over the right interval does not change the left interval. The second row shows an RBF correction, which is not Markovian. In this case the correction affects the two intervals.

This property is illustrated in Figure 5.5, which shows the result of applying the correction at $t_2 = 0.4$ for a process conditioned after having selected $t_1 = 0.5$. The first row in this figure corresponds to the example analyzed in Section 5.3, in which Z is Brownian motion. Since Brownian motion is Markovian, conditioning on the values of the process at t_2 does not alter the trajectories, $X^{[2]}(t)$, in the subinterval $t > t_1$. In the second row Z is a zero-mean Gaussian process with an RBF kernel (Rasmussen and Williams, 2005). This process is not Markovian. As a consequence, applying the correction at t_2, modifies the values of $X^{[2]}(t)$, the modified trajectories, not only for $t < t_1$, but also for $t > t_1$.

5.5.2 RMH with Brownian noise

Brownian motion is a Gauss-Markov process. Therefore, as discussed in the previous subsection, the successive RMH corrections can be applied in parallel. Furthermore, as stated in the following theorem, when μ is piecewise linear with $\mu(0) = 0$, and the distance covariance is used as the univariate relevance measure, RMH selects the variables that appear in Bayes rule and then halts.

Theorem 3. *Consider the classification problem defined by Equation (5.3) in the interval $\mathcal{T} = [0, \infty)$. Let Z be a zero-mean Brownian motion, whose covariance function is*

$$k_{BM}(s,t) = \sigma^2 \min(s,t). \tag{5.25}$$

Assume that μ is the piecewise linear function

$$\mu(t) = \begin{cases} \mu_1 \frac{t}{\tau_1} & \text{if } t \in [0, \tau_1) \\ \mu_d \left(1 - \frac{t-\tau_d}{\tau_{d+1}-\tau_d}\right) + \mu_{d+1} \frac{t-\tau_d}{\tau_{d+1}-\tau_d} & \text{if } t \in [\tau_d, \tau_{d+1}),\ d = 1, \ldots D-1 \\ \mu_D & \text{if } t \geq \tau_D \end{cases} \tag{5.26}$$

with $0 < \tau_1 < \ldots < \tau_D$. *Then,*

(i) *The class 1 mean belongs to the RKHS associated to the Brownian motion kernel and is of the form*

$$\mu(t) = \sum_{d=1}^{D} m_d k_{BM}(\tau_d, t), \quad \text{with } m_d \neq 0 \quad d = 1, \ldots, D. \tag{5.27}$$

(ii) *RMH, using distance covariance as a relevance function, selects* $\{\tau_1, \ldots, \tau_D\}$*, not necessarily in that order, and then halts.*

(iii) *The points selected by RMH are the ones that appear in Bayes rule for the classification problem defined in* (5.3) *and only those.*

The derivation of these properties, which is given in Appendix B, hinges on the fact that, in this case, the optimal classifier is Fisher's linear discriminant applied to $\{X(\tau_d)\}_{d=1}^{D}$, the projections of X at the impact points $\{\tau_d\}_{d=1}^{D}$ (Berrendero, Cuevas, and Torrecilla, 2018, Theorem 7a).

This theorem provides support for the application of RMH with Brownian corrections in Torrecilla and Suárez, 2016. However, the assumption $\mu(0) = 0$, is too restrictive: it implies that all trajectories start at zero ($X(0) = 0$), a condition that is rarely the case in practice. In the following section, we will show that similar properties are obtained when Z is an Ornstein-Uhlenbeck process with the advantage that, with in this case, model (5.3) can be used for classification problems in which $X(0)$ can be different from 0.

5.5.3 RMH with Ornstein-Uhlenbeck noise

In this section, we derive some properties of RMH in the case that the noise Z is a zero-mean Ornstein-Uhlenbeck (OU) process. The OU process has some special properties that make it useful in RMH even when in the actual Z is a general stochastic process whose form is not known. In particular, it has the Markov property. Therefore, one can take adavantage of the properties of RHM with Gauss-Markov process derived in Section 5.5.1. A second important property is its stationarity, a condition that holds in many problems of interest. Even if the data are not stationary, they can be rendered approximately stationary through the use of some transformation, such as taking derivatives, logarithms, or removing trends and seasonal components (Kokoszka and Reimherr, 2017).

Finally, assuming that Z is OU, and that the distance covariance is used as the univariate relevance measure, RMH selects the variables that appear in Bayes rule in the conditions specified by the following theorem

Theorem 4. *Consider the classification problem defined in Equation* (5.3) *defined in the whole real line* $\mathcal{T} = \mathbb{R}$. *Let Z be an Ornstein-Uhlenbeck process, whose covariance function is*

$$k_{OU}(s,t) = \sigma^2 \exp\left(-\frac{|t-s|}{l}\right). \tag{5.28}$$

Assume that μ is in the RKHS whose reproducing kernel is $k(s,t)$, and has the form

$$\mu(t) = \sum_{d=1}^{D} m_d k_{OU}(\tau_d, t), \quad m_d \neq 0 \quad d = 1, \ldots, D, \tag{5.29}$$

with $\tau_1 < \ldots < \tau_D$. Under these conditions, RMH using distance covariance as a relevance measure selects $\{\tau_d\}_{d=1}^{D}$, which are the points that appear in Bayes rule, and only those.

As in the case of Brownian motion, the optimal classifier in this case is Fisher's linear discriminant applied to $\{X(\tau_d)\}_{d=1}^{D}$ (Berrendero, Cuevas, and Torrecilla, 2018, Theorem 7a).

Note that, since the RKHS associated to the OU kernel assumption is dense in L^2, the assumption (5.29) is not overly restrictive if (5.29) provides a sufficiently accurate approximation of μ. The parameter σ^2 does not affect the correction of the trajectories (Equation (5.6)), or the update of the noise process (Equation (5.13)). Therefore, without loss of generality one can use any value of σ in Equation (5.28). By contrast, the value of the lengthscale parameter l needs to be known. If l is not known, it can be estimated from the data.

The interpretation of RMH as a process that yields a sequence of approximations of μ, which are progressively more accurate, suggests a different prescription for the choice of σ^2 and l. Specifically, the limit

$$\sigma^2 \to \infty, l \to \infty, \text{ with } \frac{\sigma^2}{l} = \text{constant}, \tag{5.30}$$

corresponds to making no prior assumptions on the form of μ. In this limit, the dominant term in the OU kernel is

$$k_{OU}(s,t) \approx \sigma^2. \tag{5.31}$$

Using this constant kernel, the correction given by Equation (5.8) at the first step of RMH is simply $X(t_1)$. Thus, the updated trajectories become

$$X^{[1]}(t) = X(t) - X(t_1). \tag{5.32}$$

Also in this limit, the process $Z^{[1]}$ is a two-sided Brownian motion emanating from t_1 (Mörters and Peres, 2010). This result is stated in the following theorem:

Theorem 5. *Let Z be a zero-mean Ornstein-Uhlenbeck process whose kernel is*

$$k_{OU}(s,t) = \sigma^2 \exp\left(-\frac{|t-s|}{l}\right). \tag{5.33}$$

In the limit $l \to \infty$, $\sigma^2 \to \infty$ with $\frac{2\sigma^2}{l} = 1$, The process $Z'(t) = [Z(t) \mid Z(\tau) = 0]$ is two-sided standard Brownian motion whose origin is τ.

Because of these properties, we will use the term *uniform Brownian* to denote the OU correction in the limit given by Equation (5.30).

The approximation of μ at the i-th step of RMH with the uniform Brownian correction is the piecewise linear function

$$\hat{\mu}_i(t) = \begin{cases} \mu_1 & \text{if } t \leq t_{[1]} \\ \mu_j \left(1 - \frac{t - t_{[j]}}{t_{[j+1]} - t_{[j]}}\right) + \mu_{j+1} \frac{t - t_{[j]}}{t_{[j+1]} - t_{[j]}} & \text{if } t \in [t_{[j]}, t_{[j+1]}],\ j = 1, \ldots i-1 \\ \mu_i & \text{if } t \geq t_{[i]} \end{cases} \tag{5.34}$$

where $t_{[1]} < \ldots < t_{[i]}$, are the ordered values $t_1, \ldots t_i$. This can be appreciated in Figure 5.4, where increasing the lengthscale of the Ornstein-Uhlenbeck process gives interpolations that approximate to this equation.

An advantage of considering this limit for Z is that it is not necessary to determine the value of any hyperparameter in RMH. The class 1 mean, μ, is approximated by a piecewise linear function, which is a simple, flexible, robust, and generally accurate interpolation method. Finally, as illustrated in the experiments carried out in Section 5.6, this prescription has excellent performance in the classification problems considered.

5.6 Empirical study

To assess the performance of recursive maxima hunting, we conducted experiments on both simulated and real datasets. Firstly, we compared different configurations of RMH, considering several alternatives for calculating the correction and stopping criteria. After selecting a representative configuration, we compared the performance of RMH with the former MH and two variable selection techniques available in the FDA literature: RKVS (Berrendero, Bueno-Larraz, and Cuevas, 2019) and mRMR (Ding and Peng, 2005).

In all cases, the predictive power of the selected variables has been evaluated using three multivariate classifiers with the reduced datasets. We consider classifiers that represent different approaches to the classification problem: *Linear Discriminant Analysis* (LDA), *k-Nearest Neighbors* (k-NN) with Euclidean distance and a *Support Vector Machine* (SVM) with Gaussian kernel. LDA is a classical and simple method that, despite its simplicity, achieves good results in many real problems of low dimension and is optimal in homoscedastic problems with Gaussian processes (Berrendero, Cuevas, and Torrecilla, 2018). On the other hand, k-NN is a simple non-parametric classification rule with reasonable overall predictive accuracy that is usually used as a benchmark in the FDA literature. Finally, SVM is one of the state-of-art classifiers in machine learning literature and presents a much more flexible approach than the previous ones.

The results shown represent the accuracy obtained with each method, dataset, and classifier, averaged over 200 independent repetitions. The methodology used presents slight differences between synthetic and real data, which will be specified in the corresponding subsections. These differences are due to the absence of restrictions to generate new simulated data, which allows us, among other things, to study the effect of sample size on the variable selection. The number of nearest neighbors k, is selected by 10-fold Cross Validation (10CV) as an odd number in the range $[1, \sqrt{N_{train}}]$, where N_{train} is the number of trajectories in the training set. Likewise, the SVM's hyperparameters γ and C are chosen by 10CV from the exponential grid $\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$.

The experiments are implemented in Python, using the variable selection algorithms provided by the **scikit-fda** library (Ramos-Carreño et al., 2023). This package will be further explained in Chapter 8. Synthetic datasets were also generated using this package. Furthermore, the validation tools and classification algorithms utilized are sourced from the **scikit-learn** package (Pedregosa et al., 2011). Let us recall that the default implementation of LDA in that library uses a singular value decomposition (SVD) solver, in which dimensions whose singular values are small are discarded. This mitigates the poor results that one would expect when applying LDA to functional data, whose covariance operator is non-invertible.

Synthetic data

We have considered different variants of the general homoscedastic model described in Equation (5.3) with $\mathbb{P}(Y=0)=\mathbb{P}(Y=1)=1/2$, using the standard Brownian motion B as the noise process for Z,

$$\begin{cases} \mathbb{P}_0 : B(t) & , \quad t \in [0,1], \\ \mathbb{P}_1 : \mu(t) + B(t) & , \quad t \in [0,1], \end{cases} \tag{5.35}$$

where μ denotes the deterministic class-1 mean function, which can be also interpreted as the difference between the means of the two classes. In addition to the importance of Brownian motion in modeling and simulation in various fields of science and engineering, the study of these models offers several advantages. The first and primary advantage is the possibility of obtaining the explicit expression of the optimal classification rule, or Bayes rule (g^*), and the associated Bayes error L^* when the probability measures are equivalent (Mörters and Peres, 2010; Baíllo, Cuevas, and Cuesta-Albertos, 2011). Following Berrendero, Cuevas, and Torrecilla, 2018 we can obtain an expression for this rule in terms of an stochastic integral,

$$g^*(x) = \mathbb{I}_{\{\eta^*(x)>0\}}, \tag{5.36}$$

with

$$\eta^*(x) = \int_0^1 \mu' dB - \frac{1}{2}\|\mu'\|^2, \tag{5.37}$$

where μ' denotes the derivative of the mean function. The associated Bayes error is

$$L^* = 1 - \Phi\left(\frac{\|\mu'\|}{2}\right) \tag{5.38}$$

where $\Phi(\cdot)$ is the cumulative distribution function of a standard normal (further details of these derivations are given in Appendix B.2). In this way, we can control the difficulty of the problems and evaluate the performance of methods accurately. We can even construct problems in which the optimal solution depends on a finite number of variables $g^*(X) = g^*(X(t_1), \ldots, X(t_D))$, and therefore, the optimal solution involves variable selection (Berrendero, Cuevas, and Torrecilla, 2018). Thus, we have defined two problems in which the Bayes rule depends only on a few variables (*peak* and *peak2*), and two others in which it depends on the complete trajectories (*square* and *sine*), which should not favor variable selection. The models differ in the specification of the mean function μ:

- *peak*: $\mu(t) = 2\Phi_{3,3}(t)$, where

$$\Phi_{m,k}(t) = \int_0^t \sqrt{2^{m-1}}\left[\mathbb{I}_{\left(\frac{2k-2}{2^m},\frac{2k-1}{2^m}\right)} - \mathbb{I}_{\left(\frac{2k-1}{2^m},\frac{2k}{2^m}\right)}\right], \quad m,k \in \mathbb{N}, 1 \le k \le 2^{m-1}. \tag{5.39}$$

  This corresponds to a piecewise linear function that forms a "peak" between $t = 1/2$ and $t = 3/4$, with its maximum at $t = 5/8$. The Bayes rule depends only on the values of the trajectories at these three points,

$$\eta^*(X) = 2X\left(\frac{5}{8}\right) - X\left(\frac{1}{2}\right) - X\left(\frac{3}{4}\right) - \frac{1}{2}, \tag{5.40}$$

  and the associated error is $L^* \simeq 0.1587$.

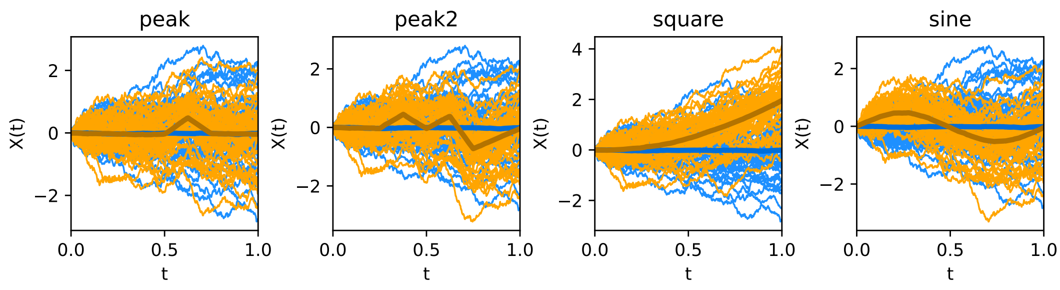


FIGURE 5.6: Some sample trajectories and class means (thick lines) of the synthetic problems.

- *peak2*: $\mu(t) = 2\Phi_{3,2}(t) + 3\Phi_{3,3}(t) - 2\Phi_{2,2}(t)$, a piecewise linear function. The optimal rule depends on six variables through

$$\begin{aligned}\eta^*(X) = 4X\left(\frac{3}{8}\right) - 2X\left(\frac{1}{4}\right) - \left(5-\sqrt{2}\right)X\left(\frac{1}{2}\right) \\ + 6X\left(\frac{5}{8}\right) - \left(3+2\sqrt{2}\right)X\left(\frac{3}{4}\right) + \sqrt{2}X(1) - \frac{17}{4},\end{aligned} \tag{5.41}$$

  and $L^* \simeq 0.0196$.

- *square*: $\mu(t) = 2t^2$. In this case, the Bayes rule depends on the whole trajectory,

$$\eta^*(X) = X(1) - \int_0^1 X(t)dt - \frac{2}{3}, \tag{5.42}$$

  with $L^* \simeq 0.1241$.

- *sine*: $\mu(t) = \frac{1}{2}\sin(2\pi t)$. Again, the optimal rule depends on the complete trajectory,

$$\eta^*(X) = X(1) + 2\int_0^1 \sin(2\pi t)X(t)dt - \frac{\pi}{4}. \tag{5.43}$$

  In this case, the Bayes error is $L^* \simeq 0.1333$.

Complete calculations of the corresponding Bayes rules and errors can be found in Appendix B.2. Examples of trajectories for each of these models are shown in Figure 5.6.

Finally, the flexibility of the simulation setting allows us to study the effect of the sample size in the performance of the methods. For each dataset, we generate incremental training sets with sizes $N_{train} = 50$, 100, 200, 500, and 1000, and independent test sets of size $N_{test} = 1000$. The trajectories were discretized in a grid of 200 regularly-spaced points. The reported accuracy values are the average of 200 independent repetitions.

Experiments on real-world data

We utilized a variety of real datasets frequently employed in functional data classification studies, chosen to represent diverse data types without striving for exhaustiveness. The datasets employed include *Phoneme*, *Wheat*, and *Australian*, which

	Australian	Berkeley	Cell	ECG	MCO	Medflies	NO_x	Phoneme	Tecator	Wheat
#Class 0	43	39	46	520	45	246	76	695	138	41
#Class 1	147	54	44	1506	44	266	39	1022	77	59
#Variables	365	31	18	85	360	30	24	50	100	701

TABLE 5.1: Number of samples per class and number of points per curve in the real-data examples.

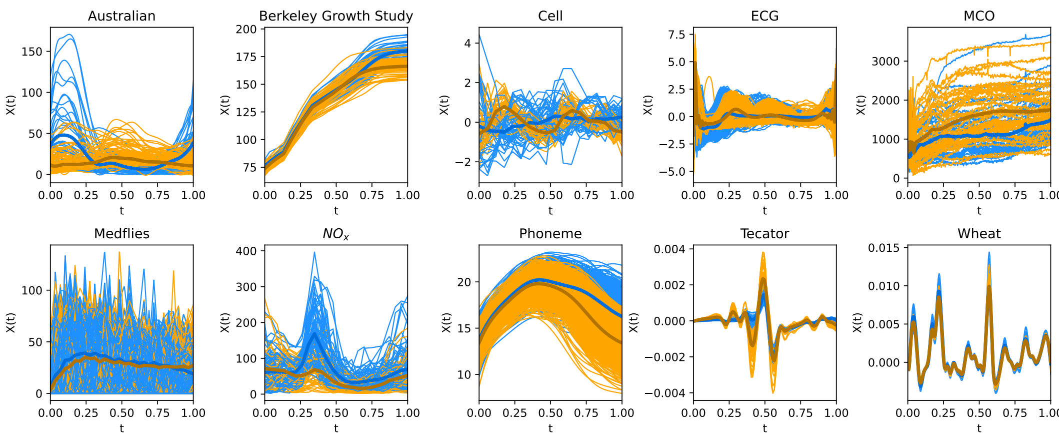


FIGURE 5.7: Trajectories of the real classification problems considered.

have been previously used in studies such as Delaigle, Hall, and Bathia, 2012 and Berrendero, Cuevas, and Torrecilla, 2018. We followed these studies by smoothing and truncating the *Phoneme* trajectories to the first 50 variables, and differentiating the *Wheat* data. In addition, we follow Delaigle and Hall, 2010 and discard the observation 190, which corresponds to a known outlier in the *Wheat* dataset. We have also included *Cell* (Leng and Müller, 2006; Rincón Hidalgo and Ruiz-Medina, 2012), NO_x (Febrero, Galeano, and González-Manteiga, 2008; Sguera, Galeano, and Lillo, 2016), *Berkeley*, *ECG*, *MCO*, *Medflies* and *Tecator* datasets (Ferraty and Vieu, 2006). For the *Medflies* dataset, we removed atypical constant zero observations. As usual, we consider the second derivatives of the *Tecator* dataset. The datasets are shown in Figure 5.7, and Table 5.1 summarizes the most relevant information about the datasets used.

All datasets were discretized to points in the interval $[0, 1]$ with a uniform separation. A random stratified split was performed to create a *train set* comprising $\frac{2}{3}$ of the observations and a *test set* comprising $\frac{1}{3}$ of the observations. This partitioning process was repeated 200 times, and the reported accuracies are averages over these different partitions.

RMH setup

The objective of this section is to propose a default configuration of RMH that exhibits good overall performance across all datasets. This configuration will serve as a reference for comparison with other variable selection methods in the next section. To this end, we discuss the election of the relevance measure and other implementation details, and evaluate the performance of the RMH algorithm with several types of corrections and stopping conditions. In the comparison, we consider two main criteria: the accuracy of the posterior prediction and the ability to select a small number of variables.

Relevance measure. We will employ the squared distance covariance $\mathcal{V}^2$ (Székely, Rizzo, and Bakirov, 2007) to calculate the relevance curve $\mathrm{Rel}(X(t), Y)$. This dependence measure has been shown to possess some optimality properties when used in combination with RMH, as stated in Theorem 4. Furthermore, Berrendero, Cuevas, and Torrecilla, 2016b, Th.2 proves the uniform convergence of the estimation $\mathcal{V}_N^2(X(t), Y)$ with respect to t, ensuring the convergence of the empirical local maxima to the true ones. Distance covariance has already been successfully utilized for variable selection in various algorithms, including MH (Li, Zhong, and Zhu, 2012; Kong, Wang, and Wahba, 2015; Berrendero, Cuevas, and Torrecilla, 2016a; Berrendero, Cuevas, and Torrecilla, 2016b), and has a simple estimator $\mathcal{V}_N^2$ that can be efficiently computed (Huo and Székely, 2016). Here we use the fast implementation provided by **dcor** library (Ramos-Carreño and Torrecilla, 2023), presented in Chapter 9.

Correction. We compare the performance of several correction methods assuming a known covariance structure for the noise process Z. The corrections under comparison include the standard Brownian motion (B) described in Section 5.5.2 (as in Torrecilla and Suárez, 2016), the Ornstein-Uhlenbeck process (OU) described in Section 5.5.3, the uniform Brownian correction (UB) defined in Theorem 5, and the radial basis function (RBF) (Rasmussen and Williams, 2005). While assuming a known covariance structure simplifies estimation, it can lead to accuracy loss if the data deviates from the assumptions. In addition, we also consider a method that estimates the correction from (5.8) using sample estimators for the covariances, (S). This proposal assumes that Z is a zero-mean Gaussian process, without any further structural assumptions. The *lengthscale* and *variance* parameters of OU and RBF kernels are estimated by maximum likelihood from the training set using the package *GPy* (GPy, 2012). Note that B and UB do not present hyperparameters by definition.

Stopping criterion. Choosing a stopping criterion for variable selection algorithms is an open problem in the literature that does not have a unique solution. One common strategy is to set the number of variables through cross-validation, although this method involves the classifier in a supposedly model-free filter selection method. Another typical approach is to establish a minimum relevance threshold so that RMH stops when the relevance of the new variable to be selected does not exceed this threshold. Nonetheless, this also introduces a new parameter that requires tuning. By chance, the properties of distance covariance for characterizing independence allow us to define a new stopping criterion that is completely data-driven. Specifically, before selecting the variable $X^{[i]}(t_i)$, we conduct a test of independence with Y (Székely and Rizzo, 2009). RMH stops if this independence cannot be rejected. The rejection condition of the test in the i-th iteration is given by

$$N\frac{\mathcal{V}_N^2(X^{[i]}(t_i), Y)}{T_N^2(X^{[i]}(t_i), Y)} \geq \chi^2_{1,1-\alpha}, \tag{5.44}$$

where N is the sample size, $\mathcal{V}_N^2$ is the sample estimator for $\mathcal{V}^2$, $\chi^2_{1,1-\alpha}$ denotes the $(1-\alpha)$ quantile of the Chi-square distribution with 1 degree of freedom, and

$$T_N^2(X^{[i]}(t_i), Y) = \frac{1}{N^2}\sum_{j,k=1}^{N} |X_j^{[i]}(t_i) - X_k^{[i]}(t_i)| \frac{1}{N^2}\sum_{j,k=1}^{N} |Y_j - Y_k|.$$

This new proposal also involves tuning a parameter, the significance level α. This parameter should be low enough to avoid excluding variables that are not very relevant marginally but are relevant in combination with others, and high enough to avoid including irrelevant variables. Exploratory analysis showed very similar results for the standard values of α considered in the literature. Therefore, in the remaining experiments, we set $\alpha = 0.01$.

Additional experiments showed that this new proposal performs similarly to the CV-based approach, with the advantage of selecting fewer variables and achieving better accuracy rates in most cases. The only exception is the RBF kernel where CV slightly outperforms the independence criterion by selecting more variables. Furthermore, the proposed independence-based criterion is able to select the exact number of variables involved in the optimal rule in *peak* and *peak2* problems. Lastly, the new proposal has a lower computational cost as the significance level can be pre-fixed without significant loss in performance in most cases, and maintains the model-free aspect of RMH. Hence, we will use the stopping criterion based on Equation (5.44) with $\alpha = 0.01$ in our future experiments.

min_redundancy. In practice, the successive corrections applied by the RMH algorithm may introduce undesired numerical artifacts that distort the variable selection. Particularly, after the correction step, contiguous variables may emerge as artificially relevant. This can occur due to numerical issues, such as divisions close to zero, or because the real maximum may lie between two discretization points, rendering the correction partially ineffective. To circumvent this problem, we leverage the high redundancy between nearby variables derived from the continuity of functional data. Specifically, after selecting the i-th variable $X(t_i)$, we exclude from future RMH iterations all variables $X(t)$ in a neighborhood of $X(t_i)$ whose redundancy with the selected variable exceeds a certain threshold `min_redundancy`. In other words,

$$Red\left(X(t), X(t_i)\right) \geq \texttt{min\_redundancy}. \tag{5.45}$$

Here, *Red* is a univariate measure of dependence that quantifies the redundancy between two variables. In this case, we cannot use $\mathcal{V}^2$ as a measure of redundancy since it is unbounded. Instead, we use the distance correlation $\mathcal{R}^2$, a dimensionless version of $\mathcal{V}^2$ rescaled between 0 and 1 (Székely, Rizzo, and Bakirov, 2007). Thus, to eliminate the most redundant variables, we must set a value of `min_redundancy` close to 1. Preliminary results showed very similar behavior for threshold values between 0.8 and 0.99 in most cases. However, extreme values of the parameter presented some difficulties with trajectories that were especially smooth or rough. To minimize these possible issues in extreme cases, in our experiments, we set `min_redundancy` $= 0.9$.

Comparison. Once the stopping criterion was fixed, we evaluated the different correction methods by examining the classification accuracy results obtained with LDA, k-NN, and SVM, presented in Tables 5.2, 5.3, and 5.4, respectively. For each

	RMH-RBF	**RMH-S**	**RMH-BM**	**RMH-OU**	**RMH-UB**
peak (1000)	0.715 ± 0.058(2.4)	0.834 ± 0.011(3.3)	**0.835 ± 0.011(3.1)**	**0.835 ± 0.011(3.1)**	**0.835 ± 0.011(3.2)**
peak2 (1000)	0.960 ± 0.007(7.7)	0.976 ± 0.005(11.1)	**0.978 ± 0.005(6.3)**	0.977 ± 0.005(6.3)	0.977 ± 0.005(6.3)
sine (1000)	0.835 ± 0.012(13.4)	**0.859 ± 0.011(7.9)**	**0.859 ± 0.011(7.5)**	**0.859 ± 0.011(8.7)**	**0.859 ± 0.011(8.5)**
square (1000)	**0.873 ± 0.011(10.9)**	0.872 ± 0.011(4.2)	**0.873 ± 0.010(4.0)**	0.872 ± 0.011(4.1)	**0.873 ± 0.010(4.7)**
Australian	0.902 ± 0.035(9.6)	**0.907 ± 0.032(5.5)***	0.898 ± 0.032(6.9)	0.897 ± 0.032(7.0)	0.898 ± 0.032(6.9)
Berkeley	0.884 ± 0.045(1.5)	0.953 ± 0.033(2.5)	0.955 ± 0.032(4.2)	0.954 ± 0.034(4.3)	**0.959 ± 0.030(3.8)***
Cell	0.856 ± 0.055(5.7)	**0.868 ± 0.057(1.1)**	0.861 ± 0.055(4.1)	0.862 ± 0.055(4.1)	0.854 ± 0.059(5.0)
ECG	0.974 ± 0.006(20.0)	**0.990 ± 0.004(20.0)***	0.982 ± 0.005(20.0)	0.984 ± 0.004(20.0)	0.982 ± 0.005(20.0)
MCO	0.893 ± 0.133(10.8)	0.958 ± 0.035(19.3)	0.961 ± 0.053(8.4)	**0.985 ± 0.023(9.5)**	0.983 ± 0.024(9.7)
Medflies	**0.610 ± 0.041(1.6)**	0.606 ± 0.036(1.4)	0.609 ± 0.037(1.9)	0.604 ± 0.037(1.8)	0.593 ± 0.034(2.1)
NOX	0.847 ± 0.069(4.8)	**0.904 ± 0.051(3.7)***	0.844 ± 0.064(4.0)	0.892 ± 0.055(3.9)	0.881 ± 0.057(4.7)
Phoneme	**0.819 ± 0.013(6.1)**	**0.819 ± 0.013(4.6)**	0.817 ± 0.013(6.0)	0.818 ± 0.013(6.0)	0.816 ± 0.013(5.5)
Tecator	0.934 ± 0.029(4.0)	0.941 ± 0.023(4.0)	0.933 ± 0.028(5.4)	**0.952 ± 0.023(19.1)***	0.936 ± 0.026(5.4)
Wheat	**1.000 ± 0.000(10.8)**	0.999 ± 0.006(11.6)	0.999 ± 0.005(10.6)	**1.000 ± 0.000(11.1)**	**1.000 ± 0.000(12.0)**
accuracy (average)	*0.865*	***0.892***	*0.886*	***0.892***	*0.889*
rank (average)	*3.429*	*2.500*	*2.714*	***2.286***	*2.571*
# vars (median)	*6.883*	***4.420***	*5.662*	*6.147*	*5.445*

TABLE 5.2: Comparison between RMH variants using the LDA classifier.

	RMH-RBF	**RMH-S**	**RMH-BM**	**RMH-OU**	**RMH-UB**
peak (1000)	0.682 ± 0.052(2.4)	0.810 ± 0.014(3.3)	**0.813 ± 0.013(3.1)**	**0.813 ± 0.013(3.1)**	0.811 ± 0.014(3.2)
peak2 (1000)	0.914 ± 0.010(7.7)	0.926 ± 0.019(11.1)	**0.952 ± 0.007(6.3)**	**0.952 ± 0.007(6.3)**	**0.952 ± 0.007(6.3)**
sine (1000)	0.814 ± 0.014(13.4)	0.833 ± 0.013(7.9)	**0.834 ± 0.012(7.5)**	0.828 ± 0.013(8.7)	0.827 ± 0.014(8.5)
square (1000)	0.844 ± 0.012(10.9)	0.855 ± 0.013(4.2)	**0.856 ± 0.012(4.0)**	0.853 ± 0.013(4.1)	0.852 ± 0.012(4.7)
Australian	0.941 ± 0.029(9.6)	**0.944 ± 0.026(5.5)**	0.931 ± 0.030(6.9)	0.932 ± 0.031(7.0)	0.931 ± 0.031(6.9)
Berkeley	0.868 ± 0.056(1.5)	0.929 ± 0.043(2.5)	0.943 ± 0.041(4.2)	0.941 ± 0.036(4.3)	**0.944 ± 0.037(3.8)**
Cell	0.879 ± 0.057(5.7)	0.830 ± 0.057(1.1)	0.857 ± 0.061(4.1)	0.857 ± 0.057(4.1)	**0.883 ± 0.049(5.0)**
ECG	**0.997 ± 0.002(20.0)**	0.996 ± 0.003(20.0)	**0.997 ± 0.002(20.0)**	**0.997 ± 0.003(20.0)**	**0.997 ± 0.002(20.0)**
MCO	0.860 ± 0.104(10.8)	0.889 ± 0.061(19.3)	0.896 ± 0.067(8.4)	**0.916 ± 0.053(9.5)**	0.914 ± 0.052(9.7)
Medflies	0.597 ± 0.042(1.6)	0.588 ± 0.041(1.4)	**0.603 ± 0.038(1.9)***	0.599 ± 0.039(1.8)	0.596 ± 0.037(2.1)
NOX	0.811 ± 0.059(4.8)	**0.885 ± 0.052(3.7)***	0.820 ± 0.062(4.0)	0.873 ± 0.054(3.9)	0.847 ± 0.058(4.7)
Phoneme	0.808 ± 0.014(6.1)	**0.815 ± 0.013(4.6)***	0.808 ± 0.015(6.0)	0.812 ± 0.013(6.0)	0.812 ± 0.014(5.5)
Tecator	0.970 ± 0.018(4.0)	**0.977 ± 0.021(4.0)**	0.975 ± 0.016(5.4)	0.974 ± 0.017(19.1)	0.973 ± 0.017(5.4)
Wheat	0.999 ± 0.004(10.8)	0.995 ± 0.012(11.6)	0.996 ± 0.013(10.6)	**1.000 ± 0.004(11.1)**	0.999 ± 0.007(12.0)
accuracy (average)	*0.856*	*0.877*	*0.877*	***0.882***	*0.881*
rank (average)	*3.857*	*3.143*	*2.286*	***2.143***	*2.571*
# vars (median)	*6.883*	***4.420***	*5.662*	*6.147*	*5.445*

TABLE 5.3: Comparison between RMH variants using the *k*-NN classifier.

dataset (rows) and correction method (columns), the average accuracy and standard deviation over 200 repetitions are shown, along with the mean number of selected variables in parentheses. The best performing method is highlighted in bold, while the second-best is underlined. Asterisks indicate statistically significant differences between the mean accuracies, as determined by a paired t-test with a 95% confidence level. In addition, for each correction method, we provide the average accuracy, positional ranking (the lower the better), and the median number of selected variables over all the considered problems. The use of the median in the latter case is an attempt to prevent the effect of the atypical ECG dataset. We only included simulated datasets with the largest sample size $N_{train} = 1000$ to avoid biasing the results with multiple versions of the same problem.

	RMH-RBF	**RMH-S**	**RMH-BM**	**RMH-OU**	**RMH-UB**
peak (1000)	0.711 ± 0.059(2.4)	0.832 ± 0.012(3.3)	**0.833 ± 0.012(3.1)**	**0.833 ± 0.011(3.1)**	0.832 ± 0.011(3.2)
peak2 (1000)	0.957 ± 0.007(7.7)	0.974 ± 0.006(11.1)	**0.976 ± 0.005(6.3)**	0.975 ± 0.005(6.3)	0.975 ± 0.005(6.3)
sine (1000)	0.832 ± 0.013(13.4)	0.856 ± 0.011(7.9)	**0.857 ± 0.012(7.5)**	**0.857 ± 0.011(8.7)**	**0.857 ± 0.011(8.5)**
square (1000)	0.870 ± 0.011(10.9)	0.870 ± 0.011(4.2)	**0.871 ± 0.011(4.0)**	0.870 ± 0.011(4.1)	**0.871 ± 0.011(4.7)**
Australian	0.942 ± 0.028(9.6)	**0.947 ± 0.029(5.5)***	0.920 ± 0.032(6.9)	0.920 ± 0.030(7.0)	0.919 ± 0.031(6.9)
Berkeley	0.869 ± 0.055(1.5)	0.940 ± 0.035(2.5)	**0.950 ± 0.035(4.2)***	0.946 ± 0.032(4.3)	0.946 ± 0.033(3.8)
Cell	**0.868 ± 0.058(5.7)**	0.861 ± 0.061(1.1)	0.850 ± 0.062(4.1)	0.855 ± 0.062(4.1)	0.865 ± 0.055(5.0)
ECG	0.998 ± 0.002(20.0)	0.998 ± 0.002(20.0)	0.998 ± 0.002(20.0)	**0.999 ± 0.002(20.0)**	0.998 ± 0.002(20.0)
MCO	0.901 ± 0.125(10.8)	0.975 ± 0.029(19.3)	0.948 ± 0.062(8.4)	**0.977 ± 0.032(9.5)**	**0.977 ± 0.033(9.7)**
Medflies	0.614 ± 0.042(1.6)	0.617 ± 0.038(1.4)	**0.623 ± 0.036(1.9)***	0.619 ± 0.038(1.8)	0.618 ± 0.037(2.1)
NOX	0.850 ± 0.072(4.8)	**0.903 ± 0.051(3.7)***	0.829 ± 0.064(4.0)	0.876 ± 0.058(3.9)	0.887 ± 0.053(4.7)
Phoneme	**0.817 ± 0.013(6.1)**	**0.817 ± 0.013(4.6)**	0.816 ± 0.014(6.0)	**0.817 ± 0.014(6.0)**	0.816 ± 0.014(5.5)
Tecator	0.982 ± 0.014(4.0)	**0.983 ± 0.017(4.0)**	0.978 ± 0.016(5.4)	0.980 ± 0.016(19.1)	0.979 ± 0.015(5.4)
Wheat	0.998 ± 0.008(10.8)	0.996 ± 0.011(11.6)	0.997 ± 0.010(10.6)	0.999 ± 0.005(11.1)	**1.000 ± 0.003(12.0)**
accuracy (average)	*0.872*	***0.898***	*0.889*	*0.895*	*0.896*
rank (average)	*3.429*	*2.786*	*2.714*	***2.071***	*2.357*
# vars (median)	*6.883*	***4.420***	*5.662*	*6.147*	*5.445*

TABLE 5.4: Comparison between RMH variants using the SVM classifier.

The results indicate that all proposed corrections exhibit similar overall performance, with the exception of the RBF method, which is clearly outperformed by the other methods (although it may perform well in specific cases). The algorithms' behavior is relatively stable across the three classifiers, with slightly lower average accuracy levels observed for the k-NN classifier. This generalization capacity is expected from the variables selected with filter methods, although it is not always achieved in practice. In this sense, it is worth noting that the number of selected variables is independent of the classifier due to the use of the stopping criterion based on the independence test.

On average, the OU correction obtains the best results, ranking first in all three classifiers, although it tends to select one or two more variables than its direct competitors, RMH-UB and RMH-S. RMH-UB behaves almost identically to RMH-OU for all datasets, selecting fewer variables and without requiring any parameter tuning. This presents several advantages in terms of interpretability and computational cost, albeit with a slight decrease in accuracy in some cases. On the other hand, RMH-S, which estimates the covariance function directly from the training sample, also performs very well in terms of accuracy by selecting the smallest number of variables. However, it is more irregular and exhibits both the most statistically significant victories and the worst performance in some problems (excluding RBF). Although the Brownian correction is the clear winner in synthetic datasets, it loses accuracy when faced with real-world problems that do not perfectly fit the model. Nevertheless, it achieves good results with a similar number of variables as RMH-UB.

Based on the results, the OU, UB, or S corrections could all be good default options for RMH. However, we choose the uniform Brownian correction due to its good theoretical properties and its ability to achieve similar accuracy rates to the best alternatives with a comparable number of variables, without the need to estimate any parameters from the sample.

Comparison between variable selection methods

The variable selection methods to be compared are as follows:

- *Recursive maxima hunting* (*RMH*): Based on the conclusions of the previous section, the representative of RMH methods in the comparison uses the uniform Brownian correction. We also use the distance covariance $\mathcal{V}^2$ and the distance correlation $\mathcal{R}^2$ to calculate relevance with the class and redundancy between variables, respectively. In both cases, we use the fast versions available in the **dcor** library (Ramos-Carreño and Torrecilla, 2023), presented in Chapter 9. The stopping criterion is based on the independence contrast defined by (5.44) with $\alpha = 0.01$, and `min_redundancy` is fixed to 0.9.

- *Maxima hunting* (*MH*): The variable selection method described in Section 5.2. We follow the implementation guidelines given in Berrendero, Cuevas, and Torrecilla, 2016b, employing the *distance covariance* as the measure for relevance. To identify local maxima, the smoothing parameter h is selected via 10-fold cross-validation from the set of values $\{1, 3, 5, 7, 10\}$. Like RMH, MH does not require tuning the number of variables since it returns as many variables as there are local maxima (up to a maximum of 20).

- *Minimum Redundancy Maximum Relevance* (*mRMR*): mRMR is a popular multivariate feature selection method proposed by Ding and Peng, 2005 which has been successfully used with functional data (Gómez-Verdejo, Verleysen, and

Fleury, 2009; Berrendero, Cuevas, and Torrecilla, 2016a). It attempts to choose a subset of the original variables that jointly maximizes the *relevance* with the class and minimizes the *redundancy* between the selected variables. We use the MID (Mutual Information Difference), which has been reported to be more stable (Gulgezen, Cataltepe, and Yu, 2009). The MI score is estimated by binning and using the estimator available in **scikit-learn**.

- *Reproducing Kernel Variable Selection* (*RKVS*): RKVS is a filter method for variable selection recently proposed by Berrendero, Cuevas, and Torrecilla, 2018 in the context of binary functional classification (see Section 2.3.2). It aims to maximize the Mahalanobis distance between the (multivariate) means of the classes resulting after variable selection. In practice, RKVS has shown excellent performance and presents optimality properties for the classification problem of homoscedastic Gaussian processes if LDA is applied after variable selection.

- *Base*: Finally, we introduce a benchmark approach that classifies the complete original trajectories (discretized) without any dimensionality reduction. For this purpose, we use a functional version of k-NN with the L^2 distance. The standard version of LDA does not perform well with functional data due to the non-invertibility of the covariance operator. However, the regularization included in the implementation of **scikit-learn** enables its utilization with discretized functions. In contrast, SVM has been successfully used with this type of data in prior literature (Rossi and Villa, 2006).

For these experiments, we followed the methodology described in previous sections. Additionally, the number of variables selected for mRMR and RKVS is chosen through 10-fold cross-validation, up to a maximum of 20 variables. The selected variables are standardized to have zero-mean and variance equals to 1 before classification using LDA, k-NN, and SVM. The variable selection methods and functional k-NN are implemented using **scikit-fda** (Ramos-Carreño et al., 2023), while the multivariate classification algorithms are from **scikit-learn** (Pedregosa et al., 2011).

In this section, we study the impact of sample size on the performance of feature selection methods using synthetic datasets. Figure 5.8 shows the results obtained with SVM. The top row displays the mean error evolution (averaged over 200 independent repetitions) of the algorithms for different sample sizes $N_{train} = 50, 100, 200,$ 500, and 1000. The horizontal dashed lines indicate the Bayes error for each problem. The bottom row plots show the average numbers of variables selected by each method for each sample size. In this case, *Base* is excluded from the graphs since it always uses the 200 variables of the problems, making it impossible to appreciate the differences between the other methods.

RMH stands out as the overall winner in terms of classification error and number of selected variables (with a data-driven stopping criterion), followed closely by RKVS, which performs slightly worse in *peak2* and *square* for small sample sizes. As expected, both methods exhibit excellent performance on the first two datasets, where they are able to identify the variables on which the optimal rule depends. However, they also obtain the best results on *square* and *sine*, which are theoretically unfavorable for variable selection since the optimal classifier involves the complete trajectories. At the other end of the spectrum is mRMR, which performs poorly in terms of both accuracy and the number of variables selected. MH exhibits somewhat irregular behavior. It is close to the performance of the *Base* method on the first two datasets, works quite well on *square*, and is the worst-performing method on *sine* asymptotically. Finally, it can be observed that the *Base* approach is only competitive

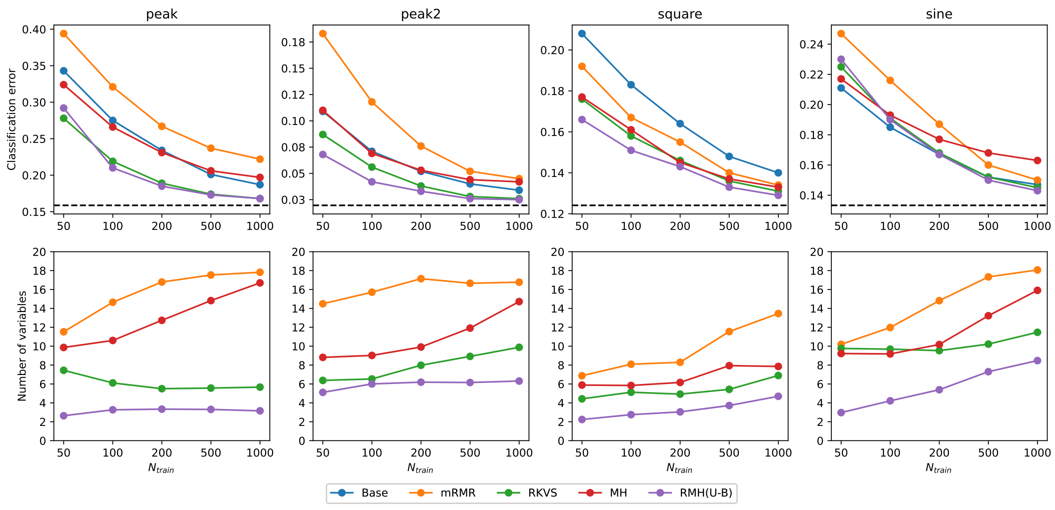


FIGURE 5.8: The results of the experiments on synthetic classification problems, using the SVM classifier. On the first row, the error of each method for each dataset is plotted against the size of the train set. Bayes error rate is shown with a dashed line. In the second row, the number of variables after the variable selection step is shown is plotted against the size of the train set.

in *sine* (although it is eventually surpassed by RMH and RKVS) and is clearly outperformed on the datasets where variable selection is optimal, as well as on *square*, which seems to be able to be explained very well with the values of the trajectories at a few points.

The results obtained with LDA and k-NN are very similar to those obtained with SVM. The main difference is a deterioration in the performance of *Base* (specially for low sample sizes), since LDA is not the best option for functional data even with regularization, and k-NN requires more observations to obtain good results. In addition to this, there is a slight synergy between LDA and RMH, and between k-NN and RKHS.

Now we compare the performance variable selection with all considered datasets. The accuracy results obtained with LDA, k-NN, and SVM, are presented in Tables 5.5, 5.6, and 5.7, respectively. As in the previous section, for each dataset (rows) and correction method (columns), the average accuracy and standard deviation over 200 repetitions are shown, along with the mean number of selected variables in parentheses. The best performing method is highlighted in bold, while the second-best is underlined. Asterisks indicate statistically significant differences between the mean accuracies, as determined by using a paired t-test with a 95% confidence level. In addition, for each correction method, we provide the average accuracy, positional ranking (the lower the better), and the median number of selected variables over all the considered problems. The use of the median in the latter case is an attempt to prevent the effect of the atypical datasets. We only included simulated datasets with the largest sample size $N_{train} = 1000$ to avoid biasing the results with multiple versions of the same problem.

The results are consistent with those observed in the synthetic data. RMH and RKVS remain the best algorithms in terms of accuracy, and along with MH, they are also the ones that use the least variables, with RMH having a slightly lower average

	Base	**mRMR**	**RKVS**	**MH**	**RMH-UB**
peak	0.790 ± 0.014(200.0)	0.785 ± 0.019(18.4)	0.833 ± 0.012(5.8)	0.808 ± 0.018(16.8)	**0.835 ± 0.011(3.2)***
peak2	0.962 ± 0.007(200.0)	0.959 ± 0.008(17.6)	**0.977 ± 0.005(9.2)**	0.961 ± 0.010(15.1)	**0.977 ± 0.005(6.3)**
sine	0.822 ± 0.013(200.0)	0.852 ± 0.014(17.9)	0.856 ± 0.011(9.5)	0.837 ± 0.022(15.6)	**0.859 ± 0.011(8.5)***
square	0.833 ± 0.013(200.0)	0.870 ± 0.010(13.5)	0.870 ± 0.011(6.1)	0.869 ± 0.011(8.1)	**0.873 ± 0.010(4.7)***
Australian	0.896 ± 0.034(365.0)	**0.901 ± 0.032(8.3)***	0.897 ± 0.034(6.1)	0.879 ± 0.028(3.3)	0.898 ± 0.032(6.9)
Berkeley	0.865 ± 0.062(31.0)	0.952 ± 0.034(4.9)	0.940 ± 0.037(4.4)	0.944 ± 0.044(3.9)	**0.959 ± 0.030(3.8)***
Cell	0.823 ± 0.065(18.0)	0.841 ± 0.062(6.8)	0.842 ± 0.057(4.4)	**0.856 ± 0.055(2.5)**	0.854 ± 0.059(5.0)
ECG	**0.994 ± 0.003(85.0)***	0.979 ± 0.006(19.4)	0.990 ± 0.004(17.8)	0.941 ± 0.022(13.0)	0.982 ± 0.005(20.0)
MCO	**0.983 ± 0.022(360.0)**	0.977 ± 0.036(11.7)	0.972 ± 0.030(6.3)	0.915 ± 0.066(13.7)	**0.983 ± 0.024(9.7)**
Medflies	0.581 ± 0.033(30.0)	**0.606 ± 0.031(4.6)**	0.600 ± 0.036(4.2)	0.602 ± 0.031(3.4)	0.593 ± 0.034(2.1)
NOX	0.852 ± 0.051(24.0)	0.880 ± 0.052(10.7)	**0.915 ± 0.044(3.6)***	0.861 ± 0.061(4.3)	0.881 ± 0.057(4.7)
Phoneme	**0.822 ± 0.013(50.0)***	0.819 ± 0.014(12.7)	0.819 ± 0.013(10.3)	0.817 ± 0.014(2.4)	0.816 ± 0.013(5.5)
Tecator	0.941 ± 0.027(100.0)	0.942 ± 0.025(12.7)	**0.943 ± 0.025(12.0)**	0.936 ± 0.024(6.7)	0.936 ± 0.026(5.4)
Wheat	0.961 ± 0.041(701.0)	0.996 ± 0.020(1.3)	**1.000 ± 0.000(1.0)**	**1.000 ± 0.000(2.2)**	**1.000 ± 0.000(12.0)**
accuracy (average)	*0.866*	*0.883*	***0.889***	*0.873*	***0.889***
rank (average)	*3.714*	*2.929*	*2.214*	*3.500*	***2.071***
# vars (median)	*150.000*	*12.180*	*6.088*	*5.498*	***5.445***

TABLE 5.5: Comparison between variable selection methods using the LDA classifier.

	Base	**mRMR**	**RKVS**	**MH**	**RMH-UB**
peak	0.713 ± 0.015(200.0)	0.642 ± 0.020(13.7)	**0.813 ± 0.013(3.5)***	0.709 ± 0.033(6.0)	0.811 ± 0.014(3.2)
peak2	0.910 ± 0.010(200.0)	0.874 ± 0.014(14.4)	**0.952 ± 0.008(6.3)**	0.930 ± 0.010(3.7)	**0.952 ± 0.007(6.3)**
sine	**0.837 ± 0.012(200.0)***	0.802 ± 0.015(16.5)	0.832 ± 0.014(8.1)	0.821 ± 0.017(14.1)	0.827 ± 0.014(8.5)
square	0.848 ± 0.012(200.0)	0.841 ± 0.012(10.1)	**0.859 ± 0.011(3.0)**	0.858 ± 0.012(3.4)	0.852 ± 0.012(4.7)
Australian	0.941 ± 0.027(365.0)	**0.942 ± 0.031(7.0)**	0.938 ± 0.028(7.4)	0.914 ± 0.031(3.5)	0.931 ± 0.031(6.9)
Berkeley	**0.952 ± 0.032(31.0)***	0.939 ± 0.037(6.3)	0.935 ± 0.041(6.7)	0.909 ± 0.056(3.9)	0.944 ± 0.037(3.8)
Cell	**0.932 ± 0.042(18.0)***	0.912 ± 0.052(10.6)	0.906 ± 0.055(12.6)	0.871 ± 0.054(4.0)	0.883 ± 0.049(5.0)
ECG	**0.998 ± 0.002(85.0)***	0.995 ± 0.003(12.2)	0.997 ± 0.002(12.7)	0.993 ± 0.003(10.5)	0.997 ± 0.002(20.0)
MCO	0.814 ± 0.072(360.0)	0.907 ± 0.059(11.3)	0.908 ± 0.056(4.6)	0.857 ± 0.066(11.4)	**0.914 ± 0.052(9.7)**
Medflies	0.550 ± 0.033(30.0)	**0.601 ± 0.041(4.1)**	0.594 ± 0.037(5.2)	0.590 ± 0.035(4.1)	0.596 ± 0.037(2.1)
NOX	0.882 ± 0.051(24.0)	0.843 ± 0.058(10.2)	**0.884 ± 0.043(4.1)**	0.830 ± 0.057(3.8)	0.847 ± 0.058(4.7)
Phoneme	**0.816 ± 0.014(50.0)***	0.810 ± 0.015(9.0)	0.814 ± 0.014(8.1)	0.803 ± 0.014(2.6)	0.812 ± 0.014(5.5)
Tecator	0.977 ± 0.016(100.0)	0.974 ± 0.020(3.8)	0.974 ± 0.020(4.7)	**0.990 ± 0.012(1.2)***	0.973 ± 0.017(5.4)
Wheat	0.956 ± 0.038(701.0)	0.996 ± 0.018(1.2)	**1.000 ± 0.000(1.0)**	**1.000 ± 0.003(2.2)**	0.999 ± 0.007(12.0)
accuracy (average)	*0.866*	*0.863*	***0.886***	*0.862*	*0.881*
rank (average)	*2.643*	*3.500*	***2.071***	*3.786*	*2.714*
# vars (median)	*150.000*	*10.143*	*5.732*	***3.828***	*5.445*

TABLE 5.6: Comparison between variable selection methods using the *k*-NN classifier.

	Base	**mRMR**	**RKVS**	**MH**	**RMH-UB**
peak	0.819 ± 0.012(200.0)	0.778 ± 0.020(17.8)	**0.832 ± 0.012(5.7)**	0.803 ± 0.019(16.7)	**0.832 ± 0.011(3.2)**
peak2	0.970 ± 0.006(200.0)	0.955 ± 0.008(16.8)	0.974 ± 0.006(9.9)	0.958 ± 0.010(14.7)	**0.975 ± 0.005(6.3)***
sine	**0.857 ± 0.012(200.0)**	0.850 ± 0.014(18.1)	0.855 ± 0.012(11.5)	0.837 ± 0.021(15.9)	**0.857 ± 0.011(8.5)**
square	0.865 ± 0.011(200.0)	0.866 ± 0.011(13.5)	0.869 ± 0.011(6.9)	0.867 ± 0.012(7.9)	**0.871 ± 0.011(4.7)***
Australian	0.846 ± 0.025(365.0)	0.938 ± 0.027(9.7)	**0.946 ± 0.029(9.0)***	0.927 ± 0.029(3.3)	0.919 ± 0.031(6.9)
Berkeley	**0.953 ± 0.031(31.0)***	0.946 ± 0.037(5.7)	0.939 ± 0.038(5.9)	0.931 ± 0.047(4.1)	0.946 ± 0.033(3.8)
Cell	**0.917 ± 0.043(18.0)***	0.890 ± 0.055(10.6)	0.895 ± 0.053(11.8)	0.853 ± 0.063(4.1)	0.865 ± 0.055(5.0)
ECG	**0.998 ± 0.002(85.0)**	0.996 ± 0.003(15.2)	0.997 ± 0.002(15.0)	0.994 ± 0.003(10.9)	**0.998 ± 0.002(20.0)**
MCO	0.500 ± 0.000(360.0)	0.966 ± 0.037(11.0)	0.964 ± 0.037(5.4)	0.905 ± 0.073(13.2)	**0.977 ± 0.033(9.7)***
Medflies	0.515 ± 0.015(30.0)	0.615 ± 0.037(4.8)	0.609 ± 0.034(5.3)	0.607 ± 0.039(4.5)	**0.618 ± 0.037(2.1)**
NOX	0.667 ± 0.000(24.0)	0.887 ± 0.051(12.8)	**0.902 ± 0.043(6.0)***	0.852 ± 0.069(4.2)	0.887 ± 0.053(4.7)
Phoneme	**0.819 ± 0.014(50.0)**	0.818 ± 0.014(12.8)	**0.819 ± 0.013(10.6)**	0.813 ± 0.015(2.6)	0.816 ± 0.014(5.5)
Tecator	0.983 ± 0.015(100.0)	0.978 ± 0.014(5.2)	0.978 ± 0.019(7.1)	**0.990 ± 0.011(2.0)***	0.979 ± 0.015(5.4)
Wheat	**1.000 ± 0.000(701.0)**	0.992 ± 0.031(1.2)	0.993 ± 0.012(1.0)	0.997 ± 0.009(2.1)	**1.000 ± 0.003(12.0)**
accuracy (average)	*0.836*	*0.891*	***0.898***	*0.881*	*0.896*
rank (average)	*2.786*	*3.357*	*2.429*	*3.929*	***1.929***
# vars (median)	*150.000*	*11.905*	*6.982*	***4.362***	*5.445*

TABLE 5.7: Comparison between variable selection methods using the SVM classifier.

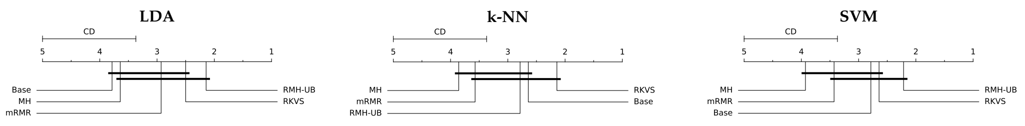


FIGURE 5.9: Critical difference (CD) diagrams comparing the results of the variable selection methods for the different classifiers. From left to right, the CD diagrams of the LDA, *k*-NN and SVM classifiers.

number of variables selected. Again, it is observed that, in general, mRMR and MH are largely outperformed by the other proposals. However, when real data is taken into consideration, mRMR outperforms MH, although MH selects the least number of variables of all proposals.

The best results are obtained by combining RMH or RKVS with LDA, which are very close to those obtained with SVM. In both scenarios, the average accuracy between RMH and RKVS is almost identical, but RMH gets better ranking results (achieving more significant victories) and selects fewer variables. When using *k*-NN, the accuracy rates are lower for all variable selection methods, and the decrease is slightly higher for RMH than for RKVS. In all three tables, variable selection techniques are superior to the *Base* approach, with better, more stable, and more interpretable results. The best results from the *Base* approach are obtained with functional *k*-NN, surpassing mRMR and MH in average accuracy, and ranking second only behind RKVS and very close to RMH. This better performance of *k*-NN makes sense since it is the only classifier fully adapted to the functional context, confirming the importance of considering the functional nature of the data in classification models. However, its performance is still very irregular in the different datasets (many significant victories and many last positions), and it does not have the advantages of dimensionality reduction and interpretability of variable selection approaches.

A different summary of these results can be seen in the critical distance (CD) diagrams presented in Figure 5.9. In these diagrams, the variable selection methods are sorted based on their average rank in the comparison, and are connected by a thick dark line when the distance between their ranks is less than the critical distance, indicating that the difference may not be significant (Demšar, 2006). From these diagrams, it is evident that RMH significantly outperforms the original MH method, which was one of the main goals of developing RMH.

5.7 Conclusions

In this chapter we have presented recursive maxima hunting (RMH), a novel feature selection algorithm in the context of binary classification with functional data. This algorithm takes into account the functional nature of the data. The method improves on the ideas outlined in the maxima hunting algorithm (Berrendero, Cuevas, and Torrecilla, 2016b). As in maxima hunting, RMH computes a dependency measure between the functional data and the class labels. However, instead of selecting the local maxima, as in maxima hunting, RMH selects the global maximum. Then, the functional data is corrected to subtract the influence of the selected point, and it proceeds iteratively, until no more relevant features are found. This procedure improves over maxima hunting in several ways. First, there is no need to estimate which points are local maxima. That was not an easy task, as in that case the data should be smoothed to prevent false local maxima to appear because the observations are noisy. As RMH only selects the global maxima, this problem is avoided. More importantly, it was shown that maxima hunting was not capable to select points that are not relevant by themselves, but are relevant once another points are selected. The iterative nature of RMH reveals those points, and this provides its biggest advantage over maxima hunting.

We have proved that RMH is optimal in an important family of problems. Specifically, in homoscedastic functional binary classification problems, where the functional data comes from an Ornstein-Uhlenbeck or Brownian process, with a different mean for each class, and where the mean difference is a finite linear combination of the covariance function of this process evaluated at some point. In that case, RMH selects just the points that appear in the Bayes rule, and thus the best possible accuracy can still be achieved after the variable selection step.

Moreover, we have tested RMH against other variable selection methods with real-world datasets. RMH consistently selected a small number of variables in the problems investigated, and achieved a very good accuracy compared with the other methods considered. This result is more impressive when taken into account that all the tested procedures except RMH required that a parameter is estimated by cross-validation, while the version of RMH that we propose does not have that advantage. We thus propose to use RMH as a preferred feature selection method in functional data, given its remarkable properties.

Chapter 6

Fuzzy clustering of functional data

Fuzzy C-means (FCM) is a generalization of the k-means clustering algorithm in which each instance can be simultaneously assigned to different clusters with a different degree of membership (Nayak, Naik, and Behera, 2015). The algorithm was previously observed to have convergence problems in the multivariate case, when the number of desired clusters or the number of dimensions of the data are high. In this chapter we present an empirical study of FCM in clustering problems with functional data (Ramos-Carreño, 2023). The data consists of trajectories sampled from stationary Gaussian processes, in which correlations are short-ranged and decay with a characteristic lengthscale, in discretized form. In the case that the grid spacing is large relative to the lengthscale at which correlations decay, one observes that FCM applied to these types of functional data has similar convergence problems as with high dimensional multivariate data. We posit that, since the values of the process at neighboring points are approximately independent, the functional observations behave as if they were samples of a high-dimensional random vector. As the dimension of the multivariate data becomes larger, it is more difficult to find initial estimates of the cluster centers for which FCM converges to the correct solution. In contrast, when the lengthscale parameter is comparable or greater than the distance between discretization points, nearby measurements are strongly correlated, which means that the functional nature of the data becomes apparent. In this case, the convergence of FCM becomes smoother, as if it were a clustering problem in lower dimensions. This not only indicates that FCM is suitable as a clustering method for functional data, but also illustrates how functional data differs from traditional multivariate ones.

The structure of this chapter is as follows: first, in Section 6.1 we explain the FCM algorithm, and the prior results that motivate this study. Second, a qualitative study of the behavior of the method with functional data in some synthetic experiments is performed in Section 6.2. A brief mathematical analysis of the observed behavior is summarized in Section 6.3. Finally, some concluding remarks are provided in Section 6.4.

6.1 Fuzzy C-means

Fuzzy clustering algorithms assign, for each datum, a degree of membership to each of the clusters of the data. This is useful for several reasons. First, in some problems the boundaries between clusters are not crisp. This is the case, for example, when the clusters correspond to concepts that can overlap, such as in document clustering, or when there is a smooth transition between clusters, as in the classification of biological organisms. Secondly, using fuzzy models make the cost function continuous, which allows the use of optimization algorithms that utilize derivatives. These algorithms have generally better numerical convergence than the algorithms that can be used when this cost function is discontinuous.

Fuzzy C-means (FCM) is one of the most widely used fuzzy clustering algorithms. It can be used for observations in a metric space $\mathcal{X}$ with a distance function d. FCM receives as input a dataset $\mathcal{D} = \{x_i \in \mathcal{X}\}_{i=1}^{N}$ and the number C of desired clusters. In addition a ω parameter, the fuzzifier, controls the degree of fuzziness. Its minimum value $\omega = 1$ corresponds to the crisp partition of k-means. The complete pseudocode of the algorithm is

Algorithm 2 Fuzzy C-means (FCM)

```
Input:
  $\mathcal{D} = \{x_i\}_{i=1}^{N}$: Dataset.
  $d$: Distance function.
  $\omega$: Fuzzifier.
Output:
  $\{q_1, \dots, q_C\}$: List of cluster centers.
1: Initialize $\{q_i\}_{i=1}^{C}$ at random
2: repeat
3:     Compute distances:
           $d_{ij} \leftarrow d(x_i, q_j).$    (6.1)
4:     Update membership matrix:
           $U_{ij} \leftarrow \left( \frac{(d_{ij})^{\frac{2}{1-\omega}}}{\sum_{k=1}^{C} (d_{ik})^{\frac{2}{1-\omega}}} \right) \quad i = 1, \dots, N \quad j = 1, \dots, C.$    (6.2)
5:     Update cluster centers:
           $q_j \leftarrow \frac{\sum_{i=1}^{N} (U_{ij})^{\omega} x_i}{\sum_{i=1}^{N} (U_{ij})^{\omega}}.$    (6.3)
6: until the cluster centers $q_j$ converge.
7: return $\{q_1, \dots, q_C\}$.
```

The algorithm as stated optimizes the inertia function

$$J(\mathcal{D}, U, \omega) = \sum_{i=1}^{N} \sum_{j=1}^{C} (U_{ij})^{\omega} d_{ij}^{2}. \tag{6.4}$$

It was observed that for high dimensional multivariate data, the algorithm failed to converge to a good solution when the number C of clusters was big enough. In that case, some initializations of the cluster centers exhibited instead convergence towards the center of gravity of the dataset, $\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$. In Winkler, Klawonn, and Kruse, 2010, this behavior was studied with the help of synthetic datasets. Instead of initializing the cluster centers randomly, they situated them manually at a location between the center of gravity and the actual clusters. Thus each cluster center q_i is initialized as

$$q_i(\alpha) = \alpha c_i + (1 - \alpha)\hat{\mu}, \tag{6.5}$$

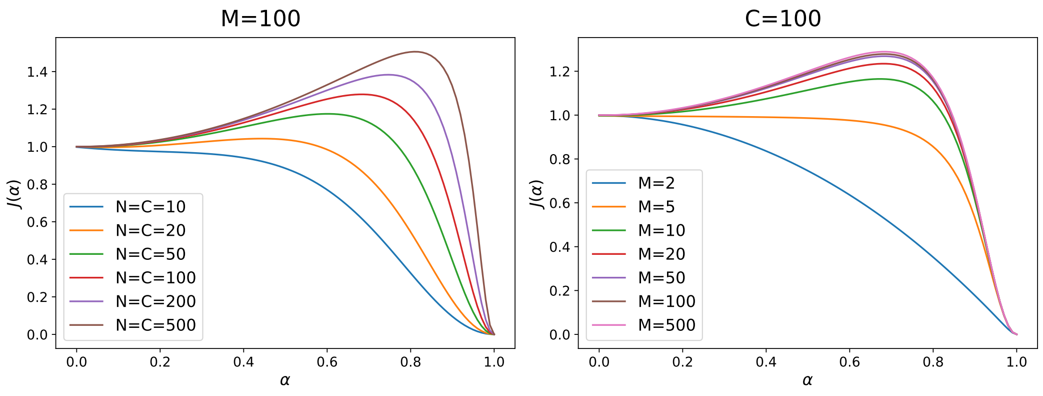


FIGURE 6.1: Plots $J(\alpha)$, the inertia function as a function of α, for a synthetic dataset (D_2), using different values of C, the number of clusters, and M, the dimension of the data. In this dataset, $C = N$, the number of observations in the sample. The cluster centers have been sampled from a standard multivariate normal distribution and then normalized. The curves of the left plot correspond to different number of clusters for a fixed dimension $M = 100$. The curves on the right plot correspond to different dimensions for a fixed number of clusters $C = 100$.

where c_i is the true location of the cluster center in the synthetic dataset and the parameter α determines the closeness of each initialization to the true cluster center. The value $\alpha = 0$ corresponds to placing all centers in the center of gravity. Setting $\alpha = 1$ corresponds to placing them in their true locations. The distances between the data and the clusters, $d_{ij}(\alpha) \quad i = 1, \dots, N, \quad j = 1, \dots, C$ depend also on this α parameter. The objective function can now be expressed as a function of α,

$$J(\mathcal{D}, \alpha) = \sum_{i=1}^{N} \sum_{j=1}^{C} \left(U_{ij}(\alpha)\right)^{\omega} \left(d_{ij}(\alpha)\right)^2 = \sum_{i=1}^{N} \sum_{j=1}^{C} \left(\frac{\left(d_{ij}(\alpha)\right)^{\frac{2}{1-\omega}}}{\sum_{k=1}^{C} \left(d_{ik}(\alpha)\right)^{\frac{2}{1-\omega}}} \right)^{\omega} d_{ij}^2(\alpha). \tag{6.6}$$

As illustrated in Figure 6.1, for multivariate data, increasing C, the number of clusters, or M, the number of dimensions of the data, results in the appearance of a local minimum for low values of α, and a global maximum in the middle. Thus, for initializations at the left side of the maximum, the algorithm converges to a degenerate configuration in which all the clusters are at the center of mass. For initializations at the right side, we have a correct convergence of the algorithm in which each cluster center converges to its true value. Note that the range of values for which the degenerate solution is obtained becomes larger as C goes to infinity.

In this chapter, we extend the study of Winkler, Klawonn, and Kruse, 2010 for functional data. In particular, we appy FCM to samples of trajectories from an Ornstein-Uhlenbeck process. As explained in 2.1.1, the Ornstein-Uhlenbeck process is a Gaussian process whose covariance function,

$$k_{OU}(s, t) = \sigma^2 \exp\left(-\frac{\|s - t\|}{l}\right), \tag{6.7}$$

depends on l, a lengthscale parameter that determines the extent of the correlations. Specifically, for $|t - s| \gg l$, the correlations are small ($\mathrm{corr}(X(s), X(t)) \ll 1$). This effect can be seen in Figure 6.2.

As shown in Section 2.1.1, when $l \to 0$, the resulting process approximates white

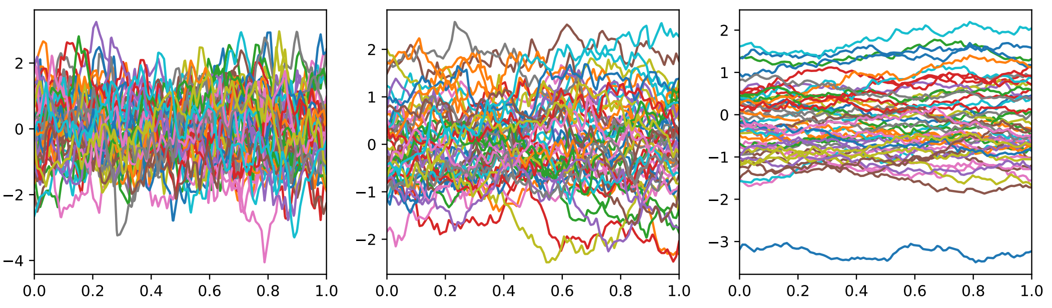


FIGURE 6.2: From left to right, plots of trajectories sampled from an Ornstein-Uhlenbeck process with lengthscales of 0.1, 1 and 10, respectively. Compare the results with those obtained in Figure 2.4, for a radial basis function (RBF) process.

noise. In this case, when the observations are discretized to M points, they can be considered as multivariate M-dimensional vectors whose coordinates are independent, as the correlation between them is 0. This case thus exhibits the same behavior as in the original study. On the contrary, when $l \to \infty$, the observations will be constant functions. In that case, as the function takes the same value in all the discretization points, the effective dimension is one. The behavior in this case is then the same as for univariate data, in which the convergence problem was not observed. Thus, it is tempting to assume that between the multivariate behavior, with $l \to 0$, in which the effective dimension of the data is M and the constant case, with $l \to \infty$, that behaves as univariate data, all other values of l will have an intermediate behavior in this problem corresponding to particular finite dimensionalities.

6.2 Study of synthetic experiments

Our goal in this section is to replicate the synthetic experiments done in Winkler, Klawonn, and Kruse, 2010 with functional data, to explore how the algorithm performs in this case. We introduce the two datasets from the original study that we are considering, as well as their extension to the functional case. The results obtained are explained after each dataset is introduced. The value of the fuzzifier parameter of FCM is fixed at $\omega = 2$ in these experiments.

In Winkler, Klawonn, and Kruse, 2010, the authors used four synthetic datasets to illustrate the shortcomings of FCM with high dimensional multivariate data. Of these, two are of particular interest, as their generalization to functional data is possible.

Dataset D_2 has one observation per cluster ($N = C$). The samples are distributed in an M-dimensional hypersphere. This can be done by generating the observations/clusters using a standard (multivariate) normal distribution and then projecting them to the unit sphere via normalization:

$$\mathbf{x}_i = \mathbf{c}_i = \frac{\mathbf{z}_i}{\|\mathbf{z}_i\|}, \quad \mathbf{z}_i \sim N(\mathbf{0}, \mathbf{I}) \quad i = 1, \dots, N. \tag{6.8}$$

In the original article, a number higher than the number of desired clusters was generated, and then k-means was used to find C well separated cluster centers. We have ommited this step, as we found that it was not necessary to achieve the same results, especially in high dimensions. This dataset presents an idealized case, in wich the

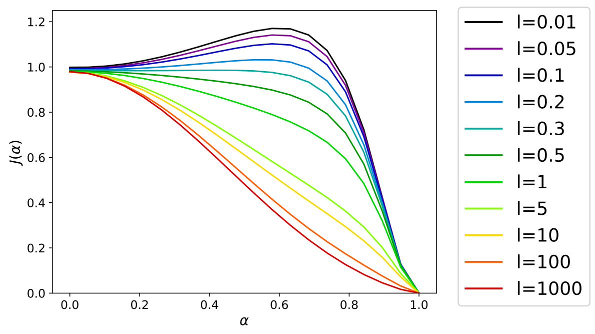


FIGURE 6.3: Inertia as a function of α, for cluster centers generated as a normalized Gaussian process with an exponential covariance function and different values for the lengthscale. The plot has $N = 50$ with one point per cluster ($N = C = 50$) The number of discretization points $M = 100$ is fixed.

clusters are compact (there is only one observation per cluster), well separated, and also far from the center of gravity of the data. Note that this center of gravity is at 0, due to the symmetrical generation of the data. The plots in Figure 6.1 explained in the previous section correspond to this model.

This dataset can be generalized to functional data by replacing the use of a multivariate normal distribution with a Gaussian process with a stationary covariance function. In particular, we have used the Ornstein-Uhlenbeck process, as mentioned before, but the same results are obtained with other stationary processes such as RBF. The lengthscale parameter l of the process controls the extent of the correlation between nearby points. As explained in Section 2.1.1, a small lengthscale would correspond to curves with low correlation, obtaining white noise in the limit towards 0. As the curves are discretized in an M-dimensional grid, this would correspond to the M-dimensional multivariate case. A larger lengthscale corresponds to smoother, more functional curves. Thus, this setting allows us to explore the transition between multivariate and functional data, and its effects in the FCM algorithm, in a smooth way.

In Figure 6.3, we can observe the evolution of the objective function of this problem by changing the lengthscale parameter l. When $l \to 0$ the same convergence problems as in the multivariate case arise. However, by increasing the lengthscale, the local minimum at the left dissapears, and the algorithm converges to the correct solution. This indicates that the algorithm works better for more functional datasets.

In Figure 6.4 we present the results obtained in this dataset D_2 by fixing the lengthscale l and varying the number of clusters, as was previously done in the left plot of Figure 6.1 for the multivariate case. Again, for a fixed l the local minimum at the center of mass can appear as the number of clusters increases, and the maximum moves to the right of the plot. This makes more difficult to obtain a good random initialization. The results for lower lengthscales are the same as in the multivariate case. For higher lengthscales, however, the number of clusters necessary for the minimum to appear is too high to be a concern in practice.

We have also studied how the number M of discretization points affects the objective function, as shown in Figure 6.5. When the number M of discretization points is small, the correlations of the points are small and thus the data behaves like multivariate data of M dimensions. However, as M increases the functional nature of the data becomes apparent, and the objective function essentially stops changing. Thus, the behavior is no longer the one of an M-dimensional vector. Instead, the behavior

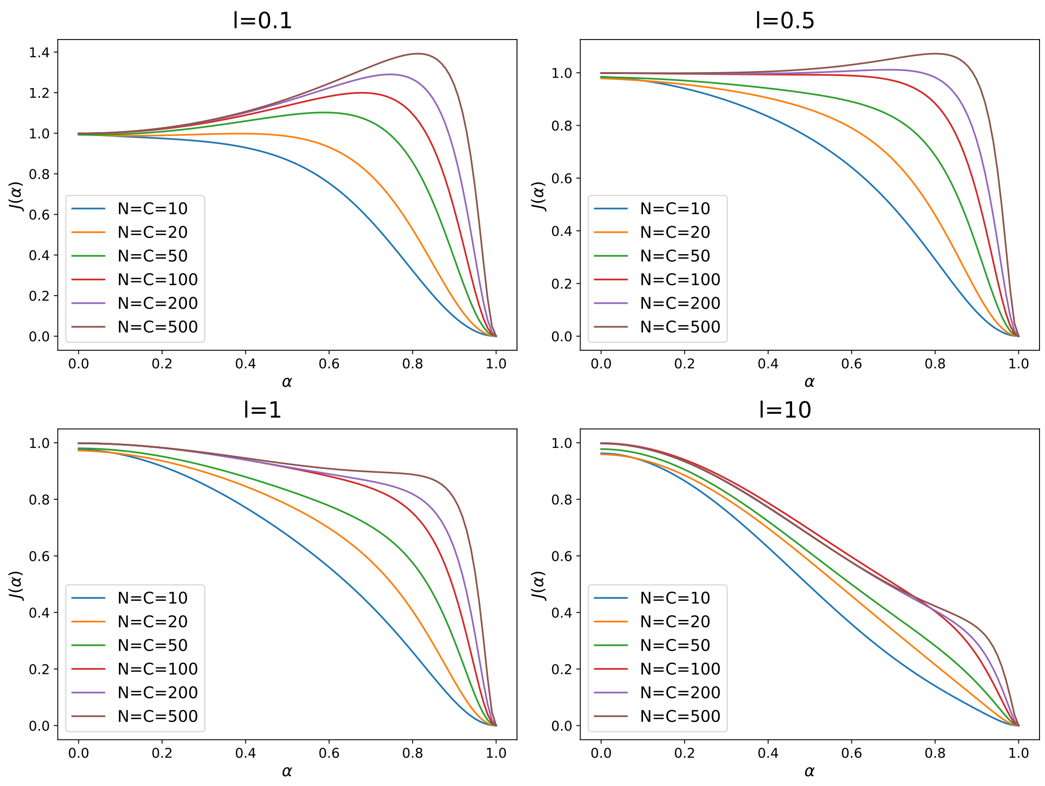


FIGURE 6.4: In this plot the effect of increasing the number of clusters C is shown for different values of the lengthscale l (0.1, 0.5, 1, and 10). When l is higher, the effect of increasing the number of clusters is much less dramatic than in the case that $l \ll \Delta t$, requiring a very large C for the convergence to fail.

of this method with functional data is similar to the one that can be found in lower dimensional multivariate observations.

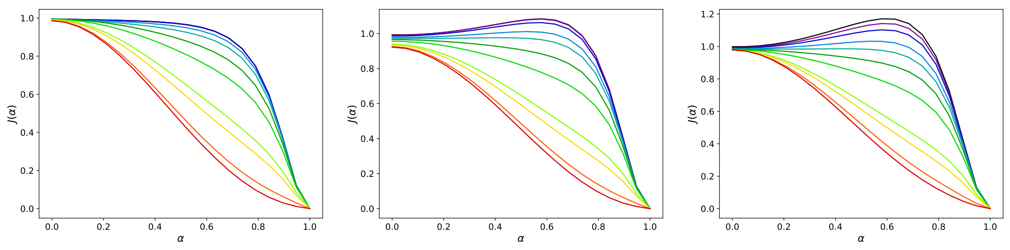


FIGURE 6.5: The plot in Figure 6.3 is recalculated for different values of M: from left to right, $M = 5$, $M = 10$, $M = 100$ (same as the original plot). As in the original plot, $N = C = 50$.

A second dataset, D_4 is also presented in Winkler, Klawonn, and Kruse, 2010. This intends to provide a more realistic example, in which the clusters can now be located anywhere and overlap. This dataset has N_C samples per cluster. The cluster centers are generated from a standard (multivariate) normal distribution $\mathbf{c}_j \sim N(\mathbf{0}, \mathbf{I}) \quad j = 1, \ldots, C$. The cluster elements are generated using a multivariate normal distribution centered at the $\mathbf{c}_j$:

$$\mathbf{x}_{ji} \sim N(\mathbf{c}_j, \sigma^2 \mathbf{I}) \quad j = 1, \ldots, C \quad i = 1, \ldots, N_C, \tag{6.9}$$

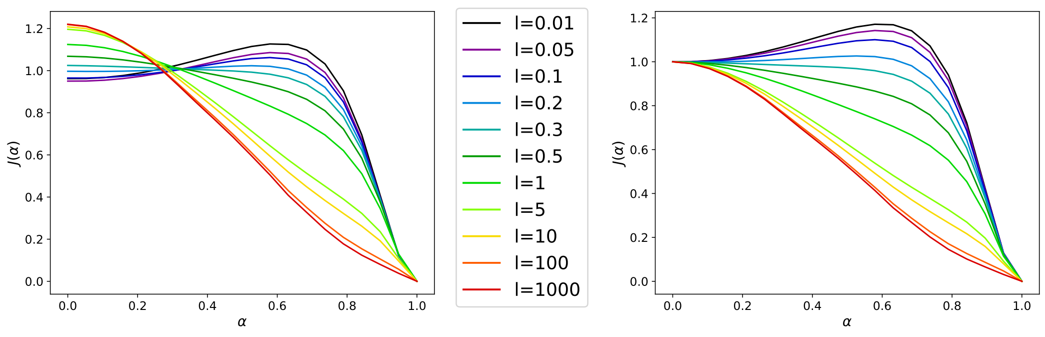


FIGURE 6.6: Inertia as a function of α, for cluster centers sampled from a Gaussian process with an exponential covariance function and different values for the lengthscale. The left plot has the original curves, while the ones in the right plot have been rescaled to have the same origin and thus be easier to compare with the ones on Figure 6.3. In both cases $N = C = 50$ and $M = 100$.

where the parameter σ controls the variance of the data.

We can see that this dataset differs from dataset D_2 in two ways. First, there is no normalization for the cluster centers to project them to a hypersphere. Second, there can be more than one sample per cluster, generated using a second normal distribution (or Gaussian process in the functional case), whose variance is controlled by σ. We have studied these two differences separately.

First, we note that without normalization one obtains similar results as in the normalized case. In Figure 6.6 we explore the variation of the lengthscale, as in Figure 6.3. We can observe that, although the curves are not exactly the same, starting now at different values, their shape is very similar. If we rescale the curves to have the same origin, we can compare them to the ones on Figure 6.3, and we observe that the two plots are alike. Thus, our analysis can focus on the normalized case, which is easier to study.

We show in Figure 6.7 the result of increasing N_C , the number of observations per cluster. This has the effect of raising the right minimum, up to a limit that depends on C, l and the dispersion inside a cluster (σ). When the dispersion is small, this has no big impact, and the results of our prior analysis are still valid. When the dispersion σ is moderately high and the lengthscale is small, the observations generated from the different clusters can mix, and the right location of the centers ceases to be a minimum of the objective function, as seen in the right plot of Figure 6.7. In this case, every initialization of the algorithm will fail to converge. Note that the effect of the dispersion is again smaller for higher lengthscales, for which the algorithm still converges to the correct solution.

In all the synthetic experiments considered, the algorithm FCM exhibits better behavior for higher lengthscales, corresponding to data in which the functional structure is more apparent, than for lower lengthscales. In this last case, the behaviour is the same as for the multivariate case, and convergence problems arise. For higher lengthscales, FCM tends to converge to the right solution. Thus, this seems to indicate that FCM works well for functional data. The convergence problems that arose for high dimensional data will appear more rarely in this context.

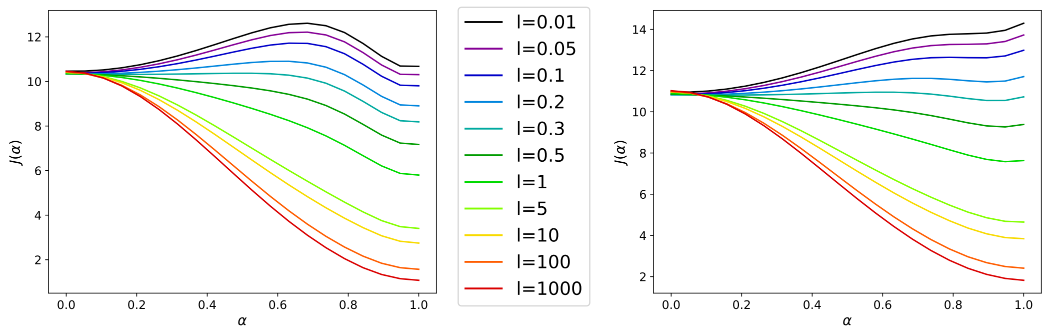


FIGURE 6.7: Inertia as a function of α, for cluster centers generated as a normalized Gaussian process with an exponential covariance function and different values for the lengthscale. The plots consists on $N = 500$ observations. The number of clusters is $C = 50$, which means that there are 10 points per cluster. They have been generated using a second Gaussian process placed at the cluster center with variances $\sigma = 0.05$ (left) and $\sigma = 0.1$ (right). The number of discretization points $M = 100$ is fixed.

6.3 Analysis

We have performed a brief analysis of the results obtained in the synthetic experiments. In particular, we analyzed the functional version of dataset D_2, with fixed fuzzifier $\omega = 2$. We explored the extreme cases $l \to \infty$ and $l \to 0$. We now present the main results obtained for these cases. The details are explained in Appendix C.

For $l \to \infty$, we studied the objective function in terms of the distribution of distances between observations. We observe that in this case the objective function approximates

$$J(\mathcal{D}, \alpha) \approx \frac{2(1-\alpha)^2(1+\alpha)^2}{((1-\alpha)^2 + (1+\alpha)^2)}, \tag{6.10}$$

which does not depend on the parameters C or M, and present no convergence problems.

For $l \to 0$, an analysis of the derivative of the objective function shows that convergence problems may start arising in this problem when the number of clusters is at least 12, which coincides to the results observed in practice. Thus, we would expect this minimum number of clusters required for the convergence problems to arise, to increase with the lengthscale. For higher lengthscales, corresponding with more functional data, the number of clusters required will be unattainable in practice.

6.4 Conclusions

The algorithm of fuzzy C-means (FCM) has been shown to present convergence problems for high dimensional multivariate data. In particular, the objective function presents an additional minimum in that case, corresponding to a degenerate solution, in which all the cluster centers are situated at the same position. The range of initializations of these centers for which the algorithm converges to this degenerate solution increases with the number of clusters.

In this chapter we have carried out an empirical study to determine whether similar difficulties appear for functional data, which is in principle infinite-dimensional.

This transformation between the multivariate and functional domains has been carried out in an experimental setting using discretized Gaussian processes with stationary covariance functions and varying the lengthscale of these processes. The lengthscale parameter measures the scale of decay of the correlations between nearby points. We observed that when this parameter is much lower than the spacing of the discretization grid, the functional FCM presents convergence problems that are similar to the multivariate case. This can be explained because in that case the values of the trajectories are approximately independent. When the scale of decay of the correlations is increased, and thus starts to be comparable to grid spacing, the functional nature becomes more apparent and the convergence is similar to low dimensional data.

The observed behavior has also been derived analytically for the limits of very small lengthscales (corresponding to the truly M dimensional case) and infinite lengthscale (corresponding to constant covariance). In that last case we see that the observed behavior is the same as for one-dimensional data.

Part II

Computational tools

Chapter 7

scikit-datasets and rdata: datasets for machine learning in Python

In this chapter we introduce the Python libraries **scikit-datasets** (Díaz-Vico and Ramos-Carreño, 2022) and **rdata** (Ramos-Carreño, 2022). The package **scikit-datasets** provides utilities for fetching and caching of datasets from several sources, such as the UCI, KEEL or LIBSVM repositories. As part of the work in this thesis, this package has been enhanced with functionality for loading functional and multivariate datasets from different sources, such as the UCR or R packages in the CRAN repository. This functionality is used to load these datasets in the **scikit-fda** package for functional data analysis, that will be presented in Chapter 8. Additionally new utility functions for running Machine Learning experiments with the fetched data and displaying the resulting scores have been created. These functions have been used to run the experiments in Section 8.6, and to create the tables in Section 8.6 and Chapter 5.

The package **rdata** (Ramos-Carreño, 2022) allows to parse data in the RData format, which is widely used in R packages, and convert it to Python objects. It is used by **scikit-datasets** to fetch datasets from CRAN. This provides direct access to most datasets included in R packages, which facilitates the comparison of the results of empirical evaluations carried out in either of these two languages.

7.1 Storage and loading of datasets

In order to meaningfully test statistics and machine learning algorithms, it is advisable to use a common set of classification and regression problems, so that different algorithms can be compared in equal terms. Thus, several data repositories and benchmarks have been collected along the years in a variety of formats (Dua and Graff, 2017; Chang and Lin, 2011; Alcalá-Fdez et al., 2010). Unfortunately most of them are not provided in a standardized format, nor can they be easily accessed programatically.

Several efforts have been made in recent years to organize and make these data sets and benchmarks available within the Statistics and Machine Learning communities. A very promising approach is OpenML (Vanschoren et al., 2014), which tries to collect existing datasets, along with the associated tasks, workflows and benchmark results. Although this is without doubt a very useful way to organize datasets and algorithms performance, OpenML currently requires that all datasets are stored in their servers in table format. Moreover sometimes the same dataset is uploaded with different modifications, which makes it more difficult to ensure that the data being tested is the same as in the original articles.

Instead the package **scikit-datasets** (Díaz-Vico and Ramos-Carreño, 2022) offers programmatic access in Python to several repositories currently in active use, such as the CRAN repository of R packages and the UCI repository. For most of them it is unreasonable to expect that they will be fully migrated to OpenML in the near future, both because they have a high number of datasets and because not all the information therein can be completely represented inside OpenML. With **scikit-datasets** those datasets can be downloaded with an API similar to the one provided in the popular Machine Learning package **scikit-learn** (Pedregosa et al., 2011; Buitinck et al., 2013), and the loaded datasets are cached and available for offline use.

The additional package **rdata** (Ramos-Carreño, 2022) has been developed in order to parse the RData file format, used in R datasets, and convert the resulting objects to their Python counterpart. This package is used by **scikit-datasets** to parse CRAN datasets, but can also be installed and used as a standalone package.

scikit-datasets has been used in the experimental setup of several scientific articles (Díaz-Vico, Figueiras-Vidal, and Dorronsoro, 2018; Díaz-Vico et al., 2019; Díaz-Vico et al., 2020; Díaz-Vico and Dorronsoro, 2020; Díaz-Vico, Fernández, and Dorronsoro, 2021), and it powers the dataset fetching module of the functional data package **scikit-fda** (Ramos-Carreño et al., 2022). **rdata** has been used to translate RData datasets in the packages **Pypath** (Saez Lab, 2021) and **Navis** (Schlegel et al., 2021). This chapter presents both packages, **scikit-datasets** and **rdata**, and illustrates their usage.

We will further explain the main functionalities of **scikit-datasets** in Section 7.2, including a description of its Application Programming Interface (API), the available repositories and the provided utility functions. The **rdata** package will be explained in Section 7.3. Finally in Section 7.4 we discuss briefly the relevance of these packages and the future plans for them.

7.2 scikit-datasets: Machine Learning repositories

scikit-datasets provides access to several repositories of data, widely used in the Machine Learning community. It downloads the data from the original source the first time it is accessed, caching it in a system folder for subsequent accesses. This way, it ensures that previously used datasets can be quickly loaded or even used offline. For most repositories, data can be coerced to a common representation, which will be explained in Section 7.2.1. This representation is based in the **scikit-learn** API for dataset fetching, with optional extra fields for information about partitions or cross validation that is sometimes provided by the repository. We also summarize the most relevant repositories that can be accesed by **scikit-datasets** at the moment of writing in Section 7.2.2, and explain their particular characteristics. Finally, we provide a brief explanation of additional utility functions available for executing experiments with the data and displaying the results in a scientific article.

7.2.1 API

For those repositories where the data is organized in a predictable way, the package **scikit-datasets** returns the data in a common format. That format is based in the one used by the popular Machine Learning package **scikit-learn** for loading its example datasets. The fetch functions return a dictionary-like class that allows attribute access and has the following keys:

- `data`: Matrix of observed data. It is a bidimensional **NumPy** array, with one observation per row and one observed feature per column.
- `target`: Target vector. It is a unidimensional **NumPy** array (except for multi-output problems), with one item per observation. This is the value that should be predicted for each observation, either a real value for regression problems or a different integer per class for classification problems.
- `DESCR`: A human readable description of the dataset.
- `feature_names`: The list of the feature names, when this information is available.
- `target_names`: The list of class names in classification problems.
- `train_indices`: For problems that include a separate set for train, the indices of the train samples.
- `validation_indices`: For problems that incorporate a validation set, the indices of the validation samples.
- `test_indices`: For problems in which a test set is available, the indices of the validation samples.
- `inner_cv`: A **scikit-learn** cross-validator object to use in combination with hyperparameter tuning utilities in order to employ specific data in validation. Only available for datasets with a validation set.
- `outer_cv`: A Python iterable over train/test partitions for different folds. Only available for datasets with explicit folds already separated.

In addition, using the parameter `return_X_y` the user can request that only the `data` and `target` attributes are returned, as a tuple.

There are repositories, such as CRAN, where the data may be in arbitrary format, and it is not possible to convert it automatically. In that case the data is returned in a format as similar as possible to the original one.

7.2.2 Available repositories

The package **scikit-datasets** provides programmatic access to several widely used repositories of Machine Learning datasets. These include datasets for classification and regression problems, both for multivariate and functional data. The repositories available from **scikit-datasets** will be briefly described now.

CRAN repository of R packages

The CRAN repository is the main repository of libraries written in the R language. Along with source code, library authors usually upload datasets that can be used to illustrate the algorithms implemented. This makes CRAN a common and widely used source of statistical data for R programmers. However, as the data is normally written in the R-specific RData format, Python users commonly have difficulties to load the data, usually by manually opening the dataset in R and exporting it again in a less specific file format, such as CSV.

The package **scikit-datasets** alleviates those problems, by automatically downloading datasets from CRAN and converting the data in RData format to suitable

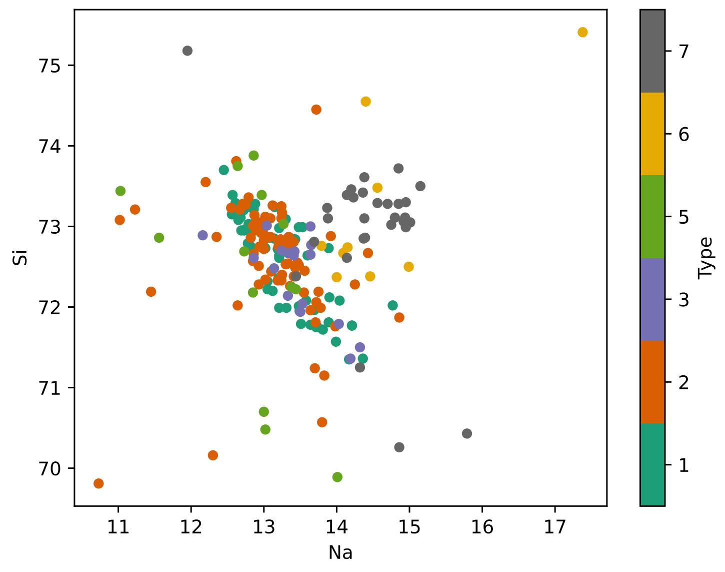


FIGURE 7.1: Scatter plot between the Na and Si columns in Glass dataset, from the **mlbench** R package.

Python objects. This last conversion step is performed by the **rdata** package, which has been built with this purpose in mind and is described in Section 7.3.

As an example, the following code loads the Glass dataset from the R package **mlbench** (Leisch and Dimitriadou, 2021). This dataset contains a classification problem in the context of criminological investigation, where the class of a particular glass fragment must be predicted from its chemical composition. Here the quantity of sodium (Na) is plotted against its silicon (Si) in a scatter plot, in Figure 7.1. Each glass category is plotted with a different color.

```
import matplotlib.pyplot as plt

from skdatasets.repositories.cran import fetch_dataset

bunch = fetch_dataset(
    dataset_name="Glass",
    package_name="mlbench",
)

bunch["Glass"].plot(
    "Na", "Si",
    kind="scatter",
    c="Type",
    colormap="Dark2",
)
plt.show()
```

Note that the datasets in the **mlbench** package are organized as one R data frame, which on Python is converted to a Pandas data frame (Pandas Development Team, 2020). Most CRAN packages, however, are not structured in this way, and the transformed data mimics the original structure. Thus, it is not possible to automatically convert them to a common format. For example, the Berkeley Growth functional dataset of Chapter 3, available in the **fda** (Ramsay et al., 2020) package, contains the growth curves for boys and girls, measured since their birth until they were 18 years

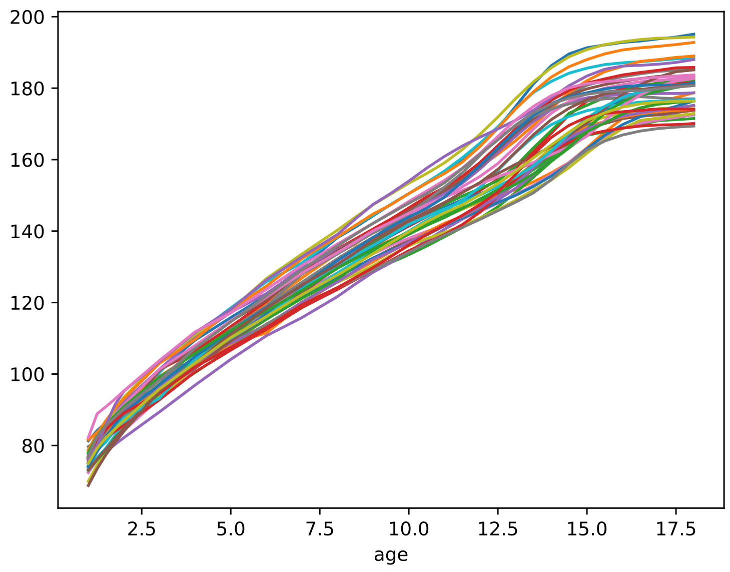


FIGURE 7.2: Growth curves for boys in the Berkeley Growth dataset, from the **fda** R package.

old. The data for boys and girls, as well as the ages when the measurements took place are stored as different attributes: "hgtm", "hgtf" and "age", respectively. Thus the following code needs to manually preprocess the data of the boys before plotting it in Figure 7.2.

```python
import matplotlib.pyplot as plt

from skdatasets.repositories.cran import fetch_dataset

bunch = fetch_dataset(
    dataset_name="growth",
    package_name="fda",
)

boys = bunch["growth"]["hgtm"]
girls = bunch["growth"]["hgtf"]
ages = bunch["growth"]["age"]

boys_new_coord = boys.rename(
    {"dim_0": "age"},
).assign_coords({"age": ages})

boys_new_coord.to_pandas().plot(legend=False)
plt.show()
```

UCI

The UCI repository (Dua and Graff, 2017) is a collection of databases and data generators hosted by the University of California Irvine (UCI). Created as an FTP archive in 1987 by David Aha and other university students, nowadays it hosts hundreds of datasets and has been cited over 1000 times. It is one of the most important sources of Machine Learning datasets, hosting widely known and used ones, such as the Iris and Wine datasets. It contains datasets for regression, classification and clustering.

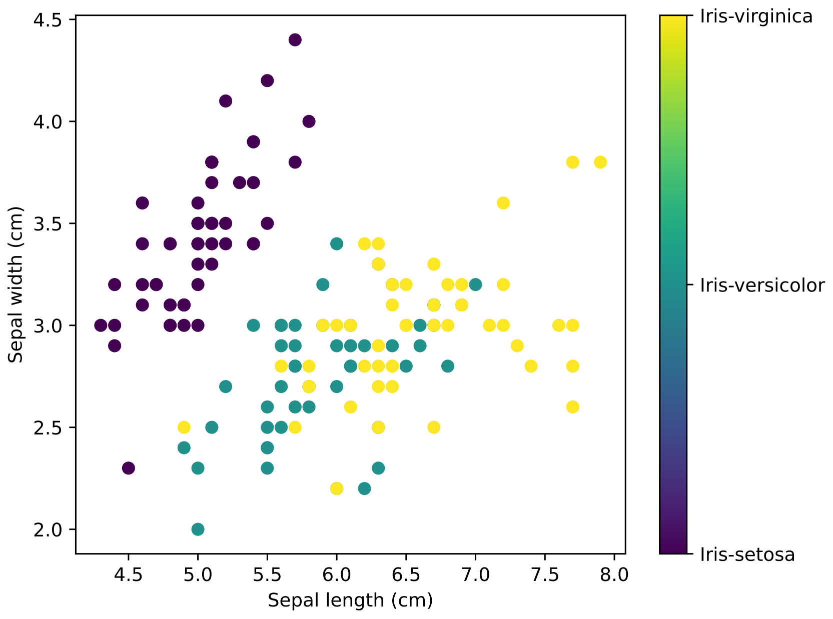


FIGURE 7.3: Scatter plot of two features of the Iris dataset, obtained from the UCI repository.

As an example, the following code can be used to plot the classical Iris dataset, obtained from the UCI. Note that the UCI repository does not provide the feature names in a structured way, so they have to be manually specified. The resulting plot is shown in Figure 7.3.

```python
import matplotlib.pyplot as plt

from skdatasets.repositories.uci import fetch

iris = fetch("iris")

formatter = plt.FuncFormatter(
    lambda i, *args: iris.target_names[int(i)]
)

plt.figure()
plt.scatter(iris.data[:, 0], iris.data[:, 1], c=iris.target)
plt.colorbar(ticks=[0, 1, 2], format=formatter)
plt.xlabel("Sepal length (cm)")
plt.ylabel("Sepal width (cm)")

plt.tight_layout()
plt.show()
```

LIBSVM

The LIBSVM data repository (Chang and Lin, 2011), hosted by the National Taiwan University contains datasets originally intended to be used along with the LIBSVM library for Support Vector Machines (SVMs). Thus, this repository is heavily oriented towards classification problems, although it provides also a few regression datasets.

In this repository some datasets have training and test partitions, and some have even a validation partition. In this last case the `inner_cv` attribute can be used to perform hyperparameter selection using the validation partition, as the following example shows. Here, the DNA multiclass dataset is being used to train an SVM from the **scikit-learn** package. In this problem a sequence of DNA is given in each observation, where each nucleobase (A, T, C, G) is represented using binary markers. The objective is to predict if the sequence corresponds to an exon/intron boundary, an intron/exon boundary, or neither. The regularization parameter C is a hyperparameter optimized over the validation set. Then, the final score is obtained comparing the predicted class of the samples in the test set with their real class, giving a result of 0.956:

```python
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC

from skdatasets.repositories.libsvm import fetch

data = fetch("multiclass", "dna.scale")

train_val_indices = data.train_indices + data.validation_indices
X_train = data.data[train_val_indices]
y_train = data.target[train_val_indices]
X_test = data.data[data.test_indices]
y_test = data.target[data.test_indices]

classifier = SVC()
grid_search = GridSearchCV(
    classifier,
    param_grid={
        "C": [1e-2, 1e-1, 1, 1e1, 1e2],
    },
    cv=data.inner_cv,
)
grid_search.fit(X_train, y_train)
score = grid_search.score(X_test, y_test)
```

KEEL

The KEEL-dataset repository (Alcalá-Fdez et al., 2010), hosted at the Universidad de Granada, contains classification, regression and time series datasets as well as semi-supervised and unsupervised learning problems. In particular, it offers a large number of classification problems that present additional difficulties, such as imbalanced classes or noise contamination.

This repository provides already separated folds for many datasets. Users can request these predefined folds passing the parameter `nfolds` with a value of 5 or 10. In that case the predefined folds will be returned in the `outer_cv` attribute. The usage of this feature is illustrated in the following code with the Titanic dataset. This is a classification problem where information about passengers aboard the Titanic is used to predict whether they survived or not. The data is already split in five folds, and a score of a linear discriminant classifier is predicted for each of them.

```python
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from skdatasets.repositories.keel import fetch

data = fetch("classification", "titanic", nfolds=5)

scores = []

```

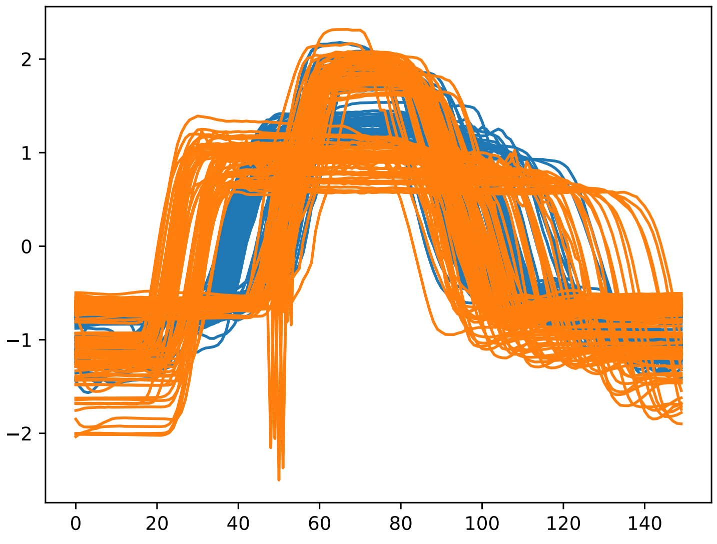


FIGURE 7.4: Curves of the GunPoint dataset from UCR, with a different color per class.

```
for X_train, y_train, X_test, y_test in data.outer_cv:

    classifier = LinearDiscriminantAnalysis()
    classifier.fit(X_train, y_train)
    scores.append(classifier.score(X_test, y_test))
```

UCR

As mentioned in Chapter 3, the UCR/UEA time series classification archive (Dau et al., 2019; Bagnall et al., 2018) is one of the most well known repositories for time series and functional datasets. Created originally by the University of California Riverside, the repository contained univariate time series classification problems. This was later expanded to include also multivariate time series with the addition of new datasets from the University of East Anglia, and it is now hosted at www.timeseriesclassification.com.

The following code illustrates the GunPoint time series classification problem from UCR. In this dataset the right hand position of different actors is tracked while they perform two different classes of tasks: pointing a replicate gun to a target or pointing their index finger instead. The curves are plotted in Figure 7.4, with a different color per class.

```
import matplotlib.pyplot as plt

from skdatasets.repositories.ucr import fetch

data = fetch("GunPoint")

plt.plot(data.data[data.target == 1].T, color="C0")
plt.plot(data.data[data.target == 2].T, color="C1")

plt.show()
```

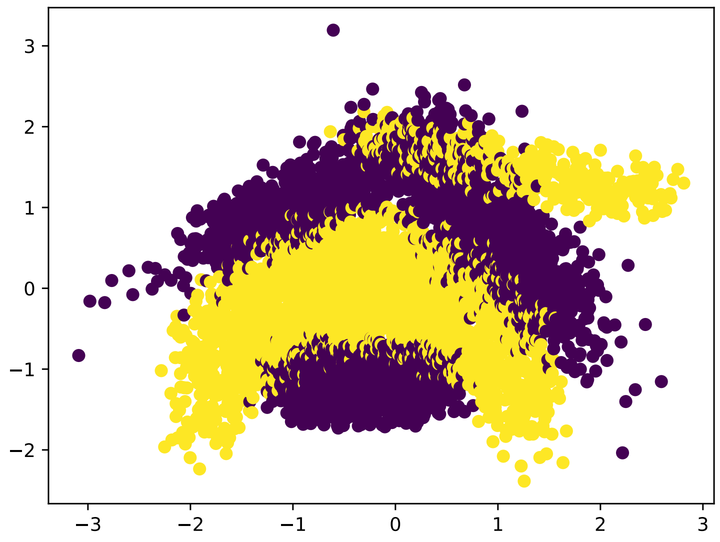


FIGURE 7.5: The synthetic Banana dataset from the Rätsch benchmark.

Rätsch

The benchmark datasets used in Rätsch, Onoda, and Müller, 2001; Mika et al., 1999 can also be managed with **scikit-datasets**. The datasets are hosted in GitHub (Diethe, 2015), and mainly consist in binary classification problems. Train-test split indices are provided to repeat each of the iterations used in the aforementioned articles. As an example, the code to plot the synthetic Banana dataset in Figure 7.5 is shown below:

```python
import matplotlib.pyplot as plt

from skdatasets.repositories.raetsch import fetch

data = fetch("banana")
plt.scatter(data.data[:, 0], data.data[:, 1], c=data.target)
plt.show()
```

Wrappers for existing libraries

In addition to the aforementioned repositories, **scikit-datasets** wraps the example datasets in the packages **scikit-learn** and **keras** (Chollet et al., 2015), exposing a similar API for manipulating them. It also provides access to currency exchange information time series from the foreign exchange market (Forex) by means of the package **forex-python** (MicroPyramid, 2021).

7.2.3 Utilities

In addition to fetching datasets, **scikit-datasets** offers several utilities to help working with them. Users can easily launch experiments for combinations of datasets and **scikit-learn** estimators by means of the **Sacred** package (Greff et al., 2017). The resulting scores can be used by **scikit-datasets** to generate comparison tables with automated ranking or to create hyphotesis testing tables, ready to be included in publications.

For example, Table 7.1, comparing cross-validation results for several classification methods in three datasets from the Rätsch benchmark, has been generated using the `scores_table()` function of the package. Note that it is possible to customize the ways in which the results are displayed. In this case different formatting is used depending on the rank (bold for best results and underline for second position). Moreover, significant results with respect to the next in the rank are marked with an asterisk. It is also possible to add additional summary rows.

	lda	**qda**	**3-nn**	**5-nn**
banana	0.38 ± 0.15 (4)	0.46 ± 0.10 (3)	**0.73 ± 0.10 (1)**	**0.73 ± 0.09 (1)**
breast_cancer	**0.73 ± 0.02 (1)**	0.70 ± 0.04 (2)	0.66 ± 0.03 (4)	0.68 ± 0.05 (3)
ringnorm	0.75 ± 0.06 (2)*	**0.98 ± 0.00 (1)***	0.70 ± 0.03 (3)	0.68 ± 0.04 (4)
Average rank	*2.33*	***2.00***	*2.67*	*2.67*

TABLE 7.1: Example of a score table generated with **scikit-datasets**.

These utility functions have been used in this work to execute the experiments in Section 8.6 and to display the results in the tables of Section 8.6 and Chapter 5.

7.3 rdata: Parsing R datasets

The datasets from the CRAN repository are stored in the R specific format RData. In Python, there were a few packages that could parse this file format, albeit all of them presented some limitations.

The package **rpy2** can be used to interact with R from Python. This includes the ability to load data in the RData format, and to convert these data to equivalent Python objects. Although this is arguably the best package to achieve interaction between both languages, it has many disadvantages if one wants to use it just to load RData datasets. In the first place, the package requires an R installation, as it relies in launching an R interpreter and communicating with it. Secondly, launching R just to load data is inefficient, both in time and memory. Finally, this package inherits the GPL license from the R language, which is not compatible with most Python packages, typically released under more permissive licenses.

The recent package **pyreadr** also provides functionality to read some R datasets. It relies in the C library **librdata** in order to perform the parsing of the RData format. This adds an additional dependency from C building tools, and requires that the package is compiled for all the desired operating systems. Moreover, this package is limited by the functionalities available in **librdata**, which at the moment of writing does not include the parsing of common objects such as R lists and S4 objects. The license can also be a problem, as it is part of the GPL family and does not allow commercial use.

As existing solutions were unsuitable for the needs of **scikit-datasets**, the package **rdata** was developed to parse data in the RData format. This is a small, extensible and very complete implementation in pure Python of a RData parser, that is able to read and convert most datasets in the CRAN repository to equivalent Python objects. It has a permissive license and can be extended to support additional conversions from custom R classes.

The package **rdata** is intended to be both flexible and easy to use. In order to be flexible, the parsing of the RData format and the conversion of the parsed structures to appropriate Python objects have been splitted in two steps. This allows advanced users to perform custom conversions without losing information. Most users, however, will want to use the default conversion routine, which attempts to convert data to a standard Python representation which preserves most part of the information.

The following code illustrates the loading of a dataset in the RData format. The function `parse_file()` of the `parser` module is used to parse the RData file, returning a tree-like structure of Python objects that contains a representation of the basic R objects conforming the dataset. The function `convert()` of the `conversion` module transforms that representation to the final Python objects, such as lists, dictionaries or dataframes, that users can manipulate.

```python
import rdata

parsed = rdata.parser.parse_file("dataset.rda")
converted = rdata.conversion.convert(parsed)
```

In addition to the aforementioned functions, the function `parse_data()` can be used when a dataset in RData format is stored in a buffer in memory, instead of on a file in disk.

Advanced users will probably require loading datasets which contain non standard S3 or S4 classes, translating each of them to a custom Python class. This is easy to achieve using **rdata** by simply creating a constructor function that receives the converted object representation and its attributes, and returns a Python object of the desired type. As an example, consider the following simple code that constructs a **Pandas** `Categorical` object from the internal representation of an R `factor`.

```python
import pandas


def factor_constructor(obj, attrs):
    values = [attrs['levels'][i - 1] if i >= 0 else None for i in obj]

    return pandas.Categorical(values, attrs['levels'], ordered=False)
```

Then, a dictionary containing as keys the original class names to convert and as values the constructor functions can be passed as the `constructor_dict` parameter of the `convert()` function. In the previous example, this could be done using the following code:

```python
converted = rdata.conversion.convert(
    parsed,
    constructor_dict={"factor": factor_constructor},
)
```

When the default conversion routine is being executed, if an object belonging to an S3 or S4 class is found, the appropriate constructor will be called passing to it the partially constructed object. If no constructor is available for that class, a warning will be emitted and the constructor of the most immediate parent class available will be called. If there are no constructors for any of the parent classes, the basic underlying Python object will be left without transformation.

By default, a dictionary named `DEFAULT_CLASS_MAP` is passed to `convert()` including constructors for commonly used classes, such as `data.frame`, `ordered` or the aforementioned `factor`. In case anyone wants different conversions for basic R objects, it would be enough to create a subclass of the `Converter` class. Several utility functions, such as the routines `convert_char()` and `convert_list()`, are exposed by the `conversion` module in order for users to be able to reuse them for that purpose.

7.4 Conclusions

We have presented the main functionalities of the **scikit-datasets** and **rdata** packages. **scikit-datasets** can be used to fetch datasets from several widely used repositories. It also provides utilities for performing experiments on that data and visualizing comparison tables with the resulting scores. Using this package, it is simple to compare new algorithms with existing ones using the same datasets.

In addition, the package **rdata** allows the parsing and conversion of datasets in the RData format to Python structures. Using this package, **scikit-datasets** can fetch almost every dataset in the CRAN repository of R packages with only one line of code. As this repository receives constant uploads of state-of-the-art statistical packages, many of them including sample datasets, we feel that this is a very valuable resource for researchers. Using **scikit-datasets** they can easily compare results for the same problems between Python-based and R-based implementations.

In the context of this work, these packages provide the necessary software support for loading the data, running the experiments and presenting the results shown in the rest of the thesis. In particular the package **scikit-fda**, presented in the next chapter, uses **scikit-datasets** for loading the common functional datasets offered by the library and presented in Chapter 3. The final comparative study of the same chapter, makes also use of the utilities of **scikit-datasets** for experiment creation and visualization of the results.

Chapter 8

scikit-fda: a Python package for functional data analysis

As part of this work, the library **scikit-fda** (Ramos-Carreño et al., 2023), a Python package for functional data analysis (FDA) has been implemented. This library provides a comprehensive set of tools for representation, preprocessing, exploratory analysis and machine learning for functional data. It is integrated in the scientific Python ecosystem, and in particular with the widely used machine learning package **scikit-learn**. As a result, it can take advantage of the functionality provided by this package, such as pipelines, model selection, and hyperparameter tuning tools. The **scikit-fda** package has been released as free and open-source software in order to encourage contributions from the FDA community. It also includes extensive documentation, with tutorials and examples of use.

The structure of this chapter is as follows: In Section 8.1 we give an overview of the libraries for FDA that are available. The tools that **scikit-fda** provides for the representation of functional data are described in Section 8.2. In Section 8.3 the tools provided for preprocessing, exploratory analysis and machine learning are described in more detail, including example code snippets. The tools for ensuring the quality of **scikit-fda**'s code and the documentation of the library are briefly explained in Section 8.4. In Section 8.5 the machine learning capabilities of the package are illustrated, using the examples presented on Ramos-Carreño et al., 2022. Finally, Section 8.6 presents a comparative study of classification methods (Ramos-Carreño, Torrecilla, and Suárez, 2022), which has been carried out using the tools available in **scikit-fda**, and thus provides a real-world example of its applicability and effectiveness in FDA research.

8.1 Software for functional data analysis

Due to the growing interest in functional data, several specialized software tools for functional data analysis (FDA) have emerged in recent years (Scheipl, 2021). One of the main references in the field is the **fda** package, which is available in R and MATLAB (Ramsay et al., 2020). This general-purpose library provides an implementation of the methods described in Ramsay and Silverman, 2005 and Ramsay, Hooker, and Graves, 2009. It utilizes a basis expansion representation of the functional observations. Another important reference in the FDA community is the **fda.usc** package (Febrero-Bande and Oviedo de la Fuente, 2012), in which the non-parametric approach developed in Ferraty and Vieu, 2006 is adopted. One of the contributions of this library is the introduction of a novel structure for the representation of functional data in discrete form, as a collection of measurements at a grid of points. This

R package provides an extensive range of FDA tools, including methods for regression and classification.

A more recent general-purpose R package is **tidyfun** (Scheipl, Goldsmith, and Wrobel, 2020). In this library, a novel data structure (vectors of class *tf*) is introduced to represent functional observations. These *tf* vectors can be included as columns in an R *data frame* alongside with other variables. Furthermore, they can be manipulated using the tools of the *tidyverse* ecosystem (Wickham et al., 2019). Another recent contribution is **funData** (Happ-Kurz, 2020). This package provides a representation for discretized univariate and multivariate functional data of arbitrary dimensions based on S4 R classes.

Finally, a variety of computational tools have been developed to address specific problems in FDA. Some relevant examples are the packages **refund** (Goldsmith et al., 2019), which, among others, provides tools for functional regression and principal component analysis (FPCA), **FDboost** (Brockhaus, Rügamer, and Greven, 2020), focused on regression problems, **funFEM** (Bouveyron, Côme, and Jacques, 2015) and **funHDDC** (Schmutz et al., 2020) for functional clustering, **fpca** (Peng and Paul, 2011) and **fdapace** (Carroll et al., 2020), which are mainly devoted to functional principal component analysis (FPCA). The **fdapace** package available in R, and its MATLAB counterpart **PACE** (Yao, Müller, and Wang, 2015), provide methods for both sparsely or densely sampled random trajectories based on FPCA, via the Principal Analysis by Conditional Estimation (PACE) algorithm. The package **fdasrvf** (Tucker, 2020b) contains tools for alignment, elastic registration, PCA, and regression with functional data based on the square-root velocity framework (SRVF) described in Srivastava and Klassen, 2016. This package is available under this name both in R and in MATLAB, as **fdasrsf** (Tucker, 2020a) in Python, and as **ElasticFDA** (Tucker, 2021) in Julia. The package **roahd** (Ieva et al., 2019) includes a collection of methods for robust analysis of functional data. Outlier dectection tools are provided also in **fdaoutlier** (Ojo, Lillo, and Fernandez Anta, 2021). Visualization tools, including interactive ones, are provided also in the packages **rainbow** (Hyndman and Shang, 2010) and **refund.shiny** (Wrobel et al., 2016). A recent contribution is the R package **mlrFDA** described in Pfisterer et al., 2021, which gives access and extends the machine learning framework **mlr** (Bischl et al., 2016) for the analysis of functional data.

In recent years, the Python language has become more relevant in statistics, data science, and machine learning. However, in contrast to the wide variety of alternatives available for FDA in R, the options in Python are much more limited both in number and functionality. Some Python libraries devoted to FDA are **fdasrsf**, which has been described earlier, and the recently released **FDApy** (Golovkine, 2021), that provides methods for principal component analysis and clustering.

In this context, we present **scikit-fda** (Ramos-Carreño et al., 2022), a general-purpose library that makes functional data processing and analysis accessible to the Python community. This package offers data structures for the representation of functional data both in discretized form and as a basis expansion, and an extensive set of tools for preprocessing (smoothing, registration and dimensionality reduction), and statistical analysis, including interactive visualization and outlier detection tools. In addition, it provides infrastructure to facilitate the application of the machine learning tools of **scikit-learn** to functional data. Comprehensive documentation is supplied that includes installation instructions, tutorials, API reference, and illustrative examples.

In the short time since its inception, several libraries have been developed using **scikit-fda** as their foundation (Bernard et al., 2021; Consagra, Venkataraman, and

Qiu, 2022). In addition, it has been employed in a number of recent investigations (Fermanian, 2022; Torrecilla et al., 2020; Tan et al., 2020; Tan et al., 2021; Pegoraro and Beraha, 2021).

8.2 Representation of functional data in scikit-fda

The **scikit-fda** package provides tools for the representation of functional observations of the form $x : \mathcal{T} \subseteq \mathbb{R}^p \to \mathbb{R}^q$, with $p \geq 1$, and $q \geq 1$. The parameter p is the dimension of the domain of the functions ($p = 1$ for curves, $p = 2$ for surfaces, etc.). The parameter q is the dimension of the codomain; that is, the number of output coordinates for vector-valued functions. For instance, a grayscale two-dimensional image can be treated as a functional datum with $p = 2$, for the location of the pixels in the image, and $q = 1$, for the intensity at each pixel. A color image consisting of three channels (e.g., red, green, and blue) would have $p = 2$ and $q = 3$. The values of p and q are the same for all the observations in a functional dataset. For the sake of clarity, we focus on the case of real-valued univariate functions (that is, $p = q = 1$) for which most of the functionality described in this section is implemented. We will further assume that the functions are defined on a compact interval in the real line. In higher dimensions, the domain is assumed to be rectangular.

Recall from Section 2 that a functional dataset $\{x_i(t), t \in \mathcal{T}\}_{i=1}^N$ can be represented either in discretized form, or as a basis expansion. In the former representation, a functional observation consists of a set of measurements at a grid of points $\mathbf{t} = (t_1, \ldots, t_M) \in \mathcal{T}^M$, which is common for all observations. This grid need not be regularly spaced. The observation x_i is then represented as the vector $\{x_i(\mathbf{t})\}_{i=1}^N$, where $x_i(\mathbf{t}) = (x_i(t_1), \ldots, x_i(t_M))$. Its representation as an expansion in a functional basis $\{\phi_b(t), t \in \mathcal{T}\}_{b \geq 1}$ would be instead

$$x_i(t) = \sum_{b \geq 1} c_{ib} \phi_b(t), \tag{8.1}$$

where the coefficients of the expansion $\{c_{ib}\}_{b \geq 1}$ are different for each observation.

The package **scikit-fda** provides data structures for both types of representation: the class `FDataGrid` for discretized data, and the class `FDataBasis` for expansions in a functional basis. They are derived from the abstract class `FData`, which provides common properties and methods. In what follows, these classes are described in detail.

8.2.1 The class `FData`

In the class `FData`, the attributes and methods shared by the discretized and the basis expansion representations of the functional dataset are collected. Thus, it provides a common interface for both `FDataGrid` and `FDataBasis` objects. Specifically, objects of the `FData` class have the following attributes:

- `dataset_name`: Name of the functional dataset.
- `n_samples`: Size (number of functional observations) of the dataset.
- `dim_domain`: Dimension of the domain in which the functions are defined ($p \geq 1$).
- `argument_names`: Names of each of the p arguments of the function (domain dimensions).

- `domain_range`: Limits of the intervals for each of the domain arguments. They are used as the default ranges for plotting and numerical integration.
- `dim_codomain`: Dimension of the codomain q (output). For scalar functions, $q = 1$. Values $q > 1$ correspond to vector-valued functions.
- `coordinate_names`: Names of the q codomain coordinates.
- `extrapolation`: Default extrapolation strategy; for instance, constant or periodic. See Section 8.2.4 for details.

As an illustration of this data structure, consider the case of observations that are bidimensional RGB images. For this functional dataset, `dim_domain` would be 2, corresponding to the two dimensions of the image. The names of these dimensions (e.g., "x" and "y", or "horizontal" and "vertical") would be stored in the attribute `argument_names`. The attribute `dim_codomain` would be 3, corresponding to the three color channels. The names of these channels ("R", "G", and "B") would be stored in the attribute `coordinate_names`.

The class `FData` provides also methods that are common to both representations; for instance, methods for evaluation, addition, multiplication by a scalar, and plotting. Since `FData` is abstract, it is not possible to directly instantiate an object of this class. Instead, objects of one of its subclasses, `FDataGrid` or `FDataBasis`, need to be created.

8.2.2 Discretized representation: The class `FDataGrid`

Functional data are often the result of monitoring a continuous process at a discrete set of points. For the general case, $x : \mathbb{R}^p \to \mathbb{R}^q$, with $p, q \geq 1$, we assume that the discretization grid in the jth dimension is $\mathbf{t}_j = (t_1, \dots, t_{M_j})$, with $j = 1, \dots, p$. In the **scikit-fda** library, the points in the grid need not be regularly spaced. The grid has to be the same for all observations in the functional dataset. The dataset $\{x_i : \mathbb{R}^p \to \mathbb{R}^q\}_{i=1}^N$ is represented as an object of the class `FDataGrid`. The values of the functional observations are stored in the tensor $\{x_i(\mathbf{t}) = (x_i(t_1), \dots, x_i(t_M))\}_{i=1}^N$ of dimension $N \times M_1 \times \cdots \times M_p \times q$. In the case $p = q = 1$, the discretized sample is simply an $N \times M$ matrix.

In addition to those inherited from `FData`, objects of this class have the following attributes:

- `grid_points`: Sequence of discretization grids, one for each domain dimension $\left(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_p\right)$. The values of the functions are specified at the Cartesian product of the 1-D arrays in the sequence.
- `data_matrix`: **NumPy** array of dimensions $N \times M_1 \times \cdots \times M_p \times q$ in which the values of the N functional observations are stored.
- `interpolation`: Default interpolation strategy for locations within the rectangular discretization grid. See Section 8.2.4 for details.

Since the attribute `data_matrix` is a **NumPy** array, it is possible to carry out pointwise operations, such as powers, exponentials, logarithms, and trigonometric functions by directly applying the corresponding **NumPy** functions (Harris et al., 2020).

As an illustration, in the following code a `FDataGrid` object is created with three functional observations measured at grid points $\mathbf{t} = (0, 0.1, 0.3, 0.4, 0.7, 1)$. These data are depicted in Figure 2.1.

```python
import skfda

grid_points = [0.0, 0.1, 0.3, 0.4, 0.7, 1.0]
data_matrix = [
    [109.5, 115.8, 121.9, 130.0, 138.2, 141.1],
    [104.6, 112.3, 118.9, 125.0, 130.1, 133.0],
    [100.4, 107.1, 112.3, 118.6, 124.0, 126.5],
]

fd = skfda.FDataGrid(
    data_matrix=data_matrix,
    grid_points=grid_points,
)
```

In the previous example, the discretization points and the values of the functions are specified manually. The values of `grid_points` and `data_matrix` can be imported from data files in standard formats, such as CSV, XLSX, ARFF, MATLAB files, with the help of the corresponding functions from **NumPy** (Harris et al., 2020), **SciPy** (Virtanen et al., 2020), **pandas** (Pandas Development Team, 2020), and similar packages. An example of how to import data from a CSV file is provided in `https://fda.readthedocs.io/importing_data`.

8.2.3 Basis expansion representation: The class `FDataBasis`

Assume that the functional observations in the daset considered belong to $\mathcal{X}$, a separable Hilbert space; for instance, L^2. Under such assumption there exists a countable basis $\{\phi_i(t)\}_{i\geq 1}$ that is complete, so that any $x(t) \in \mathcal{X}$ can be expressed as

$$x(t) = \sum_{b\geq 1} c_b \phi_b(t), \tag{8.2}$$

where $\{c_b\}_{b\geq 1}$, are the coefficients of basis expansion. The **scikit-fda** library provides support for such a representation in the constant, monomial, B-spline, and Fourier bases. In addition, other types of bases can be implemented by the user. See `https://fda.readthedocs.io/create_new_bases` for an example on how to define new bases. The choice of basis should be made taking into consideration the characteristics of the data at hand. For instance, the Fourier basis is well-suited to representing periodic functions. For non-periodic data, a representation in the B-splines basis is probably more appropriate. The monomial basis is useful to represent polynomials. Monomials are also the building blocks of the Maclaurin series. Therefore, the monomial basis can be used to represent local approximations of analytic functions. The first five elements of the different bases provided in **scikit-fda** are shown in Figure 8.1.

In general, these types of representation are infinite-dimensional. In practice, the expansion is truncated to the first B terms

$$x(t) \approx \sum_{b=1}^{B} c_b \phi_b(t). \tag{8.3}$$

Truncation often results in the smoothing of the original functional observations. This smoothing effect can be beneficial for the representation of noisy data (see Section 8.3.3).

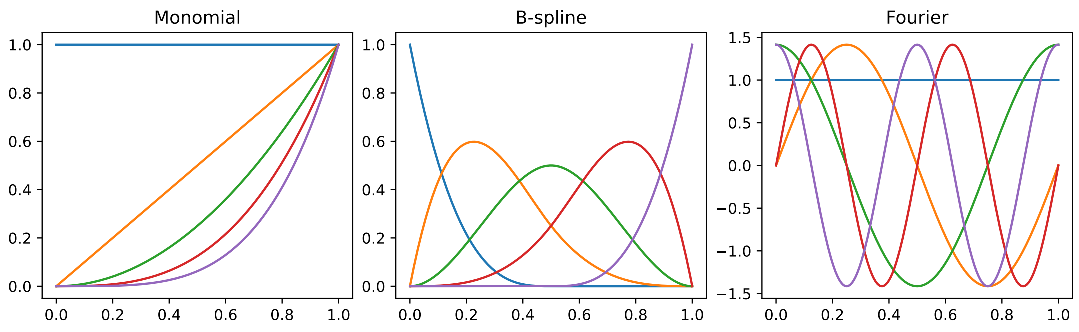


FIGURE 8.1: First five elements of the bases available in **scikit-fda**: Monomial (left), B-splines (center), and Fourier (right).

In **scikit-fda**, the class `FDataBasis` is used to represent functional data as a finite basis expansion. In addition to those inherited from `FData`, objects of this class have the following attributes:

- `basis`: The basis for the representation of the functional observations. It is an object of the class `Basis`, one of whose attributes, `n_basis` is the number of elements of the basis considered. The bases available are `Constant`, `Monomial`, `BSpline` and `Fourier`.
- `coefficients`: The $N \times B$ matrix that contains the coefficients of the basis expansion.

The following code is used to illustrate **scikit-fda**'s support for this type of representation.

```
import skfda
from skfda.representation.basis import Fourier, BSpline

X, y = skfda.datasets.fetch_phoneme(return_X_y=True)

X.to_basis(BSpline(n_basis=5))
X.to_basis(Fourier(n_basis=5))
```

In this code, the *Phoneme* dataset (Hastie, Tibshirani, and Friedman, 2009) is used. The curves in this dataset correspond to log-periodograms of the time series of utterances of five different phonemes by different speakers. The functional observations are transformed from their original discretized representation into different basis expansions. A sample of the transformed trajectories, together with the original ones, is displayed in Figure 8.2. In these plots, the smoothing effect of the transformation to the basis representations is apparent.

8.2.4 Interpolation and extrapolation

The **scikit-fda** package provides a variety of interpolation methods for functional data in discretized form. By default, linear interpolation is performed. Other types of interpolation can be specified in the attribute `interpolation` of the `FDataGrid` object. Specifically, class `SplineInterpolation` offers support for spline interpolation. It is also possible to employ other interpolation and extrapolation strategies defined by the user. An example showing how to define such custom strategies is available at `https://fda.readthedocs.io/create_new_interpolation`.

Different extrapolation strategies for `FDataGrid` and `FDataBasis` objects are available in **scikit-fda**. In particular, it is possible to specify a constant value (for instance,

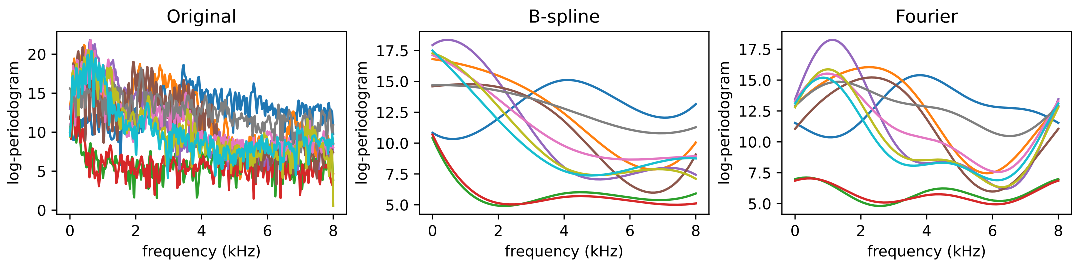


FIGURE 8.2: Different representations of the first ten trajectories of the *Phoneme* dataset. From left to right: original trajectories, B-spline, and Fourier basis representation. In both cases, 5 basis functions are considered

the value of the function at one of the limits of the domain), or to assume a periodic structure. In the case of `FDataBasis` objects, one can also directly evaluate the basis expansion outside the domain. Finally, as in interpolation, user-defined extrapolation strategies can be utilized.

8.2.5 Derivatives

The computation of derivatives is of particular importance in functional data analysis. For instance, derivatives can reveal significant information that is not apparent in the original curves. Furthermore, the norm of a derivative is a natural measure of the function's roughness (Ramsay and Silverman, 2005). For this reason, they are often employed to define penalties for regularization. In the **scikit-fda** package, the method `derivative()` can be used to perform this operation for both `FDataGrid` and `FDataBasis` objects. In the case of `FDataGrid` objects, derivatives are approximated using finite differences. For `FDataBasis` objects, they are computed exactly in terms of the derivatives of the basis functions. Therefore, if a new type of basis is designed, it is necessary to implement the derivatives of the basis functions in the corresponding class.

8.2.6 Regularization

Regularization methods consist in favoring simpler models to improve the quality and robustness of the solutions of an optimization problem. In FDA, regularization is used to obtain smooth functional approximations to noisy discrete data, for registration, and for principal component analysis, among others. For the purpose of regularization, the complexity of a function can be quantified in terms of its norm, or of a linear transformation thereof (e.g., the function derivatives). A penalty term proportional to this measure of complexity is then added to the cost function to be minimized. The package **scikit-fda** provides the necessary infrastructure to implement regularization based on the L^2-norm of the function in class `L2Regularization`. Alternatively, a linear operator can be passed as a parameter to the constructor of `L2Regularization` objects. Some common linear operators are readily available in **scikit-fda**'s operators module for that purpose. The following code illustrates this type of regularization to obtain smooth representations of a set of functions in the basis of B-splines. In this example, the L^2-norm of second derivatives of the trajectories is used to penalize their curvature.

```python
import skfda

```

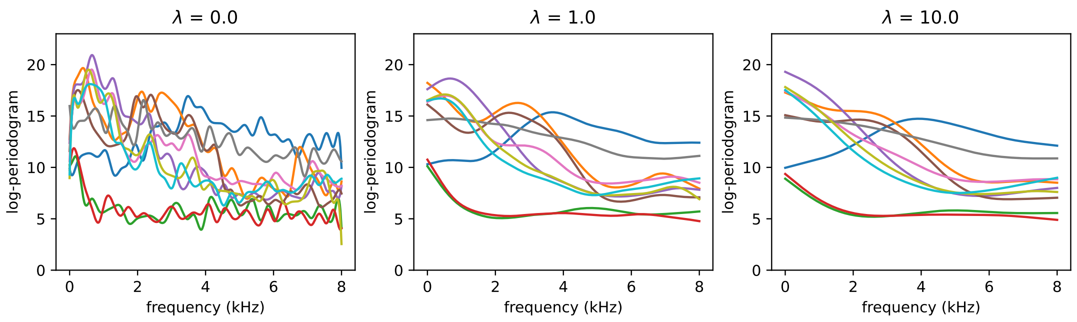


FIGURE 8.3: Smoothed representation of the first ten trajectories of the *Phoneme* dataset in a B-spline basis with 40 basis functions for different values for the regularization parameter $\lambda \geq 0$; from left to right: $\lambda = 0$ (no regularization), $\lambda = 1$, and $\lambda = 10$.

```
X, y = skfda.datasets.fetch_phoneme(return_X_y=True)
X = X.coordinates[0]

basis = skfda.representation.basis.BSpline(
    domain_range=X.domain_range,
    n_basis=40,
)

regularization = skfda.misc.regularization.L2Regularization(
    skfda.misc.operators.LinearDifferentialOperator(2),
    regularization_parameter=1,
)

smoother = skfda.preprocessing.smoothing.BasisSmoother(
    basis=basis,
    regularization=regularization,
    return_basis=True,
)

X_basis = smoother.fit_transform(X)
```

The effect of this type of smoothing is illustrated in Figure 8.3. In this figure, 10 trajectories of the *Phoneme* dataset are represented in a B-spline basis composed of 40 basis functions for different values for the regularization parameter λ; from left to right: $\lambda = 0$ (no regularization), $\lambda = 1$, and $\lambda = 10$. As shown in the left plot of this figure, when the number of basis functions is large, the non-regularized trajectories represented in this basis still exhibit significant fluctuations. Progressively smoother basis representations are obtained as the *regularization parameter* $\lambda > 0$ increases.

8.3 Functionality of scikit-fda

In this section, an overview of the utilities for functional data analysis provided by the **scikit-fda** package is given. The first step in the analysis is to generate functional datasets or to retrieve them from external sources. In the **scikit-fda** package it is possible to generate synthetic data, to simulate random trajectories from stochastic processes, and to load data from files in standard formats and from repositories of real-word datasets. The library provides also an extensive set of tools for the analysis of functional data both in discretized and basis expansion forms. In particular, it offers methods for exploratory analysis, smoothing, registration, dimensionality reduction, the computation of functional depths, outlier detection, interactive visualization, and machine learning, among others. An important feature of **scikit-fda** is the integration with the extensive collection of **scikit-learn**'s tools for machine learning, including data preprocessing, training, testing, and hyperparameter selection. Specifically, the methods provided are designed so that they can be utilized in **scikit-learn** *pipelines*. In what follows these functionalities will be described in detail.

8.3.1 Generation of synthetic data

A variety of methods for the generation of functional data, either from some simple models or sampled from stochastic processes, are available in **scikit-fda**. In particular, the function `make_multimodal_samples()` can be used to generate functions with several maxima. This is used in the synthetic registration example illustrated in Figure 8.7. The function `make_gaussian_process()` can be used to simulate trajectories sampled from a Gaussian process with a specified mean and covariance function. Several commonly employed covariance functions, such as Brownian, exponential, radial basis function (RBF), Matérn, and polynomial kernels are supplied in the package. Additional types of covariance functions can be defined by the user.

The following code illustrates the generation of 50 trajectories from standard Brownian Motion, an Ornstein-Uhlenbeck process (a Gaussian process with an exponential covariance function), and a Gaussian process with an RBF covariance function. A regular grid of 100 equally spaced points in $\mathcal{T} = [0, 1]$ is employed for the discretized representation. The simulated trajectories are displayed in Figure 8.4.

```python
import skfda
from skfda.misc.covariances import Brownian, Gaussian, Exponential

cov_dict = {
    "Brownian": Brownian(variance=1),
    "Exponential": Exponential(variance=1, length_scale=1),
    "Gaussian (RBF)": Gaussian(length_scale=0.1),
}

for i, (name, cov) in enumerate(cov_dict.items()):

    fd = skfda.datasets.make_gaussian_process(
        n_samples=50,
        n_features=100,
        mean=0,
        cov=cov,
        random_state=0,
    )

    fd.plot()
```

8.3.2 Real-world data

The package **scikit-fda** provides tools to retrieve functional datasets from other libraries and public access repositories. The datasets themselves are retrieved using the package **scikit-datasets** (Díaz-Vico and Ramos-Carreño, 2022), explained in Chapter 7. The data are downloaded only once and cached on disk, so as to reduce network traffic and make it possible to work with the data offline. They are then converted to the `FData` format for further processing and analysis. For instance, the function `fetch_cran()` can be used to retrieve datasets from R packages in the

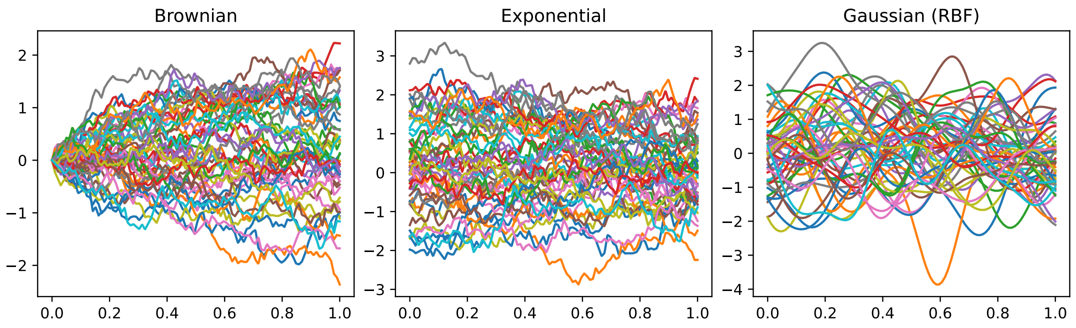


FIGURE 8.4: Trajectories sampled from Gaussian processes with different covariance functions. From left to right, standard Brownian, Exponential ($l = 1$) and RBF ($l = 0.1$).

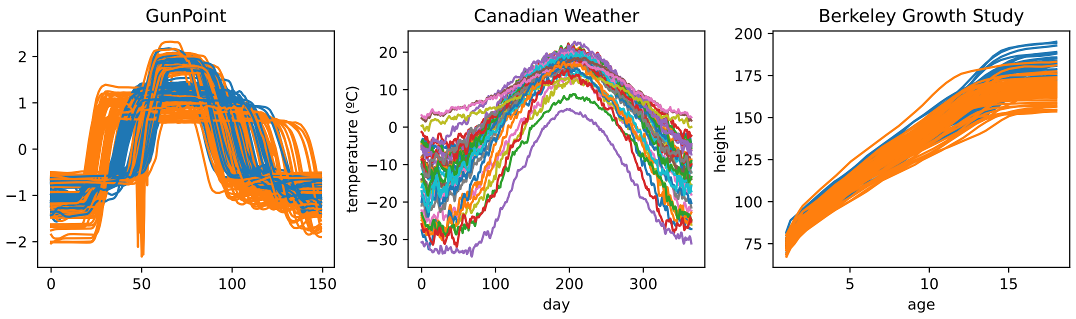


FIGURE 8.5: Real datasets fetched with **scikit-fda**. From left to right: *GunPoint*, *Canadian Weather*, and the *Berkeley Growth Study*.

CRAN repository. Datasets from the *UCR & UEA Time Series Classification Archive* (Dau et al., 2019; Bagnall et al., 2018; Bagnall et al., 2021) can be accessed making use of the function `fetch_ucr()`. Moreover, there exist specific functions to import some widely-used datasets (see Chapter 3) such as `fetch_growth()` for the *Berkeley Growth Study* dataset (Tuddenham and Snyder, 1954) and `fetch_weather()` for the *Canadian Weather* dataset (Ramsay and Silverman, 2005).

The following code illustrates the functionality described for three well-known datasets: *GunPoint* from the UCR repository, *Canadian Weather*, and the *Berkeley Growth Study*. **scikit-fda**'s functions are used to load the data and plot some of the datasets' trajectories. Note that the *Canadian Weather* dataset has two codomain dimensions ($q = 2$): temperature and precipitation. In this example, the first is selected using the `coordinates` property of an `FData` object.

```python
import skfda

dataset = skfda.datasets.fetch_ucr("GunPoint")
dataset["data"].plot(group=dataset["target"])

X, _ = skfda.datasets.fetch_weather(return_X_y=True)
X.coordinates[0].plot()

X, y = skfda.datasets.fetch_growth(return_X_y=True)
X.plot(group=y)
```

8.3.3 Preprocessing

Functional observations often need to be subject to some form of processing to facilitate ulterior manipulation. To this end, the **scikit-fda** package provides utilities for smoothing, registration, and dimensionality reduction. In what follows, these utilities are described in detail.

Smoothing

Smoothing consists in replacing the original observation $x(t)$ with a smoothed version $\hat{x}(t)$. This replacement yields a more manageable, possibly more faithful, representation of the underlying process. In particular, smoothing can be used to recover the signal component from noisy measurements. Furthermore, smoothed approximations with continuous derivatives can be used for regularization (Wang, Chiou, and Müller, 2016). The methods of **scikit-fda**'s classes `BasisSmoother` and `KernelSmoother` can be employed to this end. The package provides also utilities to determine an appropriate degree of smoothing using some form of statistical validation.

As discussed in Section 8.2, the approximation of a function by a truncated basis expansion, as in Equation (8.3), is a form of smoothing. The coefficients of the finite basis expansion can be estimated by least squares (Ramsay and Silverman, 2005). This kind of smoothing is performed in **scikit-fda** by the class `BasisSmoother`. Further smoothing can be achieved by a regularization approach based on roughness penalties, as described in Section 8.2.6 (Green and Silverman, 1993; Ramsay and Silverman, 2005).

Smoothing can be achieved also by performing a linear transformation of the original functional observations

$$\hat{x}(t) = \int_{\mathcal{T}} s_t(\tau)x(\tau)d\tau. \tag{8.4}$$

The weighting function $s_t(\tau)$ quantifies the contribution of the value of the function at τ to the smoothed value at t. This weighting function should be localized, so that the values of the function at points close to t contribute more to the average.

For functional data in discretized form, Equation (8.4) can be expressed as a matrix transformation

$$\hat{\mathbf{x}} = \mathbf{S}\mathbf{x}, \tag{8.5}$$

where $\mathbf{x} = x(\mathbf{t})$ is the vector of values of the function at the discretization points $\mathbf{t} = (t_1, \ldots, t_M)$, the vector $\hat{\mathbf{x}}$ consists of the smoothed function values at a grid of points, which can be different from the original ones, and $\mathbf{S}$ is the smoothing matrix. By default, the grid at which the smoothed values are computed is $\mathbf{t}$, the set of sampling points. The smoothing matrix $\mathbf{S}$ is sometimes referred to as the “hat” matrix, a name borrowed from regression analysis because it transforms the dependent variable vector $\mathbf{x}$ into its fitted version $\hat{\mathbf{x}}$ (Ramsay and Silverman, 2005).

Smoothing with local weights can be implemented using kernels (Wasserman, 2006). Kernel smoothing is implemented in class `KernelSmoother`. It receives as a parameter the estimator of the hat matrix, a subclass of the class `HatMatrix`. The library **scikit-fda** provides three of these: Nadaraya-Watson (`NadarayaWatsonHatMatrix`), local linear regression (`LocalLinearRegressionHatMatrix`), and k nearest neighbors

(`KNeighborsHatMatrix`) estimators. As an illustration, for the Nadaraya-Watson estimator, the hat matrix is

$$S_{ij}(h) = \frac{K\left(\frac{t_i - t_j}{h}\right)}{\sum_{m=1}^{M} K\left(\frac{t_i - t_m}{h}\right)}, \quad 1 \le i, j \le M, \tag{8.6}$$

where K is the kernel function and h is the parameter that controls the degree of smoothing. Commonly used kernels, such as Gaussian, uniform, and Epanechnikov, are available in **scikit-fda**. It is also possible to employ user-defined kernels for this type of smoothing.

In these methods, the value of the smoothing parameter needs to be carefully adjusted to avoid under- or oversmoothing. In **scikit-fda** this value can be determined by statistical validation with the help of the class `SmoothingParameterSearch`. Particular scoring criteria, such as classes for the computation of leave-one-out cross-validation (`LinearSmootherLeaveOneOutScorer`) or, alternatively, generalized cross-validation (`LinearSmootherGeneralizedCVScorer`) are provided to guide the search. The criterion that is maximized in leave-one-out cross validation is

$$CV_{loo}(h) = \frac{1}{M} \sum_{m=1}^{M} \left(\frac{x(t_m) - \hat{x}(t_m; h)}{1 - S_{mm}(h)} \right)^2, \tag{8.7}$$

where h is the smoothing parameter and $\hat{x}(t_m; h)$ is the smoothed value.

The generalized cross validation (GCV) criterion is

$$GCV(h) = \Xi(\mathbf{S}(h)) \frac{1}{M} \sum_{m=1}^{M} \left(x(t_m) - \hat{x}(t_m; h) \right)^2, \tag{8.8}$$

where Ξ is a penalty function. By default, the penalty is

$$\Xi(\mathbf{S}(h)) = \frac{1}{(1 - \mathrm{trace}(\mathbf{S}(h))/M)^2}. \tag{8.9}$$

Additional penalty functions, such as Akaike's information criterion (AIC), implemented as `akaike()`, or Shibata's model selector, implemented as `shibata()`, are provided as well.

In the code that follows, the smoothing functionality provided by **scikit-fda** is illustrated using the functions of the *Phoneme* dataset (Hastie, Tibshirani, and Friedman, 2009), which are rather noisy.

```python
import skfda
from skfda.preprocessing.smoothing import (
    KernelSmoother,
    validation,
)
from skfda.misc.hat_matrix import KNeighborsHatMatrix

X, y = skfda.datasets.fetch_phoneme(return_X_y=True)

grid = validation.SmoothingParameterSearch(
    KernelSmoother(KNeighborsHatMatrix()),
    [2, 3, 4, 5],
    scoring=validation.LinearSmootherGeneralizedCVScorer(
        validation.shibata,
    ),
    param_name="kernel_estimator__n_neighbors"
```

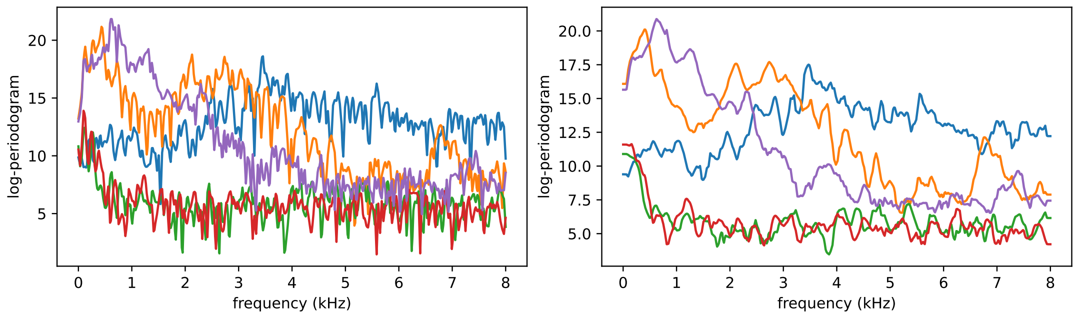


FIGURE 8.6: Five trajectories of the *Phoneme* dataset before (left) and after (right) nearest neighbors smoothing.

```
)

grid.fit(X)
X_smooth = grid.transform(X)
```

In this example, the k nearest neighbors kernel estimator is used to smooth the data. The optimal number of neighbors is selected between the values 2, 3, 4, and 5. Finally, the `LinearSmootherGeneralizedCVScorer` with the `shibata()` penalty function is used for model selection. The first five original curves and the smoothed ones are displayed in Figure 8.6.

Registration

Another type of preprocessing, which is especially relevant in FDA, is registration. As explained in Section 2.3.1, registration consists in applying transformations to the raw data so that the functional observations are properly aligned. There is a variety of reasons why misalignment can occur. In some cases, it is the result of errors in the measurement process. In others, the domain has to be warped because the functions depend on an internal parameter, which is different from the one observed. For periodic functions, such as the signal of a heartbeat, the starting time for the different measurements could be different. A number of strategies can be used for registration. For instance, maxima, minima, zeros, and other landmarks can be used as reference points for alignment. Alternatively, some measure of dispersion between the observations can be minimized. It is also possible to register a set of functional observations to a reference function. After registration, it may be necessary to evaluate the functional observations at points in the domain that are different from the ones in the original grid. This can be made utilizing the interpolation and extrapolation techniques described in Section 8.2.4. To carry out such an alignment, the package **scikit-fda** offers support for shift registration, and for elastic registration.

Shift registration consists in aligning the functional observations by a translation

$$\tilde{x}_i(t) = x_i(t + \delta_i), \quad i = 1, \dots, N, \tag{8.10}$$

where δ_i is the time shift applied to $x_i(t)$, and $\tilde{x}_i(t)$ is the registered function (Ramsay and Silverman, 2005). Shifting modifies the lower and upper bounds of the interval on which the function observations are defined. The values of the shifted functions that lie outside the original interval are discarded. For the subinterval in which the function values are not available, they are estimated by extrapolation. The method for extrapolation can be provided as an input.

The shifting constants $\{\delta_i\}_{i=1}^N$ can be determined using different procedures. If a single landmark, such as a maximum, a minimum, or a zero crossing, is present in every curve, and their locations, $\{\tau_i\}_{i=1}^N$, are known, then the i-th curve can be shifted by $\delta_i = \tau_i - \tau^*$. After registration, the location of the landmark is τ^* for every curve,

$$\tilde{x}_i(\tau^*) = x_i(\tau_i), \quad i = 1, \dots, N. \tag{8.11}$$

In **scikit-fda**, the function `landmark_shift_registration()` can be used to carry out this transformation. The function `landmark_shift_deltas()` can be used to retrieve the values of the δ_i.

Alternatively, the values δ_i can be computed by minimizing a least squares criterion (Ramsay and Silverman, 2005)

$$REGSSE = \sum_{i=1}^{N} \int_{\mathcal{T}} [\tilde{x}_i(t) - \hat{\mu}(t)]^2 dt, \tag{8.12}$$

where $\hat{\mu}(t)$ is the sample mean of the registered data $\{\tilde{x}_i(t)\}_{i=1}^N$. This type of shift registration can be performed using class `LeastSquaresShiftRegistration`. Instead of the sample mean, which is the default value, a user-defined template function can be employed. In this case, the values for the δ_i are stored as the attribute `deltas_` after the registration.

Another type of registration available in the package **scikit-fda** is elastic registration. In elastic registration, one attempts to align the data by applying a warping transformation

$$\tilde{x}_i(t) = x_i(\gamma_i(t)), \quad i = 1, \dots, N. \tag{8.13}$$

The warping γ_i is a monotonically increasing function defined in $\mathcal{T} = [a, b]$. Assuming that the values of the function at the endpoints of this interval are fixed, it obeys the constraints $\gamma_i(a) = a$ and $\gamma_i(b) = b$. If the locations of some landmarks are known, the warping function for elastic registration can be approximated by monotonically increasing splines (Ramsay and Silverman, 2005). Besides the boundary constraints specified earlier, the spline interpolator for the i-th functional observation has to satisfy

$$\tilde{x}_i(\tau_l^*) = x_i(\tau_{il}), \quad l = 1, \dots, L, \tag{8.14}$$

where $\{\tau_{il}\}_{l=1}^L$ are the landmark locations in the i-th observation, and $\tau_l^* = \gamma_i^{-1}(\tau_{il})$ is the location of the l-th landmark in the registered curves. In **scikit-fda** this type of registration can be carried out using the `landmark_elastic_registration()` procedure. Function `landmark_elastic_registration_warping()` can be used to retrieve the warpings themselves. A drawback of this approach is that the landmarks and their locations need to be identified beforehand.

An alternative type of elastic registration is to align the observations to a reference template. This has the advantage that no information on landmarks is needed. In the elastic registration method described in Srivastava et al. (2011), the template is defined in terms of the Karcher mean under the Fisher-Rao metric (Srivastava and Klassen, 2016). Then, an energy function depending on the Fisher-Rao distance between each curve and the template is minimized. To efficiently compute this distance, the square root velocity function (SRVF) transform is introduced (Joshi et al., 2007). The main reason for introducing this transform is that the Fisher–Rao distance between two functions is given by the L^2 distance between their SRVF representations. The **scikit-fda**'s class `FisherRaoElasticRegistration` includes methods for this particular type of registration. The warping functions used are stored in the

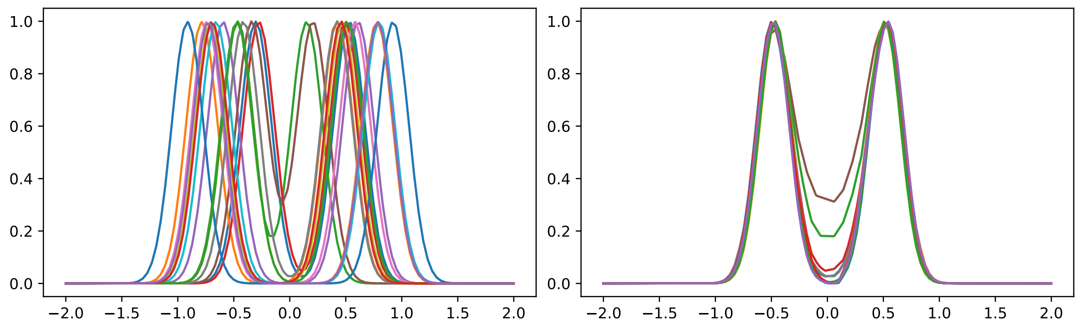


FIGURE 8.7: Elastic Fisher-Rao registration with synthetic data: The original curves are shown in the left panel. The registered curves are displayed in the right panel.

`warping_` attribute of this class. The implementation makes use of the dynamic programming routines for alignment to a template from the Python package **fdasrsf** (Tucker, 2020a). This type of elastic registration is illustrated in Figure 8.7 with synthetic data. The original trajectories, which are displayed on the left plot, exhibit two local maxima whose relative locations are different in each of the curves. In consequence, alignment cannot be achieved via a simple shift. Note that after this type of elastic registration curves are well aligned even without previous information about the location of the landmarks.

The results of applying shift registration by least squares and elastic Fisher-Rao registration to the *Berkeley Growth Study* data (Tuddenham and Snyder, 1954) are compared in Figure 8.8. This figure has been generated using the following code:

```python
import skfda
from skfda.preprocessing.registration import (
    ElasticRegistration,
    ShiftRegistration,
)

X, y = skfda.datasets.fetch_growth(return_X_y=True)

X_aligned_elastic = ElasticRegistration().fit_transform(X)
X_aligned_shift = ShiftRegistration().fit_transform(X)

X.plot()
X_aligned_shift.plot()
X_aligned_elastic.plot()
```

The curves displayed in the left panel of Figure 8.8 trace the evolution of the heights of 54 girls and 39 boys since their birth until their 18th birthday. The nominal ages at which the measurements are made coincide for all individuals. However, each child has a different growth profile. In particular, landmark features manifest themselves at different ages. For instance, even though most curves exhibit a growth spurt at puberty, the precise ages at which this occurs are different for each individual. Therefore, an elastic deformation of the actual age axis may uncover an internal age, which is more meaningful from a biological perspective. The effects of shift and elastic registration based on the Fisher-Rao distance are displayed in the middle and right panels of Figure 8.8 respectively.

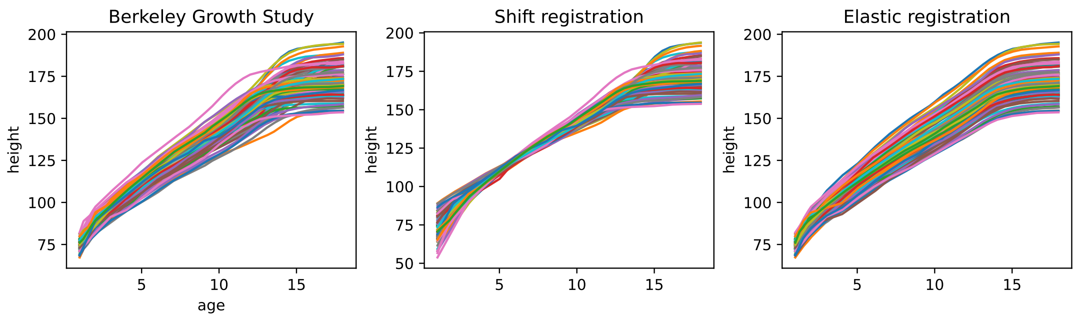


FIGURE 8.8: Registration of the *Berkeley Growth Study* data with different methods. From left to right: the original curves, shift registration by least squares and elastic Fisher-Rao registration.

Dimensionality reduction

Let us recall the relevance of dimensionality reduction techniques in FDA, as explained in Section 2.3.2. Functional data are infinite-dimensional objects. Even in the case that they are represented by a set of discrete measurements, their dimensionality is typically very high. In addition, nearby observations exhibit a large degree of dependence. Due to these characteristics, technical and computational difficulties arise in the analysis of these types of data. To alleviate such difficulties one can represent the functional data in a lower-dimensional space while preserving as much information as possible. The use of dimensionality reduction methods leads to gains in efficiency and, in some cases, improvements in interpretability and predictive capacity. Furthermore, in this lower-dimensional representation, the methods of multivariate statistics can be employed (Vieu, 2018).

A simple dimensionality-reduction method is to select a set of impact or design points that are relevant for the task at hand; for instance, the most informative points for clustering, classification, or regression (Delaigle, Hall, and Bathia, 2012; Ferraty, Hall, and Vieu, 2010; Kneip, Poss, and Sarda, 2016). Specifically, **scikit-fda**'s class `EvaluationTransformer` can be used to evaluate the functions as a set of points in the domain of the function. Alternatively, a truncated basis representation can be used (Biau, Bunea, and Wegkamp, 2005; Poskitt and Sengarapillai, 2013). The coefficients of a functional data object represented as a basis expansion can be extracted using the methods of the class `CoefficientsTransformer`. Besides these, the **scikit-fda** package provides methods for functional principal components analysis (FPCA) and variable selection methods. These types of methods are described in what follows.

Functional principal components analysis. Functional principal component analysis (FPCA) is a widely used dimensionality reduction method in FDA. With this procedure, the individual functions are represented in the orthonormal basis of eigenfunctions of the stochastic process' covariance operator. Dimensionality reduction is achieved by retaining the projections of the original functions onto the subspace of L^2 spanned by the set of eigenfunctions that correspond to the largest eigenvalues. This representation is the one, among those of the same dimension, that explains the most of the data's variance.

A random function $X \in L^2(\mathcal{T})$ can be represented as

$$X(t) = \mu(t) + \sum_{b=1}^{\infty} \xi_b \phi_b(t), \tag{8.15}$$

where $\mu(t) = \mathbb{E}[X(t)]$, $\phi_b(t)$ is the b-th principal component, and $\xi_b = \int_{\mathcal{T}}(X(t) - \mu(t))\phi_b(t)dt$, denotes the projection (score) along the b-th principal component. By the Karhunen-Loève Theorem, the scores $\{\xi_b\}_{b\geq 1}$ are uncorrelated random variables (Wang, Chiou, and Müller, 2016). Smoothed versions of the principal components can also be computed by applying the regularization penalties described in Section 8.2.6, using the procedure described in Section 9.4.2 of Ramsay and Silverman, 2005. The smooth principal components are obtained by optimizing a function that takes into account not only the sample variance but also a term that penalizes the roughness of the principal components. A reduction of the dimension of the data can be achieved by truncating the basis expansion in Equation (8.15), so that only the first B components are included

$$X(t) = \mu(t) + \sum_{b=1}^{B} \xi_b \phi_b(t). \tag{8.16}$$

In **scikit-fda**, functional principal component analysis can be carried out using the methods of class `FPCA`. The following code illustrates this functionality for the *Berkeley Growth Study* data:

```python
import skfda
import matplotlib.pyplot as plt

X, y = skfda.datasets.fetch_growth(return_X_y=True)

fpca = skfda.preprocessing.dim_reduction.feature_extraction.FPCA(
    n_components=2,
)
fpca.fit(X)

skfda.exploratory.visualization.fpca.FPCAPlot(
    X.mean(), fpca.components_, multiple=30,
).plot()

scores = fpca.transform(X)
scores_class_0 = scores[y==0]
scores_class_1 = scores[y==1]

plt.figure()
plt.scatter(scores_class_0[:, 0], scores_class_0[:, 1])
plt.scatter(scores_class_1[:, 0], scores_class_1[:, 1])
```

In this example, the first two principal components are computed. Then, the functional observations are projected onto the two-dimensional subspace spanned by these components. A numerical quadrature is used to compute the corresponding inner products. Class `FPCAPlot` is used to display the curves $\{\mu(t) \pm \phi_b(t);\ b = 1,2\}$, which are the result of adding and subtracting the b-th eigenfunction to the sample mean. In the case of the Berkely growth study, the first component captures overall deviations (either positive or negative) with respect to the mean. The second one reveals patterns associated to differences of growth speed. In particular, it exhibits a maximum followed by a sign change at around puberty. The resulting plots are shown in the left and middle panels in Figure 8.9. Finally, the projection of the curves onto the first two principal components is obtained using the `transform` method of the class `FPCA`. The scores of these two components are displayed as points in the right panel of Figure 8.9. Note that, with some exceptions, boys (blue) and girls (orange) appear grouped in two separate clusters in this plot.

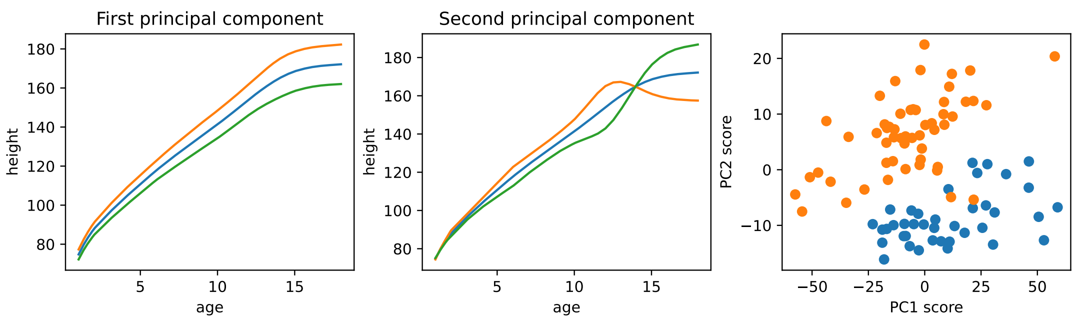


FIGURE 8.9: Principal components analysis for the *Berkeley Growth Study* data. The first and second principal components are plotted in the leftmost panel as perturbations around the mean (in blue). In the right panel, the scores of individual functional observations for the first two components are plotted. The orange and blue points correspond to girls' and boys' growth curves, respectively.

Variable selection. The package **scikit-fda** provides a variety of tools to carry out variable selection. A simple approach is to apply a multivariate variable selection method to the discretized representation of the functional observations (Berrendero, Cuevas, and Torrecilla, 2016a; Jiménez-Cordero and Maldonado, 2021). The **scikit-fda** class `EvaluationTransformer` can be used to transform `FData` objects into **NumPy** arrays. Then, any Python library for multivariate variable selection can be used. If this approach does not take into account the functional nature of the data, there can be difficulties in the analysis (Aneiros and Vieu, 2016). A multivariate method that takes into account the redundancy that arises from the continuity of functional data is minimum-redundancy-maximum-relevance (mRMR) (Ding and Peng, 2005; Peng, Long, and Ding, 2005; Berrendero, Cuevas, and Torrecilla, 2016a). This method is available in **scikit-fda**, as the `MinimumRedundancyMaximumRelevance` class. The dependence measures that quantify the relevance and the redundancy can be specified by the user.

In addition, the **scikit-fda** package includes a collection of variable selection methods that specifically take into account the functional nature of the data: reproducing kernel-based variable selection (RKVS), maxima hunting (MH), and recursive maxima hunting (RMH).

The RKVS method, implemented in the class `RKHSVariableSelection`, was introduced for binary classification problems (Berrendero, Cuevas, and Torrecilla, 2018). For a specified value of P, the goal is to identify the set of points $\mathbf{t} = (t_1, \ldots, t_P)^\top \in \mathcal{T}^P$ and select the corresponding function values, $X(t_1), \ldots, X(t_P)$, that maximize the Mahalanobis distance between groups

$$(\mu_1(\mathbf{t}) - \mu_0(\mathbf{t}))^\top k(\mathbf{t}, \mathbf{t})^{-1} (\mu_1(\mathbf{t}) - \mu_0(\mathbf{t})), \tag{8.17}$$

where $\mu_0(\mathbf{t}), \mu_1(\mathbf{t})$, and $k(\mathbf{t}, \mathbf{t})$ are the mean functions of each class and the covariance function evaluated at $t_1, \ldots, t_P$, respectively. In homoscedastic binary classification problems, with a fixed dimension P this selection is optimal in terms of classification error. In practice, the exploration of all possible combinations is often infeasible. To reduce the computational costs, a greedy search is implemented.

In MH (Berrendero, Cuevas, and Torrecilla, 2016b; Ordóñez et al., 2018) one selects the values of $t \in \mathcal{T}$ that correspond to local maxima of a non-negative dependence measure between $X(t)$ and the class label. The selected variables are thus the most relevant in a region. Furthermore, the values of $X(t)$ that are close to these local maxima, which generally provide redundant information, are automatically discarded. In Berrendero, Cuevas, and Torrecilla, 2016b, the distance correlation (Székely, Rizzo, and Bakirov, 2007) is used as the dependence measure. This variable selection method is implemented in the class `MaximaHunting`, using the implementation of distance correlation function provided by the **dcor** package (Ramos-Carreño, 2020), that will be explained in Chapter 9. MH is an interpretable, fully functional method with optimal performance in an important class of functional classification problems. In spite of its simplicity and good performance, MH has some limitations. Specifically, there can be numerical difficulties to identify the local maxima of the depence measure. Futhermore, MH takes into account only the marginal relevance of a single variable. Variables that are only relevant when selected in combination with other cannot be identified by these procedures.

RMH (Torrecilla and Suárez, 2016) addresses these limitations by assuming a particular form of the stochastic process from which the trajectories are sampled. The algorithm proceeds as follows: First the value of t that is the global maximum of the dependence between the variable $X(t)$ and the class label is selected. Let t^* be such optimum and, therefore, $X(t^*)$ the variable selected. The information conveyed by $X(t^*)$ is removed by subtracting from the trajectories the conditional expectation of the process given the value of the selected variable. Then, the global maximum of the resulting process is identified. The algorithm proceeds in this iterative manner until a pre-specified number of variables have been selected, or until a convergence criterion is fulfilled. A more complete explanation and analysis of the algorithm is provided in Chapter 5. In **scikit-fda**, the class `RecursiveMaximaHunting` provides an enhanced, very customizable implementation of this method.

8.3.4 Exploratory analysis

Exploratory analysis methods are used to identify salient features, visualize, and describe the data from a statistical point of view. Specifically, the **scikit-fda** package provides tools for the computation of summary statistics, including robust ones, interactive tools for visual analysis, and outlier detection.

Summary statistics

Common summary statistics, such as the sample mean function and the sample covariance function can be estimated using the tools provided by **scikit-fda**. Consider a set of functional observations $\{x_i(t)\}_{i=1}^{N}$. The sample mean,

$$\hat{\mu}(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t), \tag{8.18}$$

can be computed by applying the function `mean()` to the `FData` object in which the data are stored. The functional observations can be either in discrete form or in a basis representation. The resulting mean function is a `FData` object of the same type as the input (i.e., discretized or in a basis representation).

The sample covariance function $\hat{k}$,

$$\hat{k}(t,s) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i(t) - \hat{\mu}(t))(x_i(s) - \hat{\mu}(s)), \tag{8.19}$$

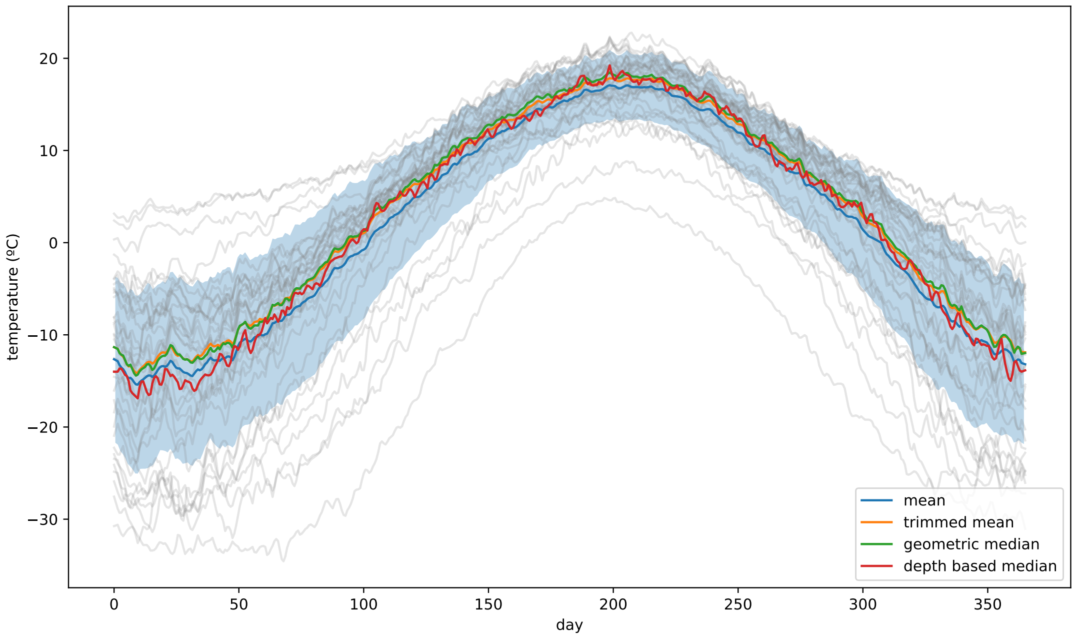


FIGURE 8.10: Centrality statistics of the *Canadian Weather* dataset. The shaded band corresponds to one standard deviation around the mean.

can be computed applying the function `cov()` to the corresponding `FData` object. Similarly, the function `var()` can be used to computed the sample variance, $\hat{k}(t,t)$. Irrespective of the representation of the functional observations, the sample variance and covariance are returned in discretized form. Instead of the functions `mean()`, `cov()` and `var()`, the `FData` methods of the same name can be used to compute these summary statistics.

Figure 8.10 presents an illustration of this functionality for the *Canadian Weather* dataset. The sample estimate of the mean temperature curves is shown as a blue curve. The shaded area corresponds to one standard deviation around the estimated mean. Other measures of centrality, such as the trimmed mean, the geometric mean, and the median, are displayed in this figure as well. These robust statistics will be described in some detail next in this section.

Depth measures

Depth measures quantify the centrality of a function in relation to a set of functions. These measures are used for exploratory analysis, to compute robust statistics, detect outliers, and for data visualization (e.g., the functional box-plot). In contrast to the univariate case, a variety of definitions of functional depth can be given that yield different orderings of the functional observations in the sample. Each of these functional depths lead to different definitions of robust statistics and of degrees of outlyingness.

Some common functional depth measures are implemented in the **scikit-fda** package. In particular, the methods of class `IntegratedDepth` can be used to compute integrated depth measures (Fraiman and Muniz, 2001), which are averages of univariate depths. Specifically, the integrated depth of the function x is

$$\mathrm{ID}(x) = \int_{\mathcal{T}} D(x(t))dt. \tag{8.20}$$

where D is an univariate depth function. This function can be selected by the user. The default is the measure proposed by Fraiman and Muniz, 2001:

$$D(x(t)) = 1 - \left| \frac{1}{2} - F_{X(t)}(x(t)) \right| , \tag{8.21}$$

where $F_{X(t)}$ denotes the distribution function of the marginal.

An alternative definition is the band depth (BD), introduced by López-Pintado and Romo, 2009. To compute this functional depth, one needs to identify the bands that are delimited by all possible pairs of functional observations in the sample. The BD value is the fraction of bands that completely encompass the curve. In **scikit-fda**, this quantity can be computed using methods of the class `BandDepth`. A related, less restrictive measure, is the modified band depth (MBD). This measure takes into account not only the number of bands that contain x, but also the time that x lies within each band. The MBD has better statistical properties than the original BD, in part because it is an integrated depth measure (Nagy et al., 2016). In **scikit-fda**, MBD is implemented in the class `ModifiedBandDepth`.

Robust statistics

The package **scikit-fda** provides support for the computation of robust statistics. Robust statistics may provide a better characterization of the data than non-robust ones (e.g., the mean or the covariance functions), especially in the presence of outliers. One of the most important robust statistics is the geometric median (Lardin-Puech, Cardot, and Goga, 2014)

$$\text{median} = \arg\min_{z \in \mathcal{X}} \sum_{i=1}^{N} \|x_i - z\| . \tag{8.22}$$

It can be computed with the function `geometric_median()`. Alternatively, the median can be defined as the deepest point in the sample. Different depth measures yield different definitions of the median. These types of medians can be computed with the function `depth_based_median()`.

Functional depth measures can be used also to define the degree of outlyingness of a function in a sample: the larger the depth value the more central the functional observation is. Finally, funcional depth measures can be used to define trimmed means (Fraiman and Muniz, 2001). A trimmed mean is a robust version of the standard mean in which the most outlying functional observations (the ones with the lowest depth values) are discarded. In **scikit-fda**, the trimmed mean is implemented in function `trim_mean()`.

The geometric median, the MBD-based median, and the MBD-based trimmed mean in which 10% of the data are discarded, of the *Canadian Weather* dataset are shown in Figure 8.10.

Interactive visualization tools and outlier detection

Visualization tools can be used to gain insight into the data. In particular, trends, salient features, and other patterns in the data can be identified simply by inspection. Visualization tools can be utilized also to single out functional observations that are markedly different from the other observations in the sample (outliers). Outlier detection is useful to identify rare events, novel patterns, anomalies, or erroneous measurements. The package **scikit-fda** provides a number of interactive tools for data visualization and outlier detection. Their implementation utilizes the functionality provided by **matplotlib** (Hunter, 2007).

Functional data objects have a `plot()` method that can be used to graph the curves. Some customization options, such as group colors or labels, are available for this method. An illustration of its use with the *Berkeley Growth Study* dataset is shown on the left-hand panel of Figure 8.11. For `FDataGrid` objects, the `scatter()` method can be used to display the values of the function as individual points in a graph. This method was used to generate Figure 2.1.

Another tool for visual exploration provided by **scikit-fda** is the functional boxplot (Sun and Genton, 2011). This is an generalization of the univariate boxplot for functional data. The functional boxplot consists of a graph of the functional median (i.e., the deepest curve in the sample) surrounded by a central envelope, which encompasses the deepest 50% of the observations, and a maximum non-outlying envelope. The width of this outer envelope is determined by scaling the central one by a constant factor. This constant factor can be selected by the user. Its default value is 1.5. In **scikit-fda**, the class `Boxplot` can be used to generate and customize functional boxplots. In this plot, a trajectory is marked as an outlier if it lies beyond the maximum non-outlying envelope for some interval. The class `BoxplotOutlierDetector` can be used for outlier detection based on this criterion. Some customizable elements of `Boxplot` objects are the depth measure, and the definition of centered bands that encompasses a user-specified fraction of the deepest observations. The following code provides an illustration of these functionalities with the *Berkeley Growth Study* dataset. The plots that result from the execution of this code are displayed in Figure 8.11.

```python
import skfda

X, _ = skfda.datasets.fetch_growth(return_X_y=True)

X.plot()

boxplot = skfda.exploratory.visualization.Boxplot(
    X,
    depth_method=skfda.exploratory.depth.ModifiedBandDepth(),
)
boxplot.plot()

boxplot = skfda.exploratory.visualization.Boxplot(
    X,
    depth_method=skfda.exploratory.depth.ModifiedBandDepth(),
    prob=[0.75, 0.5, 0.25],
)
boxplot.plot()
```

An additional tool for functional data visualization and outlier detection is the magnitude-shape plot (MS-plot) (Dai and Genton, 2018; Dai and Genton, 2019). In this method, the degree of outlyingness of a functional observation is characterized in terms of two quantities: the magnitude outlyingness (MO) and the shape outlyingness (VO). The MS-plot is the scatter plot of the values MO and VO for each functional observation. This two-dimensional representation of the data can be used, for instance, to identify clusters of functions, or detect potential outliers, either in shape or in magnitude.

The following code can be used to display the MS-plot for the temperature curves of the *Canadian Weather* dataset together with the original trajectories. Additionally, outliers are identified according to the MS-plot criterion and marked in red. The

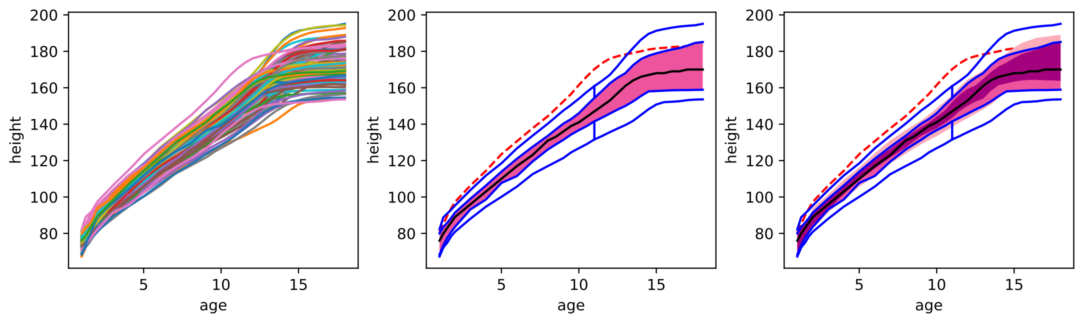


FIGURE 8.11: Functional boxplots of the *Berkeley Growth Study* dataset. The original curves are depicted in the left panel. The standard functional boxplot is shown in the central panel. In this panel, The black line stands for the functional median. The central envelope is displayed as a pink band around the median. The blue whiskers and their fences mark the maximum non-outlying envelope. Outliers are shown as a red dashed lines. In the right panel, different shades of pink are used for the deepest 25%, 50%, and 75% of the data.

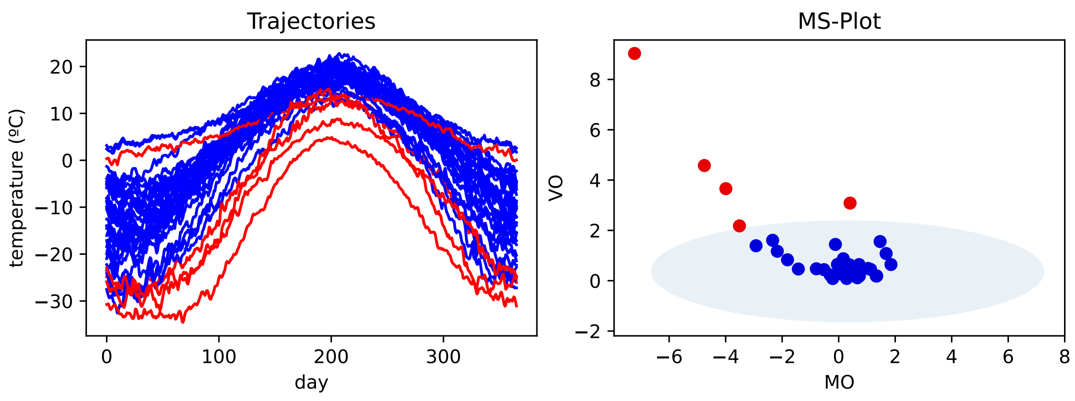


FIGURE 8.12: MS-plot and outliers of the *Canadian Weather* dataset. The original yearly temperature curves are displayed in the left panel. The corresponding MS-plot is shown in the right panel. The observations identified as outliers by the MS-plot criterion (those outside the blue ellipse) are marked in red.

class `MagnitudeShapePlot` generates the MS-plot and uses internally the methods of the class `MSPlotOutlierDetector` for outlier detection. The resulting plots are shown in Figure 8.12.

```python
import skfda

X, y = skfda.datasets.fetch_weather(return_X_y=True)
X = X.coordinates[0]

ms_plot = skfda.exploratory.visualization.MagnitudeShapePlot(X)
ms_plot.plot()

fig = X.plot(
    group=ms_plot.outliers,
    group_colors=["blue", "red"],
)
```

The class `Outliergram` provides an additional method for data visualization and detection of shape outliers (Arribas-Gil and Romo, 2014). The graph is defined in

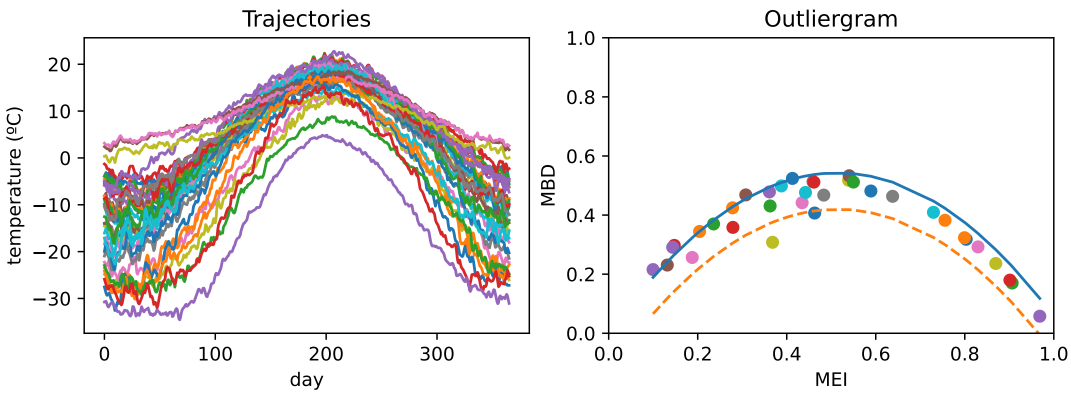


FIGURE 8.13: Outliergram of the yearly temperature curves for the *Canadian Weather* dataset. The original trajectories are in the left panel and the corresponding outliergram is shown in the right panel. The blue line corresponds to the reference parabola. The orange dashed line separates the typical curves (above) from the outliers (below).

terms of two related quantities: the modified epigraph index (MEI) and the MBD. The MEI of a trajectory is the average over time of the fraction of curves in the sample that lie above it. Each curve is a point (MEI, MBD) in the scatter plot. The outliergram takes advantage of the fact that points corresponding to typical functional observations lie on a parabola, whose analytical form is known. This parabola is used as a reference for the identification of shape outliers. Specifically, the degree of outlyingness of a curve is quantified in terms of its vertical distance to the parabola. The **scikit-fda**'s classes `Outliergram` and `OutliergramOutlierDetection` can be used to generate the outliergram and to detect outliers by using this criterion, respectively. The following code illustrates this functionality with the temperatures of the *Canadian Weather* dataset. The original trajectories and the corresponding outliergram are shown in Figure 8.13.

```
import skfda
import matplotlib.pyplot as plt

X, y = skfda.datasets.fetch_weather(return_X_y=True)
X = X.coordinates[0]

fig = X.plot()
fig = skfda.exploratory.visualization.Outliergram(X).plot()
```

In addition to standard plotting capabilities, most graphs generated with **scikit-fda** incorporate some interactive features. For example, the cursor can be placed at a point in the graph to display the actual coordinate values and the label of the observation. In addition, if different plots are used for visual exploration of some functional dataset, selecting a particular curve in one plot highlights the corresponding curve in the other active plots. Finally, widgets such as sliders can be used to select curves by some property, such as the label of the observation, or their depth in the sample.

An illustration of this interactive functionality is presented in Figure 8.14. In this figure, three different kinds of plots are displayed for the temperature curves of the *Canadian Weather* dataset: a graph of the sample trajectories, the MS-plot, and the outliergram. In the lower right side a slider has been created that displays the MBD value of the selected curve. A functional observation can be selected either

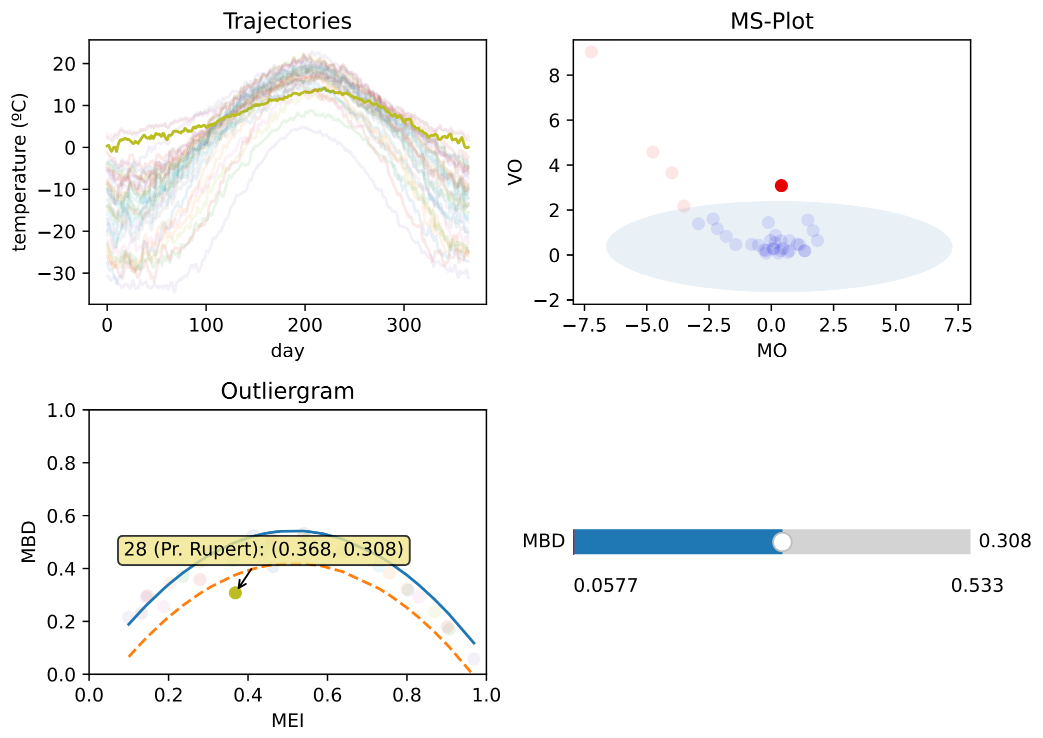


FIGURE 8.14: Interactive features in multiple plots for the *Canadian Weather* dataset.

by choosing a value in the slider widget or by clicking on the corresponding point in the MS-plot or in the outliergram. The datum selected is then highlighted in all plots. Finally, the cursor has been placed at a point in the MS-plot. This brings up a tooltip in which relevant information of the corresponding functional observation is displayed.

8.3.5 Machine learning

The package **scikit-fda** offers computational tools for both supervised (regression and classification) and unsupervised learning (clustering). We have implemented linear and kernel regression models, as well as neighbor-based algorithms for both regression and classification. In addition, logistic regression and centroid-based classification methods have been also included. The available clustering models include *k*-means, hierarchical clustering and the fuzzy C-means (FCM) algorithm discussed in Chapter 6. This part of the library is currently being extended to incorporate models such as regularized extensions of the linear and quadratic discriminants, partial least squares regression (Febrero-Bande, Galeano, and González-Manteiga, 2017) or linear regression with functional responses (Ramsay and Silverman, 2005).

The classes, methods, and functions in **scikit-fda** have been especially designed in accordance to the application programming interface (API) of the machine learning library **scikit-learn**. This design facilitates their integration in pipelines, and their use with other utilities provided by **scikit-learn**, such as cross-validation and grid search for hyperparameter selection. The package **scikit-fda** includes tools for dimensionality reduction and feature construction, by means of which the functional observations are transformed into lower-dimensional attribute vectors. This representation can then be used as input to standard multivariate machine learning methods provided by **scikit-learn**, as part of the same pipeline, allowing joint optimization of the hyperparameters.

Some of the methods provided for regression, classification and clustering require the computation of distances between functions. Examples of these are k-nearest neighbors (k-NN) and kernel regression methods, nearest centroids classifiers, and k-means clustering. Thus, several functional distances have been implemented in the package that can be employed as an hyperparameter for these methods. These distances include the L^p distances defined in Section 2.2, the angular distance

$$d_{\text{angular}}(x_1, x_2) = \frac{1}{\pi} \arccos\left(\frac{\langle x_1, x_2\rangle}{\|x_1\|\|x_2\|}\right), \tag{8.23}$$

and the α-Mahalanobis distance described in Berrendero, Bueno-Larraz, and Cuevas, 2020.

Regression

For regression, **scikit-fda** offers linear models, k-NN algorithms and kernel regression. The package includes a functional linear model with scalar response, and both multivariate and functional covariates,

$$y_i = \beta_0 + \sum_{j=1}^{p_1} \boldsymbol{\beta}_j \mathbf{x}_{ij} + \sum_{j=p_1+1}^{p} \int_{\mathcal{T}_j} \beta_j(t) x_{ij}(t) dt + \epsilon_i, \qquad i = 1, \dots, N. \tag{8.24}$$

This model is implemented as the `LinearRegression` class. This class accepts as covariates either a `FDataBasis` object, containing a functional covariate expressed in a functional basis, or a combination of `FDataBasis` objects and vectors of attributes. The regularization tools described in Section 8.2.6 can be applied to penalize the complexity of the β_j coefficients.

The use of `LinearRegression` is illustrated in the following example using the Tecator dataset. The regression problem consists in predicting the percent of fat content of several pieces of meat from their near-infrared absorbance spectra. In order to compute the β_j coefficients, the algorithm minimizes a cost function that includes both the sum of squared residuals and a term that penalizes the curvature of the β_j, related with their second derivative.

```python
import skfda
from skfda.misc.operators import LinearDifferentialOperator
from skfda.misc.regularization import L2Regularization
from skfda.ml.regression import LinearRegression
from skfda.representation.basis import BSpline

from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline


X, y = skfda.datasets.fetch_tecator(return_X_y=True)

y = y[:, 0]

basis = BSpline(n_basis=20)
X = X.to_basis(basis)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=0)

regression = LinearRegression(
    regularization=L2Regularization(
        LinearDifferentialOperator(2)))

regression.fit(X_train, y_train)

score = regression.score(X_test, y_test)

print(f"{score:.3}")
```

```
Out:
    0.951
```

The historical linear model studied on Malfait and Ramsay, 2003 is also implemented in class `HistoricalLinearRegression`, as

$$y_i(t) = \beta_0(t) + \int_{s_0(t)}^{t} \beta_1(t,s)x_i(s)ds + \epsilon_i, \qquad i = 1, \dots, N, \tag{8.25}$$

where $s_0(t) = \max(0, t - \delta)$ and δ is a predefined time lag that prevent points far in the past to affect the predicted value. This model can be used in the cases in which for a particular value $t \in \mathcal{T}$, the prediction of the value of the response at t should not be made using the values of the covariate at points $s > t$. This is commonly the case where the parameter is time, as it makes little sense to predict the response using values in the future.

The k-NN regression algorithm is also implemented in **scikit-fda**, in the class `KNeighborsRegressor`. Alternatively, the class `RadiusNeighborsRegressor` can be employed so that the neighbors that are at most at a specified distance are used for prediction. Both classes extend the ones of the same name available in **scikit-learn** to the functional case, by accepting functional data either as the covariate, the response, or both, as well as a functional distance to be used.

In addition, **scikit-fda** offers also support for kernel regression using the class `KernelRegression`. As with smoothing (see Section 8.3.3), there are several kernel estimators that can be used: Nadaraya-Watson (class `NadarayaWatsonHatMatrix`), k-nearest neighbors (class `KNeighborsHatMatrix`) and local linear regression (class `LocalLinearRegressionHatMatrix`) kernels.

Classification

Some of the classification methods described in Section 2.3.4 have also been implemented in **scikit-fda**. In particular it provides k-NN algorithms, nearest centroid methods, a logistic regression model, and classifiers based on statistical depth.

The library offers classes that wrap the nearest centroid and neighbor-based methods avaliable in **scikit-learn** and extend them to accept functional observations in both discretized and basis forms. The classes `KNeighborsClassifier` and `RadiusNeighborsClassifier` provide functional versions of k-NN and radius neighbors classifiers, respectively. The class `NearestCentroid` extends also the nearest centroid classifier in this way. In addition, the nearest centroid variant distance to trimmed means (DTM) (López Pintado and Romo, 2005) is implemented in the class `DTMClassifier`.

The package also features the class `LogisticRegression`, which implements a functional version of logistic regression for binary classification problems (Berrendero, Bueno-Larraz, and Cuevas, 2022). The method used performs variable selection as part of the classification process. It uses a greedy approach in which the points that sequentially maximize the likelihood are selected. A final multivariate logistic regression is then performed using the selected points.

Another group of classification methods supported in **scikit-fda** is the family of depth-based classifiers, introduced in Section 2.3.4. They use the statistical depth measures described in Section 8.3.4. The maximum-depth method (Cuevas, Febrero, and Fraiman, 2007), a simple classifier which predicts the class in which the new observation has a greater depth, is implemented in class `MaximumDepthClassifier`. The depth vs depth (DD) classifier proposed by Li, Cuesta-Albertos, and Liu, 2012 is available as class `DDClassifier`. Finally, the class `DDGClassifier` implements the more flexible generalized depth-depth classifier (DD^{G}) (Cuesta-Albertos, Febrero-Bande, and Oviedo de la Fuente, 2017).

The following code illustrates the usage of the DD^{G} classifier using the Berkeley Growth dataset. Here, an instance of the class `DDGClassifier` receives two depths to consider. In this case BD and MBD have been used. The multivariate classifier to be used in the depth space is specified upon instantiation as well. In this case, the multivariate k-NN classifier with a fixed number (5) of neighbors is employed.

```python
import skfda
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from skfda.exploratory.depth import BandDepth, ModifiedBandDepth

X, y = skfda.datasets.fetch_growth(return_X_y=True)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=0)

clf = skfda.ml.classification.DDGClassifier(
    depth_method=[
        ("BD", BandDepth()),
        ("MBD", ModifiedBandDepth()),
    ],
    multivariate_classifier=KNeighborsClassifier(
        n_neighbors=5,
    ),
)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

```
Out:
    0.8333333333333334
```

Clustering

The package **scikit-fda** provides generalizations of standard clustering algorithms for functional data, such as k-means, fuzzy C-means (FCM), and hierarchical clustering. In particular, the class `KMeans`, allows the use of functional distances in k-means. The following code illustrates the application of this algorithm to the yearly temperature curves of the Canadian Weather dataset, with the number of clusters equal to 3. The result is plotted in the left plot of Figure 8.15.

```python
import skfda
```

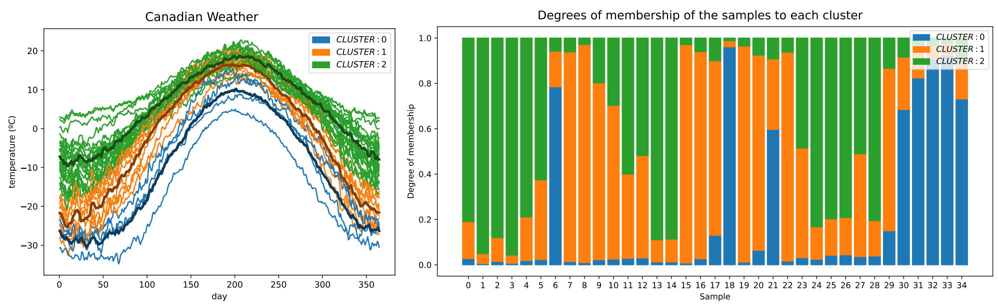


FIGURE 8.15: The left plot shows the curves of the Canadian Weather dataset. The color of each curve indicates to which cluster it is assigned in *k*-means. The darker curves are the centroids of each cluster. The right plot shows the degrees of membership of each of the curves to each cluster, when fuzzy C-means (FCM) is applied. The length of a segment indicates the degrees of membership of a particular curve to the corresponding cluster.

```

X, y = skfda.datasets.fetch_weather(return_X_y=True)
X = X.coordinates[0]

kmeans = skfda.ml.clustering.KMeans(n_clusters=3, random_state=0)
kmeans.fit(X)

fig = skfda.exploratory.visualization.clustering.ClusterPlot(
    kmeans,
    X,
).plot()
```

FCM, the fuzzy version of *k*-means described in Chapter 6, is also implemented in **scikit-fda**, in the class `FuzzyCMeans`. Visualization functions are also provided to plot the degrees of membership to each of the clusters of the curves. The following code illustrates the application of the FCM algorithm to the Canadian Weather dataset, again with 3 clusters.

```
import skfda
import matplotlib.pyplot as plt

X, y = skfda.datasets.fetch_weather(return_X_y=True)
X = X.coordinates[0]

fuzzycmeans = skfda.ml.clustering.FuzzyCMeans(
    n_clusters=3,
    random_state=0,
)
fuzzycmeans.fit(X)

fig = plt.figure(figsize=(10, 5))
skfda.exploratory.visualization.clustering.ClusterMembershipPlot(
    fuzzycmeans,
    X,
    fig=fig,
).plot()
```

The result of the execution of this code is displayed in Figure 8.15.

Finally the class `AgglomerativeClustering` for agglomerative hierarchical clustering is also provided. This is a wrapper of the **scikit-learn** class of the same name so that it can be used with functional inputs and functional distances.

Integration with scikit-learn for machine learning

The **scikit-fda** package has been especially designed for seamless integration with **scikit-learn** (Pedregosa et al., 2011). Specifically, there are a number of methods that transform the functional data into a two-dimensional array so that **scikit-learn**'s machine learning algorithms can be applied. For instance, the method `transform()` of the `EvaluationTransformer` class returns an array whose elements are the values of the functions in the sample at a specified set of points. If the data are in a basis representation, the expansion coefficients can be extracted using the methods of the class `CoefficientsTransformer`. Additionally, other **scikit-fda** methods, such as variable selection, can be used to provide a multivariate characterization of the functional observations.

The classes and methods provided for preprocessing and machine learning conform to **scikit-learn**'s API (Buitinck et al., 2013). An advantage of adopting this standard is that they can be employed in **scikit-learn** pipelines (class `Pipeline`). The following code illustrates how to build such a pipeline for a classification problem with functional data:

```python
import skfda

from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC

import skfda.preprocessing.smoothing as smoothing
import skfda.preprocessing.dim_reduction as dimred

X, y = skfda.datasets.fetch_phoneme(return_X_y=True)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=0)

smoothing_step = smoothing.kernel_smoothers.KNeighborsSmoother()
dimred_step = dimred.feature_extraction.FPCA()
classification_step = SVC()

pipeline = Pipeline([
    ('smoothing', smoothing_step),
    ('dimred', dimred_step),
    ('classification', classification_step)])

grid = GridSearchCV(
    pipeline,
    param_grid={
        'smoothing__smoothing_parameter': [3, 5, 7],
        'dimred__n_components': [1, 2, 3],
        'classification__C': [0.001, 0.01, 0.1, 1, 10],
    })

grid.fit(X_train, y_train)

score = grid.score(X_test, y_test)

print(f"{score:.3}")
```

```
Out:
    0.879
```

In this pipeline, the following sequence of operations is applied: kernel smoothing using the nearest-neighbors estimator, dimensionality reduction by functional principal components analysis (FPCA), and, finally, a standard (multivariate) support vector machine (SVM) classifier with an RBF kernel. The hyperparameters of the complete model, including those that correspond to preprocessing, are tuned by cross-validation using **scikit-learn**'s class `GridSearchCV`. Specifically, the number of neighbors in the nearest-neighbors kernel smoother is selected between the values 3, 5, and 7. The number of principal components considered ranges from 1 to 3. The grid $[0.001, 0.01, 0.1, 1.0, 10.0]$ is explored to determine the regularization parameter of the SVM. Then, the best values of the hyperparameters are used to fit the model using the complete training set. Finally, the accuracy of the classifier is computed and printed.

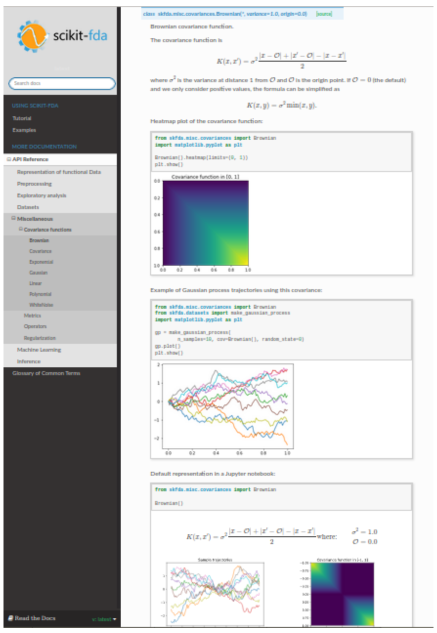
8.4 Code quality and documentation

The **scikit-fda** library has been designed to provide a powerful, self-contained, flexible, stable, and easy-to-use framework for the analysis of functional data in Python. The package is built as a SciPy Toolkit (SciKit)[1]. It is fully integrated in the Scientific Python software ecosystem[2]. Scientific Python is a collection of free and open-source Python packages for scientific and technical computing (Oliphant, 2007; Millman and Aivazis, 2011). Standard coding and naming practices are used throughout the project (Van Rossum, Warsaw, and Coghlan, 2001; Goodger and Van Rossum, 2001). This not only improves the legibility of the code and simplifies its maintenance, but also facilitates external contributions to the development of the package. Whenever appropriate, the design conforms to **scikit-learn** specifications (Pedregosa et al., 2011), so that the machine learning tools implemented in that package can be readily applied to functional data. To ensure the quality and robustness of the software, a comprehensive suite of unit and integration tests is provided. These automated tests are executed regularly in a continuous integration environment. We have attempted also to minimize the number of dependencies and to provide flexible interfaces that are intuitive and consistent throughout the application.

The **scikit-fda** package is free and open-source software distributed under the OSI-approved 3-Clause BSD license[3]. The GitHub page of **scikit-fda** (`https://github.com/GAA-UAM/scikit-fda`) is the main communication channel with the developers of the package for questions, bug reports, and feature requests. Contributions from the members of the FDA community are encouraged, as are comments and suggestions to improve the quality of the software.

To facilitate the use of **scikit-fda**, exhaustive documentation, including installation instructions, tutorials, API references, and illustrative examples are provided. The documentation, which is available online at `https://fda.readthedocs.io`, is built with the Python tool **Sphinx** (Sphinx Development Team, 2020). The examples and tutorials have been devised with **Sphinx-Gallery** (Nájera et al., 2020). They can be viewed online or downloaded as interactive **Jupyter** notebooks (Kluyver et al., 2016). In Figure 8.16, two screenshots of the documentation pages are shown: A reference page for the Brownian covariance function and an example of use of the functional boxplot functionality are displayed on the left and right panels of the figure, respectively.

[1]See `https://svn.scipy.org/scikits.html` for further details on SciKits (Accessed 2023-05-18)
[2]`https://scientific-python.org/` (Accessed 2023-05-18)
[3]`https://opensource.org/licenses/BSD-3-Clause` (Accessed 2023-05-18)



class skfda.misc.covariances.Brownian(*, variance=1.0, origin=0.0) [source]

Brownian covariance function.

The covariance function is

$$K(x, x') = \sigma^2 \frac{|x - \mathcal{O}| + |x' - \mathcal{O}| - |x - x'|}{2}$$

where σ^2 is the variance at distance 1 from $\mathcal{O}$ and $\mathcal{O}$ is the origin point. If $\mathcal{O} = 0$ (the default) and we only consider positive values, the formula can be simplified as

$$K(x, y) = \sigma^2 \min(x, y).$$

Heatmap plot of the covariance function:

```
from skfda.misc.covariances import Brownian
import matplotlib.pyplot as plt

Brownian().heatmap(limits=(0, 1))
plt.show()
```

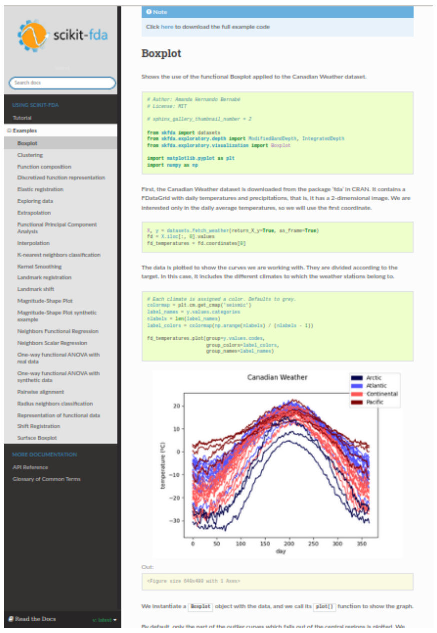
Example of Gaussian process trajectories using this covariance:

```
from skfda.misc.covariances import Brownian
from skfda.datasets import make_gaussian_process
import matplotlib.pyplot as plt

gp = make_gaussian_process(
        n_samples=10, cov=Brownian(), random_state=0)
gp.plot()
plt.show()
```

Default representation in a Jupyter notebook:

```
from skfda.misc.covariances import Brownian

Brownian()
```

$$K(x, x') = \sigma^2 \frac{|x - \mathcal{O}| + |x' - \mathcal{O}| - |x - x'|}{2} \text{where:} \quad \sigma^2 = 1.0 \quad \mathcal{O} = 0.0$$



Note

Click here to download the full example code

Boxplot

Shows the use of the functional Boxplot applied to the Canadian Weather dataset.

```
# Author: Amanda Hernando Bernabé
# License: MIT

# sphinx_gallery_thumbnail_number = 2

from skfda import datasets
from skfda.exploratory.depth import ModifiedBandDepth, IntegratedDepth
from skfda.exploratory.visualization import Boxplot

import matplotlib.pyplot as plt
import numpy as np
```

First, the Canadian Weather dataset is downloaded from the package 'fda' in CRAN. It contains a FDataGrid with daily temperatures and precipitations, that is, it has a 2-dimensional image. We are interested only in the daily average temperatures, so we will use the first coordinate.

```
X, y = datasets.fetch_weather(return_X_y=True, as_frame=True)
fd = X.iloc[:, 0].values
fd_temperatures = fd.coordinates[0]
```

The data is plotted to show the curves we are working with. They are divided according to the target. In this case, it includes the different climates to which the weather stations belong to.

```
# Each climate is assigned a color. Defaults to grey.
colormap = plt.cm.get_cmap('seismic')
label_names = y.values.categories
nlabels = len(label_names)
label_colors = colormap(np.arange(nlabels) / (nlabels - 1))

fd_temperatures.plot(group=y.values.codes,
                     group_colors=label_colors,
                     group_names=label_names)
```

Out:

```
<Figure size 640x480 with 1 Axes>
```

We instantiate a `Boxplot` object with the data, and we call its `plot()` function to show the graph.

FIGURE 8.16: **scikit-fda** documentation: On the left panel, the reference page of the Brownian covariance function is shown. On the right one, an illustration of the functional boxplot functionality.

8.5 Examples of use

The goal of this section is to illustrate the machine learning capabilities of **scikit-fda**, by addressing a representative set of machine learning problems with functional data, including clustering, regression, and classification, from different areas of application. These problems also serve to illustrate the special properties and the difficulties posed by the analysis of these types of data.

This section is based on the work presented at Ramos-Carreño et al., 2022. For the sake of reproducibility, the code used in this section is available at https://fda.readthedocs.io/ictai-examples, as part of the documentation of the package, and can be executed in the cloud using the link in the bottom part of each example. Import statements are ommited in the code for clarity.

8.5.1 Meteorological data: data visualization, clustering, and FPCA

We now use the annual temperature curves of the AEMET dataset (see Chapter 3) to illustrate some of the functionalities offered by **scikit-fda** for visualization, clustering and functional principal component anaysis (FPCA). We can download the data directly with **scikit-fda** by using the following code. The result is an object of class `FDataGrid` which contains the functional data in a discretized form.

```
X, _ = fetch_aemet(return_X_y=True)
```

This is an example of the **scikit-fda** tools for fetching datasets, based on the package **scikit-datasets** (Díaz-Vico and Ramos-Carreño, 2022), explained in Chapter 7.

Now, we select the temperature curves (the first coordinate function) and plot them in the upper part of Figure 8.17.

```
X = X.coordinates[0]
X.plot()
```

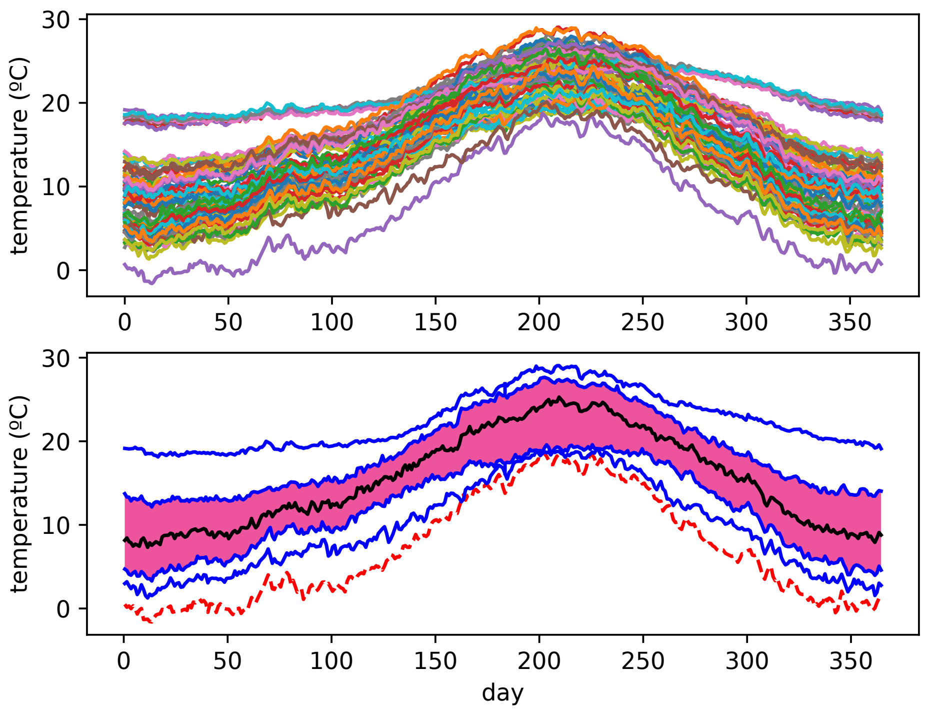


FIGURE 8.17: Functional temperature observations from the AEMET dataset (up) as well as its corresponding boxplot (bottom).

In this figure, one can see that all stations present the typical behavior of the northern hemisphere, with higher temperatures in summer that descend during winter. However, temperatures at some stations are atypical with respect to the majority: a purple curve with a significantly lower temperature than the others, and a set of flatter curves with warmer winters. In fact, these are outliers of magnitude and shape, respectively. The first one belongs to the Navacerrada station, at 1894 meters in height near a ski resort, while the others correspond to stations in the Canary Islands, known for their subtropical climate.

One way to detect and visualize magnitude outliers is to use the functional boxplot proposed in Sun and Genton, 2011.

```
Boxplot(
    X, depth_method=ModifiedBandDepth(),
).plot()
```

The resulting plot is in the bottom part of Figure 8.17. This is an extension of the classical univariate boxplot to the functional case. In pink, the central envelope of the data contains the deepest 50% of observations. The outlying envelope, bounded by the most external blue curves, separates the typical trajectories from magnitude outliers such as Navacerrada, in red.

The centrality, or depth, of a curve is quantified by statistical depths measures. For instance, the deepest observation in a dataset corresponds to the median (depicted in black in Figure 8.17), while outliers have depth values tending to zero. There are many proposals of functional depths, each of which defines different median and envelopes, and verifies different properties (Gijbels and Nagy, 2017). In **scikit-fda** are available integrated depths (Fraiman and Muniz, 2001), as well as the band and the modified band depths (López-Pintado and Romo, 2009). The last one is used in the previously shown boxplot.

The functional boxplot is not very suitable for detecting shape outliers. To this end, the magnitude-shape plot (Dai and Genton, 2018) and the outliergram (Arribas-Gil and Romo, 2014), both available in the library, can be used. All these visual tools can be combined and used interactively.

We now center our attention in grouping the stations by climate using the annual temperatures. This is a clustering problem that can be addressed adapting the classical k-means algorithm to the functional setting, using a distance between functions. The library **scikit-fda** also provides hierarchical clustering, and fuzzy C-means (FCM), a variant of k-means in which each observation has a degree of membership to each cluster. The package offers a variety of functional distances. Here, we use the L^2 distance. In the following code, the k-means algorithm is executed for 5 clusters (climatic regions). As the algorithm is sensitive to the initialization of the cluster centers, the best of 10 different initializations is chosen.

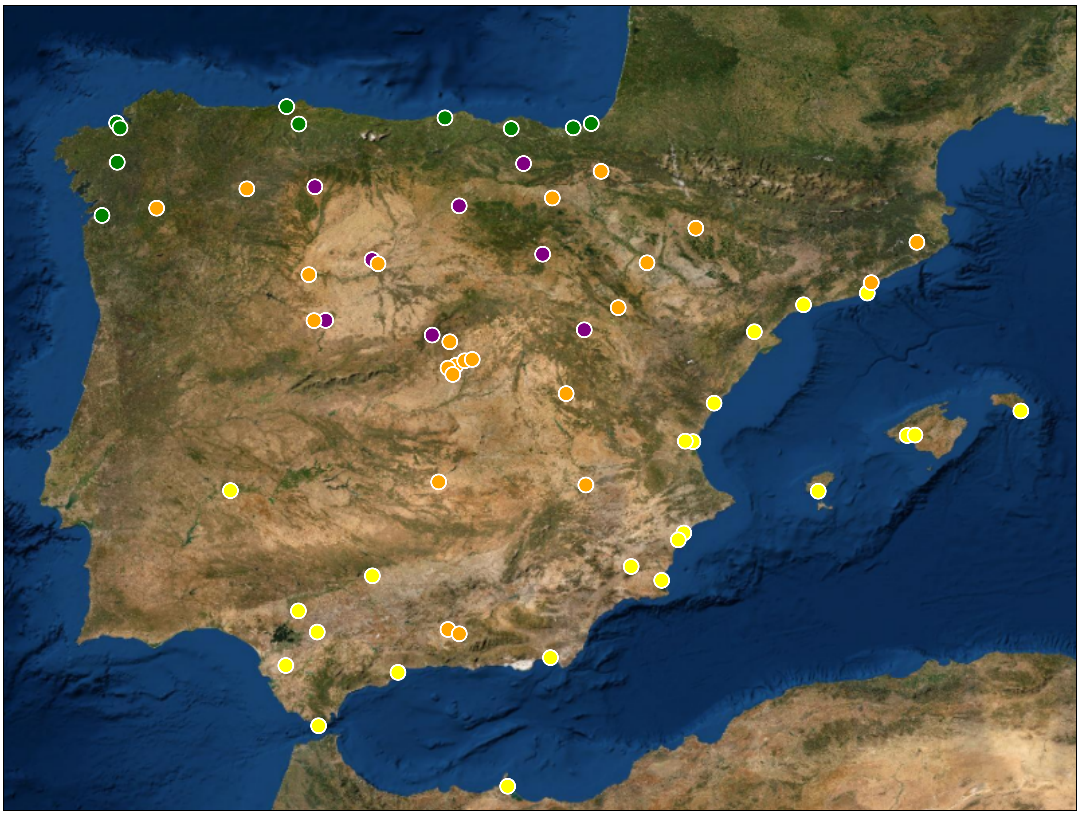
```
kmeans = KMeans(
    n_clusters=5,
    n_init=10,
    metric=l2_distance,
)
clusters = kmeans.fit_predict(X)
```

A map of Spain with the location of the weather stations is displayed in Figure 8.18. The colors indicate to which cluster each station is assigned. The clusters are in good correspondence with the climatic regions of Spain. The red points, located only in the Canary Islands, would correspond to the subtropical climate. The green points, in the north of mainland Spain, could represent the Atlantic climate. Yellow stations are located mostly along the Balearic Islands, the south, and the western coast of the Iberian peninsula, suggesting the so-called Mediterranean climate. Orange points generally appear at inland locations whose climate is continental. Finally, the purple stations are scattered on the coldest points of Spain, including some mountain ranges, and are thus examples of cold or high mountain climates.

Focusing on the Canary Islands, we can observe the presence of two points (yellow-mediterranean and purple-mountain) that, at a first glance, seem to be mislabeled. A closer inspection, however, shows that the “Mediterranean” station is the airport of Los Rodeos, characterized for having dense fog and a lower temperature than their surroundings (Romeo and Marzol Jaén, 2014). The cold-mountain station corresponds to an observatory located on an altitude of 2390 meters above sea level, at the slopes of Mount Teide, the highest mountain in Spain.

Although the clustering is strongly aligned with the climes of Spain, it is difficult to understand how the labels have been assigned. Dimensionality reduction techniques may be useful, because they allow us to identify features for the interpretation of the results. The most popular dimensionality reduction method is, probably, principal components analysis (PCA). In PCA, the data are projected along the (orthogonal) directions of maximal variance. In functional principal components analysis (FPCA), the idea remains the same, but calculations need to be adapted. As an example, projections are made with the L^2 inner product (see Section 2.2) $\langle x_1, x_2\rangle_{L^2} = \int_{\mathcal{T}} x_1(t)x_2(t)dt$. The following code obtains the first two principal components (`fit`) and projects the data on these directions (`transform`). Finally, to get some information about the components, the mean of temperatures (blue lines), and the result of adding and subtracting the principal components (orange and green lines) are plotted in Figure 8.19.

```
fpca = FPCA(n_components=2)
fpca.fit(X)
```



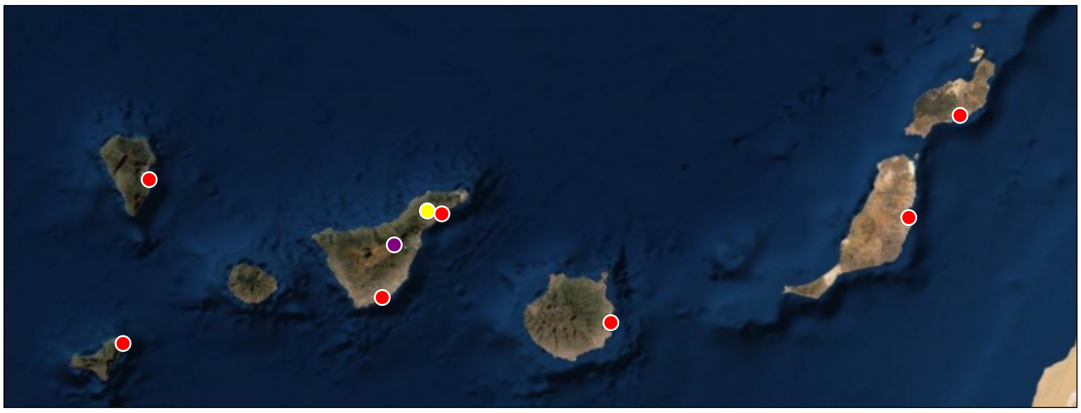


FIGURE 8.18: Weather stations clusters based on temperature curves in mainland Spain (top) and on the Canary Islands (bottom).

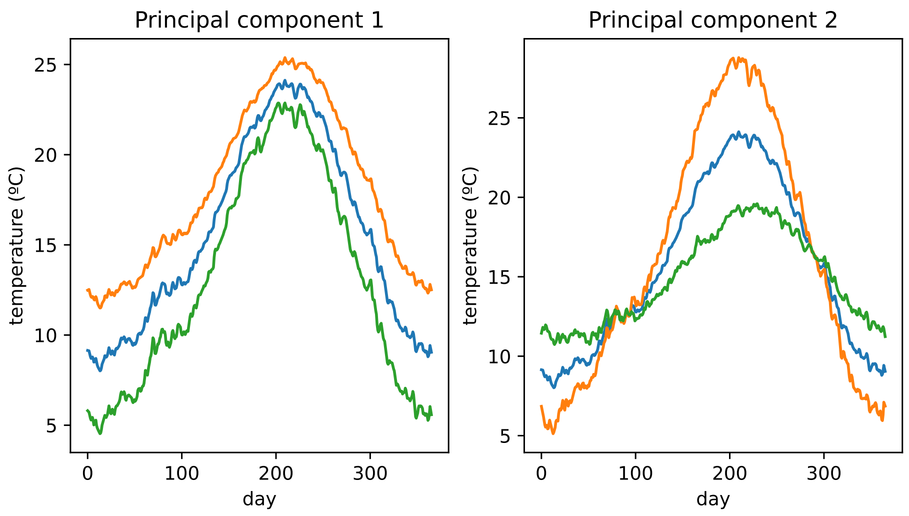


FIGURE 8.19: Mean of AEMET temperatures (blue) ± principal components (orange and green)

```
X_red = fpca.transform(X)
FPCAPlot(
    fpca.mean_, fpca.components_,
).plot()
```

The first component captures the average temperature. The second one marks the difference between summer (maximum of temperatures) and winter (minimum). In some sense, it measures the thermal amplitude. Moreover, dimensionality reduction allows us to visualize the data in the reduced space. For example, Figure 8.20 shows a scatter plot of the temperatures in the space of the first two principal components. Colors are those of the clusters obtained previously with *k*-means. Following the previous discussion, we can interpret the first component (x axis) as the average temperature, from colder (left) to warmer (right) locations. On the other hand, the second component (y axis) represents the thermal amplitude, from small variations (bottom) to higher differences (top) between summer and winter. We can see that stations from Canary Islands (red points) are quite different to the others (as seen in Figure 8.17), with warmer average temperatures and low variation, which are characteristics of the subtropical climate. Green points have colder temperatures, but have low thermal amplitude too. These points correspond to the northern coast of mainland Spain, having an Atlantic climate. Continental and Mediterranean climates (orange and yellow) present a much more pronounced amplitude, with the difference that continental climate has lower overall temperatures. However, two stations identified as Mediterranean by *k*-means seem to be mislabeled in this PCA representation. With a closer inspection, we see that one of these outliers is, precisely, the already mentioned airport of Los Rodeos, which has “Mediterranean” average temperature, with the low annual variations of the Canary Islands, where it is located. The other corresponds to Tarifa, in the Strait of Gibraltar. It is a very windy place which receives the cold water from the Atlantic. This causes Tarifa to have its own micro-climate characterized by smoother annual temperatures, as shown in the scatter plot. Finally, purple locations (cold-mountain climate) exhibit a variety of amplitudes, but are characterized for the lowest average temperatures. In particular, the leftmost point corresponds to Navacerrada station, the magnitude outlier detected in Figure 8.17.

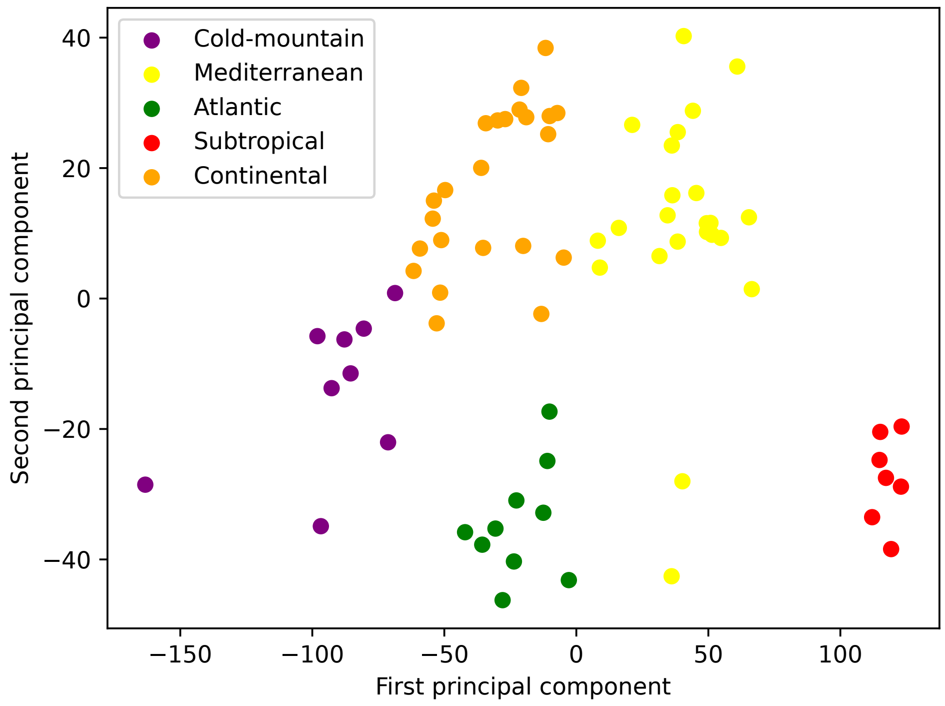


FIGURE 8.20: AEMET data projected to their first two principal components.

8.5.2 Spectrometric data: derivatives, regression, and variable selection

In this section we use the Tecator dataset (see Chapter 3) to illustrate the regression problem and some new functionalities. In this case, response variables are retrieved jointly with the functional data. We only keep the fat content as target for regression (Ferraty and Vieu, 2006). Trajectories are plotted in the upper part of Figure 8.21 using a color gradient for the fat content.

```
X, y = fetch_tecator(return_X_y=True)
y = y[:, 0]
X.plot(gradient_criteria=y)
```

We can appreciate that the magnitude of each curve bears almost no correlation with its fat contents. Nevertheless, it is well known in the literature that this problem becomes easier by using the second derivative of the trajectories (Ferraty and Vieu, 2006). This can be a convenient preprocessing which is exclusive of functional data. So, we compute the second derivative of the data:

```
X_der = X.derivative(order=2)
```

The resulting derivatives are in the middle plot of Figure 8.21. After differentiation, the fat content is clearly proportional to the magnitude of the curves at several points.

Before performing regression, as explained in Ramsay and Silverman, 2005, a common approach to prevent overfitting when working with continuous regression coefficients is to use the representation in a basis expansion. As an example, we represent the original data in a B-spline basis with $B = 10$ elements with **scikit-fda**:

```
basis = BSpline(n_basis=10)
X_der_basis = X_der.to_basis(basis)
```

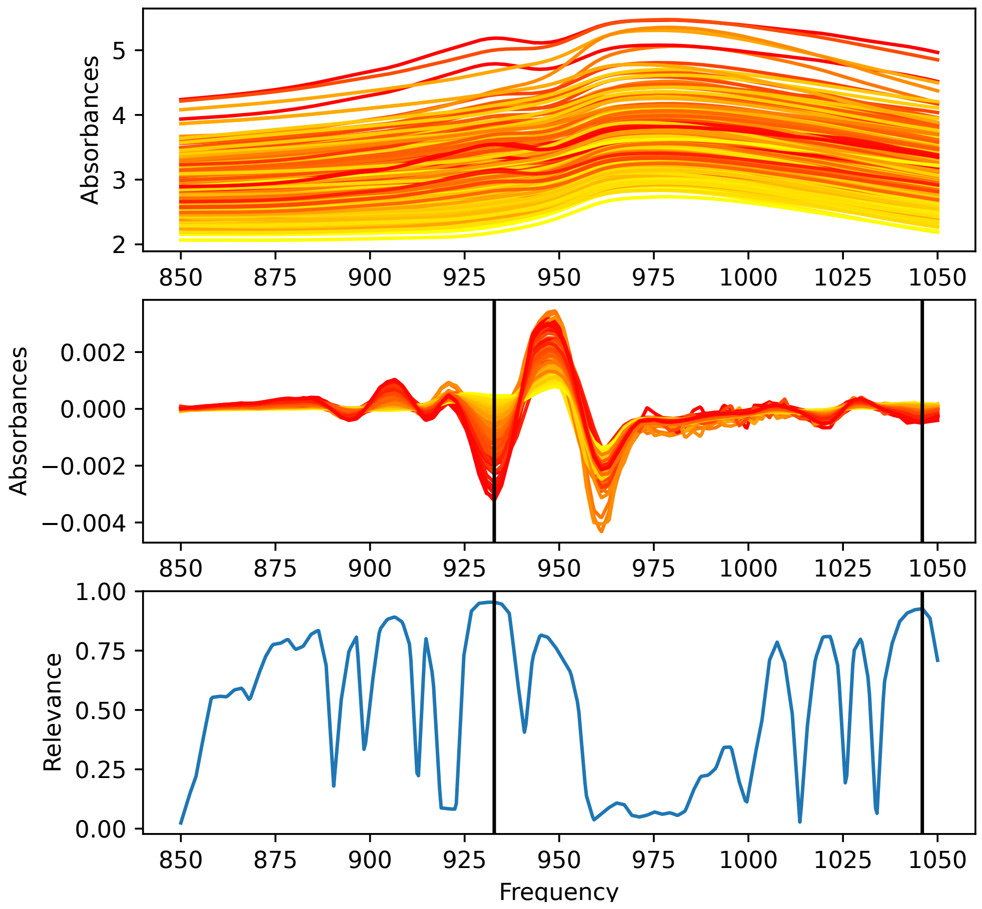


FIGURE 8.21: On top, the curves of the Tecator dataset. A color gradient indicates the fat content. The middle plot shows the second derivative of these curves. The bottom plot shows the dependency of the target on each function point. The two points of higher dependency have been marked with vertical black lines.

We can split the data in train and test partitions using, for example, the function `train_test_split` from **scikit-learn**, which is able to deal with the functional objects of **scikit-fda**. Then we use the standard functional linear regression model with scalar response $y = \beta_0 + \int_{\mathcal{T}} \beta_1(t)x(t)dt$ (Ramsay and Silverman, 2005). The following code fits the regression model, obtains the predicted values for the test partition, and compute the score of the prediction in terms of the coefficient of determination R^2:

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
score = r2_score(y_test, y_pred)
```

Some alternatives available in the library include nonparametric models based on nearest neighbors or kernel regression. It is even possible to add additional regularization terms for the coefficients, if needed. Nevertheless, even with this simple model, we are able to obtain an R^2 score of 0.951 by using the second derivatives of the Tecator trajectories.

Although this model has a good performance, it lacks interpretability. In the plot of the derivatives, it is easy to identify a couple of points that should provide a good prediction by themselves. Thus, we would expect that a variable selection procedure could find those points of interest and reduce the dimensionality of the data obtaining a more interpretable model. Many classical multivariate methods for variable selection do not work with continuous functional data due to the high redundancy between close variables. The package **scikit-fda** offers several variable selection methods that prevent this problem, from adaptations of multivariate methods that take redundancies into account, such as minimum-redundancy-maximum-relevance (mRMR) (Berrendero, Cuevas, and Torrecilla, 2016a), to purely functional methods that consider the nature of these data, such as a proposal based on reproducing kernel Hilbert spaces (Berrendero, Cuevas, and Torrecilla, 2018) or recursive maxima hunting (Torrecilla and Suárez, 2016). Here, we use the method maxima hunting (MH), that computes the relevance of each point $x(t)$ as its relation with the response variable (quantified with a measure of statistical dependence). MH selects the local maxima of the resultant relevance function, what automatically removes redundant variables (Berrendero, Cuevas, and Torrecilla, 2016b). In the following example we use MH to select the two most relevant local maxima.

```
var_sel = MaximaHunting(
    local_maxima_selector=(
        RelativeLocalMaximaSelector(
            max_points=2,
        )
    )
)
X_mv = var_sel.fit_transform(X_der, y)
```

In the bottom plot of Figure 8.21 we can see the relevance function for Tecator derivatives with the distance correlation measure (Székely, Rizzo, and Bakirov, 2007), and the two selected variables. These are also marked over the derivative curves in the middle plot, where we can appreciate that they correspond to points where the fat content was proportional to the magnitude of the curves.

After selection we can apply any multivariate regression model from the **scikit-learn** library. For example, the standard linear model obtains a R^2 score of 0.917. This is slightly worse than the score obtained with the whole trajectories, but it is using only two variables, what entails a significant gain in interpretability. Moreover, we could even use regression trees or any other method available in the **scikit-learn** to try to improve the performance without losing interpretability.

8.5.3 Voice signals: smoothing, registration, and classification

The binary version of the Phoneme dataset, restricted to the first 150 variables (see Chapter 3) was chosen to illustrate smoothing, functional data registration (alignment) and classification. Analogously to previous examples, we download the data using the `fetch_phoneme` function. The top plot in Figure 8.22 shows the first 20 curves with a different color per class.

```
X[:20].plot(group=y)
```

We can observe that phoneme trajectories are particularly noisy, which may complicate further analysis. This can be solved by smoothing the curves, for example, using a weighted average of neighbouring points $\hat{x}(t) = \int_T w_t(s)x(s)ds$, where $\hat{x}$ is a smoothing estimation of the underlying signal and $w_t(s)$ has its maximum at $s = t$ and decreases monotonically from that point. Several smoothing strategies are available in **scikit-fda** ranging from different kernel smoothers, widely used in density estimation, to smoothing via representation in a basis, which also allows penalizing the curvature to achieve additional smoothness. Here, we smooth the Phoneme

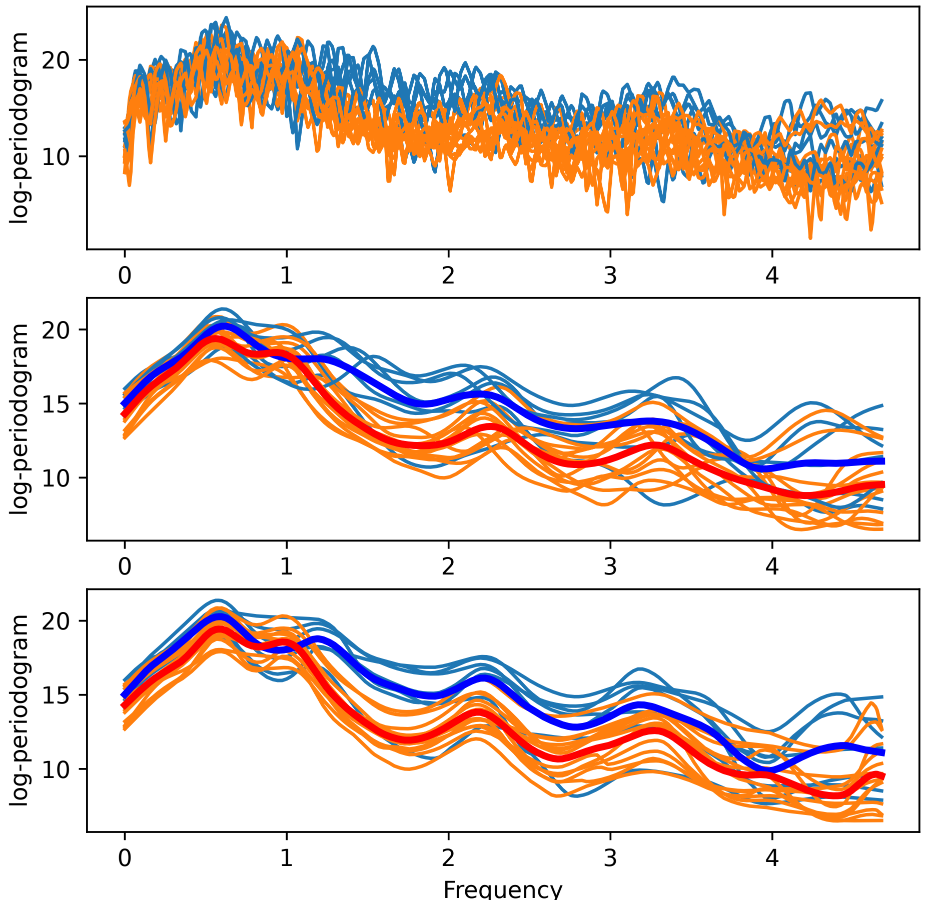


FIGURE 8.22: Ten curves from the Phoneme dataset with classes "aa" (in blue) and "ao" (in orange). We can see original curves (top), smoothed curves (middle), and the curves after per-class registration (bottom). Class means are plotted in thick darker lines in the last two plots.

observations with a Nadaraya-Watson kernel smoother. The bandwith parameter controls the degree of smoothing, and has to be adjusted to prevent infra or over-smoothing.

```
smoother = KernelSmoother(
    NadarayaWatsonHatMatrix(
        bandwidth=0.1,
        kernel=normal,
    ),
  )
)
X_smooth = smoother.fit_transform(X)
```

The first 20 smoothed curves are shown in the middle plot of Figure 8.22. We can see that the per-class means have much less pronounced maxima and minima than the individual observations. This can be an indicator of misalignment. Misalignment is a new challenge that did not appear in multivariate statistics. It affects not only to the mean estimation, but methods such as variable selection, that assume aligned data to start with. Functional data registration is the process for which the data is aligned so that maxima, minima, or other relevant landmarks appear at the same points in each observation (Marron et al., 2015). The package **scikit-fda** offers shifting and elastic registration procedures. In this case we do not have information about the landmarks of interest, so we can use an advanced elastic registration procedure based on the properties of the Fisher-Rao distance (Srivastava et al., 2011). For example, the following code registers together all curves from the first class:

```
reg = FisherRaoElasticRegistration(
    penalty=0.01,
)
X_reg = reg.fit_transform(X_smooth[y==0])
```

The result of the per-class registration is displayed in the bottom plot of Figure 8.22. We can see that the registered curves have now their landmarks properly aligned, and the class means more closely resemble a typical trajectory of the corresponding class. More importantly, we can appreciate more clearly that the disposition of maxima and minima is different between classes. Hence, we should not attempt to register all curves together.

As we have just seen, aligning all the curves at the same time entails a loss of discriminant information. Therefore, for the classification task, we consider the unaligned smoothed functions of the middle plot in Figure 8.22. Moreover, a classifier that is resilient to unaligned data should be preferable. The package **scikit-fda** offers a variety of classification algorithms for functional data: distance-based classifiers such as nearest centroids or k-nearest neighbors; depth-based classifiers, including maximum-depth (Cuevas, Febrero, and Fraiman, 2007) and the DD^G classifier (Cuesta-Albertos, Febrero-Bande, and Oviedo de la Fuente, 2017); or even functional logistic regression (Berrendero, Bueno-Larraz, and Cuevas, 2022). As an example, we use a k-nearest neighbors classifier with the functional Mahalanobis distance proposed in Berrendero, Bueno-Larraz, and Cuevas, 2020.

```
classifier = KNeighborsClassifier(
    n_neighbors=67,
    metric=MahalanobisDistance())
```

The sample is split in train and test, leaving the 30% of the data as the test partition. We then fit the model and compute the predictions, obtaining an accuracy score of 0.805:

```
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
score = accuracy_score(y_test, y_pred)
```

For simplicity, we have fixed the number of neighbors to $\sqrt{N}$. In a complete analysis it is often better to choose the values for the number of neighbors and the smoothing parameter via a cross validation procedure. As most objects in **scikit-fda** have the same application programming interface (API) as **scikit-learn** classes (Buitinck et al., 2013), the hyperparameter selection and cross validation utilities of that package can be directly applied, making this an easy task.

8.6 A comparative study of functional classifiers with scikit-fda

The goal of this section, based on the work presented in Ramos-Carreño, Torrecilla, and Suárez, 2022, is to compare the predictive performance of different classifiers for these types of data, using the tools provided in **scikit-fda**. Most off-the-shelf methods, such as the ones implemented in the Python library **scikit-learn** (Pedregosa et al., 2011), assume that the instances to be classified are characterized by vectors of

TABLE 8.1: Characteristics of the datasets considered.

	N	M	C	Majority class (%)	Reference
ArrowHead	211	251	3	38.39	Dau et al., 2019
Australian	190	365	2	77.37	Bureau of Meteorology, 1992
Cell	90	18	2	51.11	Spellman et al., 1998
Coffee	56	286	2	51.79	Dau et al., 2019
ECG	2026	85	2	74.33	Baíllo, Cuevas, and Fraiman, 2010
Fish	350	463	7	14.29	Dau et al., 2019
Growth	93	31	2	58.06	Baíllo, Cuevas, and Fraiman, 2010
GunPoint	200	150	2	50.00	Dau et al., 2019
MCO	89	360	2	50.56	Baíllo, Cuevas, and Fraiman, 2010
Medflies	534	30	2	52.06	Baíllo, Cuevas, and Fraiman, 2010
NOx	115	24	2	66.09	Febrero, Galeano, and González-Manteiga, 2008
Phoneme	4509	256	5	25.79	Baíllo, Cuevas, and Fraiman, 2010
Phoneme (bin)	1717	50	2	59.52	Baíllo, Cuevas, and Fraiman, 2010
Plane	210	144	7	14.29	Dau et al., 2019
Symbols	1020	398	6	17.75	Dau et al., 2019
Tecator	215	100	2	64.19	Ferraty and Vieu, 2006
Tecator (2$^{\text{nd}}$ der)	215	100	2	64.19	Ferraty and Vieu, 2006
Yoga	3300	426	2	53.64	Dau et al., 2019

attributes. The direct application of these methods to functional data in discretized form does not take advantage of their continuous structure, and can be problematic from a theoretical and practical perspective. A possible approach is to use a dimensionality reduction method and then apply a standard multivariate classifier (Berrendero, Cuevas, and Torrecilla, 2018). Alternatively, the classification method can be adapted to the functional setting. For instance, a nearest-centroid classifier can be built using the per-class functional means as prototypes and a functional metric to compute distances (Delaigle and Hall, 2012). It is also possible to utilize functional measures of centrality in a sample to design depth-based classifiers (Cuevas, Febrero, and Fraiman, 2007; Cuesta-Albertos, Febrero-Bande, and Oviedo de la Fuente, 2017). In k-NN, a functional distance can be used to identify the nearest neighbors (Baíllo, Cuevas, and Fraiman, 2010). Random forest (Breiman, 2001) can be directly employed with functional data by using the values of the functions as attributes.

The performance of these different types of classifiers has been tested using a wide range of functional classification problems from different areas of application. A total of 5 multiclass datasets (arrowheads, fish, phoneme, plane and symbols) and 13 binary classification problems are used in this comparison. These datasets themselves are some of the ones described in Chapter 3. A summary of the characteristics of these datasets is presented in Table 8.1. In this table, N is the sample size, M the size of the grid, and C the number of classes. For the Phoneme dataset the data has been smoothed by applying a Nadaraya-Watson smoother to the original curves. The binary version of this dataset truncated to the first 50 features has also been included in the study (Delaigle and Hall, 2012). We also include the second derivatives of Tecator, as for this dataset they contain most of the information (Ferraty and Vieu, 2006). To compute the derivatives, the original curves have been approximated using B-splines.

In this study, four different families of classifiers are considered: nearest centroid classifiers (Delaigle and Hall, 2012), a functional variable selection method (Berrendero, Cuevas, and Torrecilla, 2018) used in combination with different multivariate classifiers, classifiers based on the notion of depth (Cuevas, Febrero, and Fraiman, 2007; Cuesta-Albertos, Febrero-Bande, and Oviedo de la Fuente, 2017), and k-NN classifiers that employ different functional distances (Baíllo, Cuevas, and Fraiman, 2010). In what follows, we first compare classifiers within each of these families. From each of these families the classifier that has the best overall predictive performance in the problems considered is selected. Finally, the selected predictors are compared between each other and with random forest, which is one of the best off-the-shelf classifiers (Fernández-Delgado et al., 2014).

The infrastructure for the empirical evaluation is provided by the scikit-datasets Python package (Díaz-Vico and Ramos-Carreño, 2022). The classifiers are built using **scikit-fda** (Ramos-Carreño et al., 2023), a Python package that offers a comprehensive set of tools for statistical analysis and machine learning for functional data, in combination with **scikit-learn** (Pedregosa et al., 2011). For each of the classification problems, stratified sampling is used to partition the data into a training set, which includes 70% of the instances available for learning, and a test set with the remaining ones. When necessary, 5-fold cross-validation within the training set is used to determine the values of the hyperparameters of the different classifiers.

The results reported consist of the mean accuracy and standard deviation over 100 random partitions for each particular classifier-dataset combination. To account for sample variability, the classifiers are trained and tested using the same partitions. For each dataset, the scores of the best and second best predictors are highlighted in boldface and underlined, respectively. An asterisk is used to indicate statistically significant differences at the 5% level using a paired t-test.

An overall comparison of the different classification methods is made in terms of their average accuracy and rank. A Friedman test is used to determine whether the overall differences between average ranks are statistically significant. If significant differences are detected by this test a pairwise post-hoc Nemenyi test is used (Demšar, 2006). The results of these tests are summarized in critical distance (CD) diagrams generated with the **autorank** Python package (Herbold, 2020).

8.6.1 Nearest centroid classifiers

Nearest centroid classifier (NC), defined in Section 2.3.4, conform the first family of methods in the comparison. In all these methods, the class of a particular observation is predicted as the one whose mean (computed from the training data) is closer to it. The closeness is measured using a metric. Four different metrics are considered (see Section 2.2 for the definitions):

- NC-L^1 uses the L^1-distance
$$d_{L^1}(x_1, x_2) = \int_{\mathcal{T}} |x_1(t) - x_2(t)| dt. \tag{8.26}$$
- NC-L^2 with the L^2-distance:
$$d_{L^2}(x_1, x_2) = \sqrt{\int_{\mathcal{T}} |x_1(t) - x_2(t)|^2 dt}. \tag{8.27}$$
- NC-Mah uses the α-Mahalanobis distance (Berrendero, Bueno-Larraz, and Cuevas, 2020). In the multivariate case, the Mahalanobis distance is
$$d_{\mathrm{Mah}}(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \Sigma^{-1} (\mathbf{x}_1 - \mathbf{x}_2)} = \sqrt{\langle \Sigma^{-1/2}(\mathbf{x}_1 - \mathbf{x}_2), \Sigma^{-1/2}(\mathbf{x}_1 - \mathbf{x}_2) \rangle}, \tag{8.28}$$

TABLE 8.2: Comparison between nearest centroids classifiers.

	NC-L^1	**NC-L^2**	**NC-Mah**	**NC-Ang**
ArrowHead	0.734 ± 0.045 (4)	0.765 ± 0.054 (3)	**0.782 ± 0.043 (1)***	0.765 ± 0.054 (2)
Australian	0.864 ± 0.038 (3)	0.852 ± 0.041 (4)	0.903 ± 0.040 (2)	**0.906 ± 0.034 (1)**
Cell	0.861 ± 0.069 (3)	0.861 ± 0.065 (2)	**0.869 ± 0.063 (1)**	0.818 ± 0.067 (4)
Coffee	0.955 ± 0.049 (4)	0.959 ± 0.048 (2)	**1.000 ± 0.000 (1)***	0.959 ± 0.048 (2)
ECG	0.780 ± 0.025 (4)	0.840 ± 0.016 (2)	**0.971 ± 0.006 (1)***	0.790 ± 0.028 (3)
Fish	0.616 ± 0.041 (4)	0.635 ± 0.045 (3)	**0.716 ± 0.041 (1)***	0.640 ± 0.044 (2)
Growth	0.742 ± 0.102 (4)	0.777 ± 0.093 (3)	**0.958 ± 0.032 (1)***	0.934 ± 0.039 (2)
GunPoint	0.712 ± 0.060 (2)	0.706 ± 0.058 (3)	**0.770 ± 0.051 (1)***	0.684 ± 0.055 (4)
MCO	0.635 ± 0.079 (4)	0.637 ± 0.078 (3)	**0.986 ± 0.026 (1)***	0.839 ± 0.068 (2)
Medflies	0.551 ± 0.031 (3)	**0.556 ± 0.030 (1)**	0.548 ± 0.034 (4)	0.553 ± 0.032 (2)
NOx	0.726 ± 0.077 (4)	0.782 ± 0.079 (3)	**0.898 ± 0.052 (1)***	0.851 ± 0.057 (2)
Phoneme	0.851 ± 0.008 (4)	0.867 ± 0.007 (3)	**0.910 ± 0.006 (1)***	0.869 ± 0.008 (2)
Phoneme (binary)	0.750 ± 0.017 (4)	0.762 ± 0.016 (3)	**0.812 ± 0.015 (1)***	0.798 ± 0.015 (2)
Plane	0.954 ± 0.028 (2)	0.953 ± 0.027 (3)	**0.969 ± 0.020 (1)***	0.949 ± 0.028 (4)
Symbols	0.887 ± 0.015 (3)	0.891 ± 0.015 (2)	**0.920 ± 0.016 (1)***	0.878 ± 0.015 (4)
Tecator	0.681 ± 0.046 (4)	0.685 ± 0.045 (3)	**0.953 ± 0.021 (1)***	0.860 ± 0.036 (2)
Tecator (2nd derivative)	0.961 ± 0.020 (2)	0.960 ± 0.022 (3)	0.959 ± 0.021 (4)	**0.971 ± 0.017 (1)***
Yoga	0.519 ± 0.035 (4)	0.528 ± 0.038 (2)	**0.578 ± 0.016 (1)***	0.527 ± 0.032 (3)
Average accuracy	*0.766*	*0.779*	***0.861***	*0.811*
Average rank	*3.444*	*2.667*	***1.389***	*2.444*

where Σ is its covariance matrix of a data distribution. However, it is not posible to compute an equivalent functional version of the Mahalanobis distance because, in general, the covariance operator is not invertible. The α-Mahalanobis distance is a regularized version of the Mahalanobis distance for functional data, that depends on a regularization parameter α. The regularization parameter α is fixed by cross validation in the range $\{10^{-i}, i = 0, \dots, 6\}$.

- NC-Ang utilizes the angular distance

$$d_{\text{angular}}(x_1, x_2) = \frac{1}{\pi} \arccos\left(\frac{\langle x_1, x_2\rangle}{\|x_1\|\|x_2\|}\right). \tag{8.29}$$

The results of these experiments are presented in Table 8.2 and summarized in a CD diagram (Figure 8.23). NC with the α-Mahalanobis distance clearly outperforms the other competitors. The average rank of NC-Mah is significantly better than NC-L^1 and NC-L^2. It also obtains significant victories in a number of datasets.

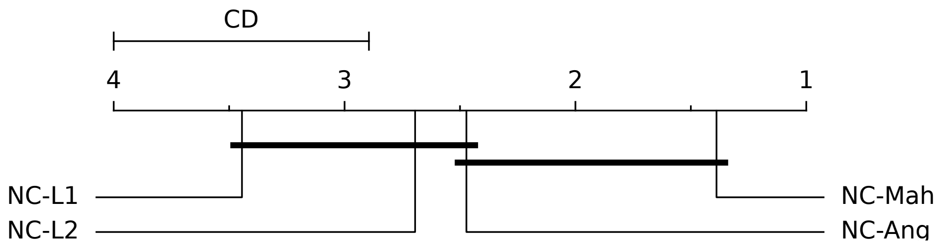


FIGURE 8.23: CD diagram of the nearest centroid classifiers. Classifiers are grouped with a thick dark line if the differences among their average ranks are not statistically significant.

8.6.2 Functional variable selection

A usual strategy to deal with functional data consists in applying a dimensionality reduction method to transform the original trajectories into vectors. Then, any standard multivariate classifier can be used. Variable selection methodology provides

TABLE 8.3: Comparison between RKVS-based classifiers.

	RKVS+LDA	RKVS+QDA	RKVS+k-NN	RKVS+RF
ArrowHead	0.787 ± 0.044 (4)	0.832 ± 0.043 (3)	**0.855 ± 0.045 (1)***	0.834 ± 0.043 (2)
Australian	0.889 ± 0.035 (4)	0.890 ± 0.039 (3)	0.936 ± 0.029 (2)	**0.937 ± 0.030 (1)**
Cell	0.842 ± 0.059 (3)	0.841 ± 0.062 (4)	0.883 ± 0.062 (2)	**0.891 ± 0.061 (1)**
Coffee	**0.974 ± 0.044 (1)**	0.970 ± 0.045 (2)	0.968 ± 0.048 (3)	0.951 ± 0.052 (4)
ECG	0.982 ± 0.005 (4)	0.987 ± 0.006 (3)	**0.997 ± 0.002 (1)***	0.993 ± 0.005 (2)
Fish	0.804 ± 0.039 (2)	**0.824 ± 0.037 (1)***	0.786 ± 0.038 (3)	0.770 ± 0.041 (4)
Growth	0.951 ± 0.034 (2)	**0.952 ± 0.038 (1)**	0.944 ± 0.040 (3)	0.916 ± 0.054 (4)
GunPoint	0.881 ± 0.038 (4)	0.909 ± 0.042 (2)	0.892 ± 0.041 (3)	**0.928 ± 0.039 (1)***
MCO	**0.961 ± 0.044 (1)**	0.959 ± 0.040 (2)	0.881 ± 0.060 (3)	0.855 ± 0.072 (4)
Medflies	0.591 ± 0.033 (2)	0.589 ± 0.033 (3)	0.568 ± 0.033 (4)	**0.599 ± 0.032 (1)**
NOx	0.914 ± 0.048 (2)	**0.919 ± 0.037 (1)**	0.890 ± 0.050 (3)	0.877 ± 0.049 (4)
Phoneme	0.922 ± 0.006 (4)	**0.927 ± 0.005 (1)***	0.923 ± 0.005 (3)	0.924 ± 0.006 (2)
Phoneme (binary)	**0.820 ± 0.015 (1)***	0.815 ± 0.015 (2)	0.806 ± 0.013 (3)	0.806 ± 0.014 (4)
Plane	0.971 ± 0.019 (2)	**0.974 ± 0.020 (1)**	0.960 ± 0.020 (4)	0.971 ± 0.021 (3)
Symbols	0.870 ± 0.015 (4)	0.950 ± 0.016 (3)	0.959 ± 0.010 (2)	**0.961 ± 0.010 (1)**
Tecator	0.938 ± 0.026 (2)	**0.974 ± 0.019 (1)***	0.879 ± 0.035 (3)	0.833 ± 0.045 (4)
Tecator (2nd derivative)	0.936 ± 0.026 (4)	0.978 ± 0.015 (2)	**0.980 ± 0.015 (1)**	0.978 ± 0.017 (2)
Yoga	0.686 ± 0.019 (4)	0.752 ± 0.027 (3)	**0.902 ± 0.011 (1)***	0.899 ± 0.010 (2)
Average accuracy	*0.873*	***0.891***	*0.889*	*0.884*
Average rank	*2.778*	***2.111***	*2.500*	*2.556*

interpretable reductions by replacing the original functions by their values at several well chosen points. In this work, we have chosen the reproducing kernel-based variable selection (RKVS) method proposed in Berrendero, Cuevas, and Torrecilla, 2018, and explained in Section 2.3.2, as a representative of this family of techniques.

Here, we use the greedy implementation of RKVS available in **scikit-fda** with four standard multivariate classifiers included in **scikit-learn**: linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), k-nearest neighbors (k-NN) with the Euclidean distance and random forest (RF). The number of variables selected and the k of k-NN are chosen by cross-validation between 1 and 10. The number of trees in RF is fixed to 100. Finally, as RKVS is defined for binary problems, we follow a one-versus-rest strategy for the multiclass datasets.

Accuracy results and ranking for each dataset are shown in Table 8.3. In general terms, there are no such big differences as in the previous section. This is illustrated in the associated CD diagram shown in Figure 8.24. However, RKVS+QDA and RKVS+k-NN obtain better results with significant victories in three datasets each. In particular, the application of QDA exhibits better global results as indicated by the average rank. This difference it is not appreciated in the average accuracy by the effect of the Yoga problem where the performance of RKVS+QDA is bad, maybe because of a lack of Gaussianity.

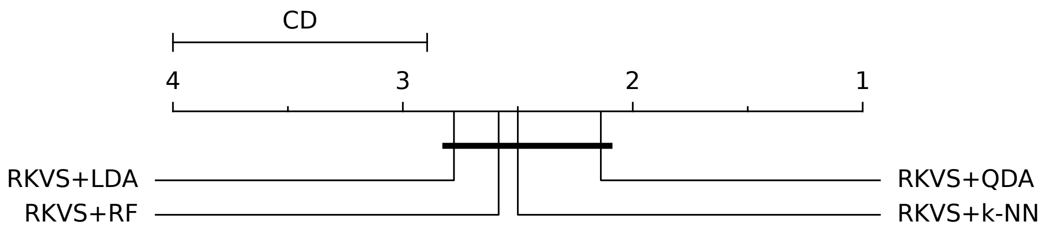


FIGURE 8.24: Critical distance diagram of the multivariate classifiers applied after RKVS variable selection.

TABLE 8.4: Comparison between depth-based classifiers.

	MD-MBD	**MD-Mah**	**DD^G-MBD**	**DD^G-Mah**
ArrowHead	0.722 ± 0.051 (4)	0.813 ± 0.048 (2)	0.739 ± 0.045 (3)	**0.825 ± 0.043 (1)***
Australian	0.747 ± 0.055 (4)	0.870 ± 0.047 (3)	0.875 ± 0.040 (2)	**0.898 ± 0.043 (1)***
Cell	0.764 ± 0.081 (4)	0.869 ± 0.063 (3)	**0.878 ± 0.054 (1)**	0.875 ± 0.058 (2)
Coffee	0.895 ± 0.068 (3)	**0.978 ± 0.039 (1)**	0.893 ± 0.065 (4)	0.972 ± 0.049 (2)
ECG	0.740 ± 0.024 (4)	0.909 ± 0.021 (2)	0.874 ± 0.010 (3)	**0.942 ± 0.013 (1)***
Fish	0.453 ± 0.057 (4)	0.619 ± 0.042 (3)	0.633 ± 0.041 (2)	**0.687 ± 0.044 (1)***
Growth	0.739 ± 0.100 (3)	**0.941 ± 0.040 (1)***	0.718 ± 0.080 (4)	0.927 ± 0.044 (2)
GunPoint	0.536 ± 0.026 (4)	0.793 ± 0.058 (3)	0.829 ± 0.049 (2)	**0.835 ± 0.053 (1)**
MCO	0.634 ± 0.077 (4)	0.877 ± 0.070 (2)	0.636 ± 0.088 (3)	**0.904 ± 0.053 (1)***
Medflies	0.529 ± 0.025 (2)	0.489 ± 0.037 (4)	**0.552 ± 0.039 (1)***	0.515 ± 0.038 (3)
NOx	0.700 ± 0.072 (4)	0.809 ± 0.055 (2)	0.778 ± 0.052 (3)	**0.863 ± 0.053 (1)***
Phoneme	0.813 ± 0.009 (4)	0.901 ± 0.006 (2)	0.847 ± 0.008 (3)	**0.905 ± 0.006 (1)***
Phoneme (binary)	0.745 ± 0.018 (3)	0.797 ± 0.015 (2)	0.721 ± 0.022 (4)	**0.804 ± 0.016 (1)***
Plane	0.901 ± 0.044 (4)	0.948 ± 0.027 (2)	**0.963 ± 0.028 (1)***	0.937 ± 0.029 (3)
Symbols	0.722 ± 0.022 (4)	0.926 ± 0.016 (2)	0.922 ± 0.013 (3)	**0.957 ± 0.011 (1)***
Tecator	0.684 ± 0.043 (3)	0.947 ± 0.028 (2)	0.621 ± 0.047 (4)	**0.967 ± 0.024 (1)***
Tecator (2nd derivative)	0.964 ± 0.023 (4)	0.970 ± 0.018 (2)	0.967 ± 0.017 (3)	**0.978 ± 0.017 (1)***
Yoga	0.588 ± 0.020 (4)	0.640 ± 0.016 (3)	0.659 ± 0.014 (2)	**0.698 ± 0.015 (1)***
Average accuracy	*0.715*	*0.839*	*0.784*	***0.860***
Average rank	*3.667*	*2.278*	*2.667*	***1.389***

8.6.3 Depth-based classifiers

The next group to consider is the family of depth-based classifiers, described in Section 2.3.4. We now compare the maximum depth (MD) methods and the generalized depth-depth classifier (DD^G) approach. In the latter case, we only consider the simplest case with one depth measure in the comparison $G = 1$.

We have tested these two techniques with two different depth measures: modified band depth (MBD) (López-Pintado and Romo, 2009), and a depth measure based on the previously commented α-Mahalanobis distance (Mah) (Berrendero, Bueno-Larraz, and Cuevas, 2020). DD^G is combined with a simple k-NN classifier with the Euclidean distance. The number of neighbors is set by cross-validation between 1 and 10. Accuracy and rank positions by dataset are shown in Table 8.4. The associated CD diagram is shown in Figure 8.25.

These results are quite straightforward. On the one hand, DD^G methods clearly beat their same depth MD counterparts. On the other hand, given a classifier, the α-Mahalanobis depth versions outperform those using MBD. In summary, DD^G-Mah is the undoubted winner in this family with better performance in average and significant victories in most datasets.

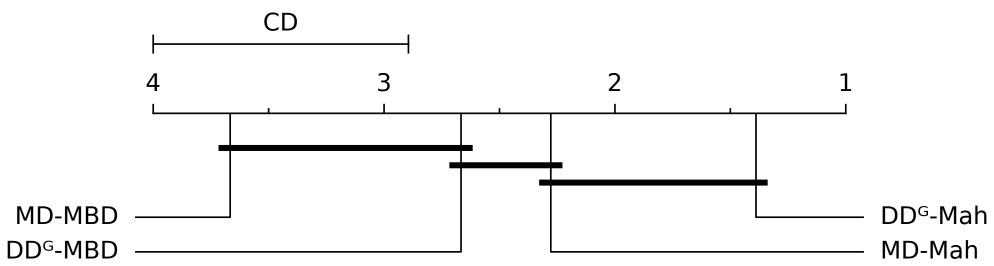


FIGURE 8.25: Critical distance diagram of the depth-based classifiers.

8.6.4 Functional k-NN classifiers

As mentioned in Section 2.3.4, it is possible to extend the k-nearest neighbors (k-NN) classifier to the functional setting by using a distance between functions to find the

TABLE 8.5: Comparison between functional k-NN classifiers.

	k-NN-L^1	k-NN-L^2	k-NN-Mah	k-NN-Ang
ArrowHead	0.883 ± 0.038 (4)	0.892 ± 0.034 (3)	**0.903 ± 0.031 (1)***	0.896 ± 0.034 (2)
Australian	0.945 ± 0.029 (3)	0.944 ± 0.024 (4)	0.951 ± 0.026 (2)	**0.958 ± 0.028 (1)***
Cell	0.918 ± 0.050 (3)	**0.935 ± 0.047 (1)***	0.924 ± 0.051 (2)	0.877 ± 0.055 (4)
Coffee	0.971 ± 0.041 (3)	0.978 ± 0.033 (2)	0.962 ± 0.049 (4)	**0.979 ± 0.033 (1)**
ECG	0.996 ± 0.002 (4)	0.998 ± 0.002 (2)	0.998 ± 0.002 (3)	**0.999 ± 0.002 (1)***
Fish	0.796 ± 0.037 (3)	0.816 ± 0.032 (2)	0.776 ± 0.033 (4)	**0.818 ± 0.032 (1)**
Growth	**0.957 ± 0.033 (1)**	0.956 ± 0.036 (2)	0.921 ± 0.052 (3)	0.918 ± 0.042 (4)
GunPoint	**0.951 ± 0.028 (1)***	0.938 ± 0.026 (3)	0.916 ± 0.031 (4)	0.938 ± 0.026 (2)
MCO	0.774 ± 0.079 (4)	0.801 ± 0.072 (3)	**0.986 ± 0.025 (1)***	0.941 ± 0.036 (2)
Medflies	0.541 ± 0.036 (3)	0.540 ± 0.034 (4)	0.544 ± 0.035 (2)	**0.550 ± 0.035 (1)**
NOx	**0.894 ± 0.046 (1)***	0.877 ± 0.046 (2)	0.871 ± 0.054 (3)	0.858 ± 0.049 (4)
Phoneme	0.909 ± 0.007 (2)	0.908 ± 0.006 (3)	**0.911 ± 0.006 (1)***	0.894 ± 0.007 (4)
Phoneme (binary)	**0.809 ± 0.014 (1)**	0.808 ± 0.015 (2)	0.803 ± 0.015 (4)	0.806 ± 0.015 (3)
Plane	0.968 ± 0.021 (2)	0.964 ± 0.022 (4)	**0.983 ± 0.014 (1)***	0.964 ± 0.021 (3)
Symbols	**0.966 ± 0.008 (1)**	0.962 ± 0.009 (4)	0.966 ± 0.009 (2)	0.962 ± 0.009 (3)
Tecator	0.798 ± 0.051 (4)	0.825 ± 0.048 (3)	**0.952 ± 0.024 (1)***	0.939 ± 0.028 (2)
Tecator (2nd derivative)	**0.980 ± 0.018 (1)**	0.973 ± 0.019 (4)	0.980 ± 0.019 (3)	0.980 ± 0.017 (2)
Yoga	0.932 ± 0.008 (3)	**0.933 ± 0.008 (1)**	0.916 ± 0.009 (4)	0.933 ± 0.008 (2)
Average accuracy	*0.888*	*0.892*	***0.903***	*0.901*
Average rank	*2.444*	*2.722*	*2.500*	***2.333***

neighbors. We have tested the k-NN classifier implemented in **scikit-fda** with the same metrics used for the NC classifiers (Subsection 8.6.1): L^1, L^2, α-Mahalanobis and angular distance. Parameters k and α are selected by cross-validation as before. The results by classifier-dataset can be seen in Table 8.5, and the corresponding critical distance diagram is shown in Figure 8.26.

Rank results show a certain equality between all the proposals, with no significant differences (see Figure 8.26). However, versions with Mahalanobis and angular distances obtain better results in terms of accuracy. These differences are mostly motivated by the bad performances of L^1 and L^2 proposals in MCO and Tecator datasets, which probably need a more global approach. Finally, we choose k-NN-Mah over k-NN-Ang because of the number of significant victories.

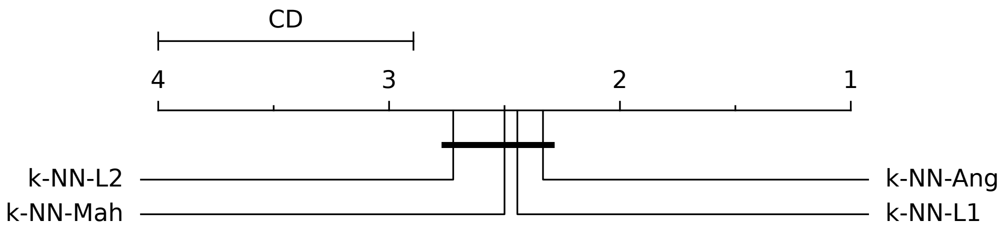


FIGURE 8.26: Critical distance diagram of the functional k-NN classifiers.

8.6.5 Final comparison

In this section, a final comparison is made between the best classifiers of each family and a random forest composed of 100 trees. The results are summarized in Table 8.6 and in Figure 8.27.

A first conclusion of this study is that the α-Mahalanobis distance, recently proposed in Berrendero, Bueno-Larraz, and Cuevas, 2020, has very good properties for classification when used by a distance-based functional classifier, such as nearest centroids or k-NN. Furthermore, it has also proven to be useful to design depth-based classifiers.

TABLE 8.6: Final comparison between the classifiers.

	NC-Mah	**RKVS+QDA**	**DD^G-Mah**	***k*-NN-Mah**	**RF**
ArrowHead	0.782 ± 0.043 (5)	0.832 ± 0.043 (3)	0.825 ± 0.043 (4)	**0.903 ± 0.031 (1)***	0.852 ± 0.041 (2)
Australian	0.903 ± 0.040 (3)	0.890 ± 0.039 (5)	0.898 ± 0.043 (4)	0.951 ± 0.026 (2)	**0.961 ± 0.026 (1)***
Cell	0.869 ± 0.063 (4)	0.841 ± 0.062 (5)	0.875 ± 0.058 (3)	**0.924 ± 0.051 (1)***	0.911 ± 0.048 (2)
Coffee	**1.000 ± 0.000 (1)***	0.970 ± 0.045 (4)	0.972 ± 0.049 (3)	0.962 ± 0.049 (5)	0.979 ± 0.029 (2)
ECG	0.971 ± 0.006 (4)	0.987 ± 0.006 (3)	0.942 ± 0.013 (5)	**0.998 ± 0.002 (1)***	0.992 ± 0.003 (2)
Fish	0.716 ± 0.041 (4)	**0.824 ± 0.037 (1)***	0.687 ± 0.044 (5)	0.776 ± 0.033 (3)	0.806 ± 0.037 (2)
Growth	**0.958 ± 0.032 (1)**	0.952 ± 0.038 (2)	0.927 ± 0.044 (3)	0.921 ± 0.052 (4)	0.917 ± 0.059 (5)
GunPoint	0.770 ± 0.051 (5)	0.909 ± 0.042 (3)	0.835 ± 0.053 (4)	0.916 ± 0.031 (2)	**0.964 ± 0.022 (1)***
MCO	**0.986 ± 0.026 (1)**	0.959 ± 0.040 (3)	0.904 ± 0.053 (4)	0.986 ± 0.025 (2)	0.783 ± 0.081 (5)
Medflies	0.548 ± 0.034 (3)	0.589 ± 0.033 (2)	0.515 ± 0.038 (5)	0.544 ± 0.035 (4)	**0.621 ± 0.027 (1)***
NOx	0.898 ± 0.052 (2)	**0.919 ± 0.037 (1)***	0.863 ± 0.053 (5)	0.871 ± 0.054 (3)	0.869 ± 0.054 (4)
Phoneme	0.910 ± 0.006 (4)	**0.927 ± 0.005 (1)***	0.905 ± 0.006 (5)	0.911 ± 0.006 (3)	0.924 ± 0.006 (2)
Phoneme (binary)	0.812 ± 0.015 (2)	**0.815 ± 0.015 (1)***	0.804 ± 0.016 (4)	0.803 ± 0.015 (5)	0.810 ± 0.015 (3)
Plane	0.969 ± 0.020 (4)	0.974 ± 0.020 (3)	0.937 ± 0.029 (5)	**0.983 ± 0.014 (1)**	**0.983 ± 0.017 (1)**
Symbols	0.920 ± 0.016 (5)	0.950 ± 0.016 (4)	0.957 ± 0.011 (3)	0.966 ± 0.009 (2)	**0.967 ± 0.011 (1)**
Tecator	0.953 ± 0.021 (3)	**0.974 ± 0.019 (1)***	0.967 ± 0.024 (2)	0.952 ± 0.024 (4)	0.810 ± 0.048 (5)
Tecator (2nd derivative)	0.959 ± 0.021 (5)	0.978 ± 0.015 (3)	0.978 ± 0.017 (4)	0.980 ± 0.019 (2)	**0.990 ± 0.012 (1)***
Yoga	0.578 ± 0.016 (5)	0.752 ± 0.027 (3)	0.698 ± 0.015 (4)	0.916 ± 0.009 (2)	**0.932 ± 0.008 (1)***
Average accuracy	*0.861*	*0.891*	*0.860*	***0.903***	*0.893*
Average rank	*3.389*	*2.667*	*4.000*	*2.611*	***2.278***

From the results, it is apparent that the best overall accuracy is achieved by the functional version of k-NN that employs the α-Mahalanobis distance, random forest, and the classifier that uses RKVS followed by QDA.

The depth-based classifier has the poorest performance between the methods considered. This means that depth, while being a useful statistical concept, does not capture all the information necessary for classification. It should therefore be used in combination with additional features.

The performance of the nearest centroid classifier is affected by the poor accuracy obtained when the mean is not a good prototype of the curves of a given class; for example, if the distribution of the curves is asymmetric, multimodal or the curves are not aligned, as in ArrowHead, GunPoint and Yoga.

Random forest (RF) achieves the best average rank in the problems considered. This means that it is one of the best classifiers not only for multivariate, but also for functional classification problems. A possible explanation is that RF takes advantage of the high dimensionality of the data to build decision trees whose predictions are complementary, in the sense that their individual errors tend to be independent and are therefore averaged out in the final forest prediction. An interesting avenue of exploration is to enhance the design of RF to take advantage of the functional nature of the data.

The multivariate classifiers LDA, QDA, k-NN, and RF have an excellent overall performance when used in combination with RKVS. This is probably a consequence of the functional nature of this variable selection procedure. A further advantage is that the data, in principle infinite dimensional, is represented in a low dimensional space. This dimensionality reduction entails significant gains in efficiency, simplifies the analysis, and improves the interpretability of the classifiers learned.

The functional k-NN classifier is robust and performs consistently well in the classification problems considered, as evidenced by the fact that it has the highest average accuracy. This provides further empirical support to the proposal made in Baíllo, Cuevas, and Fraiman, 2010 for k-NN to be used as a benchmark method for comparison in functional classification problems.

8.7 Conclusions

We have presented **scikit-fda**, a Python package that implements the useful techniques for working with functional data developed in the area of functional data analysis (FDA). This package include classes for representing functional data in a

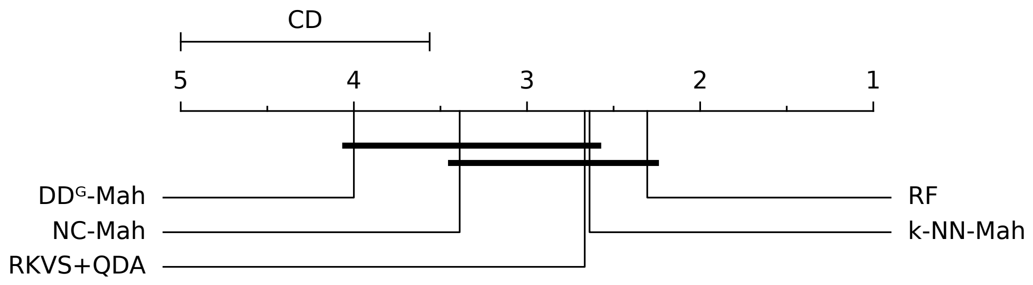


FIGURE 8.27: Critical distance diagram of the classifiers.

computer, in both discretized and basis expansion forms. It also provides tools for preprocessing this data, including smoothing, registration and dimensionality reduction methods. Exploratory analysis techniques, such as robust statistics, outlier detection algorithms and interactive visualization tools, have also been developed. In addition, the package provides several machine learning methods, for performing regression, classification and clustering tasks with functional data. These methods have been designed to be compatible with and integrated in the scientific Python ecosystem, one of the programming environments most widely used by machine learning practitioners. In particular, it conforms to the application programming interface (API) of **scikit-learn**, the go-to package for machine learning in Python, and thus can reuse much of the functionality in that package. Furthermore, the **scikit-fda** has been endowed with a comprehensive documentation, which includes examples, tutorials and reference material.

The usage of **scikit-fda** has been illustrated with three real-data examples. Moreover, we performed an analysis of functional classification methods from different families using the functionalities from this package. The results of this analysis show that, at least in the problems considered, the family of functional k-NN classifiers obtains very good accuracy, which is also fairly consistent accross datasets. Moreover, we show that a random forest classifier can be applied to functional data, obtaining also very good results. Finally, the strategy of performing functional variable selection followed by a multivariate classifier is also very competitive against these approaches.

Chapter 9

dcor: distance correlation and energy statistics in Python

This chapter presents **dcor**, an open-source Python package that provides tools to compute statistics related to the energy distance, such as the distance covariance and the distance correlation, and perform homogeneity and independence tests that utilize these quantities (Ramos-Carreño, 2020; Ramos-Carreño and Torrecilla, 2023). In particular, the **dcor** package was initially created in order to provide a fast implementation of the distance covariance and the distance correlation, necessary to implement the maxima hunting (MH) and recursive maxima hunting (RMH) algorithms for functional dimensionality reduction discussed in Chapter 5. The package was later enhanced with additional tools to compute related energy statistics. These statistics include distances between distributions and the associated tests for homogeneity and independence. Some of the most efficient algorithms for the estimation of these measures have been implemented relying on optimization techniques such as vectorization, compilation, and parallelization. The performance of these estimators is evaluated by comparison with alternative implementations in other packages. The package is also designed to be compatible with the packages conforming the scientific Python ecosystem. With that purpose in mind, **dcor** is an early adopter of the Python array API standard.

9.1 Energy statistics

The so-called energy distance is a metric that can be used to compare the distributions of two random vectors (Szekely, 1989). The name energy arises from the connections of this metric with Newton's potential energy (Szekely, 2002). It has a number of desirable properties, such as rotational invariance, scale equivariance, and characterizing the equivalence of distributions (i.e., it is equal to zero if and only if the distributions are identical) (Székely and Rizzo, 2013a; Rizzo and Székely, 2016). The first application of a distance between distributions is usually testing homogeneity. In this sense, nonparametric tests based on the energy distance have been proposed for testing the equality of multiple multivariate distributions (Székely and Rizzo, 2004) or for change point detection in time series (Kim et al., 2009), among others. The energy distance has also been used to provide a nonparametric extension of the classical ANOVA (Rizzo and Székely, 2010) and goodness-of-fit tests for several distributions (Székely and Rizzo, 2005; Rizzo, 2009; Yang, 2012). Meanwhile, this metric can be also considered on its own, for example in hierarchical clustering (Szekely and Rizzo, 2005).

In general, the statistics related to the energy distance are called energy statistics, or E-statistics (Rizzo and Székely, 2016). The most popular energy statistics are distance covariance and correlation, which measure independence between two random vectors (Székely, Rizzo, and Bakirov, 2007). They can be seen as a generalization of the classical notions of covariance and Pearson's correlation as they are able to capture nonlinear dependencies and are defined for vectors of arbitrary dimensions. These measures have become very popular in recent years and have been used in many application areas such causal analysis (Zhang et al., 2014), feature selection (Yenigün and Rizzo, 2015; Berrendero, Cuevas, and Torrecilla, 2016b), robust statistics (Kasieczka and Shih, 2020), and testing independence (Rizzo and Székely, 2016).

In this chapter we present the functionalities of **dcor**, a Python package dedicated to E-statistics (Ramos-Carreño, 2020). The library **dcor** is an open source project that seeks to provide statistical tools based on energy statistics to the Python community in an easy-to-use way. It contains different estimators for some E-statistics, with special attention to distance correlation and covariance, and nonparametric tests for both homogeneity and independence. These statistical tools are briefly described in Section 9.2. A complete description of all functionalities is provided in the package documentation, available online at `https://dcor.readthedocs.io/`.

During the design of the library, we have tried to maximize compatibility with other tools of the scientific Python ecosystem. Another focal point has been the quality and efficiency of the code, paying special attention to automated testing, code vectorization and compilation, or parallelization, among others. Section 9.3 is devoted to the implementation details related to performance and extensibility. Moreover, the computational efficiency of the principal measures in **dcor** is compared with the reference R package **energy** (Rizzo and Szekely, 2022) and recent alternative implementations in both R and Python. Finally, the impact of the package is addressed in Section 9.4, and some general conclusions are drawn in Section 9.5.

9.2 Functionalities of the package

The main goal of the **dcor** package is to provide efficient estimators for distance correlation and other E-statistics, such as the energy distance and partial distance correlation. In addition, hypothesis tests for homogeneity and independence of distributions based on these measures are provided. A brief description of these functionalities is given below.

9.2.1 Energy distance

Energy distance is a metric between the distributions of two random vectors $\mathbf{X}, \mathbf{Y}$ that take values in $\mathbb{R}^d$ (Székely and Rizzo, 2013a). It is defined as

$$\mathcal{E}(\mathbf{X}, \mathbf{Y}) = 2\mathbb{E}(\|\mathbf{X} - \mathbf{Y}\|) - \mathbb{E}(\|\mathbf{X} - \mathbf{X}'\|) - \mathbb{E}(\|\mathbf{Y} - \mathbf{Y}'\|), \tag{9.1}$$

where $\|\cdot\|$ stands for the Euclidean norm, and $\mathbf{X}'$ and $\mathbf{Y}'$ denote independent and identically distributed copies of $\mathbf{X}$ and $\mathbf{Y}$, respectively.

Alternatively, denoting their respective characteristic functions by $\phi_{\mathbf{X}}(\mathbf{t}) = \mathbb{E}\left[e^{i\mathbf{t}\mathbf{X}}\right]$ and $\phi_{\mathbf{Y}}(\mathbf{t}) = \mathbb{E}\left[e^{i\mathbf{t}\mathbf{Y}}\right]$, this measure can be rewritten as

$$\mathcal{E}(\mathbf{X}, \mathbf{Y}) = \frac{1}{c_d} \int_{\mathbb{R}^d} \frac{|\phi_{\mathbf{X}}(\mathbf{t}) - \phi_{\mathbf{Y}}(\mathbf{t})|^2}{\|\mathbf{t}\|^{d+1}} d\mathbf{t}, \tag{9.2}$$

where $c_d = \frac{\pi^{(1+d)/2}}{\Gamma((1+d)/2)}$ is half the surface area of the unit sphere in $\mathbb{R}^d$.

Let $\{\mathbf{x}_i\}_{i=1}^{N_\mathbf{X}}$ and $\{\mathbf{y}_i\}_{i=1}^{N_\mathbf{Y}}$ be two samples of $\mathbf{X}$ and $\mathbf{Y}$ with sizes $N_\mathbf{X}$ and $N_\mathbf{Y}$, respectively. An estimator of the energy distance based on the sample means can be defined as

$$\begin{aligned}\mathcal{E}_{N_\mathbf{X},N_\mathbf{Y}}(\mathbf{X},\mathbf{Y}) = {} & \frac{2}{N_\mathbf{X}N_\mathbf{Y}}\sum_{i=1}^{N_\mathbf{X}}\sum_{j=1}^{N_\mathbf{Y}}\|\mathbf{x}_i - \mathbf{y}_j\| \\ & - \frac{1}{N_\mathbf{X}^2}\sum_{i=1}^{N_\mathbf{X}}\sum_{j=1}^{N_\mathbf{X}}\|\mathbf{x}_i - \mathbf{x}_j\| - \frac{1}{N_\mathbf{Y}^2}\sum_{i=1}^{N_\mathbf{Y}}\sum_{j=1}^{N_\mathbf{Y}}\|\mathbf{y}_i - \mathbf{y}_j\|.\end{aligned} \tag{9.3}$$

The function `energy_distance()` implements this estimator. In order to improve the robustness of the estimation, the **dcor** package allows the use of other centrality measures, such as a trimmed mean or the median, as proposed in James, Kejariwal, and Matteson, 2016. Moreover, an unbiased version of this estimator based on the use of U-statistics is also available, as proposed in Matteson and James, 2014; James, Kejariwal, and Matteson, 2016.

9.2.2 Distance covariance and correlation

Distance covariance $\mathcal{V}$, and distance correlation $\mathcal{R}$, are dependency measures between two random vectors, $\mathbf{X}$ and $\mathbf{Y}$, with finite first moments (Székely, Rizzo, and Bakirov, 2007; Székely and Rizzo, 2009). Unlike the classical Pearson correlation, these measures can detect nonlinear dependencies. Indeed, they are equal to 0 if and only if the random vectors are independent. Furthermore, $\mathcal{V}$ and $\mathcal{R}$ can be defined for vectors of arbitrary dimensions, $\mathbf{X} \in \mathbb{R}^p$ and $\mathbf{Y} \in \mathbb{R}^q$.

In this context, the squared distance covariance $\mathcal{V}^2(\mathbf{X},\mathbf{Y})$ is defined as a weighted distance between the joint characteristic function $\phi_{\mathbf{X},\mathbf{Y}}(\mathbf{t},\mathbf{s})$ and the product of the marginals $\phi_\mathbf{X}(\mathbf{t})$ and $\phi_\mathbf{Y}(\mathbf{s})$. The distance covariance $\mathcal{V}(\mathbf{X},\mathbf{Y})$ is then the nonnegative number that verifies

$$\mathcal{V}^2(\mathbf{X},\mathbf{Y}) = \int_{\mathbb{R}^{p+q}} |\phi_{\mathbf{X},\mathbf{Y}}(\mathbf{t},\mathbf{s}) - \phi_\mathbf{X}(\mathbf{t})\phi_\mathbf{Y}(\mathbf{s})|^2 w(\mathbf{t},\mathbf{s})d\mathbf{t}d\mathbf{s}, \tag{9.4}$$

where $w(\mathbf{t},\mathbf{s}) = (c_p c_q \|\mathbf{t}\|_p^{1+p}\|\mathbf{s}\|_q^{1+q})^{-1}$, $\|\cdot\|_d$ is the euclidean norm in $\mathbb{R}^d$ and c_d is again half the surface area of the unit sphere in $\mathbb{R}^d$.

Analogously to classical Pearson correlation, distance correlation $\mathcal{R}(\mathbf{X},\mathbf{Y})$ is defined from the distance covariance as

$$\mathcal{R}^2(\mathbf{X},\mathbf{Y}) = \begin{cases} \frac{\mathcal{V}^2(\mathbf{X},\mathbf{Y})}{\sqrt{\mathcal{V}^2(\mathbf{X},\mathbf{X})\mathcal{V}^2(\mathbf{Y},\mathbf{Y})}} & \text{if } \mathcal{V}^2(\mathbf{X},\mathbf{X})\mathcal{V}^2(\mathbf{Y},\mathbf{Y}) > 0, \\ 0 & \text{if } \mathcal{V}^2(\mathbf{X},\mathbf{X})\mathcal{V}^2(\mathbf{Y},\mathbf{Y}) = 0. \end{cases} \tag{9.5}$$

In spite of the apparent complexity of these definitions, distance covariance and correlation have a simple parameter-free estimator. This is an advantage over other popular dependency measures, such as mutual information, which require the estimation of additional smoothing parameters (Vergara and Estévez, 2014; Laarne, Zaidan, and Nieminen, 2021). Given a sample $\{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^N$ of N observations of the joint random vector $(\mathbf{X},\mathbf{Y})$, we define the double centered distance matrices $\mathbf{A}$ and

B as

$$\mathbf{A}_{i,j} = a_{i,j} - \frac{1}{N}\sum_{l=1}^{N} a_{il} - \frac{1}{N}\sum_{k=1}^{N} a_{kj} + \frac{1}{N^2}\sum_{k,l=1}^{N} a_{kl},$$
$$\mathbf{B}_{i,j} = b_{i,j} - \frac{1}{N}\sum_{l=1}^{N} b_{il} - \frac{1}{N}\sum_{k=1}^{N} b_{kj} + \frac{1}{N^2}\sum_{k,l=1}^{N} b_{kl},$$

where $a_{ij} = \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_p$ and $b_{ij} = \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_q$. Then, the sample distance covariance is the square root of

$$\mathcal{V}_N^2(\mathbf{X}, \mathbf{Y}) = \frac{1}{N^2}\sum_{i,j=1}^{N} \mathbf{A}_{i,j}\mathbf{B}_{i,j}. \tag{9.6}$$

Likewise, the (squared) sample distance correlation is the standardized (squared) sample covariance

$$\mathcal{R}_N^2(\mathbf{X}, \mathbf{Y}) = \begin{cases} \frac{\mathcal{V}_N^2(\mathbf{X},\mathbf{Y})}{\sqrt{\mathcal{V}_N^2(\mathbf{X},\mathbf{X})\mathcal{V}_N^2(\mathbf{Y},\mathbf{Y})}}, & \text{if } \mathcal{V}_N^2(\mathbf{X}, \mathbf{X})\mathcal{V}_N^2(\mathbf{Y}, \mathbf{Y}) > 0, \\ 0, & \text{if } \mathcal{V}_N^2(\mathbf{X}, \mathbf{X})\mathcal{V}_N^2(\mathbf{Y}, \mathbf{Y}) = 0. \end{cases} \tag{9.7}$$

Both estimators $\mathcal{V}_N$ and $\mathcal{R}_N$ converge almost surely to their population counterparts $\mathcal{V}$ and $\mathcal{R}$ when N tends to infinity (Székely, Rizzo, and Bakirov, 2007).

Functions `distance_covariance()` and `distance_correlation()` implement the estimators in Equation (9.6) and Equation (9.7), respectively. The library also provides unbiased and bias-corrected estimators of measures $\mathcal{V}^2(\mathbf{X}, \mathbf{Y})$ and $\mathcal{R}^2(\mathbf{X}, \mathbf{Y})$, as functions `u_distance_covariance_sqr()` and `u_distance_correlation_sqr()` (Székely and Rizzo, 2014). Note that in this case one cannot take the square root as the estimator can take negative values. In addition, an affinely invariant version of distance correlation is available as the `distance_correlation_af_inv()` function (Dueck et al., 2014).

Despite of its simplicity, the main drawback of the estimator defined in Equation (9.6) is its high computational cost. Due to the evaluation of the distance matrices, the complexity in both time and memory is $O(N^2)$. As an alternative, Huo and Székely, 2016 and Chaudhuri and Hu, 2019 propose two different estimators of the distance covariance based on the AVL-tree structure (Adelson-Velskii and Landis, 1962) and the mergesort algorithm, respectively. Both proposals have complexity $O(N \log N)$, although they are restricted to univariate distributions ($p = q = 1$). These fast algorithms are available in the distance covariance and correlation functions by means of the parameter `method`, and are used by default when possible.

9.2.3 Partial distance correlation

Partial distance covariance and correlation are extensions of the aforementioned dependency measures that allow for controlling for the effects of a third random vector **Z** of arbitrary dimension (Székely and Rizzo, 2017).

Partial distance covariance is defined as

$$\mathcal{V}^*(\mathbf{X}, \mathbf{Y}; \mathbf{Z}) = \begin{cases} \mathcal{V}^2(\mathbf{X}, \mathbf{Y}) - \frac{\mathcal{V}^2(\mathbf{X},\mathbf{Z})\mathcal{V}^2(\mathbf{Y},\mathbf{Z})}{\mathcal{V}^2(\mathbf{Z},\mathbf{Z})} & \text{if } \mathcal{V}^2(\mathbf{Z}, \mathbf{Z}) \neq 0, \\ \mathcal{V}^2(\mathbf{X}, \mathbf{Y}) & \text{if } \mathcal{V}^2(\mathbf{Z}, \mathbf{Z}) = 0, \end{cases}$$

while partial distance correlation is

$$\mathcal{R}^*(\mathbf{X},\mathbf{Y};\mathbf{Z}) = \begin{cases} \frac{\mathcal{R}^2(\mathbf{X},\mathbf{Y})-\mathcal{R}^2(\mathbf{X},\mathbf{Z})\mathcal{R}^2(\mathbf{Y},\mathbf{Z})}{\sqrt{1-\mathcal{R}^4(\mathbf{X},\mathbf{Z})}\sqrt{1-\mathcal{R}^4(\mathbf{Y},\mathbf{Z})}} & \text{if } \mathcal{R}^4(\mathbf{X},\mathbf{Z}) \neq 1 \text{ and } \mathcal{R}^4(\mathbf{Y},\mathbf{Z}) \neq 1, \\ 0 & \text{if } \mathcal{R}^4(\mathbf{X},\mathbf{Z}) = 1 \text{ or } \mathcal{R}^4(\mathbf{Y},\mathbf{Z}) = 1. \end{cases}$$

The estimators of these quantities proposed in Székely and Rizzo, 2014 are provided in `partial_distance_covariance()` and `partial_distance_correlation()` functions, respectively.

Finally, note that these dependency measures should be used carefully as they present some undesirable or counterintuitive properties (Székely and Rizzo, 2014, Sec. 4.2). In particular, $\mathcal{R}^*(\mathbf{X},\mathbf{Y};\mathbf{Z}) = 0$ does not always imply that $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$ and vice versa.

9.2.4 Hypothesis testing

One of the main applications of the distances between distributions is hypothesis testing. In this sense, measures based on E-statistics are no exception. For example, Székely and Rizzo, 2005 proposes a goodness-of-fit test for multivariate normality based on the energy distance, and a test of independence can be found in Bakirov, Rizzo, and Székely, 2006, both implemented in the **energy** R package. Meanwhile, in the **dcor** package we can find three different tests: one for homogeneity and two for independence.

First, the energy distance is used to define a permutation test of homogeneity (Székely and Rizzo, 2004). This test is based on the properties of the statistic

$$T = \frac{N_{\mathbf{X}}N_{\mathbf{Y}}}{N_{\mathbf{X}}+N_{\mathbf{Y}}}\mathcal{E}_{N_{\mathbf{X}},N_{\mathbf{Y}}}(\mathbf{X},\mathbf{Y}), \tag{9.8}$$

where $\mathcal{E}_{N_{\mathbf{X}},N_{\mathbf{Y}}}$ is the estimator of the energy distance defined in Equation (9.3). This test, and its extension for more than two populations, is implemented in the function `homogeneity.energy_test()`.

Second, a permutation test for detecting independence between two distributions is constructed using the distance covariance (Székely, Rizzo, and Bakirov, 2007). The statistic used for this test is $N\mathcal{V}_N^2$, where $\mathcal{V}_N^2$ is the estimator of the distance covariance defined in Equation (9.6). This test is implemented by function `independence.distance_covariance_test()`.

Finally, function `independence.distance_correlation_t_test()` provides an asymptotic test of independence for high dimensional random vectors (Székely and Rizzo, 2013b). This test relies on the convergence of the statistic

$$T_N = \frac{\mathcal{R}_N^*(\mathbf{X},\mathbf{Y})}{\sqrt{1-(\mathcal{R}_N^*(\mathbf{X},\mathbf{Y}))^2}}\sqrt{\frac{N(N-3)}{2}-1}, \tag{9.9}$$

where $\mathcal{R}_N^*(\mathbf{X},\mathbf{Y})$ is the bias-corrected version of the estimator of the squared distance correlation proposed in Székely and Rizzo, 2014.

Some additional parameters of these tests can be adjusted by the user, including the number of repetitions used in the permutation tests.

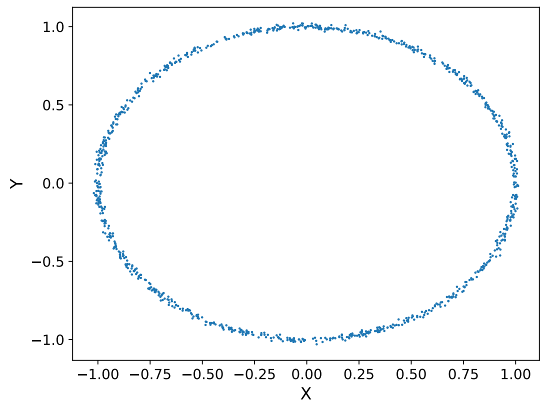


FIGURE 9.1: Ring of data generated for the illustrative example.

9.2.5 Illustrative example

This section shows how to use the **dcor** package through a brief example with a toy dataset.

1. Generation of the dataset In this case, we generate 1000 pairs of coordinates (x, y) conforming an annulus (see Figure 9.1).

```python
import numpy as np
import dcor

n_samples = 1000
random_state = np.random.default_rng(123456)
u = random_state.uniform(-1, 1, size=n_samples)

y = (np.cos(u * np.pi)
    + random_state.normal(0, 0.01, size=n_samples))
x = (np.sin(u * np.pi)
    + random_state.normal(0, 0.01, size=n_samples))
```

Hence, the random vectors X and Y are non independent, but identically distributed.

2. Distances Computation of $\mathcal{E}(X, Y)$ and $\mathcal{R}(X, Y)$ with the AVL algorithm. Remember that fast implementations are only valid for one-dimensional variables.

```python
dcor.energy_distance(x, y)
```

```
Out:
    0.00152....
```

```python
dcor.distance_correlation(x, y, method="avl")
```

```
Out:
    0.2097....
```

The close-to-zero value of the energy distance reflects the homogeneity of the distributions of X and Y. Meanwhile, the distance correlation indicates a small dependency between them.

3. Hypothesis testing Finally, we test homogeneity and independence to check if the above conclusions are statistically significant.

```
dcor.homogeneity.energy_test(
    x, y, num_resamples=100, random_state=random_state)
```

```
Out:
    HypothesisTest(pvalue=0.3069..., statistic=0.7649...)
```

```
dcor.independence.distance_covariance_test(
    x, y, num_resamples=100, random_state=random_state)
```

```
Out:
    HypothesisTest(pvalue=0.0099..., statistic=13.5334...)
```

Tests in **dcor** return an object with the p-value and the value of the statistic. As expected, with 95% confidence, we can accept the homogeneity (p-value $\simeq$ 0.31) and reject the independence (p-value $\simeq$ 0.099).

To conclude, let us point out that the **dcor** package owns detailed examples for most functionalities in the online documentation[1]. These examples include different datasets, measurements of error and performance, and background about the statistical methods.

9.3 Implementation details

The package **dcor** is implemented in Python 3.8 and released in Github (Ramos-Carreño, 2020) under a MIT license. Releases are available both in PyPI[2] and the conda-forge channel from Anaconda[3]. In order to guarantee the quality of the library, it includes a comprehensive and automated collection of tests, and a complete documentation. The code contains also type annotations for static analyzers. Some additional features aiming at improving performance and extensibility are described below.

9.3.1 Performance

The functions of the **dcor** package make a careful use of vectorization to guarantee improved performance. In addition, fast algorithms for distance covariance and correlation described in Section 9.2.2 make use of the **Numba** library (Lam, Pitrou, and Seibert, 2015) to compile and optimize them, what entails further speed gains.

In this section we compare the performance of the **dcor** implementations of the energy distance and distance correlation with their counterparts in other R and Python proposals. The R package **energy** (Rizzo and Szekely, 2022) is the reference library when working with E-statistics. It is more similar to **dcor** than any other software in terms of approach and scope. A detailed comparison between **dcor** and **energy** is available in the documentation. The recent **dcortools** (Edelmann and Fiedler, 2022) addresses the subset of E-statistics relying on distance covariance and correlation with a very careful implementation. On the other hand, there are no Python packages devoted to energy statistics or subsets of them. Nonetheless, some isolated functionalities can be found in some statistical libraries. This is the case

[1] https://dcor.readthedocs.io/en/latest/auto_examples/index.html
[2] https://pypi.org/project/dcor
[3] https://anaconda.org/conda-forge/dcor

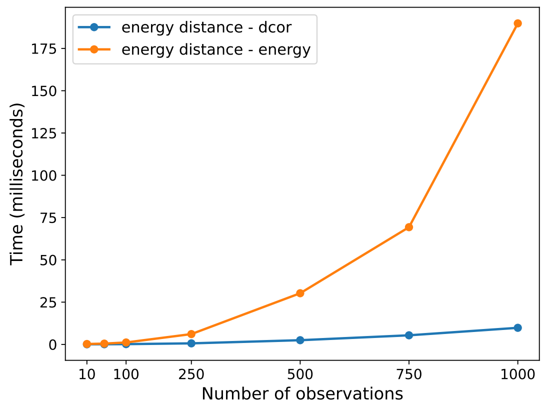


FIGURE 9.2: Performance comparison between Python (**dcor** package) and R (**energy** package) implementations of the energy distance.

of **statsmodels** (**seabold+perktold_2010_statsmodels**), **hyppo** (Panda et al., 2021), and **pingouin** (Vallat, 2018), which include some few E-statistics among extensive catalogs of statistical tools.

Since the complexity in terms of the dimensions, p and q, of $\mathbf{X}$ and $\mathbf{Y}$ does not depend on the distance algorithms themselves, but to the implementation of the Euclidean distance in each language (R and Python), we study only the effect of varying the sample size. Then, we consider $p = q = 1$ and the following sample sizes $N = N_X = N_Y$: 10, 50, 100, 250, 500, 750, 1000. Datasets are generated randomly form a standard normal distribution, as data distribution has no effect in the computational cost. For each example, we show the minimum of 100 independent single calls of each estimator for each sample size. The choice of the minimum instead of the sample mean or other statistics, responds to the objective of discarding the effects of other running processes (see, e.g. Chen and Revels, 2016 for further details).

On the one hand, Figure 9.2 shows the comparison between **dcor** (Python) and **energy** (R) implementations of the energy distance following the expression in Equation (9.3). At this moment, no other package in this study has a valid method for the energy distance. On the other hand, Figure 9.3 is dedicated to distance correlation implementations. Left panel presents the implementations of the original estimator defined in Equation (9.6) joint with fast versions following both the AVL-based algorithm (**dcor**, **energy**, and **dcortools**) proposed in Huo and Székely, 2016, and the fast mergesort-based implementation (**dcor**, and **hyppo**) proposed in Chaudhuri and Hu, 2019. The right panel focuses on fast algorithms, using a more appropriate scale to appreciate the differences. Distance covariance is not explicitly considered in the comparison as some of these packages do not expose that functionality to end users. Nonetheless, the performance of distance correlation directly depends on the performance of the distance covariance.

In all cases, **dcor** implementations clearly outperforms those in the reference package **energy** and their Python counterparts (**statsmodels**, **hyppo**, and **pingouin**). The recent R library **dcortools** is the only comparable with **dcor**. It exhibits also a very good performance, being slightly inferior to **dcor** with small sample sizes, but obtaining the best results with more than 250 observations. As expected, fast algorithms obtain the best results, although they are limited to univariate random vectors. In this context, **dcor** is the only library that implements the two fast algorithms, AVL and mergesort. Their results are very similar although the mergesort-based implementation shows a slightly, but consistently, better performance than the AVL approach.

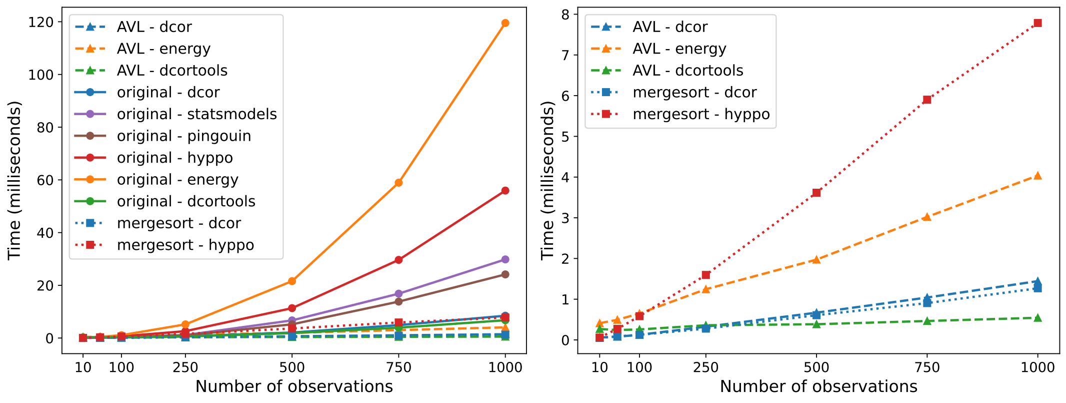


FIGURE 9.3: Performance comparison between Python (packages **dcor**, **statsmodels**, **hyppo**, and **pingouin**) and R (packages **energy** and **dcortools**) implementations of distance correlation. On the left, all estimators available in these packages. On the right, only fast algorithms.

Finally, the **dcor** package can exploit some additional opportunities for parallelization. A typical example is the repeated computation of a dependence measure, such as distance covariance, which naturally arises in variable selection or in correlation analysis. In this context, the function `rowwise()` can be used to handle each pair of variables in parallel (employing several CPUs by means of the **Numba** library) when a fast algorithm for distance covariance is used. Another occasion for parallelization appears in permutation tests, which also involve repeating calls to the same function. The difference here is that the distances involved are the same for each permutation, although in a different order, so they need to be computed only once. We can then leverage the **Joblib** library (Joblib Development Team, 2020) to launch the permutations in parallel with a configurable number of jobs. Note that these examples belong to the so called embarrasingly parallel problems, where the resultant speedup is directly proportional to the number of physical CPUs available.

9.3.2 Extensibility

A major concern in the scientific Python ecosystem is the proliferation of different frameworks, each with its own similar, but incompatible, data structures. As a result, several implementations of the same non-trivial algorithms may arise, with the subsequent mainteinance burden and fragmentation of the community. In the development process of the **dcor** package we have addressed this problem in two ways.

First and foremost, **dcor** adheres to the Python array API standard (Consortium for Python Data API Standards, 2022). This specification was created to offer a common interface for different implementations of arrays and tensors in Python. This is a joint effort of the Python scientific community that tries to integrate, among others, **NumPy** CPU-based arrays (Harris et al., 2020), **CuPy** GPU-based ones (Okuta et al., 2017), **Dask** distributed arrays (Dask Development Team, 2016) and the different types of tensors available in deep learning libraries, such as **Pytorch** (Paszke et al., 2019) or **Tensorflow** (Abadi et al., 2015). The package **dcor** accepts objects that follow this standard, and therefore, it could be used in combination with these libraries when they finish their standarization effort. In order to guarantee that integration, **dcor** has been tested against the `numpy.array_api` module, which contains a minimal reference implementation of the Python array API standard.

Furthermore, special care has been taken in **dcor** to allow the use of arrays of arbitrary numeric type, including non-floating point types such as `Fraction` and `Decimal`. As a consequence, the original types will be preserved along all computations and results.

The only exception to these approaches are those few functions which are compiled, as currently **Numba** supports only a subset of array and number types.

9.4 Impact and applications

During the last few years, E-statistics, and in particular distance correlation, have attracted quite a bit of attention, accumulating thousands of citations and being used in many different application areas. The package **dcor** gives the Python scientific community access to a comprehensive set of significant measures and tests based on E-statistics in a unified framework, previously available only in R through the **energy** package. In addition, the library presents some differential characteristics such as the adoption of the Python array API standard, and fast algorithms and code optimizations which lead to better performances than those of their R and Python counterparts in most cases. As a result, **dcor** could be very useful in the wide variety of situations where E-statistics are. Indeed, it is already being used in a number of relevant scientific publications, as well as several open source scientific packages.

Up to now, the use of **dcor** has been particularly frequent in areas related to machine learning. For example, distance correlation is used in causal inference (Markham, Chivukula, and Grosse-Wentrup, 2020; Runge, 2022) to detect the dependence structure of the data. The energy distance and distance correlation have been considered for word embeddings in natural language processing problems (Zhelezniak et al., 2019; Kayal, 2021). Recent works in bias detection (Synthesized, 2022) make use of the distance correlation to uncover attributes that act as proxies for sensitive data. Furthermore, distance covariance and correlation are commonly used in dimensionality reduction strategies, both in multivariate (Menvouta, Serneels, and Verdonck, 2023; Böhm, Berens, and Kobak, 2022) and functional (Ramos-Carreño et al., 2023) frameworks.

Meanwhile, the package has also proven to be useful in identifying medicinal plants (Kharyuk et al., 2018), analysing correlations between the Sustainable Development Goals of the United Nations (Laumann et al., 2022) or in a financial context, for either diversifying investments (Benowitz, 2020) or optimizing technical indicators for prediction (John Richardson, 2022).

9.5 Conclusions

This chapter has introduced the package **dcor**, a package that provides functionalities based on E-statistics to the Python scientific community. In addition to include a varied set of statistical measures and hypothesis tests, the library has been designed with efficiency and extensibility in mind. Some examples of this approach are the flexibility about different array and numeric types and the auxiliary tools for parallelization. The efficiency criterion is also reflected in code optimizations like vectorization and compilation, which allow **dcor** to obtain better performance than the alternatives in R.

The interest in these statistical tools has been shown by the good reception of the package in many different application areas since its first release in 2017. This attention made **dcor** a reference package for E-statistics in Python, and solidified our commitment with the development and expansion of the package. To this respect, our future plans include incorporating further developments derived from the theory of E-statistics, enriching the existing documentation with more tutorials and additional usage examples, and continue optimizing the existing functionalities.

To conclude, we want to emphasize our involvement with the open-source community, both addressing user comments and encouraging practitioners to contribute to the development of the **dcor** package.

Chapter 10

Conclusions and future work

Functional data are ubiquitous in many areas of application. Evidence of their importance is given by the flurry of activity in both the statistics and machine learning research communities. These types of data present special characteristics that are markedly different from multivariate observations. On the one hand, they are inherently infinite-dimensional, which entails difficulties for their statistical analysis: key quantities, such as the probability density function, or the likelihood ratio, are, in most cases, ill-defined for random functions. On the other hand, they are highly redundant, in the sense that nearby points in a function provide similar information.

A first approach to learning from functions is to apply multivariate methods to the observations, which are either in discretized form or as a truncated basis expansion. However, it should be possible to design more effective machine learning methods that take advantage of the continuous nature and underlying smoothness of the functional observations. An idea that pervades this thesis is the importance of incorporating the tools of FDA, the branch of statistics that deals with random functions, into the design of machine learning methods with these types of data. Based on this overarching principle, we have designed, implemented, and analyzed statistical and machine learning models for functional data. In particular, the thesis presents novel methods and extensive analysis for classification, dimensionality reduction, and clustering problems. Additionaly, open-source libraries have been developed for statistical analysis and machine learning with functional data.

A first methodological contribution is the design of optimal classification rules for the classification of Gaussian processes (GPs). The approach followed consists in considering the quadratic discriminant classifier, which achieves the Bayes error when the functional observations are given in discretized form; i.e., as tables of values. In the limit that the discretization becomes denser, the discriminant defines a classification rule for Gaussian processes, which is optimal. The analysis carried out provides a novel perspective of how the near-perfect classification phenomenon (Delaigle and Hall, 2012) occurs in this context: the quadratic discriminant contains terms that diverge as the discretization grid becomes finer. When the GPs are equivalent these divergences cancel out and the optimal rule is given in terms of the Radon-Nikodym derivative between the corresponding distributions (Baíllo, Cuevas, and Cuesta-Albertos, 2011). When the GPs are orthogonal, an optimal rule results from retaining only the singular terms in the quadratic discriminant, which become dominant in this limit. This singular rule achieves, asymptotically, zero error in the population limit. Finally, as part of this work, the optimal classification rules for some cases of interest have been derived.

A second contribution is the analysis of recursive maxima hunting (RMH), a filter variable selection method, in the context of functional binary classification problems. This method selects variables that are relevant not only by themselves but also when considered in combination with others. Furthermore, we show that RMH selects precisely the points involved in the optimal classification rule for some homoscedastic classification problems. These include the problems in which the common noise is Brownian motion or an Ornstein-Uhlenbeck process, and the mean difference between classes belongs to the associated RKHS. An extensive empirical evaluation of the method both in synthetic and real-world problems serves as an illustration of the effectiveness of the method, which is comparable to other benchmark variable selection methods.

A third contribution is an analysis of the clustering algorithm fuzzy C-means when applied to functional data in discretized form. In particular, we consider Gaussian processes whose correlations have a characteristic lengthscale; that is, the values of the trajectories are approximately independent for distances much larger than this lengthscale. By varying this parameter, we can explore different regimes: when the characteristic length of the correlations is much smaller than the grid spacing, the algorithm exhibits poor convergence. This behavior observed is similar to the multivariate case analyzed in Winkler, Klawonn, and Kruse, 2010. The reason is that, since the grid points are further than the extent of the correlations, the function values at those points are uncorrelated. Therefore, the discretized trajectories are akin to multivariate attribute vectors whose components are independent. By contrast, the convergence improves as the correlation lengthscale increases and becomes comparable to the size of the discretization grid, a limit at which the functional nature of the data is manifest.

These contributions are complemented with the design and development of computational tools for statistical analysis and machine learning with functional data. In particular, we have developed two Python libraries that facilitate the retrieval of both multivariate and functional datasets. The package **scikit-datasets** allows the user to download datasets from several widely used online repositories, such as UCR, for functional data, or UCI, for multivariate data. It also provides utility functions for carrying out large-scale comparative studies using these data and tools to summarize and analyze the results. The **rdata** library can be used to convert datasets from the internal format used in the R programming language to Python objects. In addition to being useful by itself, it is also used by the **scikit-datasets** package, in order to allow the direct loading of datasets available in CRAN, the R packages repository.

The Python package **scikit-fda** is the most prominent practical contribution of this thesis. This library provides a large set of computational tools for the analysis of functional data. These include classes to represent the trajectories in either discretized form or as truncated basis expansions, preprocessing and exploratory analysis tools, and machine learning models for functional data. The library has been developed according to the specifications of the scientific Python ecosystem. Furthermore, it is compatible with the widely-used machine learning library **scikit-learn**. In particular, the tools offered in **scikit-fda** can be used in **scikit-learn** pipelines and hyperparameter selection methods.

The functionality of the **scikit-fda** package is illustrated through several examples from real-world applications. In addition, we carried out a comparative study of different classification methods. This study not only illustrates the application of the tools developed (**scikit-datasets** and **scikit-fda**), but also provides insights into the weaknesses and strengths of the different methods.

Finally, motivated by the implementation of RMH, the **dcor** Python library was developed. This library implements the distance covariance and distance correlation, two measures of dependence between random vectors (Székely, Rizzo, and Bakirov, 2007). Related measures of homogeneity and dependence, and their corresponding hypothesis tests are also provided. The **dcor** package is being used in several scientific studies (Kharyuk et al., 2018; Laumann et al., 2022; Böhm, Berens, and Kobak, 2022) and software packages (John Richardson, 2022; Runge, 2022).

10.1 Future work

During the elaboration of this thesis, we have addressed many problems of interest in the area of machine learning with functional data. Some of the work lends itself to in-depth studies and extensions. This thesis has also opened up new avenues of research.

In particular, the exploration of recursive maxima hunting has focused on binary classification problems. Nevertheless, the algorithm can be readily applied to multiclass classification and regression problems. Additionally, the algorithm can be extended to multivariate functional data, or to functional data that depend on a continuous parameter in higher dimensions, such as surfaces or volumes.

Another interesting direction is an in-depth analysis of the relation between multivariate and functional problems. This has been done using two techniques. First, in Chapter 4 we considered the functional case in a discretized representation. When the discretization was coarse, we could analyze the data as a multivariate vector, obtaining an expression for the optimal classification rule. We studied how this rule changed with finer and finer grids, obtaining in the limit the expressions for the functional case. Secondly, in Chapter 6 we studied the variation of the lengthscale parameter that characterized the decay of the correlations for Gaussian processes. When the lengthscale was small, the behavior observed was similar than the one exhibited by multivariate data, while for larger lengthscales the functional nature of the data became apparent. We believe that the techniques employed in this thesis to study this transition between the multivariate and functional worlds ought to be analyzed in additional contexts in which there is a discrepance between the behavior in the multivariate case and the one observed for functional data. For example, one could analyze the behaviour of logistic regression in this transition to the functional case, in which the maximum likelihood estimator does not exist in general (Berrendero, Bueno-Larraz, and Cuevas, 2022), as well as other methods that depend in quantities that are ill-defined in the functional context, such as the mode.

From a practical perspective, there are many possible ways to extend the contributions in this thesis. In particular, we intend to make several improvements of the **scikit-fda** package. Specifically, we plan to include in the package tools for handling irregular functional data in which the observed points are different for each functional observation. These kinds of data require special techniques for their efficient representation and treatment, specially when the measurements are sparse. We also want to extend the package functionalities to support cases in which each observation consists of several functions and/or multivariate data. Moreover, we would want to improve support for handling functional observations that depend on several variables, such as images or surfaces.

New functional distances and statistical depth measures should also be considered for their inclusion in the package. Some examples are the extremal, or infimal, depth (Mosler, 2013; Narisetty and Nair, 2016) or its flexible modifications (Nagy et al., 2021). We also should define distances and depths for multivariate functional data. Additional functional methods for clustering, classification and regression should also be developed and included in the package, such as support vector machines (SVMs) (Rossi and Villa, 2006), generalized regression models (Scheipl, Gertheiss, and Greven, 2016), Gaussian process regression, or neural networks (Rossi and Conan-Guez, 2005; Kovachki et al., 2023). It would also be of interest to study the possibility of developing generative functional data models, such as variational autoencoders, that could be incorporated to the package.

The compatibility of **scikit-fda** with other software packages is also one of our priorities. Most advances in FDA made by the statistics community end having implementations in the R programming language. Even as we try to offer most part of these functionalities in **scikit-fda**, it is certain that not all available tools will be included, especially those that are very specific. Thus, it would be interesting to explore ways in which we could combine the functional data tools in R and Python in an effective way. In particular, it would be useful to integrate with the tools that allow direct communication between Python and R interpreters in the same process, such as **rpy2**, by offering efficient conversion functions between functional representations in both languages. This would allow to move between these languages at different points in the analysis and thus combine the strengths of both.

Finally, during the development of **scikit-fda** we noticed that functions in either discretized form or as a basis expansions arise frequently in other areas. These include branches of mathematics and physics, such as simulations of differential equations, or computer graphics. While in some cases only one function or a vector of functions are necessary, other contexts, such as the computation of the Jacobian, the Hessian, and higher derivatives require matrices and tensors of functions. Thus, we will consider creating a new Python package that can represent vectors, matrices or tensors whose elements are themselves general functions, in discretized or basis forms. This package should include also functions for operating with these objects, performing algebra and computing quantities such as functional distances. The package could then play the role for functions that **NumPy** (Harris et al., 2020) plays for multivariate data. By carefully designing it around the Python array API standard (Consortium for Python Data API Standards, 2022), we could also use these functions in more contexts in the future, such as with deep learning frameworks like **PyTorch** (Paszke et al., 2019).

Capítulo 10

Conclusiones y trabajo futuro

Los datos funcionales tienen gran presencia en muchas áreas de aplicación. Podemos encontrar evidencia de su importancia en la intensa actividad en las comunidades de estadística e investigación en aprendizaje automático. Estos tipos de datos presentan características especiales que son marcadamente distintas de las observaciones multivariantes. Por una parte, son inherentemente infinito-dimensionales, lo cual lleva a dificultades para su análisis estadístico: cantidades clave, como la función de densidad de probabilidad o el cociente de verosimilitudes se encuentran, en la mayoría de casos, mal definidas en el caso de funciones aleatorias. Por otro lado, son altamente redundantes, en el sentido de que puntos cercanos de la función proporcionan información similar.

Una primera aproximación para aprender a partir de funciones es aplicar métodos multivariantes a las observaciones, que se encuentran bien de forma discretizada o bien en una expansion en bases truncada. No obstante, debería ser posible diseñar métodos de aprendizaje automático que aprovechen la naturaleza continua y la suavidad subyacente de las observaciones funcionales. Una idea que impregna esta tesis es la importancia de incorporar las herramientas de FDA, la rama de la estadística que trata con funciones aleatorias, en el diseño de métodos de aprendizaje automático con este tipo de datos. Basándonos en este principio general, hemos diseñado, implementado y analizado modelos estadísticos y de aprendizaje automático para datos funcionales. En particular, la tesis presenta métodos novedosos y un análisis extenso para problemas de clasificación, reducción de dimensionalidad y agrupamiento. Adicionalmente, se han desarrollado bibliotecas de código abierto para el análisis estadístico y aprendizaje automático con datos funcionales.

Una primera contribución metodológica es el diseño de reglas de clasificación óptima para la clasificación de procesos gaussianos (GPs). La aproximación seguida consiste en considerar el clasificador de discriminante cuadrático, que obtiene el error de Bayes cuando las observaciones funcionales son dadas en forma discretizada, esto es, como tablas de valores. En el límite en el que la discretización se vuelve más densa, el discriminante define una regla de clasificación para procesos gaussianos que es óptima. El análisis llevado a cabo proporciona una perspectiva novedosa de como el fenómeno de clasificación casi perfecta (Delaigle y Hall, 2012) ocurre en este contexto: el discriminante cuadrático contiene términos que divergen cuando la rejilla de discretización se vuelve más fina. Cuando los procesos gaussianos son equivalentes estas divergencias se cancelan, y la regla óptima se da en términos de la derivada de Radon-Nikodym entre las distribuciones respectivas (Baíllo, Cuevas y Cuesta-Albertos, 2011). Cuando son ortogonales, una regla óptima puede obtenerse preservando solo los términos singulares del discriminante cuadrático, que se vuelven dominantes en este límite. Asintóticamente, esta regla logra un error nulo en el límite poblacional. Finalmente, como parte de este trabajo, las reglas de clasificación óptima se han derivado para algunos casos de interés.

Una segunda contribución es el análisis de recursive maxima hunting (RMH), un método de filtro para selección de variables, en el contexto de problemas de clasificación binaria con datos funcionales. Este método selecciona variables que son relevantes para la clasificación no solo por sí mismas, sino también cuando son tomadas junto a otras. Por añadido, RMH selecciona exactamente los puntos involucrados en la regla de clasificación óptima en algunos problemas de clasificación homoscedásticos. Estos incluyen aquellos cuyo proceso de ruido común es movimiento browniano o un proceso Ornstein-Uhlenbeck, y en los que la diferencia entre las medias de ambas clases pertenece al espacio de Hilbert con núcleo reproductor asociado. Una evaluación empírica extensa del método, en problemas tanto sintéticos como con datos reales, sirve para ilustrar la efectividad del método, que se muestra comparable con otros métodos de selección de variables usados como referencia.

Una tercera contribución es un análisis del algoritmo fuzzy C-means al ser aplicado a datos funcionales discretizados. En particular, consideramos procesos gaussianos cuya correlación tiene una escala de longitud característica, es decir, los valores de las trayectorias son aproximadamente independientes para distancias mucho mayores que su escala de longitud. Variando este parámetro podemos explorar distintos regímenes de comportamiento. Cuando la escala de longitud característica es mucho menor que el espaciado de la rejilla, el algoritmo muestra problemas de convergencia. Este comportamiento observado es similar al del caso multivariante analizado en Winkler, Klawonn y Kruse, 2010. La razón es que, como los puntos de la rejilla están más alejados que el alcance de las correlaciones, los valores de la función en esos puntos no guardan correlación. Por tanto, las trayectorias discretizadas son similares a vectores de atributos multivariantes cuyas componentes son independientes. En contraste, la convergencia mejora conforme la escala de longitud se incrementa y se vuelve comparable con el tamaño de la rejilla de discretización, un límite en el que la naturaleza funcional de los datos se pone de manifiesto.

Estas contribuciones son complementadas con el diseño y desarrollo de herramientas computacionales para el análisis estadístico y el aprendizaje automático con datos funcionales. En particular, hemos desarrollado dos bibliotecas de Python que facilitan la carga de conjuntos de datos, tanto multivariantes como funcionales. El paquete **scikit-datasets** permite al usuario descargar conjuntos de datos desde varios repositorios online ampliamente utilizados, como UCR, para datos funcionales, o UCI, para datos multivariantes. También proporciona funciones de utilidad para realizar estudios comparativos a gran escala usando estos datos y herramientas para resumir y analizar los resultados. La biblioteca **rdata** puede usarse para convertir conjuntos de datos desde el formato interno usado en el lenguaje de programación R a objetos de Python. Además de ser útil por sí misma, es también usada por el paquete **scikit-datasets**, con el objetivo de permitir la carga directa de conjuntos de datos disponibles en CRAN, el repositorio de paquetes de R.

El paquete de Python **scikit-fda** es la contribución práctica más prominente de esta tesis. Esta biblioteca proporciona un gran conjunto de herramientas computacionales para el análisis de datos funcionales. Éstas incluyen clases para representar las trayectorias en forma discretizada o como una expansión en una base, herramientas de preprocesamiento y análisis exploratorio, y modelos de aprendizaje automático para datos funcionales. La biblioteca se ha desarrollado de acuerdo a las especificaciones del ecosistema científico de Python. Además, ésta es compatible con la popular herramienta de aprendizaje automático **scikit-learn**. Como consecuencia, las herramientar proporcionadas por **scikit-fda** pueden emplearse en los flujos de trabajo y métodos de selección de hiperparámetros de **scikit-learn**.

La funcionalidad del paquete **scikit-fda** se ha ilustrado a través de varios ejemplos en aplicaciones reales. Además, hemos realizado un estudio comparativo de los métodos de clasificación de referencia. Este estudio no solo ilustra la aplicación de las herramientas desarrolladas (**scikit-datasets** and **scikit-fda**), sino que también proporciona intuición sobre las debilidades y fortalezas de los distintos métodos.

Finalmente, a raíz de la implementación de RMH, se ha desarrollado la biblioteca de Python **dcor**. Esta biblioteca implementa la covarianza de la distancia y la correlación de la distancia, dos medidas de dependencia entre vectores aleatorios (Székely, Rizzo y Bakirov, 2007). También se incluyen medidas relacionadas de homogeneidad y dependencia, y sus correspondientes tests de hipótesis. El paquete **dcor** está siendo usado en varios estudios científicos (Kharyuk y col., 2018; Laumann y col., 2022; Böhm, Berens y Kobak, 2022) y paquetes de software (John Richardson, 2022; Runge, 2022).

10.1 Trabajo futuro

Durante la elaboración de esta tesis hemos abordado muchos problemas de interés en el área de aprendizaje automático con datos funcionales. Parte del trabajo realizado se presta a realizar estudios en profundidad y nuevas extensiones. Además, esta tesis abre nuevas vías de investigación.

En particular, la exploración del algoritmo recursive maxima hunting (RMH) se ha ceñido problemas de clasificación binaria. No obstante, dicho algoritmo es en principio aplicable a problemas multiclase o incluso de regresión. Además, queda pendiente explorar una versión de dicho algoritmo aplicable a datos funcionales multivariantes o a datos funcionales cuyo parámetro continuo sea de mayores dimensiones, como superficies o volúmenes.

Otra dirección interesante es el análisis en profundidad de la relación entre problemas multivariantes y funcionales. Esto se ha realizado usando dos técnicas. Primero, en el Capítulo 4 consideramos el caso funcional en una representación discretizada. Cuando la rejilla de discretización era gruesa, podíamos analizar los datos como un vector multivariante, obteniendo una expresión para la regla de clasificación óptima. Estudiamos como cambiaba esta regla con rejillas más y más finas, obteniendo en el límite las expresiones para el caso funcional. Segundo, en el Capítulo 6 estudiamos la variación del parámetro de escala de longitud que caracteriza el descenso de las correlaciones en procesos gaussianos. Cuando la escala de longitud era pequeña, el comportamiento observado era similar al mostrado por datos multivariantes, mientras que para valores más altos se hacía aparente la naturaleza funcional de los datos. Creemos que las técnicas empleadas en esta tesis para estudiar la transición entre los mundos multivariante y funcional debería ser analizada en contextos adicionales en los que haya una discrepancia entre el comportamiento en el caso multivariante y el observado para datos funcionales. Por ejemplo, se podría analizar el comportamiento de la regresión logística en la transición al caso funcional, en el que el estimador de máxima verosimilitud no existe en general (Berrendero, Bueno-Larraz y Cuevas, 2022), así como otros métodos que dependan de cantidades mal definidas en el contexto funcional, como la moda.

Desde el punto de vista práctico, existen multitud de posibles caminos para ampliar las aportaciones de esta tesis. En particular, son varias las mejoras que pretendemos realizar sobre el paquete **scikit-fda**. En concreto, queremos incluir herramientas para trabajar con datos funcionales irregulares en los que los puntos de observación sean diferentes para cada observación funcional. Estos tipos de datos requieren técnicas especiales para su eficiente representación y tratamiento, especialmente cuando las mediciones son dispersas. También queremos extender las funcionalidades del paquete para soportar casos en los que cada observación conste de varias funciones y/o datos multivariantes. Además, nos gustaría mejorar la capacidad del paquete para manejar observaciones funcionales que dependan de varias variables, como imágenes o superficies.

Se debe considerar también la inclusión en el paquete de nuevas distancias funcionales y medidas de profundidad. Ejemplos de esto son la profundidad extrema, o ínfima (Mosler, 2013; Narisetty y Nair, 2016) y sus flexibles modificaciones (Nagy y col., 2021). También deberíamos definir distancias y profundidades para datos funcionales multivariantes. Se deben desarrollar e incluir en el paquete nuevos métodos de agrupamiento, clasificación y regresión, como máquinas de vectores de soporte (SVMs) (Rossi y Villa, 2006), modelos de regresión generalizados (Scheipl, Gertheiss y Greven, 2016), regresión con procesos gaussianos, o redes neuronales (Rossi y Conan-Guez, 2005; Kovachki y col., 2023). También sería de interés estudiar la posibilidad de desarollar modelos generativos para datos funcionales, como variational autoencoders, que puedan incorporarse al paquete.

La compatibilidad de **scikit-fda** con otros paquetes de software es también una de nuestras prioridades. La mayor parte de los avances en FDA realizados por la comunidad estadística acaban teniendo implementaciones en el lenguaje de programación R. Aunque intentemos ofrecer la mayor parte de dichas funcionalidades en **scikit-fda**, seguramente no se incluyan todas las herramientas disponibles, en especial aquellas que sean demasiado específicas. Por lo tanto, sería interesante explorar formas en las que poder combinar las herramientas para datos funcionales de R y Python de forma efectiva. En particular, sería útil integrar la biblioteca con las herramientas que permiten comunicación directa entre intérpretes de Python y R en el mismo proceso, como **rpy2**, ofreciendo funciones de conversión eficientes entre representaciones de datos funcionales en ambos lenguajes. Esto permitiría moverse entre estos lenguajes en distintos puntos del análisis y, por tanto, combinar las fortalezas de ambos.

Finalmente, durante el desarrollo de **scikit-fda** nos dimos cuenta de que las funciones tanto discretizadas como en bases aparecen con frecuencia en otras áreas. Estas incluyen ramas de las matemáticas y la física, como simulaciones de ecuaciones diferenciales o gráficos por ordenador. Aunque en algunos casos solo se requiere usar una función o un vector de funciones, en otros contextos, como el cálculo de las matrices jacobiana y hessiana, o derivadas de orden superior, se requiere el uso de matrices y tensores de funciones. Por tanto consideraremos crear un nuevo paquete de Python que pueda representar vectores, matrices o tensores cuyos elementos sean a su vez funciones generales, en forma discretizada o como expansión en una base. Este paquete debería incluir también funciones para operar con estos objetos, realizar álgebra y computar cantidades como las distancias funcionales. El paquete jugaría entonces un rol similar al que **NumPy** (Harris y col., 2020) tiene para datos multivariantes. Si se realiza un diseño cuidadoso, aprovechando la API estándar de arrays en Python (Consortium for Python Data API Standards, 2022), podremos usar también estas funciones en más contextos en el futuro, por ejemplo en frameworks de aprendizaje profundo como **PyTorch** (Paszke y col., 2019).

Appendices

Appendix A

Discrete monitoring

In the derivations carried out, a process X is monitored at a set of appropriately chosen discrete times $\{t_m\}_{m=1}^{M} \in \mathcal{T}^M$. Here we consider that these points are equispaced, and we denote as ΔT the distance between subsequent points. The integrals that appear (e.g., in the definitions of the inner products) are then approximated by Riemann sums

$$\int_{t \in \mathcal{T}} h(t)dt \approx \frac{1}{M} \sum_{m=1}^{M} h(t_m). \tag{A.1}$$

For functions that are continuous in $\mathcal{T}$, these Riemman sums converge to the corresponding definite integrals in the limit of dense monitoring

$$\lim_{M\to\infty} \frac{1}{M} \sum_{m=1}^{M} h(t_m) = \int_{t \in \mathcal{T}} h(t)dt \quad \forall h \in \mathcal{C}\left[\mathcal{T}\right], \tag{A.2}$$

where $\mathcal{C}\left[\mathcal{T}\right]$ is the space of continuous functions in $\mathcal{T}$.

Let k_0 and k_1 be symmetric, strictly positive kernels that are continuous in $\mathcal{T}$. Let the corresponding reproducing kernel Hilbert space (RKHS) be infinite dimensional. In the discretized representation, the kernel functions $\{k_i(s,t); s,t \in \mathcal{T}\}_{i=0}^{1}$ are approximated by $\mathbf{K}_i$, the corresponding $M \times M$ Gram matrices, whose elements are

$$(\mathbf{K}_i)_{mn} = k_i(t_n, t_m), \quad n, m = 1, 2, \ldots M, \tag{A.3}$$

for $i = 0, 1$. Let $\{\nu_{im} =\}_{m=1}^{M}$ be the (positive) eigenvalues of matrix $\mathbf{K}_i$. Theorem 3.4 of Baker, 1977 can be used to show that, in the limit of dense monitoring,

$$\lim_{M\to\infty} \frac{\nu_{im}}{\Delta T} = \lambda_{im}, \quad m = 1, 2, \ldots, M \tag{A.4}$$

where $\{\lambda_{i1} \geq \lambda_{i2} \geq \ldots \geq \lambda_{iM} > 0\}$ are the largest M eigenvalues of $\mathcal{K}_i$, the covariance operator associated to the kernel k_i.

Therefore, the spectrum of the Gram matrix $\mathbf{K}_i$ converges to the spectrum of the covariance operator $\mathcal{K}_i$. In particular, the ratio of the determinants of the Gram matrix

$$\begin{aligned} \lim_{M\to\infty} \frac{|\mathbf{K}_1|}{|\mathbf{K}_0|} &= \lim_{M\to\infty} \prod_{m=1}^{M} \frac{\nu_{1m}}{\nu_{0m}} \\ &= \lim_{M\to\infty} \prod_{m=1}^{M} \frac{\lambda_{1m}}{\lambda_{0m}} \equiv \frac{|\mathcal{K}_1|}{|\mathcal{K}_0|}, \end{aligned} \tag{A.5}$$

can be used to define the ratio $\frac{|\mathcal{K}_1|}{|\mathcal{K}_0|}$ when the corresponding Gaussian processes are equivalent ($\mathbb{P}_0 \sim \mathbb{P}_1$), in which case the limit exists (is finite) and is different from zero.

Appendix B

RMH proofs and results

B.1 Proofs of the theorems and propositions

In this section we group the proofs of the theorems and propositions that were proposed in Chapter 5.

B.1.1 Convexity of $\mathcal{V}^2$

Theorem 6 (Berrendero, Cuevas, and Torrecilla, 2016b)**.** *In the setting of the binary functional classification problems described in Equation (5.3) the squared distance covariance between a point of X and the class labels Y, $\mathcal{V}^2(X(t),Y)$, can be alternatively calculated as*

$$\mathcal{V}^2(X(t),Y) = 4p^2(1-p)^2\left[I_{01}(t) - \frac{I_{00}(t)+I_{11}(t)}{2}\right], \tag{B.1}$$

where $p = P(Y=1)$, $I_{ij}(t) = \mathbb{E}(|X(t)-X'(t)| \mid Y=i, Y'=j)$ and (X',Y') is an independent copy of (X,Y).

Corollary 1. *Under the model given by Equation (5.3), $\mathcal{V}^2(X(t),Y)$ has the following expression:*

$$\begin{split}\mathcal{V}^2(X(t),Y) = 4p^2(1-p)^2\Bigg[\frac{2\sigma(t)}{\sqrt{\pi}}\left(e^{-\frac{\mu(t)^2}{4\sigma(t)^2}}-1\right)+\\ \mu(t)\left(2\,\mathrm{cdf}\left(\frac{\mu(t)}{\sqrt{2}\sigma(t)}\right)-1\right)\Bigg],\end{split} \tag{B.2}$$

where $p = P(Y=1)$, cdf *is the cumulative distribution function of a standard normal random variable, $\mu(t) = \mu_1(t)-\mu_0(t)$ and $\sigma(t)$ is the standard deviation of the noise process $Z(t)$ at point t.*

Proof.
Consider the random variables given by the marginal distributions

$$\begin{aligned}X_0(t) &= X(t) \mid Y=0 \sim N(0,\sigma(t)^2),\\ X_1(t) &= X(t) \mid Y=1 \sim N(\mu(t),\sigma(t)^2),\end{aligned} \tag{B.3}$$

for any $t \in [0,1]$.

Let $X_0'(t)$ and $X_1'(t)$ be independent copies of $X_0(t)$ and $X_1(t)$ respectively. Then,

$$\begin{aligned}X_0(t)-X_0'(t) &\sim N(0,2\sigma(t)^2),\\ X_1(t)-X_1'(t) &\sim N(0,2\sigma(t)^2),\\ X_1(t)-X_0(t) &\sim N(\mu(t),2\sigma(t)^2).\end{aligned} \tag{B.4}$$

Let X be a Gaussian random variable $X \sim N(\mu, \sigma^2)$. Then,

$$\mathbb{E}|X| = \sigma\sqrt{\frac{2}{\pi}}e^{-\frac{\mu^2}{2\sigma^2}} + \mu\left(2\,\mathrm{cdf}\left(\frac{\mu}{\sigma}\right) - 1\right). \tag{B.5}$$

Therefore,

$$\begin{aligned} I_{00}(t) &= \mathbb{E}|X_0(t) - X_0'(t)| = \frac{2\sigma(t)}{\sqrt{\pi}}, \\ I_{11}(t) &= \mathbb{E}|X_1(t) - X_1'(t)| = \frac{2\sigma(t)}{\sqrt{\pi}}, \\ I_{01}(t) &= I_{10}(t) = \mathbb{E}|X_1(t) - X_0(t)| \\ &= \frac{2\sigma(t)}{\sqrt{\pi}}e^{-\frac{\mu(t)^2}{4\sigma(t)^2}} + \mu(t)\left(2\,\mathrm{cdf}\left(\frac{\mu(t)}{\sqrt{2}\sigma(t)}\right) - 1\right). \end{aligned} \tag{B.6}$$

Substituting these expressions in Equation (B.1) we obtain

$$\begin{aligned} \mathcal{V}^2(X(t), Y) = 4p^2(1-p)^2\Bigg[&\frac{2\sigma(t)}{\sqrt{\pi}}\left(e^{-\frac{\mu(t)^2}{4\sigma(t)^2}} - 1\right) + \\ &\mu(t)\left(2\,\mathrm{cdf}\left(\frac{\mu(t)}{\sqrt{2}\sigma(t)}\right) - 1\right)\Bigg]. \end{aligned} \tag{B.7}$$

□

Remark 1. *From Corollary 1, using the relation* $\frac{d}{dt}\mathrm{cdf}(t) = \mathrm{pdf}(t) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}t^2}$, *the formulas for the first and second derivatives of* $\mathcal{V}^2(X(t), Y)$ *are*

$$\begin{aligned} \frac{d}{dt}\mathcal{V}^2(X(t), Y) = 4p^2(1-p)^2\Bigg[&\frac{2\sigma'(t)}{\sqrt{\pi}}\left(e^{-\frac{\mu(t)^2}{4\sigma^2(t)}} - 1\right) + \\ &\mu'(t)\left(2\,\mathrm{cdf}\left(\frac{\mu(t)}{\sqrt{2}\sigma(t)}\right) - 1\right)\Bigg], \end{aligned} \tag{B.8}$$

and

$$\begin{aligned} \frac{d^2}{dt^2}\mathcal{V}^2(X(t), Y) = 4p^2(1-p)^2\Bigg[&\frac{2\sigma''(t)}{\sqrt{\pi}}\left(e^{-\frac{\mu(t)^2}{4\sigma^2(t)}} - 1\right) + \\ &\frac{1}{\sqrt{\pi}}e^{-\frac{\mu(t)^2}{4\sigma^2(t)}}\left(\frac{(\mu(t)\sigma'(t) - \sigma(t)\mu'(t))^2}{\sigma^3(t)}\right) + \\ &\mu''(t)\left(2\,\mathrm{cdf}\left(\frac{\mu(t)}{\sqrt{2}\sigma(t)}\right) - 1\right)\Bigg], \end{aligned} \tag{B.9}$$

provided that all the derivatives involved exist.

Lemma 1. *Under the model given by Equation (5.3), with functions defined in an open interval* (a, b), *let* σ *and* $|\mu|$ *be twice differentiable a.e. Let* $\sigma(t)$ *be concave* ($\sigma''(t) \leq 0$) *and* $|\mu(t)|$ *convex* ($(|\mu(t)|)'' \geq 0$) *at every point* $t \in (a, b)$ *in which they are twice differentiable. Then* $\mathcal{V}^2(X(t), Y)$ *is convex in* t. *Thus, a local maximum of* $\mathcal{V}^2(X(t), Y)$ *cannot be at a point* $t \in (a, b)$.

Proof.
The proof is based on the study of the sign of $\frac{d^2}{dt^2}\mathcal{V}^2(X(t),Y)$ in Equation (B.9). This sign is determined by a sum of three terms. The first term,

$$\frac{2\sigma''(t)}{\sqrt{\pi}}\left(e^{-\frac{\mu(t)^2}{4\sigma^2(t)}}-1\right), \tag{B.10}$$

is non-negative, given that $\sigma''(t) \leq 0$. The second term,

$$\frac{1}{\sqrt{\pi}}e^{-\frac{\mu(t)^2}{4\sigma^2(t)}}\left(\frac{(\mu(t)\sigma'(t)-\sigma(t)\mu'(t))^2}{\sigma^3(t)}\right), \tag{B.11}$$

is always positive since $\sigma(t)$ is positive. Finally, the third term

$$\mu''(t)\left(2\,\mathrm{cdf}\left(\frac{\mu(t)}{\sqrt{2}\sigma(t)}\right)-1\right), \tag{B.12}$$

is positive whenever $\mu''(t)$ and $\mu(t)$ have the same sign, as $\mathrm{cdf}(x) > 0.5$ if and only if $x > 0$. Since $(|\mu(t)|)'' \geq 0$, this is always the case:

$$\begin{aligned}\mu''(t) &= (\mu'(t))' = \left(\mathrm{sign}(\mu(t))(|\mu(t)|)'\right)' \\ &= \mathrm{sign}(\mu(t))(|\mu(t)|)''.\end{aligned} \tag{B.13}$$

Given that the three terms are positive, the second derivative of $\mathcal{V}^2(X(t),Y)$ is also positive. Thus, $\mathcal{V}^2(X(t),Y)$ is convex in all differentiable points t.

□

B.1.2 RMH properties

Theorem 1. *Consider the process*

$$X(t) = \begin{cases} Z(t) & \text{if } Y = 0, \\ \mu(t) + Z(t) & \text{if } Y = 1, \quad i \geq 1, \end{cases} \tag{5.10}$$

with μ a deterministic function, and Z a zero-mean Gaussian process whose covariance function is $k(s,t)$. The modified process

$$X^{[i]}(t) = X^{[i-1]}(t) - \mathbb{E}\left[Z^{[i-1]}(t) \mid Z^{[i-1]}(t_i) = X^{[i-1]}(t_i)\right], \quad i \geq 1 \tag{5.11}$$

with $X^{[0]} = X$ and $Z^{[0]} = Z$ is of the same form as the original one:

$$X^{[i]}(t) = \begin{cases} Z^{[i]}(t) & \text{if } Y = 0, \\ \mu^{[i]}(t) + Z^{[i]}(t) & \text{if } Y = 1, \quad i \geq 1, \end{cases} \tag{5.12}$$

with $\mu^{[i]}$ a deterministic function, and $Z^{[i]}$ a zero-mean Gaussian process.

Proof.
Equation (5.12) holds, by the definition of the problem, when $i = 0$.

Lets first check that the proposition works for the first iteration. Since Z is a Gaussian process, which are completely determined by their marginal distributions, we know that

$$\begin{aligned} X^{[1]}(t) =& X(t) - \mathbb{E}\left[Z(t) \mid Z(t_1) = X(t_1)\right] \\ =& X(t) - \frac{k(t,t_1)}{k(t_1,t_1)} X(t_1). \end{aligned} \tag{B.14}$$

Let $Y = 1$, then we have:

$$\begin{aligned} X^{[1]}(t) =& \mu(t) + Z(t) - \frac{k(t,t_1)}{k(t_1,t_1)} \left(\mu(t_1) + Z(t_1)\right) \\ =& \left(\mu(t) - \frac{k(t,t_1)}{k(t_1,t_1)}\mu(t_1)\right) + \left(Z(t) - \frac{k(t,t_1)}{k(t_1,t_1)} Z(t_1)\right) \\ =& \mu^{[1]}(t) + Z^{[1]}(t). \end{aligned} \tag{B.15}$$

The result for $Y = 0$ follows by substituting 0 for $\mu(\cdot)$.

As $Z^{[1]}(t)$ is a linear combination of Gaussian processes that are jointly Gaussian, it is itself a Gaussian process. Its mean is then

$$\begin{aligned} \mathbb{E}\left[Z^{[1]}(t)\right] =& \mathbb{E}\left[Z(t) - \mathbb{E}\left[Z(t) \mid Z(t_1)\right]\right] \\ =& \mathbb{E}\left[Z(t)\right] - \mathbb{E}\left[\mathbb{E}\left[Z(t) \mid Z(t_1)\right]\right] \\ =& \mathbb{E}\left[Z(t)\right] - \mathbb{E}\left[Z(t)\right] = 0. \end{aligned} \tag{B.16}$$

The second equality is a consequence of the linearity of the expectation. The third one is the law of total expectation.

Its covariance is

$$\begin{aligned} k^{[1]}(s,t) &= \mathbb{E}\left[Z^{[1]}(s) Z^{[1]}(t)\right] \\ &= \mathbb{E}\left[\left(X(s) - \frac{k(s,t_1)}{k(t_1,t_1)} X(t_1)\right)\left(X(t) - \frac{k(t,t_1)}{k(t_1,t_1)} X(t_1)\right)\right] \\ &= \mathbb{E}\left[X(s)X(t)\right] - \frac{k(t,t_1)}{k(t_1,t_1)} \mathbb{E}\left[X(s)X(t_1)\right] \\ &- \frac{k(s,t_1)}{k(t_1,t_1)} \mathbb{E}\left[X(t_1)X(t)\right] + \frac{k(s,t_1)}{k(t_1,t_1)} \frac{k(t,t_1)}{k(t_1,t_1)} \mathbb{E}\left[X(t_1)X(t_1)\right] \\ &= k(s,t) - \frac{k(t,t_1)k(s,t_1)}{k(t_1,t_1)}. \end{aligned} \tag{B.17}$$

The argument for $i > 1$ follows by induction. Note that the equality $Z^{[i]}(t) = \left[Z^{[i-1]}(t) \mid Z^{[i-1]}(t_i) = 0\right]$ holds for all i as both processes are Gaussian and their mean and covariance are the same. □

Theorem 2. *Under the conditions specified in Theorem 1, if*

$$\mu(t) = \sum_{d \geq 1} m_d k(t_d, t), \tag{5.22}$$

with $m_d > 0$ *and* $t_d \neq t_{d'}$ *for* $d \neq d'$*, and one applies the RMH corrections for* $\{t_1, \ldots, t_i\}$*, then*

$$\mu^{[i]}(t) = \sum_{d \geq i+1} m_d k^{[i]}(t_d, t). \tag{5.23}$$

Proof.

We can prove that for $i = 1$ and the rest follows by induction, due to the recursive form of the problem. We can separate the first term of the sum as

$$\mu(t) = m_1 k(t_1, t) + \sum_{d \geq 2} m_d k(t_d, t). \tag{B.18}$$

We have now

$$\begin{aligned}
\mu^{[1]}(t) &= \mu(t) - \frac{k(t,t_1)}{k(t_1,t_1)}\mu(t_1) \\
&= m_1 k(t_1,t) + \sum_{d\geq 2} m_d k(t_d,t) - \frac{k(t,t_1)}{k(t_1,t_1)}\left(m_1 k(t_1,t_1) + \sum_{d\geq 2} m_d k(t_d,t_1)\right) \\
&= \sum_{d\geq 2} m_d k(t_d,t) - \frac{k(t,t_1)}{k(t_1,t_1)}\sum_{d\geq 2} m_d k(t_d,t_1) \qquad \text{(B.19)} \\
&= \sum_{d\geq 2} m_d \left(k(t_d,t) - \frac{k(t,t_1)k(t_d,t_1)}{k(t_1,t_1)}\right) \\
&= \sum_{d\geq 2} m_d k^{[1]}(t_d,t).
\end{aligned}$$

□

Lemma 2 (Lamperti, 1977). *Let Z be a Gaussian Markov process with covariance function k. Then for all s, τ, t with $s < \tau < t$ it must verify:*

$$k(s,t) = \frac{k(s,\tau)k(\tau,t)}{k(\tau,\tau)}. \tag{B.20}$$

The converse is also true: if the covariance of a Gaussian process verifies the above then it is a Markov process.

Proposition 1. *Let Z be a zero-mean Gaussian process with the Markov property. The process $\tilde{Z}(t) = Z(t) - \mathbb{E}\left[Z(t) \mid Z(\tau)\right]$, with $s < \tau < t$ verifies that $\tilde{Z}(s)$ and $\tilde{Z}(t)$ are independent.*

Proof.
We denote as $\tilde{k}$ the covariance function of the process $\tilde{Z}$. We have

$$\tilde{k}(s,t) = k(s,t) - \frac{k(s,\tau)k(\tau,t)}{k(\tau,\tau)}. \tag{B.21}$$

From Lemma 2 we have:

$$\tilde{k}(s,t) = k(s,t) - \frac{k(s,\tau)k(\tau,t)}{k(\tau,\tau)} = k(s,t) - k(s,t) = 0. \tag{B.22}$$

As $\tilde{Z}(s)$ and $\tilde{Z}(t)$ are jointly Gaussian, their covariance equal to 0 implies their independence. □

Theorem 3. *Consider the classification problem defined by Equation (5.3) in the interval $\mathcal{T} = [0, \infty)$. Let Z be a zero-mean Brownian motion, whose covariance function is*

$$k_{BM}(s,t) = \sigma^2 \min(s,t). \tag{5.25}$$

Assume that μ is the piecewise linear function

$$\mu(t) = \begin{cases} \mu_1 \frac{t}{\tau_1} & \textit{if } t \in [0, \tau_1) \\ \mu_d \left(1 - \frac{t-\tau_d}{\tau_{d+1}-\tau_d}\right) + \mu_{d+1}\frac{t-\tau_d}{\tau_{d+1}-\tau_d} & \textit{if } t \in [\tau_d, \tau_{d+1}),\ d = 1, \ldots D-1 \\ \mu_D & \textit{if } t \geq \tau_D \end{cases} \tag{5.26}$$

with $0 < \tau_1 < \ldots < \tau_D$. *Then,*

(i) The class 1 mean belongs to the RKHS associated to the Brownian motion kernel and is of the form

$$\mu(t) = \sum_{d=1}^{D} m_d k_{BM}(\tau_d, t), \quad \textit{with } m_d \neq 0 \quad d = 1, \ldots, D. \tag{5.27}$$

(ii) RMH, using distance covariance as a relevance function, selects $\{\tau_1, \ldots, \tau_D\}$, *not necessarily in that order, and then halts.*

(iii) The points selected by RMH are the ones that appear in Bayes rule for the classification problem defined in (5.3) *and only those.*

Proof. (i) We can take

$$\begin{aligned}\begin{pmatrix} m_1 \\ \vdots \\ m_D \end{pmatrix} &= \begin{pmatrix} k_{BM}(\tau_1,\tau_1) & k_{BM}(\tau_1,\tau_2) & \ldots & k_{BM}(\tau_1,\tau_D) \\ k_{BM}(\tau_2,\tau_1) & k_{BM}(\tau_2,\tau_2) & \ldots & k_{BM}(\tau_2,\tau_D) \\ \vdots & \vdots & & \vdots \\ k_{BM}(\tau_D,\tau_1) & k_{BM}(\tau_D,\tau_2) & \ldots & k_{BM}(\tau_D,\tau_D) \end{pmatrix}^{-1} \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_D \end{pmatrix} \\ &= \begin{pmatrix} \tau_1 & \tau_1 & \ldots & \tau_1 \\ \tau_1 & \tau_2 & \ldots & \tau_2 \\ \vdots & \vdots & & \vdots \\ \tau_1 & \tau_2 & \ldots & \tau_D \end{pmatrix}^{-1} \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_D \end{pmatrix}. \end{aligned} \tag{B.23}$$

Thus,

$$\begin{aligned}\mu(t) = \sum_{d=1}^{D} m_d k_{BM}(\tau_d, t) &= \begin{pmatrix} m_1 & \ldots & m_D \end{pmatrix} \begin{pmatrix} k_{BM}(\tau_1, t) \\ \vdots \\ k_{BM}(\tau_D, t) \end{pmatrix} \\ &= (\mu_1 \ldots \mu_D) \begin{pmatrix} \tau_1 & \tau_1 & \ldots & \tau_1 \\ \tau_1 & \tau_2 & \ldots & \tau_2 \\ \vdots & \vdots & & \vdots \\ \tau_1 & \tau_2 & \ldots & \tau_D \end{pmatrix}^{-1} \begin{pmatrix} \min(\tau_1, t) \\ \vdots \\ \min(\tau_D, t) \end{pmatrix}. \end{aligned} \tag{B.24}$$

If $t \in [0, \tau_1)$ we have

$$\begin{aligned}\mu(t) &= \begin{pmatrix}\mu_1 & \dots & \mu_D\end{pmatrix} \begin{pmatrix}\tau_1 & \tau_1 & \dots & \tau_1 \\ \tau_1 & \tau_2 & \dots & \tau_2 \\ \vdots & \vdots & & \vdots \\ \tau_1 & \tau_2 & \dots & \tau_D\end{pmatrix}^{-1} \begin{pmatrix}t \\ \vdots \\ t\end{pmatrix} \\ &= \begin{pmatrix}\mu_1 & \dots & \mu_D\end{pmatrix} \begin{pmatrix}\tau_1 & \tau_1 & \dots & \tau_1 \\ \tau_1 & \tau_2 & \dots & \tau_2 \\ \vdots & \vdots & & \vdots \\ \tau_1 & \tau_2 & \dots & \tau_D\end{pmatrix}^{-1} \begin{pmatrix}\tau_1 \\ \vdots \\ \tau_1\end{pmatrix} \frac{t}{\tau_1} \\ &= \mu_1 \frac{t}{\tau_1}.\end{aligned} \tag{B.25}$$

If $t \in [\tau_d, \tau_{d+1})$ we have

$$\begin{aligned}\mu(t) &= \begin{pmatrix}\mu_1 & \dots & \mu_D\end{pmatrix} \begin{pmatrix}\tau_1 & \tau_1 & \dots & \tau_1 \\ \tau_1 & \tau_2 & \dots & \tau_2 \\ \vdots & \vdots & & \vdots \\ \tau_1 & \tau_2 & \dots & \tau_D\end{pmatrix}^{-1} \begin{pmatrix}\tau_1 \\ \vdots \\ \tau_d \\ t \\ \vdots \\ t\end{pmatrix} \\ &= \begin{pmatrix}\mu_1 & \dots & \mu_D\end{pmatrix} \begin{pmatrix}\tau_1 & \tau_1 & \dots & \tau_1 \\ \tau_1 & \tau_2 & \dots & \tau_2 \\ \vdots & \vdots & & \vdots \\ \tau_1 & \tau_2 & \dots & \tau_D\end{pmatrix}^{-1} \left[\begin{pmatrix}\tau_1 \\ \vdots \\ \tau_d \\ \tau_d \\ \vdots \\ \tau_d\end{pmatrix} + \frac{t-\tau_d}{\tau_{d+1}-\tau_d} \left(\begin{pmatrix}\tau_1 \\ \vdots \\ \tau_d \\ \tau_{d+1} \\ \vdots \\ \tau_{d+1}\end{pmatrix} - \begin{pmatrix}\tau_1 \\ \vdots \\ \tau_d \\ \tau_d \\ \vdots \\ \tau_d\end{pmatrix} \right) \right] \\ &= \mu_d + \frac{t-\tau_d}{\tau_{d+1}-\tau_d}(\mu_{d+1} - \mu_d) \\ &= \mu_d \left(1 - \frac{t-\tau_d}{\tau_{d+1}-\tau_d}\right) + \mu_{d+1} \frac{t-\tau_d}{\tau_{d+1}-\tau_d}.\end{aligned} \tag{B.26}$$

If $t \geq \tau_D$ we have

$$\begin{aligned}\mu(t) &= \begin{pmatrix}\mu_1 & \dots & \mu_D\end{pmatrix} \begin{pmatrix}\tau_1 & \tau_1 & \dots & \tau_1 \\ \tau_1 & \tau_2 & \dots & \tau_2 \\ \vdots & \vdots & & \vdots \\ \tau_1 & \tau_2 & \dots & \tau_D\end{pmatrix}^{-1} \begin{pmatrix}\tau_1 \\ \vdots \\ \tau_D\end{pmatrix} \\ &= \mu_D.\end{aligned} \tag{B.27}$$

(ii) The mean is piecewise linear, so the only non-differentiable points are $\tau_1, \dots, \tau_D$. As the mean is piecewise linear, in each point t in which the mean is differentiable we have $(|\mu(t)|)'' = 0$. In the first correction the process is Brownian motion. Thus, its standard deviation is $\sigma(t) = \sqrt{t}$, which has $\sigma''(t) = -\frac{1}{4t^{3/2}} \leq 0$. At $t = 0$ the process has a fixed value 0. The distance covariance at this point will then be 0, and thus this point cannot be a maximum. We can now apply Lemma 1 in the interval $(0, \infty)$ to deduce that the maximum of the distance covariance cannot be in a point in which the mean is differentiable. Thus, the first selected point should be one of $\tau_1, \dots, \tau_D$. By Theorem 2, we know that in the next step the mean has a similar shape, with all non-selected points still present, but a different corrected process.

We know that, for the Brownian motion, after the sucessive corrections the process still keeps a Brownian motion in the rightmost part and several Brownian bridges between the selected points. The standard deviation of a Brownian bridge between τ_d and τ_{d+1} is $\sigma(t) = \sqrt{\frac{(\tau_{d+1}-t)(t-\tau_d)}{\tau_{d+1}-\tau_d}}$, and in the same way as with the Brownian motion we have $\sigma''(t) \leq 0$. Thus, we can apply sequentially Lemma 1 and Theorem 2 to show that RMH necessarily selects the points $\tau_1, \dots, \tau_D$. After these points are selected, Theorem 2 indicates that RMH should halt.

(iii) Following Berrendero, Cuevas, and Torrecilla, 2018 we know that the Bayes rule in a binary homoscedastic problem following Equation (5.3) is

$$g^*(x) = \mathbb{I}_{\{\eta^*(x)>0\}}, \tag{B.28}$$

with

$$\eta^*(x) = \langle x, \mu \rangle_k - \frac{1}{2} \langle \mu, \mu \rangle_k - \log\left(\frac{1-p}{p}\right). \tag{B.29}$$

Here $\langle \cdot, \cdot \rangle_k$ is the inner product in the RKHS associated to the kernel of Z. Note also that the only term involving the trajectories is $\langle x, \mu \rangle_k$.

When the mean has the form $\mu(t) = \sum_{d=1}^{D} m_d k_{BM}(\tau_d, t)$, we can use the reproducing property to prove that the only relevant points are $\tau_1, \dots, \tau_D$, as follows

$$\begin{aligned} \langle x, \mu \rangle_k &= \left\langle x(t), \sum_{d=1}^{D} m_d k_{BM}(\tau_d, t) \right\rangle_k \\ &= \sum_{d=1}^{D} m_d \langle x(t), k_{BM}(\tau_d, t) \rangle_k \\ &= \sum_{d=1}^{D} m_d x(\tau_d). \end{aligned} \tag{B.30}$$

□

Lemma 3. *Let*

$$\mu(t) = \sum_{d=1}^{D} m_d k_{OU}(\tau_d, t), \quad d = 1, \dots, D \tag{B.31}$$

with $m_d \neq 0$ and $\tau_1 < \ldots < \tau_D$ and where k_{OU} is the covariance function of an Ornstein-Uhlenbeck process. Then $\mu(t)$ is only non-differentiable at points $\tau_1, \dots, \tau_D$. Everywhere else, $\mu(t)$ is infinitely differentiable and $|\mu(t)|$ is convex, that is, $\frac{d^2|\mu(t)|}{dt^2} > 0$.

Proof. As k_{OU} is the covariance function of an Ornstein-Uhlenbeck stochastic process, we have:

$$k_{OU}(s, t) = \sigma^2 e^{-\frac{|s-t|}{l}} \tag{B.32}$$

where σ^2 and l are positive constants. For $x \neq 0$, we know that $|x|$ is infinitely differentiable and $\frac{d|x|}{x} = \text{sign}(x)$ Thus, for a fixed t if $s \neq t$, $k_{OU}(s, t)$ is infinitely differentiable at point s, and its first derivative is

$$\frac{\partial k_{OU}(s, t)}{\partial s} = -\frac{\sigma^2}{l} e^{-\frac{|s-t|}{l}} \text{sign}(s - t). \tag{B.33}$$

If $s = t$, then $k_{OU}(s,t)$ is non differentiable at point s.

Now, we have that

$$\mu(t) = \sum_{d=1}^{D} m_d k_{OU}(\tau_d, t). \tag{B.34}$$

If $t \neq \tau_d\ \forall d$, then $\mu(t)$ is a sum of functions infinitely differentiable at point t and it is also infinitely differentiable at t. Otherwise, $\exists!\, d\ \ t = \tau_d$ (because all τ_d are different). Thus $\mu(t)$ is a sum of functions differentiable at point t and one function non-differentiable at point t. That makes $\mu(t)$ non-differentiable at point t.

Let us now prove the convexity of $|\mu(t)|$ on the differentiable points.

We can rewrite $\mu(t)$ as

$$\begin{aligned}\mu(t) &= \begin{pmatrix} k_{OU}(\tau_1,t) & \dots & k_{OU}(\tau_D,t)\end{pmatrix}\begin{pmatrix} m_1 \\ \vdots \\ m_D\end{pmatrix} \\ &= \begin{pmatrix} k_{OU}(\tau_1,t) \dots & k_{OU}(\tau_D,t)\end{pmatrix}\begin{pmatrix} k_{OU}(\tau_1,\tau_1) & \dots & k_{OU}(\tau_1,\tau_D) \\ k_{OU}(\tau_2,\tau_1) & \dots & k_{OU}(\tau_2,\tau_D) \\ \vdots & & \vdots \\ k_{OU}(\tau_D,\tau_1) & \dots & k_{OU}(\tau_D,\tau_D)\end{pmatrix}^{-1}\begin{pmatrix} \mu(\tau_1) \\ \vdots \\ \mu(\tau_D)\end{pmatrix}\end{aligned} \tag{B.35}$$

where we used the fact that for every d

$$\begin{pmatrix} k_{OU}(\tau_d,\tau_1) & \dots & k_{OU}(\tau_d,\tau_D)\end{pmatrix}\begin{pmatrix} m_1 \\ \vdots \\ m_D\end{pmatrix} = \mu(\tau_d). \tag{B.36}$$

This formula corresponds to the expectation of an Ornstein-Uhlenbeck process $Z(t)$ with covariance $k_{OU}(s,t)$ conditioned at the values $\tau_1, \dots, \tau_D$:

$$\mu(t) = \mathbb{E}[Z \mid Z(\tau_1) = \mu(\tau_1), \dots, Z(\tau_D) = \mu(\tau_D)]. \tag{B.37}$$

Now, because an Ornstein-Uhlenbeck process is a Markov process, we know that for a point $t \in (\tau_d, \tau_{d+1})$, the value of $\mu(t)$ only depends on $\mu(\tau_d)$ and $\mu(\tau_{d+1})$:

$$\mu(t) = \mathbb{E}[Z \mid Z(\tau_d) = \mu(\tau_d), Z(\tau_{d+1}) = \mu(\tau_{d+1})] \quad t \in (\tau_d, \tau_{d+1}). \tag{B.38}$$

In a similar way, if $t < \tau_1$ or $t > \tau_D$, the value of $\mu(t)$ only depends on $\mu(\tau_1)$ and $\mu(\tau_D)$, respectively. Thus, we only need to check the cases where the conditioning is done in one or two points.

We first consider the case where the conditioning is done in just one point t_d, with $d \in \{1, D\}$. We have

$$\mu(t) = \frac{\sigma^2 e^{-\frac{|t-\tau_d|}{l}} \mu(\tau_d)}{\sigma^2} = e^{-\frac{|t-\tau_d|}{l}} \mu(\tau_d). \tag{B.39}$$

Then, for any differentiable point,

$$\begin{aligned}\frac{d|\mu(t)|}{dt} &= e^{-\frac{|t-\tau_d|}{l}} \left(-\frac{\operatorname{sign}(t-\tau_d)}{l}\right) |\mu(\tau_d)| \\ \frac{d^2|\mu(t)|}{dt^2} &= e^{-\frac{|t-\tau_d|}{l}} \cdot \frac{|\mu(\tau_d)|}{l^2} > 0.\end{aligned} \tag{B.40}$$

In the second case we condition only to two points τ_d and τ_{d+1}. We have now

$$\begin{aligned}\mu(t) &= \sigma^2 \begin{pmatrix} e^{-\frac{|t-\tau_d|}{l}} & e^{-\frac{|t-\tau_{d+1}|}{l}} \end{pmatrix} \frac{1}{\sigma^2} \begin{pmatrix} 1 & e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} \\ e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} & 1 \end{pmatrix}^{-1} \begin{pmatrix} \mu(\tau_d) \\ \mu(\tau_{d+1}) \end{pmatrix} \\ &= \frac{1}{1-e^{-2\frac{|\tau_d-\tau_{d+1}|}{l}}} \begin{pmatrix} e^{-\frac{|t-\tau_d|}{l}} & e^{-\frac{|t-\tau_{d+1}|}{l}} \end{pmatrix} \begin{pmatrix} \mu(\tau_d) - \mu(\tau_{d+1})e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} \\ \mu(\tau_{d+1}) - \mu(\tau_d)e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} \end{pmatrix}.\end{aligned} \tag{B.41}$$

Then for any differentiable point

$$\begin{aligned}\frac{d|\mu(t)|}{dt} =& \frac{1}{1-e^{-2\frac{|\tau_d-\tau_{d+1}|}{l}}} \\ &\cdot \operatorname{sign}\left(\begin{pmatrix} e^{-\frac{|t-\tau_d|}{l}} & e^{-\frac{|t-\tau_{d+1}|}{l}} \end{pmatrix} \begin{pmatrix} \mu(\tau_d) - \mu(\tau_{d+1})e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} \\ \mu(\tau_{d+1}) - \mu(\tau_d)e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} \end{pmatrix} \right) \\ &\cdot \begin{pmatrix} e^{-\frac{|t-\tau_d|}{l}} \cdot \frac{-\operatorname{sign}(t-\tau_d)}{l} & e^{-\frac{|t-\tau_{d+1}|}{l}} \cdot \frac{-\operatorname{sign}(t-\tau_{d+1})}{l} \end{pmatrix} \\ &\cdot \begin{pmatrix} \mu(\tau_d) - \mu(\tau_{d+1})e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} \\ \mu(\tau_{d+1}) - \mu(\tau_d)e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} \end{pmatrix}.\end{aligned} \tag{B.42}$$

Finally

$$\begin{aligned}\frac{d^2|\mu(t)|}{dt^2} =& \frac{1}{l^2} \frac{1}{1-e^{-2\frac{|\tau_d-\tau_{d+1}|}{l}}} \\ &\cdot \left| \begin{pmatrix} e^{-\frac{|t-\tau_d|}{l}} & e^{-\frac{|t-\tau_{d+1}|}{l}} \end{pmatrix} \begin{pmatrix} \mu(\tau_d) - \mu(\tau_{d+1})e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} \\ \mu(\tau_{d+1}) - \mu(\tau_d)e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} \end{pmatrix} \right| > 0.\end{aligned} \tag{B.43}$$

Thus $\frac{d^2|\mu(t)|}{dt^2} > 0$ in for any differentiable point t, as we wanted to prove.

□

Lemma 4. *Let Z be an Ornstein-Uhlenbeck process, with covariance function*

$$k_{OU}(s,t) = \sigma^2 e^{-\frac{|s-t|}{l}}. \tag{B.44}$$

If Z_{cond} is this process conditioned to be 0 in an arbitrary number of points, with covariance function $k_{cond}(s,t)$, then $\sigma_{cond}(t) = \sqrt{k_{cond}(t,t)}$ is concave at every differentiable point t, that is, $\frac{d^2\sigma_{cond}(t)}{dt^2} \leq 0$.

Proof. Using the chain rule

$$\begin{aligned}\frac{d^2}{dt^2}\sigma_{cond}(t) = \frac{d^2}{dt^2}\sqrt{k_{cond}(t,t)} &= \frac{d}{dt}\left(\frac{1}{2}\frac{1}{\sqrt{k_{cond}(t,t)}}\frac{d}{dt}k_{cond}(t,t)\right) \\ &= \frac{1}{2}\Bigg(-\frac{1}{2}\frac{1}{\sqrt{(k_{cond}(t,t))^3}}\left(\frac{d}{dt}k_{cond}(t,t)\right)^2 \\ &\quad + \frac{1}{2}\frac{1}{\sqrt{k_{cond}(t,t)}}\frac{d^2}{ds^2}k_{cond}(t,t)\Bigg).\end{aligned} \tag{B.45}$$

It thus suffices to prove that $k_{cond}(t,t)$ itself is concave.

As the conditioned process is Markov, we can compute separately $\frac{d^2}{dt^2}k_{cond}(t,t)$ on each interval between conditioning points. Thus, we only have to prove this property for the cases with zero, one or two conditioning points.

If there are zero conditioning points, then

$$k_{cond}(t,t) = k(t,t) = \sigma^2. \tag{B.46}$$

Thus, $k_{cond}(t,t)$ is constant and $\frac{d^2}{dt^2}k_{cond}(t,t) = 0$.

If we have only one conditioning point τ_d then

$$\begin{aligned} k_{cond}(t,t) &= k(t,t) - k(t,\tau_d)k(\tau_d,\tau_d)^{-1}k(t,\tau_d) \\ &= \sigma^2\left(1 - e^{-2\frac{|t-\tau_d|}{l}}\right). \end{aligned} \tag{B.47}$$

Thus

$$\begin{aligned} \frac{d}{dt}k_{cond}(t,t) &= \sigma^2 \cdot \frac{2}{l} \cdot \operatorname{sign}(t-\tau_d) \cdot e^{-2\frac{|t-\tau_d|}{l}} \\ \frac{d^2}{dt^2}k_{cond}(t,t) &= -\sigma^2 \cdot \frac{4}{l^2} \cdot e^{-2\frac{|t-\tau_d|}{l}} < 0. \end{aligned} \tag{B.48}$$

If we have two conditioning points τ_d and τ_{d+1}, with $\tau_d < t < \tau_{d+1}$, then

$$\begin{aligned} k_{cond}(t,t) = k(t,t) - &\begin{pmatrix} k(t,\tau_d) & k(t,\tau_{d+1}) \end{pmatrix} \\ &\cdot \begin{pmatrix} k(\tau_d,\tau_d) & k(\tau_d,\tau_{d+1}) \\ k(\tau_d,\tau_{d+1}) & k(\tau_{d+1},\tau_{d+1}) \end{pmatrix}^{-1} \begin{pmatrix} k(t,\tau_d) \\ k(t,\tau_{d+1}) \end{pmatrix} \\ &= \sigma^2\left[1 - \frac{1}{1-e^{-2\frac{|\tau_d-\tau_{d+1}|}{l}}} \begin{pmatrix} e^{-\frac{|t-\tau_d|}{l}} & e^{-\frac{|t-\tau_{d+1}|}{l}} \end{pmatrix}\right. \\ &\left.\cdot \begin{pmatrix} 1 & -e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} \\ -e^{-\frac{|\tau_d-\tau_{d+1}|}{l}} & 1 \end{pmatrix} \begin{pmatrix} e^{-\frac{|t-\tau_d|}{l}} \\ e^{-\frac{|t-\tau_{d+1}|}{l}} \end{pmatrix}\right] \\ &= \sigma^2\left[1 - \frac{1}{1-e^{-2\frac{|\tau_d-\tau_{d+1}|}{l}}}\left[e^{-2\frac{|t-\tau_d|}{l}} - 2e^{-\frac{2|\tau_d-\tau_{d+1}|}{l}} + e^{-2\frac{|t-\tau_{d+1}|}{l}}\right]\right]. \end{aligned} \tag{B.49}$$

Then

$$\begin{aligned} \frac{d}{dt}k_{cond}(t,t) = -\sigma^2 &\left[\frac{1}{1-e^{-2\frac{|\tau_d-\tau_{d+1}|}{l}}}\right. \\ &\left.\cdot\left[-e^{-2\frac{|t-\tau_d|}{l}} \cdot \frac{2}{l} \cdot \operatorname{sign}(t-\tau_d) - e^{-2\frac{|t-\tau_{d+1}|}{l}} \cdot \frac{2}{l} \cdot \operatorname{sign}(t-\tau_{d+1})\right]\right] \end{aligned} \tag{B.50}$$

$$\frac{d^2}{dt^2}k_{cond}(t,t) = -\sigma^2\left[\frac{1}{1-e^{-2\frac{|\tau_d-\tau_{d+1}|}{l}}}\left[e^{-2\frac{|t-\tau_d|}{l}} \cdot \frac{4}{l^2} + e^{-2\frac{|t-\tau_{d+1}|}{l}} \cdot \frac{4}{l^2}\right]\right] < 0.$$

□

Theorem 4. *Consider the classification problem defined in Equation (5.3) defined in the whole real line $\mathcal{T} = \mathbb{R}$. Let Z be an Ornstein-Uhlenbeck process, whose covariance function is*

$$k_{OU}(s,t) = \sigma^2 \exp\left(-\frac{|t-s|}{l}\right). \tag{5.28}$$

Assume that μ is in the RKHS whose reproducing kernel is $k(s,t)$, and has the form

$$\mu(t) = \sum_{d=1}^{D} m_d k_{OU}(\tau_d, t), \quad m_d \neq 0 \quad d = 1, \ldots, D, \tag{5.29}$$

with $\tau_1 < \ldots < \tau_D$. Under these conditions, RMH using distance covariance as a relevance measure selects $\{\tau_d\}_{d=1}^{D}$, which are the points that appear in Bayes rule, and only those.

Proof. The proof follows the same schema as the one for Theorem 3.

From Lemma 3 we know that the only non-differentiable points of the mean are $\tau_1, \ldots, \tau_D$, and that it is convex everywhere else. We also know from Lemma 4 that in each iteration the standard deviation is concave at every differentiable point. Applying Lemma 1 to the corresponding open intervals between selected points we deduce that the maximum of the distance covariance at each step of the algorithm cannot be in a point in which the mean is differentiable. Thus, at each step the point selected should be one of $\tau_1, \ldots, \tau_D$.

By Theorem 2, we know that in the next step the mean has a similar shape, with all non-selected points still present, and where the process is a corrected version of Ornstein-Uhlenbeck. We can then continue applying the same reasoning to find that all $\tau_1, \ldots, \tau_D$ should be selected. After these points are selected, Theorem 2 indicates that RMH should halt. □

Theorem 5. *Let Z be a zero-mean Ornstein-Uhlenbeck process whose kernel is*

$$k_{OU}(s,t) = \sigma^2 \exp\left(-\frac{|t-s|}{l}\right). \tag{5.33}$$

In the limit $l \to \infty$, $\sigma^2 \to \infty$ with $\frac{2\sigma^2}{l} = 1$, The process $Z'(t) = [Z(t) \mid Z(\tau) = 0]$ is two-sided standard Brownian motion whose origin is τ.

Proof.
We will first consider this process before taking limits. As Z is stationary and with zero mean, we can assume without loss of generalization that $\tau = 0$. Then, Z' is a zero-mean Gaussian process whose covariance function is

$$\begin{aligned} k'(s,t) =& k(s,t) - \frac{k(s,\tau)k(\tau,t)}{k(\tau,\tau)} \\ =& \sigma^2 \left[\exp\left(-\frac{|s-t|}{l}\right) - \exp\left(-\frac{|s|+|t|}{l}\right)\right]. \end{aligned} \tag{B.51}$$

Let us take the limit of this process as $l \to \infty$ and $\sigma^2 \to \infty$ with $\sigma^2 = \frac{1}{2}l$, so that the variance tends to infinity as the lengthscale tends to infinity:

$$k'(s,t) = \lim_{l\to\infty} \frac{1}{2} l \left[\exp\left(-\frac{|s-t|}{l}\right) - \exp\left(-\frac{|s|+|t|}{l}\right)\right]. \tag{B.52}$$

Replacing the exponential functions by their Taylor series expansions, we get

$$\begin{aligned} k'(s,t) =& \lim_{l\to\infty} \frac{1}{2} l \left[1 - \frac{|s-t|}{l} + O\left(\frac{1}{l^2}\right) - \left(1 - \frac{|s|+|t|}{l} + O\left(\frac{1}{l^2}\right)\right)\right] \\ =& \frac{|s|+|t|-|s-t|}{2}. \end{aligned} \tag{B.53}$$

In this limit, if $s < \tau = 0 < t$ or $t < \tau = 0 < s$ then $k'(s,t) = 0$. Also, if $\tau < s, t$ or $s, t < \tau$ then:

$$k'(s,t) = \frac{|s| + |t| - |s - t|}{2} = \begin{cases} \min(|s|, |t|) & \text{if } st > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{B.54}$$

This is precisely the kernel of two-sided Brownian motion that emanates from $\tau = 0$. Therefore, upon conditioning to a particular observed value, in the limit of large l, the process is a standard Brownian motion that emanates from the point at which the process has been conditioned. The same reasoning can be applied assuming a different linear relation between lengthscale and variance, $\sigma^2 = Cl$, with C constant, to obtain a Brownian motion with rescaled covariance.

□

B.2 Simulations

The synthetic datasets used follow the simplified Brownian model in Equation (5.35),

$$\begin{cases} \mathbb{P}_0 : B(t) & , \quad t \in [0,1] \\ \mathbb{P}_1 : \mu(t) + B(t) & , \quad t \in [0,1] \end{cases}, \tag{B.55}$$

with $B(t)$ being a standard Brownian motion, $\mu(t)$ a deterministic trend and $\mathbb{P}(Y = 0) = \mathbb{P}(Y = 1) = 1/2$.

Following Baíllo, Cuevas, and Cuesta-Albertos, 2011; Berrendero, Cuevas, and Torrecilla, 2018, when the probability measures are equivalent, the Bayes classification rule for this model can be computed as

$$g^*(X) = \mathbb{I}_{\{\eta^*(X) > 0\}} \tag{B.56}$$

where

$$\eta^*(X) = \int \mu' dX - \frac{1}{2}\|\mu'\|^2 \tag{B.57}$$

and the Bayes error rate is

$$L^* = 1 - \Phi\left(\frac{\|\mu'\|}{2}\right) \tag{B.58}$$

where $\Phi(\cdot)$ is the cumulative distribution function of a standard normal random variable.

B.2.1 Peak functions

We first defined what we called a "peak" function, obtained as the integral of an element $\varphi_{m,k}(t)$ of a Haar basis:

$$\begin{aligned} \Phi_{m,k}(t) &= \int_0^t \varphi_{m,k}(s) ds \\ &= \int_0^t \sqrt{2^{m-1}} \left[\mathbb{I}_{\left(\frac{2k-2}{2^m}, \frac{2k-1}{2^m}\right)}(s) - \mathbb{I}_{\left(\frac{2k-1}{2^m}, \frac{2k}{2^m}\right)}(s) \right] ds, \end{aligned} \tag{B.59}$$

for $m, k \in \mathbb{N}, 1 \leq k \leq 2^{m-1}$.

These functions are relevant, because they are a orthonormal basis of the Dirichlet space (Mörters and Peres, 2010), that is, the reproducing kernel Hilbert space

(RKHS) associated with the Brownian covariance kernel. Let us remind that $\mathbb{P}_0 \sim \mathbb{P}_1 \iff \mu(t) \in \mathcal{H}_k$, as explained in Section 2.3.4. Thus, in any nonsingular problem (with positive error) $\mu(t)$ can be expressed has a linear combination of elements of the form $\Phi_{m,k}(t)$. Under the model in Equation (5.35) with $\mu(t) = \Phi_{m,k}(t)$, then

$$\eta^*(X) = 2X\left(\frac{2k-1}{2^m}\right) - X\left(\frac{2k-2}{2^m}\right) - X\left(\frac{2k}{2^m}\right) - \frac{1}{\sqrt{2^{m+1}}}. \tag{B.60}$$

Thus, there are only three relevant variables that appear in the optimal rule, and the associated Bayes error rate in this case is:

$$L^* = 1 - \Phi\left(\frac{\| \Phi'_{m,k}(t) \|}{2}\right) = 1 - \Phi\left(\frac{1}{2}\right) \simeq 0.3085. \tag{B.61}$$

However, if we compute $\mathcal{V}^2(X(t), Y)$ using Corollary 1, we obtain

$$\begin{aligned}\mathcal{V}^2(X(t), Y) = 2\sqrt{\frac{t}{\pi}}&\left(e^{-\frac{\Phi^2_{m,k}(t)}{4t}} - 1\right)\\ &+ \Phi_{m,k}(t)\left(2\,\mathrm{cdf}\left(\frac{\Phi_{m,k}(t)}{\sqrt{2t}}\right) - 1\right)\end{aligned} \tag{B.62}$$

which has only one maximum at $t = \frac{2k-1}{2^m}$. Thus maxima hunting would ignore two variables that are necessary for a correct classification, while RMH would be able to select these.

B.2.2 Bayes rule and error for the synthetic datasets

We now compute Bayes rule and error for the selected synthetic experiments. The mean in the first synthetic example is just a (scaled) peak function, $\mu(t) = 2\Phi_{3,3}(t)$, as defined above. Thus we have

$$\begin{aligned}\mu'(t) &= 2\varphi_{3,3}(t)\\ \|\mu'\|^2 &= \int_0^1 (2\varphi_{3,3}(t))^2 dt = 4\\ L^* &= 1 - \Phi\left(1\right) \simeq 0.1587\\ \eta^*(X) &= 2X\left(\frac{5}{8}\right) - X\left(\frac{1}{2}\right) - X\left(\frac{3}{4}\right) - \frac{1}{2},\end{aligned} \tag{B.63}$$

which again depends only on three variables.

The mean in the second example is a linear combination of the peak functions defined above, $\mu(t) = 2\Phi_{3,2}(t) + 3\Phi_{3,3}(t) - 2\Phi_{2,2}(t)$. We have now

$$\begin{aligned}
\mu'(t) &= 2\varphi_{3,2}(t) + 3\varphi_{3,3}(t) - 2\varphi_{2,2}(t) \\
\|\mu'\|^2 &= \int_0^1 (2\varphi_{3,2}(t) + 3\varphi_{3,3}(t) - 2\varphi_{2,2}(t))^2 dt = 17 \\
L^* &= 1 - \Phi\left(\frac{\sqrt{17}}{2}\right) \simeq 0.0196 \\
\eta^*(X) &= 4X\left(\frac{3}{8}\right) - 2X\left(\frac{1}{4}\right) - \left(5 - \sqrt{2}\right) X\left(\frac{1}{2}\right) \\
&\quad + 6X\left(\frac{5}{8}\right) - \left(3 + 2\sqrt{2}\right) X\left(\frac{3}{4}\right) + \sqrt{2}X(1) - \frac{17}{4},
\end{aligned} \tag{B.64}$$

which now depends on six variables.

The following examples are more complex functions. As they are not piecewise linear, Bayes rule will no longer depend on a finite number of variables. The first example is a (scaled) square function, $\mu(t) = 2t^2$. In this case

$$\begin{aligned}
\mu'(t) &= 4t \\
\|\mu'\|^2 &= \int_0^1 (4t)^2 dt = \frac{16}{3} \\
L^* &= 1 - \Phi\left(\frac{4}{2\sqrt{3}}\right) \simeq 0.1241 \\
\eta^*(X) &= X(1) - \int_0^1 X(t)dt - \frac{2}{3}.
\end{aligned} \tag{B.65}$$

The last synthetic example as the sinusoidal function $\mu(t) = \frac{1}{2}\sin 2\pi t$ as its mean. Thus we have

$$\begin{aligned}
\mu'(t) &= \pi \cos 2\pi t \\
\|\mu'\|^2 &= \pi^2 \int_0^1 \frac{1 + \cos 4\pi t}{2} dt = \frac{\pi^2}{2} + \frac{\pi}{8}\sin 4\pi = \frac{\pi^2}{2} \\
L^* &= 1 - \Phi\left(\frac{\pi}{2\sqrt{2}}\right) \simeq 0.1333 \\
\eta^*(X) &= X(1) + 2\int_0^1 \sin(2\pi t) X(t) dt - \frac{\pi}{4}.
\end{aligned} \tag{B.66}$$

Appendix C

Analysis of fuzzy C-means

In this section we present an analysis of the results obtained for fuzzy C-means (FCM) in the functional case of dataset D_2, as explained in Chapter 6. We explain mathematically the observed behavior in the limits $l \to \infty$ and $l \to 0$. In this last limit, which exhibits the same behavior as the multivariate case, we also find the lowest value of C for which the convergence to the center of mass is theoretically possible in the D_2 dataset, corresponding with normalized Gaussian data with one observation per cluster.

Consider the simplest case, in which there is a single observation per cluster. We also fix the fuzzifier parameter $\omega = 2$. The expected value of the center of gravity of the data is 0, due to the symmetrical generation of the clusters. In fact, as we are interested in the behavior for a large number of clusters, we can assume that it will be close to 0. Thus, for a fixed value of α, the i-th initial cluster center will then be placed at $q_i = \alpha c_i$. The inertia function $J(\mathcal{D})$ depends on the data only through the squared distances between the initial centers q_i and the data points (the real cluster centers c_i), as

$$J(\mathcal{D}) = \sum_{i=1}^{N} \sum_{j=1}^{C} U_{ij}^{\omega} d_{ij}^2 = \sum_{i=1}^{N} \frac{1}{\sum_{j=1}^{C} \frac{1}{d_{ij}^2}}. \tag{C.1}$$

We can compute these distances as

$$d_{ij}^2 = \langle c_i - q_j, c_i - q_j \rangle = \langle c_i, c_i \rangle - 2\alpha \langle c_i, c_j \rangle + \alpha^2 \langle c_j, c_j \rangle. \tag{C.2}$$

We consider the normalized case, where $\langle c_i, c_i \rangle = 1$. Thus

$$d_{ij}^2 = 1 - 2\alpha \langle c_i, c_j \rangle + \alpha^2. \tag{C.3}$$

If we substitute Equation (C.3) in Equation (C.1), we obtain

$$J(\mathcal{D}, \alpha) = \sum_{i=1}^{N} \frac{1}{\sum_{j=1}^{C} \frac{1}{1-2\alpha\langle c_i, c_j \rangle + \alpha^2}} = \sum_{i=1}^{N} \frac{1}{\frac{1}{1-2\alpha+\alpha^2} + \sum_{\substack{j=1 \\ j \neq i}}^{C} \frac{1}{1-2\alpha\langle c_i, c_j \rangle + \alpha^2}}. \tag{C.4}$$

Thus, the inertia depends only on the distribution of the distances of each q_i to the cluster centers. The mean of this distribution is independent of the covariance

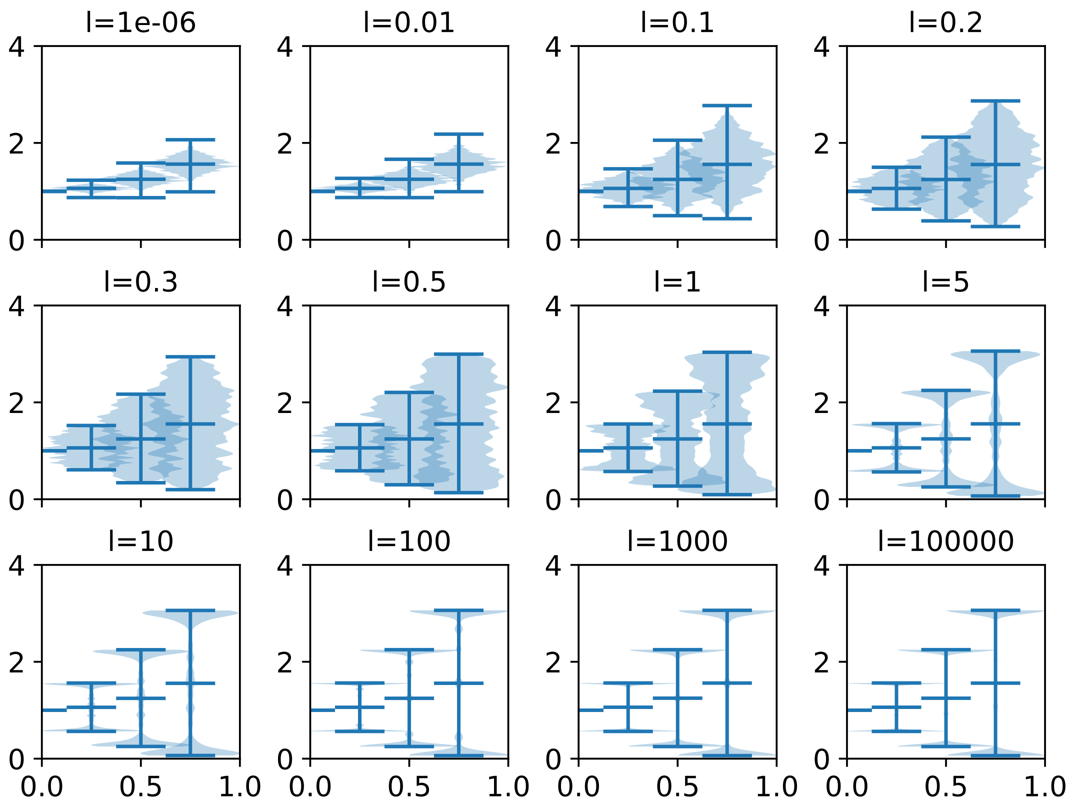


FIGURE C.1: Violin plots showing how the values of l and α affect the distribution of distances in the normalized dataset. The position of the vertical lines in the x axis correspond to each value of α. The y axis represent the possible values for the distance between two points in the sample. A vertical histogram of these distances is shown for each l and α. In these plots, $N = C = 50$ and $M = 100$.

function that generates the centers, as

$$\begin{aligned} \mathbb{E}_{j\neq i}[d_{ij}^2] &= \mathbb{E}[1 - 2\alpha\langle c_i, c_j\rangle + \alpha^2] \\ &= 1 - 2\alpha\mathbb{E}\left[\int_{\mathcal{T}} c_i(t)c_j(t)dt\right] + \alpha^2 \\ &= 1 - 2\alpha \int_{\mathcal{T}} \mathbb{E}[c_i(t)]\mathbb{E}[c_j(t)]dt + \alpha^2 \text{ (independence)} \\ &= 1 + \alpha^2. \end{aligned} \tag{C.5}$$

It is also easy to see that, for the normalized case, the squared distances are in the interval $[1 - 2\alpha + \alpha^2, 1 + 2\alpha + \alpha^2]$, symmetric around the mean. When $\alpha = 1$ this is the interval $[0, 4]$ as expected.

Is interesting to see how the distribution of distances in the dataset changes with respect to the parameters l and α. This can be seen in Figure C.1 through the use of violin plots. Each plot corresponds with different values of l. The distribution of distances is shown for four different values of α: 0, 0.25, 0.5 and 0.75. We can observe that by increasing the lengthscale the distribution of distances becomes bimodal with the probability masses strongly concentrated around the modes. This means that initially, the points are either very close to each other or very far apart.

There is a natural explanation of why increasing the lengthscale causes this bimodal distribution to appear. In fact, increasing the lengthscale increases the correlation of the different points inside a curve. As point evaluations are increasingly correlated, the data cloud (seen as points in M dimensions) becomes more ellipsoidal, with observations tending to lay near the line corresponding to constant trajectories ($x_i(t_1) = x_i(t_2) = \ldots = x_i(t_M)$). Thus, most of the observations in the cloud will be situated at the nonnegative orthant (the part of the space corresponding to functions that are always nonnegative) or at the nonpositive orthant (corresponding to nonpositive functions). When normalization is applied, all data points in the nonnegative orthant will be close together, and all the points in the nonpositive orthant will also be close together, but the two sets of points will be far away.

Indeed, as we mentioned in Section 2.1.1, when $l \to \infty$ the covariance function tends to a constant, and the trajectories are thus also constant. In this particular case, after normalization approximately half cluster centers will have the minimum distance $1 - 2\alpha + \alpha^2$ with a given q_i, and the other half will have the maximum distance $1 + 2\alpha + \alpha^2$. Thus the inertia will approximate

$$J(\mathcal{D}, \alpha) \approx \sum_{i=1}^{N} \frac{1}{\frac{C/2}{1-2\alpha+\alpha^2} + \frac{C/2}{1+2\alpha+\alpha^2}} = \frac{2(1-\alpha)^2(1+\alpha)^2}{((1-\alpha)^2 + (1+\alpha)^2)}. \tag{C.6}$$

We can see that, as the features are all the same, the effective dimension is one, and there is no dependence on C.

In order to explore the presence of the local maximum that causes the convergence problems, we can try to analyze the derivative of the inertia. We first compute the derivative of the squared distance and its inverse, as they will be needed:

$$\frac{d}{d\alpha} d_{ij}^2 = -2\langle c_i, c_j \rangle + 2\alpha, \tag{C.7}$$

$$\frac{d}{d\alpha} \frac{1}{d_{ij}^2} = \frac{2\langle c_i, c_j \rangle - 2\alpha}{d_{ij}^4}. \tag{C.8}$$

Using those, we can compute the derivative of the objective function as

$$\begin{aligned}\frac{d}{d\alpha} J(\mathcal{D}, \alpha) &= \sum_{i=1}^{N} \left(\frac{2}{\left(\sum_{j=1}^{C} \frac{1}{d_{ij}^2}\right)^2} \sum_{j=1}^{C} \frac{-\langle c_i, c_j \rangle + \alpha}{d_{ij}^4} \right) \\ &= \sum_{i=1}^{N} \left(\frac{2}{\left(\sum_{j=1}^{C} \frac{1}{d_{ij}^2}\right)^2} \left(\frac{\alpha - 1}{d_{ii}^4} + \sum_{\substack{j=1 \\ j \neq i}}^{C} \frac{\alpha - \langle c_i, c_j \rangle}{d_{ij}^4} \right) \right).\end{aligned} \tag{C.9}$$

In order for the maximum to appear, the derivative must be positive in an interval. This is guaranteed if

$$0 \leq \frac{\alpha - 1}{d_{ii}^4} + \sum_{\substack{j=1 \\ j \neq i}}^{C} \frac{\alpha - \langle c_i, c_j \rangle}{d_{ij}^4}. \tag{C.10}$$

The term $\frac{\alpha-1}{d_{ii}^4}$ is always negative, as $\alpha < 1$. For small values of C, this term dominates in the expression and thus the derivative is negative and FCM converges. When C is bigger, however, the term is very small, and the behavior of FCM will depend on the other term. We can compute which values of C are too small for the problems to arise. In order to do that, we can check the value of C for which the inequality holds:

$$\begin{aligned} \frac{1-\alpha}{d_{ii}^4} &\leq \sum_{\substack{j=1 \\ j\neq i}}^{C} \frac{\alpha - \langle c_i, c_j\rangle}{d_{ij}^4} \\ \frac{1}{(1-\alpha)^3} &\leq \sum_{\substack{j=1 \\ j\neq i}}^{C} \frac{\alpha - \langle c_i, c_j\rangle}{(1-2\alpha\langle c_i, c_j\rangle + \alpha^2)^2}. \end{aligned} \tag{C.11}$$

As the functions are discretized to M points, the inner product can be expressed as the Riemann sum

$$\langle c_i, c_j\rangle = \frac{1}{M}\sum_{m=1}^{M} c_{im}c_{jm}, \tag{C.12}$$

where c_{im} is the m-th feature of c_i. We consider the worst case, namely that $l \to 0$ and $M \to \infty$. Then the features of c_i are independent (because $l \to 0$) and thus

$$\langle c_i, c_j\rangle = \frac{1}{M}\sum_{m=1}^{M} c_{im}c_{jm} \xrightarrow[M\to\infty]{} \mathbb{E}(c_{im}c_{jm}) = \mathbb{E}c_{im}\mathbb{E}c_{jm} = 0. \tag{C.13}$$

Replacing that in the expression from Equation (C.11) and performing algebraic manipulations we arrive at

$$C \geq 1 + \frac{(1+\alpha^2)^2}{\alpha(1-\alpha)^3}. \tag{C.14}$$

The term in the right is greater than 11 for $\alpha \in (0,1)$, as its minimum value in that region is $\frac{1}{27}(163+56\sqrt{7}) \approx 11.52$. Thus, we need $C \geq 12$ for the maximum to appear. This has been observed in practice for high values of M and small enough l.

When $l \gg 0$ or $M \ll \infty$, the term $\langle c_i, c_j\rangle$ does not longer cancel out. However, we know that $-1 \leq \langle c_i, c_j\rangle \leq 1$. Due to the symmetry of the data we would expect that $\langle c_i, c_j\rangle$ presents roughly the same distribution of negative and positive values. However, the effect of its sign in Equation (C.11) is not the same. For example, when $\langle c_i, c_j\rangle > \alpha$ the numerator $\alpha - \langle c_i, c_j\rangle$ becomes negative, and the denominator $(1-2\alpha\langle c_i, c_j\rangle + \alpha^2)^2$ becomes smaller. When $\langle c_i, c_j\rangle < 0$ the numerator is positive, but the denominator is larger. Thus, the negative terms in the sum carry more weight than the positive ones, making the ocurrence of a local maximum more difficult.

In the extreme case, with $l \to \infty$, approximately half of the terms in the sum will have a value of $\langle c_i, c_j\rangle = 1$ and the other half will have $\langle c_i, c_j\rangle = -1$. In that case

$$\begin{aligned} \frac{C}{2}\frac{1}{(1-\alpha)^3} &\leq \frac{C}{2}\frac{\alpha+1}{(1+2\alpha+\alpha^2)^2} \\ 0 &\geq C. \end{aligned} \tag{C.15}$$

Thus, the inequality never holds, as we already expected from Equation (C.6). In this case the derivative of the inertia function is always negative and FCM always converges, independently of C and M. Between these extreme cases, we obtain different behaviors for the different values of l, requiring increasingly more clusters for the two minima to appear as l increases.

Bibliography

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng (2015). *TensorFlow, Large-scale Machine Learning on Heterogeneous Systems*. DOI: 10.5281/zenodo.4724125.

Adelson-Velskii, G. M. and E. M. Landis (1962). "An algorithm for organization of information". In: *Proceedings of the USSR Academy of Sciences* 146.2, pp. 263–266.

Aguilera-Morillo, M. C., I. Buño, R. E. Lillo, and J. Romo (2020). "Variable Selection with P-splines in Functional Linear Regression: Application in Graft-versus-Host Disease". In: *Biometrical Journal* 62.7, pp. 1670–1686. ISSN: 1521-4036. DOI: 10.1002/bimj.201900189.

Alcalá-Fdez, J., A. Fernández, J. Luengo, J. Derrac, S García, L. Sanchez, and F. Herrera (2010). "KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework". In: *Journal of Multiple-Valued Logic and Soft Computing* 17, pp. 255–287.

Aneiros, G. and P. Vieu (2016). "Sparse Nonparametric Model for Regression with Functional Covariate". In: *Journal of Nonparametric Statistics* 28.4, pp. 839–859. ISSN: 1048-5252. DOI: 10.1080/10485252.2016.1234050.

Aneiros, G., S. Novo, and P. Vieu (2022). "Variable Selection in Functional Regression Models: A Review". In: *Journal of Multivariate Analysis*. 50th Anniversary Jubilee Edition 188, p. 104871. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2021.104871.

Aneiros, G. and P. Vieu (2014). "Variable Selection in Infinite-Dimensional Problems". In: *Statistics & Probability Letters* 94, pp. 12–20. ISSN: 0167-7152. DOI: 10.1016/j.spl.2014.06.025.

Arribas-Gil, A. and J. Romo (2014). "Shape Outlier Detection and Visualization for Functional Data: The Outliergram". In: *Biostatistics* 15.4, pp. 603–619. ISSN: 1465-4644. DOI: 10.1093/biostatistics/kxu006.

Assunção, R. V., A. C. Silva, A. Roy, B. Bourlès, C. H. S. Silva, J.-F. Ternon, M. Araujo, and A. Bertrand (2020). "3D Characterisation of the Thermohaline Structure in the Southwestern Tropical Atlantic Derived from Functional Data Analysis of in Situ Profiles". In: *Progress in Oceanography* 187, p. 102399. ISSN: 0079-6611. DOI: 10.1016/j.pocean.2020.102399.

Bagnall, A., H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh (2018). *The UEA Multivariate Time Series Classification Archive, 2018*. DOI: 10.48550/arXiv.1811.00075.

Bagnall, A., J. Lines, A. Bostrom, J. Large, and E. Keogh (2017). "The Great Time Series Classification Bake off: A Review and Experimental Evaluation of Recent Algorithmic Advances". In: *Data Mining and Knowledge Discovery* 31.3, pp. 606–660. ISSN: 1573-756X. DOI: 10.1007/s10618-016-0483-9.

Bagnall, A., J. Lines, W. Vickers, and E. Keogh (2021). *The UEA & UCR Time Series Classification Repository*. URL: www.timeseriesclassification.com.

Baíllo, A. and A. Cuevas (2008). “Supervised Classification for Functional Data: A Theoretical Remark and Some Numerical Comparisons”. In: *Functional and Operatorial Statistics*. Contributions to Statistics. Heidelberg: Physica-Verlag HD, pp. 43–46. ISBN: 978-3-7908-2062-1. DOI: 10.1007/978-3-7908-2062-1_7.

Baíllo, A., A. Cuevas, and J. A. Cuesta-Albertos (2011). “Supervised Classification for a Family of Gaussian Functional Models”. In: *Scandinavian Journal of Statistics* 38.3, pp. 480–498. ISSN: 1467-9469. DOI: 10.1111/j.1467-9469.2011.00734.x.

Baker, C. T. H. (1977). *The Numerical Treatment of Integral Equations*. Clarendon Press. ISBN: 978-0-19-853406-8.

Bakirov, N. K., M. L. Rizzo, and G. J. Székely (2006). “A Multivariate Nonparametric Test of Independence”. In: *Journal of Multivariate Analysis* 97.8, pp. 1742–1756. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2005.10.005.

Bauer, A., F. Scheipl, H. Küchenhoff, and A.-A. Gabriel (2018). “An Introduction to Semiparametric Function-on-Scalar Regression”. In: *Statistical Modelling* 18.3-4, pp. 346–364. ISSN: 1471-082X. DOI: 10.1177/1471082X17748034.

Baíllo, A., A. Cuevas, and R. Fraiman (2010). “Classification Methods for Functional Data”. In: *The Oxford Handbook of Functional Data Analysis*. DOI: 10.1093/oxfordhb/9780199568444.013.10.

Belli, E. (2022). “Smoothly Adaptively Centered Ridge Estimator”. In: *Journal of Multivariate Analysis* 189, p. 104882. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2021.104882.

Benowitz, M. (2020). *Hedgecraft: A Portfolio Management Algorithm for the 21st Century.* URL: https://github.com/mayabenowitz/Hedgecraft (visited on 05/31/2022).

Berlinet, A. and C. Thomas-Agnan (2004). *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Boston, MA: Springer US. ISBN: 978-1-4419-9096-9. DOI: 10.1007/978-1-4419-9096-9_1.

Bernard, G., C. Lejeune, S. Ferrieres, and O. Teste (2021). *Curve Shape Analysis*. GitHub. URL: https://github.com/Clej/curve_shape_analysis (visited on 05/22/2023).

Berrendero, J. R., B. Bueno-Larraz, and A. Cuevas (2020). “On Mahalanobis Distance in Functional Settings”. In: *Journal of Machine Learning Research* 21.9, pp. 1–33. ISSN: 1533-7928. URL: http://jmlr.org/papers/v21/18-156.html (visited on 06/29/2022).

Berrendero, J. R., B. Bueno-Larraz, and A. Cuevas (2022). “On Functional Logistic Regression: Some Conceptual Issues”. In: *TEST*. ISSN: 1863-8260. DOI: 10.1007/s11749-022-00836-9.

Berrendero, J. R. and J. Cárcamo (2019). “Linear Components of Quadratic Classifiers”. In: *Advances in Data Analysis and Classification* 13.2, pp. 347–377. ISSN: 1862-5355. DOI: 10.1007/s11634-018-0321-6.

Berrendero, J. R., B. Bueno-Larraz, and A. Cuevas (2019). “An RKHS Model for Variable Selection in Functional Linear Regression”. In: *Journal of Multivariate Analysis*. Special Issue on Functional Data Analysis and Related Topics 170, pp. 25–45. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2018.04.008.

Berrendero, J. R., A. Cuevas, and J. L. Torrecilla (2016a). “The mRMR Variable Selection Method: A Comparative Study for Functional Data”. In: *Journal of Statistical Computation and Simulation* 86.5, pp. 891–907. ISSN: 0094-9655. DOI: 10.1080/00949655.2015.1042378.

Berrendero, J. R., A. Cuevas, and J. L. Torrecilla (2016b). “Variable Selection in Functional Data Classification: A Maxima-Hunting Proposal”. In: *Statistica Sinica* 26.2, pp. 619–638. ISSN: 10170405. DOI: 10.5705/ss.202014.0014.

Berrendero, J. R., A. Cuevas, and J. L. Torrecilla (2018). "On the Use of Reproducing Kernel Hilbert Spaces in Functional Classification". In: *Journal of the American Statistical Association* 113.523, pp. 1210–1218. ISSN: 0162-1459. DOI: 10.1080/01621459.2017.1320287.

Beyaztas, U. and Z. M. Yaseen (2019). "Drought Interval Simulation Using Functional Data Analysis". In: *Journal of Hydrology* 579, p. 124141. ISSN: 0022-1694. DOI: 10.1016/j.jhydrol.2019.124141.

Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Boston, MA: Springer US. ISBN: 978-1-4757-0452-5 978-1-4757-0450-1. DOI: 10.1007/978-1-4757-0450-1.

Biau, G., F. Bunea, and M. Wegkamp (2005). "Functional Classification in Hilbert Spaces". In: *IEEE Transactions on Information Theory* 51.6, pp. 2163–2172. ISSN: 1557-9654. DOI: 10.1109/TIT.2005.847705.

Bischl, B., M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, and Z. M. Jones (2016). "mlr: Machine Learning in R". In: *Journal of Machine Learning Research* 17.170, pp. 1–5. ISSN: 1533-7928. URL: http://jmlr.org/papers/v17/15-066.html (visited on 07/20/2022).

Blanquero, R., E. Carrizosa, A. Jiménez-Cordero, and B. Martín-Barragán (2019). "Variable Selection in Classification for Multivariate Functional Data". In: *Information Sciences* 481, pp. 445–462. ISSN: 0020-0255. DOI: 10.1016/j.ins.2018.12.060.

Blanquero, R., E. Carrizosa, A. Jiménez-Cordero, and B. Martín-Barragán (2020). "Selection of Time Instants and Intervals with Support Vector Regression for Multivariate Functional Data". In: *Computers & Operations Research* 123, p. 105050. ISSN: 0305-0548. DOI: 10.1016/j.cor.2020.105050.

Böhm, J. N., P. Berens, and D. Kobak (2022). "Attraction-Repulsion Spectrum in Neighbor Embeddings". In: *Journal of Machine Learning Research* 23.95, pp. 1–32. ISSN: 1533-7928.

Bollerslev, T., R. Y. Chou, and K. F. Kroner (1992). "ARCH Modeling in Finance: A Review of the Theory and Empirical Evidence". In: *Journal of Econometrics* 52.1, pp. 5–59. ISSN: 0304-4076. DOI: 10.1016/0304-4076(92)90064-X.

Borggaard, C. and H. H. Thodberg (1992). "Optimal Minimal Neural Interpretation of Spectra". In: *Analytical Chemistry* 64.5, pp. 545–551. ISSN: 0003-2700. DOI: 10.1021/ac00029a018.

Boschi, T., J. Di Iorio, L. Testa, M. A. Cremona, and F. Chiaromonte (2021). "Functional Data Analysis Characterizes the Shapes of the First COVID-19 Epidemic Wave in Italy". In: *Scientific Reports* 11.1, p. 17054. ISSN: 2045-2322. DOI: 10.1038/s41598-021-95866-y.

Bouveyron, C., E. Côme, and J. Jacques (2015). "The Discriminative Functional Mixture Model for a Comparative Analysis of Bike Sharing Systems". In: *Annals of Applied Statistics* 9.4, pp. 1726–1760. ISSN: 1932-6157, 1941-7330. DOI: 10.1214/15-AOAS861.

Breiman, L. (2001). "Random Forests". In: *Machine Learning* 45.1, 5–32. ISSN: 0885-6125.

Brockhaus, S., D. Rügamer, and S. Greven (2020). "Boosting Functional Regression Models with FDboost". In: *Journal of Statistical Software* 94, pp. 1–50. ISSN: 1548-7660. DOI: 10.18637/jss.v094.i10.

Buitinck, L., G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux (2013). "API Design for Machine Learning Software: Experiences from the scikit-learn Project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122.

Burba, F., F. Ferraty, and P. Vieu (2009). "K-Nearest Neighbour Method in Functional Nonparametric Regression". In: *Journal of Nonparametric Statistics* 21.4, pp. 453–469. ISSN: 1048-5252. DOI: 10.1080/10485250802668909.

Bureau of Meteorology, A. (1992). *Long-Term Historical Rainfall Data for Australia*. URL: http://rda.ucar.edu/datasets/ds482.1/.

Carey, J. R., P. Liedo, H. G. Müller, J. L. Wang, and J. M. Chiou (1998). "Relationship of Age Patterns of Fecundity to Mortality, Longevity, and Lifetime Reproduction in a Large Cohort of Mediterranean Fruit Fly Females". In: *The Journals of Gerontology. Series A, Biological Sciences and Medical Sciences* 53.4, B245–251. ISSN: 1079-5006.

Carroll, C., A. Gajardo, Y. Chen, X. Dai, J. Fan, P. Z. Hadjipantelis, K. Han, H. Ji, H.-G. Mueller, and J.-L. Wang (2020). *fdapace: Functional Data Analysis and Empirical Dynamics*. Manual. R package version 0.5.4. URL: https://CRAN.R-project.org/package=fdapace.

Cérou, F. and A. Guyader (2006). "Nearest Neighbor Classification in Infinite Dimension". In: *ESAIM: Probability and Statistics* 10, pp. 340–355. ISSN: 1292-8100, 1262-3318. DOI: 10.1051/ps:2006014.

Chakraborty, A. and V. M. Panaretos (2021). "Functional Registration and Local Variations: Identifiability, Rank, and Tuning". In: *Bernoulli* 27.2, pp. 1103–1130. ISSN: 1350-7265. DOI: 10.3150/20-BEJ1267.

Chang, C.-C. and C.-J. Lin (2011). "LIBSVM: A Library for Support Vector Machines". In: *ACM Transactions on Intelligent Systems and Technology* 2.3, 27:1–27:27. ISSN: 2157-6904. DOI: 10.1145/1961189.1961199.

Chaudhuri, A. and W. Hu (2019). "A Fast Algorithm for Computing Distance Correlation". In: *Computational Statistics Data Analysis* 135, pp. 15–24. ISSN: 0167-9473. DOI: 10.1016/j.csda.2019.01.016.

Chen, D., M. A. Cremona, Z. Qi, R. D. Mitra, F. Chiaromonte, and K. D. Makova (2020). "Human L1 Transposition Dynamics Unraveled with Functional Data Analysis". In: *Molecular Biology and Evolution* 37.12, pp. 3576–3600. ISSN: 0737-4038. DOI: 10.1093/molbev/msaa194.

Chen, J. and J. Revels (2016). "Robust Benchmarking in Noisy Environments". In: *Proceedings of the 20th Annual IEEE High Performance Extreme Computing Conference*.

Chollet, F. et al. (2015). *Keras*. URL: https://keras.io.

Consagra, W., A. Venkataraman, and X. Qiu (2022). *Efficient Multidimensional Functional Data Analysis Using Marginal Product Basis Systems*. DOI: 10.48550/arXiv.2107.14728.

Consortium for Python Data API Standards (2022). *Python Array API Standard*. URL: https://data-apis.org/array-api (visited on 05/27/2022).

Cont, R. (2001). "Empirical Properties of Asset Returns: Stylized Facts and Statistical Issues". In: *Quantitative Finance* 1.2, pp. 223–236. ISSN: 1469-7688. DOI: 10.1080/713665670.

Crawford, L., A. Monod, A. X. Chen, S. Mukherjee, and R. Rabadán (2020). "Predicting Clinical Outcomes in Glioblastoma: An Application of Topological and Functional Data Analysis". In: *Journal of the American Statistical Association* 115.531, pp. 1139–1150. ISSN: 0162-1459. DOI: 10.1080/01621459.2019.1671198.

Cremona, M. A., H. Xu, K. D. Makova, M. Reimherr, F. Chiaromonte, and P. Madrigal (2019). "Functional Data Analysis for Computational Biology". In: *Bioinformatics* 35.17, pp. 3211–3213. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz045.

Cucker, F. and S. Smale (2002). "On the Mathematical Foundations of Learning". In: *Bulletin of the American Mathematical Society* 39.1, pp. 1–49. ISSN: 0273-0979, 1088-9485. DOI: 10.1090/S0273-0979-01-00923-5.

Cucker, F. and D. X. Zhou (2007). *Learning Theory: An Approximation Theory Viewpoint*. Cambridge University Press. ISBN: 978-1-139-46286-0.

Cuesta-Albertos, J. A., M. Febrero-Bande, and M. Oviedo de la Fuente (2017). "The DD^G-Classifier in the Functional Setting". In: *TEST* 26.1, pp. 119–142. ISSN: 1863-8260. DOI: 10.1007/s11749-016-0502-6.

Cuesta-Albertos, J. A. and S. Dutta (2022). *On Perfect Classification and Clustering for Gaussian Processes*. DOI: 10.48550/arXiv.1602.04941.

Cuesta-Albertos, J. A. and A. Nieto-Reyes (2010). "Functional Classification and the Random Tukey Depth. Practical Issues". In: *Combining Soft Computing and Statistical Methods in Data Analysis*. Ed. by C. Borgelt, G. González-Rodríguez, W. Trutschnig, M. A. Lubiano, M. Á. Gil, P. Grzegorzewski, and O. Hryniewicz. Advances in Intelligent and Soft Computing. Springer Berlin Heidelberg, pp. 123–130. ISBN: 978-3-642-14746-3.

Cuevas, A. (2014). "A Partial Overview of the Theory of Statistics with Functional Data". In: *Journal of Statistical Planning and Inference* 147, pp. 1–23. ISSN: 0378-3758. DOI: 10.1016/j.jspi.2013.04.002.

Cuevas, A., M. Febrero, and R. Fraiman (2004). "An ANOVA Test for Functional Data". In: *Computational Statistics & Data Analysis* 47.1, pp. 111–122. ISSN: 0167-9473. DOI: 10.1016/j.csda.2003.10.021.

Cuevas, A., M. Febrero, and R. Fraiman (2007). "Robust Estimation and Classification for Functional Data via Projection-Based Depth Notions". In: *Computational Statistics* 22.3, pp. 481–496. ISSN: 1613-9658. DOI: 10.1007/s00180-007-0053-0.

Dai, W. and M. G. Genton (2018). "Multivariate Functional Data Visualization and Outlier Detection". In: *Journal of Computational and Graphical Statistics* 27.4, pp. 923–934. ISSN: 1061-8600. DOI: 10.1080/10618600.2018.1473781.

Dai, W. and M. G. Genton (2019). "Directional Outlyingness for Multivariate Functional Data". In: *Computational Statistics & Data Analysis*. High-Dimensional and Functional Data Analysis 131, pp. 50–65. ISSN: 0167-9473. DOI: 10.1016/j.csda.2018.03.017.

Dai, X., H.-G. Müller, and F. Yao (2017). "Optimal Bayes Classifiers for Functional Data and Density Ratios". In: *Biometrika* 104.3, pp. 545–560. ISSN: 0006-3444. DOI: 10.1093/biomet/asx024.

Dask Development Team (2016). *Dask: Library for Dynamic Task Scheduling*. Manual. URL: https://dask.org.

Dau, H. A., A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh (2019). "The UCR Time Series Archive". In: *IEEE/CAA Journal of Automatica Sinica* 6.6, pp. 1293–1305. ISSN: 2329-9274. DOI: 10.1109/JAS.2019.1911747.

Delaigle, A., P. Hall, and N. Bathia (2012). "Componentwise Classification and Clustering of Functional Data". In: *Biometrika* 99.2, pp. 299–313. ISSN: 0006-3444. DOI: 10.1093/biomet/ass003.

Delaigle, A. and P. Hall (2010). "Defining Probability Density for a Distribution of Random Functions". In: *The Annals of Statistics* 38.2, pp. 1171–1193. ISSN: 0090-5364, 2168-8966. DOI: 10.1214/09-AOS741.

Delaigle, A. and P. Hall (2012). "Achieving near Perfect Classification for Functional Data". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 74.2, pp. 267–286. ISSN: 1467-9868. DOI: 10.1111/j.1467-9868.2011.01003.x.

Delaigle, A. and P. Hall (2013). "Classification Using Censored Functional Data". In: *Journal of the American Statistical Association* 108.504, pp. 1269–1283. ISSN: 0162-1459. DOI: 10.1080/01621459.2013.824893.

Demšar, J. (2006). "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *Journal of Machine Learning Research* 7.Jan, pp. 1–30. ISSN: ISSN 1533-7928.

Devroye, L., L. Györfi, and G. Lugosi (1996). *A Probabilistic Theory of Pattern Recognition*. Ed. by I. Karatzas and M. Yor. Vol. 31. Stochastic Modelling and Applied Probability. New York, NY: Springer. ISBN: 978-1-4612-6877-2 978-1-4612-0711-5. DOI: 10.1007/978-1-4612-0711-5.

Díaz-Vico, D. and J. R. Dorronsoro (2020). "Deep Least Squares Fisher Discriminant Analysis". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.8, pp. 2752–2763. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2019.2906302.

Díaz-Vico, D., A. Fernández, and J. R. Dorronsoro (2021). "Companion Losses for Deep Neural Networks". In: *Hybrid Artificial Intelligent Systems*. Ed. by H. Sanjurjo González, I. Pastor López, P. García Bringas, H. Quintián, and E. Corchado. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 538–549. ISBN: 978-3-030-86271-8. DOI: 10.1007/978-3-030-86271-8_45.

Díaz-Vico, D., A. R. Figueiras-Vidal, and J. R. Dorronsoro (2018). "Deep MLPs for Imbalanced Classification". In: *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. DOI: 10.1109/IJCNN.2018.8489504.

Díaz-Vico, D., J. Prada, A. Omari, and J. Dorronsoro (2020). "Deep Support Vector Neural Networks". In: *Integrated Computer-Aided Engineering* 27.4, pp. 389–402. ISSN: 1069-2509. DOI: 10.3233/ICA-200635.

Díaz-Vico, D., J. Prada, A. Omari, and J. R. Dorronsoro (2019). "Deep Support Vector Classification and Regression". In: *From Bioinspired Systems and Biomedical Applications to Machine Learning*. Ed. by J. M. Ferrández Vicente, J. R. Álvarez-Sánchez, F. de la Paz López, J. Toledo Moreo, and H. Adeli. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 33–43. ISBN: 978-3-030-19651-6. DOI: 10.1007/978-3-030-19651-6_4.

Díaz-Vico, D. and C. Ramos-Carreño (2022). *scikit-datasets: scikit-learn-compatible Datasets*. DOI: 10.5281/zenodo.6383047. URL: https://github.com/daviddiazvico/scikit-datasets (visited on 05/22/2023).

Diethe, T. (2015). *13 Benchmark Datasets Derived from the UCI, DELVE and STATLOG Repositories*. DOI: 10.5281/zenodo.18110. URL: https://github.com/tdiethe/gunnar_raetsch_benchmark_datasets/ (visited on 05/22/2023).

Ding, C. and H. Peng (2005). "Minimum Redundancy Feature Selection from Microarray Gene Expression Data". In: *Journal of Bioinformatics and Computational Biology* 03.02, pp. 185–205. ISSN: 0219-7200. DOI: 10.1142/S0219720005001004.

Doob, J. L. (1942). "The Brownian Movement and Stochastic Equations". In: *Annals of Mathematics* 43.2, pp. 351–369. ISSN: 0003-486X. DOI: 10.2307/1968873.

Doob, J. L. (1990). *Stochastic Processes*. 1st ed. WILEY. ISBN: 978-0-471-52369-7.

Dua, D. and C. Graff (2017). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml.

Dueck, J., D. Edelmann, T. Gneiting, and D. Richards (2014). "The Affinely Invariant Distance Correlation". In: *Bernoulli* 20.4, pp. 2305–2330. ISSN: 1350-7265. DOI: 10.3150/13-BEJ558.

Dunn, J. C. (1973). "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters". In: *Journal of Cybernetics* 3.3, pp. 32–57. ISSN: 0022-0280. DOI: 10.1080/01969727308546046.

Edelmann, D. and J. Fiedler (2022). *Dcortools: Providing Fast and Flexible Functions for Distance Correlation Analysis*. Manual. URL: https://CRAN.R-project.org/package=dcortools.

Epifanio, I. and N. Ventura-Campos (2014). "Hippocampal Shape Analysis in Alzheimer's Disease Using Functional Data Analysis". In: *Statistics in Medicine* 33.5, pp. 867–880. ISSN: 1097-0258. DOI: 10.1002/sim.5968.

Fama, E. F. (1965). "The Behavior of Stock-Market Prices". In: *The Journal of Business* 38.1, pp. 34–105. ISSN: 0021-9398. URL: https://www.jstor.org/stable/2350752 (visited on 09/26/2018).

Febrero, M., P. Galeano, and W. González-Manteiga (2008). "Outlier Detection in Functional Data by Depth Measures, with Application to Identify Abnormal NO_x Levels". In: *Environmetrics* 19.4, pp. 331–345. ISSN: 1099-095X. DOI: 10.1002/env.878.

Febrero-Bande, M., P. Galeano, and W. González-Manteiga (2017). "Functional Principal Component Regression and Functional Partial Least-Squares Regression: An Overview and a Comparative Study". In: *International Statistical Review* 85.1, pp. 61–83. ISSN: 1751-5823. DOI: 10.1111/insr.12116.

Febrero-Bande, M. and M. Oviedo de la Fuente (2012). "Statistical Computing in Functional Data Analysis: The R Package fda.usc". In: *Journal of Statistical Software* 51.1, pp. 1–28. ISSN: 1548-7660. DOI: 10.18637/jss.v051.i04.

Feldman, J. (1958). "Equivalence and Perpendicularity of Gaussian Processes." In: *Pacific Journal of Mathematics* 8.4, pp. 699–708. ISSN: 0030-8730. URL: https://projecteuclid.org/journals/pacific-journal-of-mathematics/volume-8/issue-4/Equivalence-and-perpendicularity-of-Gaussian-processes/pjm/1103039696.full (visited on 01/18/2023).

Feng, S., M. Zhang, and T. Tong (2022). "Variable Selection for Functional Linear Models with Strong Heredity Constraint". In: *Annals of the Institute of Statistical Mathematics* 74.2, pp. 321–339. ISSN: 1572-9052. DOI: 10.1007/s10463-021-00798-z.

Fermanian, A. (2022). "Functional Linear Regression with Truncated Signatures". In: *Journal of Multivariate Analysis* 192, p. 105031. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2022.105031.

Fernández-Delgado, M., E. Cernadas, S. Barro, and D. Amorim (2014). "Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?" In: *Journal of Machine Learning Research* 15.1, pp. 3133–3181. ISSN: 1532-4435.

Ferrando, L., N. Ventura-Campos, and I. Epifanio (2020). "Detecting and Visualizing Differences in Brain Structures with SPHARM and Functional Data Analysis". In: *NeuroImage* 222, p. 117209. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2020.117209.

Ferraty, F., P. Hall, and P. Vieu (2010). "Most-Predictive Design Points for Functional Data Predictors". In: *Biometrika* 97.4, pp. 807–824. ISSN: 0006-3444. URL: https://www.jstor.org/stable/29777138 (visited on 10/15/2018).

Ferraty, F. and P. Vieu (2006). *Nonparametric Functional Data Analysis: Theory and Practice*. Springer Series in Statistics. New York: Springer-Verlag. ISBN: 978-0-387-30369-7. URL: https://www.springer.com/gp/book/9780387303697 (visited on 09/10/2019).

Ferreira, L. and D. B. Hitchcock (2009). “A Comparison of Hierarchical Methods for Clustering Functional Data”. In: *Communications in Statistics - Simulation and Computation* 38.9, pp. 1925–1949. ISSN: 0361-0918. DOI: 10.1080/03610910903168603.

Fraiman, R., Y. Gimenez, and M. Svarc (2016). “Feature Selection for Functional Data”. In: *Journal of Multivariate Analysis*. Special Issue on Statistical Models and Methods for High or Infinite Dimensional Spaces 146, pp. 191–208. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2015.09.006.

Fraiman, R. and G. Muniz (2001). “Trimmed Means for Functional Data”. In: *TEST* 10.2, pp. 419–440. ISSN: 1863-8260. DOI: 10.1007/BF02595706.

Frois Caldeira, J., R. Gupta, M. T. Suleman, and H. S. Torrent (2020). “Forecasting the Term Structure of Interest Rates of the BRICS: Evidence from a Nonparametric Functional Data Analysis”. In: *Emerging Markets Finance and Trade* 0.0, pp. 1–18. ISSN: 1540-496X. DOI: 10.1080/1540496X.2020.1808458.

Galeano, P., E. Joseph, and R. E. Lillo (2015). “The Mahalanobis Distance for Functional Data with Applications to Classification”. In: *Technometrics* 57.2, pp. 281–291. ISSN: 0040-1706. DOI: 10.1080/00401706.2014.902774.

Ghosal, R. and A. Maity (2022). “Variable Selection in Nonparametric Functional Concurrent Regression”. In: *Canadian Journal of Statistics* 50.1, pp. 142–161. ISSN: 1708-945X. DOI: 10.1002/cjs.11654.

Ghumman, A. R., Ateeq-ur-Rauf, H. Haider, and Md. Shafiquzamman (2019). “Functional Data Analysis of Models for Predicting Temperature and Precipitation under Climate Change Scenarios”. In: *Journal of Water and Climate Change* 11.4, pp. 1748–1765. ISSN: 2040-2244. DOI: 10.2166/wcc.2019.172.

Gijbels, I. and S. Nagy (2017). “On a General Definition of Depth for Functional Data”. In: *Statistical Science* 32.4, pp. 630–639. ISSN: 0883-4237, 2168-8745.

Goia, A. and P. Vieu (2016). “An Introduction to Recent Advances in High/Infinite Dimensional Statistics”. In: *Journal of Multivariate Analysis*. Special Issue on Statistical Models and Methods for High or Infinite Dimensional Spaces 146, pp. 1–6. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2015.12.001.

Goldsmith, J., F. Scheipl, L. Huang, J. Wrobel, C. Di, J. Gellar, J. Harezlak, M. W. McLean, B. Swihart, L. Xiao, C. Crainiceanu, and P. T. Reiss (2019). *refund: Regression with Functional Data*. Manual. R package version 0.1-21. URL: https://CRAN.R-project.org/package=refund.

Golovkine, S. (2021). *FDApy: A Python Package for Functional Data*. DOI: 10.48550/arXiv.2101.11003.

Gómez-Verdejo, V., M. Verleysen, and J. Fleury (2009). “Information-Theoretic Feature Selection for Functional Data Classification”. In: *Neurocomputing*. Financial Engineering 72.16, pp. 3580–3589. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2008.12.035.

Gong, X., Y. Wang, and B. Lin (2021). “Assessing Dynamic China’s Energy Security: Based on Functional Data Analysis”. In: *Energy* 217, p. 119324. ISSN: 0360-5442. DOI: 10.1016/j.energy.2020.119324.

Goodger, D. and G. Van Rossum (2001). *PEP 257: Docstring Conventions*. Accessed: 2020-09-28. URL: https://www.python.org/dev/peps/pep-0257/.

GPy (2012). *GPy: A Gaussian Process Framework in Python*. URL: http://github.com/SheffieldML/GPy (visited on 05/22/2023).

Green, P. J. and B. W. Silverman (1993). *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. CRC Press. ISBN: 978-0-412-30040-0.

Greff, K., A. Klein, M. Chovanec, F. Hutter, and Jürgen Schmidhuber (2017). “The Sacred Infrastructure for Computational Research”. In: *Proceedings of the 16th Python* *in Science Conference*. Ed. by K. Huff, D. Lippa, D. Niederhut, and M Pacer, pp. 49–56. DOI: 10.25080/shinma-7f4c6e7-008.

Grosenick, L., S. Greer, and B. Knutson (2008). "Interpretable Classifiers for fMRI Improve Prediction of Purchases". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 16.6, pp. 539–548. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2008.926701.

Gulgezen, G., Z. Cataltepe, and L. Yu (2009). "Stable and Accurate Feature Selection". In: *Machine Learning and Knowledge Discovery in Databases*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 455–468. ISBN: 978-3-642-04180-8.

Guo, X., A. Srivastava, and S. Sarkar (2021). "A Quotient Space Formulation for Generative Statistical Analysis of Graphical Data". In: *Journal of Mathematical Imaging and Vision* 63.6, pp. 735–752. ISSN: 1573-7683. DOI: 10.1007/s10851-021-01027-1.

Guyon, I. and A. Elisseeff (2003). "An Introduction to Variable and Feature Selection". In: *Journal of Machine Learning Research* 3.Mar, pp. 1157–1182. ISSN: ISSN 1533-7928. URL: http://www.jmlr.org/papers/v3/guyon03a.html (visited on 10/18/2018).

Guyon, I., S. Gunn, M. Nikravesh, and L. A. Zadeh, eds. (2006). *Feature Extraction: Foundations and Applications*. Studies in Fuzziness and Soft Computing. Berlin Heidelberg: Springer-Verlag. ISBN: 978-3-540-35487-1. URL: https://www.springer.com/gp/book/9783540354871 (visited on 09/09/2019).

Hájek, J. (1958). "A Property of *J*-Divergences of Marginal Probability Distributions". In: *Czechoslovak Mathematical Journal* 08.3, pp. 460–463. ISSN: 0011-4642. URL: https://eudml.org/doc/11945 (visited on 01/18/2023).

Happ-Kurz, C. (2020). "Object-Oriented Software for Functional Data". In: *Journal of Statistical Software* 93.1, pp. 1–38. ISSN: 1548-7660. DOI: 10.18637/jss.v093.i05.

Harris, C. R., K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant (2020). "Array Programming with NumPy". In: *Nature* 585.7825, pp. 357–362. ISSN: 1476-4687. DOI: 10.1038/s41586-020-2649-2.

Hartigan, J. A. and M. A. Wong (1979). "A K-Means Clustering Algorithm". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1, pp. 100–108. ISSN: 0035-9254. DOI: 10.2307/2346830.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second. Springer Series in Statistics. New York: Springer-Verlag. ISBN: 978-0-387-84857-0. URL: //www.springer.com/la/book/9780387848570 (visited on 12/09/2018).

Herbold, S. (2020). "Autorank: A Python Package for Automated Ranking of Classifiers". In: *Journal of Open Source Software* 5.48, p. 2173. ISSN: 2475-9066.

Horsley, K. J., J. O. Ramsay, B. Ditto, and D. Da Costa (2021). "Maternal Blood Pressure Trajectories and Associations with Gestational Age at Birth: A Functional Data Analytic Approach". In: *Journal of Hypertension*. ISSN: 0263-6352. DOI: 10.1097/HJH.0000000000002995.

Hsing, T. and R. Eubank (2015). *Theoretical Foundations of Functional Data Analysis, with an Introduction to Linear Operators*. Wiley Series in Probability and Statistics. Chichester, UK: John Wiley & Sons. ISBN: 978-1-118-76254-7 978-0-470-01691-6. DOI: 10.1002/9781118762547.

Hu, X., Y. Yuan, X. Zhu, H. Yang, and K. Xie (2019). "Behavioral Responses to Pre-Planned Road Capacity Reduction Based on Smartphone GPS Trajectory Data: A Functional Data Analysis Approach". In: *Journal of Intelligent Transportation Systems* 23.2, pp. 133–143. ISSN: 1547-2450. DOI: 10.1080/15472450.2018.1488133.

Hubert, M., P. Rousseeuw, and P. Segaert (2017). "Multivariate and Functional Classification Using Depth and Distance". In: *Advances in Data Analysis and Classification* 11.3, pp. 445–466. ISSN: 1862-5355. DOI: 10.1007/s11634-016-0269-3.

Hunter, J. D. (2007). "Matplotlib: A 2D Graphics Environment". In: *Computing in Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Huo, X. and G. J. Székely (2016). "Fast Computing for Distance Covariance". In: *Technometrics* 58.4, pp. 435–447. ISSN: 0040-1706. DOI: 10.1080/00401706.2015.1054435.

Hyndman, R. J. and M. Shahid Ullah (2007). "Robust Forecasting of Mortality and Fertility Rates: A Functional Data Approach". In: *Computational Statistics & Data Analysis* 51.10, pp. 4942–4956. ISSN: 0167-9473. DOI: 10.1016/j.csda.2006.07.028.

Hyndman, R. J. and H. L. Shang (2010). "Rainbow Plots, Bagplots, and Boxplots for Functional Data". In: *Journal of Computational and Graphical Statistics* 19.1, pp. 29–45. ISSN: 1061-8600. DOI: 10.1198/jcgs.2009.08158.

Ieva, F., A. M. Paganoni, J. Romo, and N. Tarabelloni (2019). "roahd Package: Robust Analysis of High Dimensional Data". In: *The R Journal* 11.2, pp. 291–307.

Ivanescu, A. E., A.-M. Staicu, F. Scheipl, and S. Greven (2015). "Penalized Function-on-Function Regression". In: *Computational Statistics* 30.2, pp. 539–568. ISSN: 1613-9658. DOI: 10.1007/s00180-014-0548-4.

Jacques, J. and C. Preda (2014). "Functional Data Clustering: A Survey". In: *Advances in Data Analysis and Classification* 8.3, pp. 231–255. ISSN: 1862-5355. DOI: 10.1007/s11634-013-0158-y.

James, N. A., A. Kejariwal, and D. S. Matteson (2016). "Leveraging Cloud Data to Mitigate User Experience from 'Breaking Bad'". In: *2016 IEEE International Conference on Big Data (IEEE Big Data 2016)*, pp. 3499–3508. DOI: 10.1109/BigData.2016.7841013.

Jiménez-Cordero, A. and S. Maldonado (2021). "Automatic Feature Scaling and Selection for Support Vector Machine Classification with Functional Data". In: *Applied Intelligence* 51.1, pp. 161–184. ISSN: 1573-7497. DOI: 10.1007/s10489-020-01765-6.

Joblib Development Team (2020). *Joblib: Running Python Functions as Pipeline Jobs*. URL: https://joblib.readthedocs.io/.

John Richardson (2022). *TuneTA: Intelligently Optimizes Technical Indicators and Optionally Selects the Least Intercorrelated for Use in Machine Learning Models*. URL: https://github.com/jmrichardson/tuneta (visited on 05/31/2022).

Joshi, S. H., E. Klassen, A. Srivastava, and I. Jermyn (2007). "A Novel Representation for Riemannian Analysis of Elastic Curves in $\mathbb{R}^n$". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–7. DOI: 10.1109/CVPR.2007.383185.

Kailath, T. (1966). "Some Results on Singular Detection". In: *Information and Control* 9.2, pp. 130–152. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(66)90202-6.

Kailath, T. (1971). "RKHS Approach to Detection and Estimation Problems–I: Deterministic Signals in Gaussian Noise". In: *IEEE Transactions on Information Theory* 17.5, pp. 530–549. ISSN: 1557-9654. DOI: 10.1109/TIT.1971.1054673.

Kalivas, J. H. (1997). “Two Data Sets of near Infrared Spectra”. In: *Chemometrics and Intelligent Laboratory Systems* 37.2, pp. 255–259. ISSN: 0169-7439. DOI: 10.1016/S0169-7439(97)00038-5.

Kasieczka, G. and D. Shih (2020). “Robust Jet Classifiers through Distance Correlation”. In: *Physical Review Letters* 125.12, p. 122001. DOI: 10.1103/PhysRevLett.125.122001.

Kayal, S. (2021). “Unsupervised Sentence-embeddings by Manifold Approximation and Projection”. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, pp. 1–11. DOI: 10.18653/v1/2021.eacl-main.1.

Kharyuk, P., D. Nazarenko, I. Oseledets, I. Rodin, O. Shpigun, A. Tsitsilin, and M. Lavrentyev (2018). “Employing Fingerprinting of Medicinal Plants by Means of LC-MS and Machine Learning for Species Identification Task”. In: *Scientific Reports* 8.1, p. 17053. ISSN: 2045-2322. DOI: 10.1038/s41598-018-35399-z.

Kim, A. Y., C. Marzban, D. B. Percival, and W. Stuetzle (2009). “Using Labeled Data to Evaluate Change Detectors in a Multivariate Streaming Environment”. In: *Signal Process.* Special Section: Visual Information Analysis for Security 89.12, pp. 2529–2536. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2009.04.011.

Kluyver, T., B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing (2016). “Jupyter Notebooks – a Publishing Format for Reproducible Computational Workflows”. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt. IOS Press, pp. 87–90.

Kneip, A., D. Poss, and P. Sarda (2016). “Functional Linear Regression with Points of Impact”. In: *The Annals of Statistics* 44.1, pp. 1–30. ISSN: 00905364. URL: http://www.jstor.org/stable/43818898.

Kneip, A. and J. O. Ramsay (2008). “Combining Registration and Fitting for Functional Models”. In: *Journal of the American Statistical Association* 103.483, pp. 1155–1165. ISSN: 0162-1459. DOI: 10.1198/016214508000000517.

Kneip, A. and P. Sarda (2011). “Factor Models and Variable Selection in High-Dimensional Regression Analysis”. In: *The Annals of Statistics* 39.5, pp. 2410–2447. ISSN: 0090-5364, 2168-8966. DOI: 10.1214/11-AOS905.

Kokoszka, P. and M. Reimherr (2017). *Introduction to Functional Data Analysis*. New York: Chapman and Hall/CRC. ISBN: 978-1-315-11741-6. DOI: 10.1201/9781315117416.

Kong, J., S. Wang, and G. Wahba (2015). “Using Distance Covariance for Improved Variable Selection with Application to Learning Genetic Risk Models”. In: *Statistics in Medicine* 34.10, pp. 1708–1720. ISSN: 1097-0258. DOI: 10.1002/sim.6441.

Kovachki, N., Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar (2023). “Neural Operator: Learning Maps between Function Spaces with Applications to PDEs”. In: *Journal of Machine Learning Research* 24.89, pp. 1–97. ISSN: 1533-7928. URL: http://jmlr.org/papers/v24/21-1524.html (visited on 05/09/2023).

Kuhn, M. (2008). “Building Predictive Models in R Using the Caret Package”. In: *Journal of Statistical Software* 28, pp. 1–26. ISSN: 1548-7660. DOI: 10.18637/jss.v028.i05.

Laarne, P., M. A. Zaidan, and T. Nieminen (2021). “Ennemi: Non-linear Correlation Detection with Mutual Information”. In: *SoftwareX* 14, p. 100686. ISSN: 2352-7110. DOI: 10.1016/j.softx.2021.100686.

Lam, S. K., A. Pitrou, and S. Seibert (2015). “Numba: A LLVM-based Python JIT Compiler”. In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. LLVM ’15. New York, NY, USA: Association for Computing Machinery, pp. 1–6. ISBN: 978-1-4503-4005-2. DOI: 10.1145/2833157.2833162.

Lamperti, J. (1977). *Stochastic Processes: A Survey of the Mathematical Theory*. Applied Mathematical Sciences. New York: Springer-Verlag. ISBN: 978-0-387-90275-3.

Lardin-Puech, P., H. Cardot, and C. Goga (2014). “Analysing Large Datasets of Functional Data: a Survey Sampling Point of View”. In: *Journal de la société française de statistique* 155.4, pp. 70–94. URL: http://www.numdam.org/item/JSFS_2014__155_4_70_0/ (visited on 10/24/2021).

Laumann, F., J. von Kügelgen, T. H. Kanashiro Uehara, and M. Barahona (2022). “Complex Interlinkages, Key Objectives, and Nexuses among the Sustainable Development Goals and Climate Change: A Network Analysis”. In: *Lancet Planet. Health* 6.5, e422–e430. ISSN: 2542-5196. DOI: 10.1016/S2542-5196(22)00070-5.

Lee, E. R. and B. U. Park (2012). “Sparse Estimation in Functional Linear Regression”. In: *Journal of Multivariate Analysis* 105.1, pp. 1–17. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2011.08.005.

Leisch, F. and E. Dimitriadou (2021). *mlbench: Machine Learning Benchmark Problems*. Manual. R package version 2.1-3.

Leng, X. and H.-G. Müller (2006). “Classification Using Functional Data Analysis for Temporal Gene Expression Data”. In: *Bioinformatics* 22.1, pp. 68–76. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bti742.

Li, J., J. A. Cuesta-Albertos, and R. Y. Liu (2012). “DD-Classifier: Nonparametric Classification Procedure Based on DD-Plot”. In: *Journal of the American Statistical Association* 107.498, pp. 737–753. ISSN: 0162-1459. DOI: 10.1080/01621459.2012.688462.

Li, R., W. Zhong, and L. Zhu (2012). “Feature Screening via Distance Correlation Learning”. In: *Journal of the American Statistical Association* 107.499, pp. 1129–1139. ISSN: 0162-1459. DOI: 10.1080/01621459.2012.695654.

Lian, H. (2011). “Convergence of Functional K-Nearest Neighbor Regression Estimate with Functional Responses”. In: *Electronic Journal of Statistics* 5, pp. 31–40. ISSN: 1935-7524, 1935-7524. DOI: 10.1214/11-EJS595.

Lindquist, M. A. and I. W. McKeague (2009). “Logistic Regression with Brownian-like Predictors”. In: *Journal of the American Statistical Association* 104.488, pp. 1575–1585. ISSN: 0162-1459. DOI: 10.1198/jasa.2009.tm08496.

Litjens, G., T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, and C. I. Sánchez (2017). “A Survey on Deep Learning in Medical Image Analysis”. In: *Medical Image Analysis* 42, pp. 60–88. ISSN: 1361-8415. DOI: 10.1016/j.media.2017.07.005.

López Pintado, S. and J. Romo (2005). “Depth-Based Classification for Functional Data”. In: *DES - Working Papers. Statistics and Econometrics. WS* ws055611. DOI: 10.1090/dimacs/072/08.

López-Pintado, S. and J. Romo (2009). “On the Concept of Depth for Functional Data”. In: *Journal of the American Statistical Association* 104.486, pp. 718–734. ISSN: 0162-1459.

Lukić, M. and J. Beder (2001). “Stochastic Processes with Sample Paths in Reproducing Kernel Hilbert Spaces”. In: *Transactions of the American Mathematical Society* 353.10, pp. 3945–3969. ISSN: 0002-9947, 1088-6850. DOI: 10.1090/S0002-9947-01-02852-5.

Malfait, N. and J. O. Ramsay (2003). “The Historical Functional Linear Model”. In: *The Canadian Journal of Statistics / La Revue Canadienne de Statistique* 31.2, pp. 115–128. ISSN: 0319-5724. DOI: 10.2307/3316063.

Markham, A., A. Chivukula, and M. Grosse-Wentrup (2020). “MeDIL: A Python Package for Causal Modelling”. In: *Proceedings of the 10th International Conference on Probabilistic Graphical Models*. PMLR, pp. 621–624.

Marks, S. and O. J. Dunn (1974). “Discriminant Functions When Covariance Matrices Are Unequal”. In: *Journal of the American Statistical Association* 69.346, pp. 555–559. ISSN: 0162-1459. DOI: 10.1080/01621459.1974.10482992.

Marron, J. S. and I. L. Dryden (2021). *Object Oriented Data Analysis*. New York: Chapman and Hall/CRC. ISBN: 978-1-351-18967-5. DOI: 10.1201/9781351189675.

Marron, J. S., J. O. Ramsay, L. M. Sangalli, and A. Srivastava (2015). “Functional Data Analysis of Amplitude and Phase Variation”. In: *Statistical Science* 30.4, pp. 468–484. ISSN: 0883-4237, 2168-8745. DOI: 10.1214/15-STS524.

Martin-Barragan, B., R. Lillo, and J. Romo (2014). “Interpretable Support Vector Machines for Functional Data”. In: *European Journal of Operational Research* 232.1, pp. 146–155. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2012.08.017.

Matsui, H. (2014). “Variable and Boundary Selection for Functional Data via Multiclass Logistic Regression Modeling”. In: *Computational Statistics & Data Analysis* 78, pp. 176–185. ISSN: 0167-9473. DOI: 10.1016/j.csda.2014.04.015.

Matteson, D. S. and N. A. James (2014). “A Nonparametric Approach for Multiple Change Point Analysis of Multivariate Data”. In: *Journal of the American Statistical Association* 109.505, pp. 334–345. ISSN: 0162-1459. DOI: 10.1080/01621459.2013.849605.

Maturo, F. and R. Verde (2022). “Pooling Random Forest and Functional Data Analysis for Biomedical Signals Supervised Classification: Theory and Application to Electrocardiogram Data”. In: *Statistics in Medicine* 41.12, pp. 2247–2275. ISSN: 1097-0258. DOI: 10.1002/sim.9353.

McKeague, I. W. and B. Sen (2010). “Fractals with Point Impact in Functional Linear Regression”. In: *Annals of Statistics* 38.4, pp. 2559–2586. ISSN: 0090-5364. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3684058/ (visited on 09/10/2019).

McLean, M. W., G. Hooker, A.-M. Staicu, F. Scheipl, and D. Ruppert (2014). “Functional Generalized Additive Models”. In: *Journal of Computational and Graphical Statistics* 23.1, pp. 249–269. ISSN: 1061-8600. DOI: 10.1080/10618600.2012.729985.

Menvouta, E. J., S. Serneels, and T. Verdonck (2023). “direpack: A Python 3 Package for State-of-the-Art Statistical Dimensionality Reduction Methods”. In: *SoftwareX* 21, p. 101282. ISSN: 2352-7110. DOI: 10.1016/j.softx.2022.101282.

MicroPyramid (2021). *Forex-Python*. MicroPyramid. URL: https://github.com/MicroPyramid/forex-python (visited on 12/15/2021).

Mika, S., G. Ratsch, J. Weston, B. Scholkopf, and K. Mullers (1999). “Fisher Discriminant Analysis with Kernels”. In: *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*, pp. 41–48. DOI: 10.1109/NNSP.1999.788121.

Millman, K. J. and M. Aivazis (2011). “Python for Scientists and Engineers”. In: *Computing in Science Engineering* 13.2, pp. 9–12. ISSN: 1558-366X. DOI: 10.1109/MCSE.2011.36.

Mishura, Y. and G. Shevchenko (2017). *Theory and Statistical Applications of Stochastic Processes*. John Wiley & Sons. ISBN: 978-1-119-47663-4.

Moody, G. and R. Mark (2001). “The Impact of the MIT-BIH Arrhythmia Database”. In: *IEEE Engineering in Medicine and Biology Magazine* 20.3, pp. 45–50. ISSN: 1937-4186. DOI: 10.1109/51.932724.

Morris, J. S. (2015). “Functional Regression”. In: *Annual Review of Statistics and Its Application* 2.1, pp. 321–359. DOI: 10.1146/annurev-statistics-010814-020413.

Mörters, P. and Y. Peres (2010). *Brownian Motion*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press. ISBN: 978-0-521-76018-8. DOI: 10.1017/CBO9780511750489.

Mosler, K. (2013). “Depth Statistics”. In: *Robustness and Complex Data Structures: Festschrift in Honour of Ursula Gather*. Ed. by C. Becker, R. Fried, and S. Kuhnt. Berlin, Heidelberg: Springer, pp. 17–34. ISBN: 978-3-642-35494-6. DOI: 10.1007/978-3-642-35494-6_2.

Mosler, K. and P. Mozharovskyi (2017). “Fast DD-classification of Functional Data”. In: *Statistical Papers* 58.4, pp. 1055–1089. ISSN: 1613-9798. DOI: 10.1007/s00362-015-0738-3.

Müller, H.-G., J. R. Carey, D. Wu, P. Liedo, and J. W. Vaupel (2001). “Reproductive Potential Predicts Longevity of Female Mediterranean Fruitflies”. In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 268.1466, pp. 445–450. DOI: 10.1098/rspb.2000.1370.

Müller, H.-G. and U. Stadtmüller (2005). “Generalized Functional Linear Models”. In: *The Annals of Statistics* 33.2, pp. 774–805. ISSN: 0090-5364, 2168-8966. DOI: 10.1214/009053604000001156.

Nagy, S., I. Gijbels, M. Omelka, and D. Hlubinka (2016). “Integrated Depth for Functional Data: Statistical Properties and Consistency”. In: *ESAIM: Probability and Statistics* 20, pp. 95–130. ISSN: 1292-8100, 1262-3318. DOI: 10.1051/ps/2016005.

Nagy, S., S. Helander, G. V. Bever, L. Viitasaari, and P. Ilmonen (2021). “Flexible Integrated Functional Depths”. In: *Bernoulli* 27.1, pp. 673–701. ISSN: 1350-7265. DOI: 10.3150/20-BEJ1254.

Nájera, Ó., E. Larson, L. Estève, G. Varoquaux, L. Liu, J. Grobler, E. S. de Andrade, C. Holdgraf, A. Gramfort, M. Jas, J. Nothman, O. Grisel, N. Varoquaux, E. Gouillart, M. Luessi, A. Lee, J. Vanderplas, T. Hoffmann, T. A. Caswell, B. Sullivan, A. Batula, jaeilepp, T. Robitaille, S. Appelhoff, P. Kunzmann, M. Geier, Lars, K. Sunden, D. Stańczak, and A. Y. Shih (2020). *sphinx-gallery: Release v0.7.0*. Version v0.7.0. DOI: 10.5281/zenodo.3741780.

Narisetty, N. N. and V. N. Nair (2016). “Extremal Depth for Functional Data and Applications”. In: *Journal of the American Statistical Association* 111.516, pp. 1705–1714. ISSN: 0162-1459. DOI: 10.1080/01621459.2015.1110033.

Nayak, J., B. Naik, and H. S. Behera (2015). “Fuzzy C-Means (FCM) Clustering Algorithm: A Decade Review from 2000 to 2014”. In: *Computational Intelligence in Data Mining - Volume 2*. Ed. by L. C. Jain, H. S. Behera, J. K. Mandal, and D. P. Mohapatra. Smart Innovation, Systems and Technologies. New Delhi: Springer India, pp. 133–149. ISBN: 978-81-322-2208-8. DOI: 10.1007/978-81-322-2208-8_14.

Ojo, O. T., R. E. Lillo, and A. Fernandez Anta (2021). *fdaoutlier: Outlier Detection Tools for Functional Data Analysis*. Manual. URL: https://CRAN.R-project.org/package=fdaoutlier.

Okuta, R., Y. Unno, D. Nishino, S. Hido, and C. Loomis (2017). “CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations”. In: *Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Thirty-First Annual Conference on Neural Information Processing Systems (NIPS)*. URL: http://learningsys.org/nips17/assets/papers/paper_16.pdf.

Oliphant, T. E. (2007). “Python for Scientific Computing”. In: *Computing in Science Engineering* 9.3, pp. 10–20. ISSN: 1558-366X. DOI: 10.1109/MCSE.2007.58.

Ordóñez, C., M. Oviedo de la Fuente, J. Roca-Pardiñas, and J. R. Rodríguez-Pérez (2018). “Determining Optimum Wavelengths for Leaf Water Content Estimation from Reflectance: A Distance Correlation Approach”. In: *Chemometrics and Intelligent Laboratory Systems* 173, pp. 41–50. ISSN: 0169-7439. DOI: 10.1016/j.chemolab.2017.12.001.

Osborne, M. F. M. (1959). “Brownian Motion in the Stock Market”. In: *Operations Research* 7.2, pp. 145–173. ISSN: 0030-364X. URL: https://www.jstor.org/stable/167153 (visited on 09/26/2018).

Oshinubi, K., F. Ibrahim, M. Rachdi, J. Demongeot, K. Oshinubi, F. Ibrahim, M. Rachdi, and J. Demongeot (2022). “Functional Data Analysis: Application to Daily Observation of COVID-19 Prevalence in France”. In: *AIMS Mathematics* 7.math-07-04-298, pp. 5347–5385. ISSN: 2473-6988. DOI: 10.3934/math.2022298.

Panda, S., S. Palaniappan, J. Xiong, E. Bridgeford, R. Mehta, C. Shen, and J. Vogelstein (2021). *hyppo: A Multivariate Hypothesis Testing Python Package*. URL: https://github.com/neurodata/hyppo (visited on 05/22/2023).

Pandas Development Team (2020). *pandas-dev/pandas: pandas*. Version latest. DOI: 10.5281/zenodo.3509134.

Parzen, E. (1959). *Statistical Inference on Time Series by Hilbert Space Methods*. Tech. rep. 23. Stanford, California: Applied Mathematics and Statistics Laboratory, Stanford University.

Parzen, E. (1961a). “An Approach to Time Series Analysis”. In: *The Annals of Mathematical Statistics* 32.4, pp. 951–989. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177704840.

Parzen, E. (1961b). “Regression Analysis of Continuous Parameter Time Series”. In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. Vol. 4.1. University of California Press, pp. 469–490. URL: https://projecteuclid.org/ebooks/berkeley-symposium-on-mathematical-statistics-and-probability/Proceedings-of-the-Fourth-Berkeley-Symposium-on-Mathematical-Statistics-and/chapter/Regression-Analysis-of-Continuous-Parameter-Time-Series/bsmsp/1200512178 (visited on 01/18/2023).

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay (2011). “scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830. ISSN: 1533-7928. URL: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html (visited on 06/03/2019).

Pegoraro, M. and M. Beraha (2021). “Fast PCA in 1-D Wasserstein Spaces via B-Splines Representation and Metric Projection”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.10, pp. 9342–9349. ISSN: 2374-3468. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17126 (visited on 09/15/2021).

Peng, H., F. Long, and C. Ding (2005). “Feature Selection Based on Mutual Information Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy”. In:

IEEE Transactions on Pattern Analysis and Machine Intelligence 27.8, pp. 1226–1238. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2005.159.

Peng, J. and D. Paul (2011). *fpca: Restricted MLE for Functional Principal Components Analysis*. Manual. R package version 0.2-1. URL: https://CRAN.R-project.org/package=fpca.

Pfisterer, F., L. Beggel, X. Sun, F. Scheipl, and B. Bischl (2021). *Benchmarking Time Series Classification – Functional Data vs Machine Learning Approaches*. DOI: 10.48550/arXiv.1911.07511.

Picheny, V., R. Servien, and N. Villa-Vialaneix (2019). "Interpretable Sparse SIR for Functional Data". In: *Statistics and Computing* 29.2, pp. 255–267. ISSN: 1573-1375. DOI: 10.1007/s11222-018-9806-6.

Poskitt, D. S. and A. Sengarapillai (2013). "Description Length and Dimensionality Reduction in Functional Data Analysis". In: *Computational Statistics & Data Analysis*. The Third Special Issue on Statistical Signal Extraction and Filtering 58, pp. 98–113. ISSN: 0167-9473. DOI: 10.1016/j.csda.2011.03.018.

Poß, D., D. Liebl, A. Kneip, H. Eisenbarth, T. D. Wager, and L. F. Barrett (2020). "Superconsistent Estimation of Points of Impact in Non-Parametric Regression with Functional Predictors". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 82.4, pp. 1115–1140. ISSN: 1369-7412. DOI: 10.1111/rssb.12386.

Ramos-Carreño, C. (2020). *dcor: Distance Correlation and Related E-Statistics in Python*. DOI: 10.5281/zenodo.3468124. URL: https://github.com/vnmabus/dcor (visited on 05/22/2023).

Ramos-Carreño, C. (2023). "Fuzzy C-means: Differences on Clustering Behavior between High Dimensional and Functional Data (Student Abstract)". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37.

Ramos-Carreño, C. and J. L. Torrecilla (2023). "dcor: Distance Correlation and Energy Statistics in Python". In: *SoftwareX* 22, p. 101326. ISSN: 2352-7110. DOI: 10.1016/j.softx.2023.101326.

Ramos-Carreño, C., J. L. Torrecilla, M. Carbajo-Berrocal, P. Marcos, and A. Suárez (2023). "scikit-fda: A Python Package for Functional Data Analysis". In: *Journal of Statistical Software*. (Accepted for publication. Preprint available at http://arxiv.org/abs/2211.02566).

Ramos-Carreño, C., J. L. Torrecilla, Y. Hong, and A. Suárez (2022). "scikit-fda: Computational Tools for Machine Learning with Functional Data". In: *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 213–218. DOI: 10.1109/ICTAI56018.2022.00038.

Ramos-Carreño, C., J. L. Torrecilla, and A. Suárez (2022). "Classification of Functional Data: A Comparative Study". In: *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 866–871. DOI: 10.1109/ICMLA55696.2022.00143.

Ramos-Carreño, C. (2022). *rdata: Read R datasets from Python*. DOI: 10.5281/zenodo.6382237. URL: https://github.com/vnmabus/rdata (visited on 05/22/2023).

Ramos-Carreño, C., A. Suárez, J. L. Torrecilla, M. Carbajo Berrocal, P. Marcos Manchón, P. Pérez Manso, A. Hernando Bernabé, D. García Fernández, Y. Hong, P. M. Rodríguez-Ponga Eyriès, Á. Sánchez Romero, and E. Petrunina (2022). *scikit-fda: Functional Data Analysis in Python*. DOI: 10.5281/zenodo.3468127. URL: https://github.com/GAA-UAM/scikit-fda (visited on 05/22/2023).

Ramsay, J. O. and X. Li (1998). "Curve Registration". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 60.2, pp. 351–363. ISSN: 1467-9868. DOI: 10.1111/1467-9868.00129.

Ramsay, J. O., H. Wickham, S. Graves, and G. Hooker (2020). *fda: Functional Data Analysis*. Manual. R package version 2.4.8.1. URL: https://CRAN.R-project.org/package=fda.

Ramsay, J. and B. W. Silverman (2005). *Functional Data Analysis*. Second. Springer Series in Statistics. New York: Springer-Verlag. ISBN: 978-0-387-40080-8. DOI: 10.1007/b98888.

Ramsay, J. O., G. Hooker, and S. Graves (2009). *Functional Data Analysis with R and MATLAB*. Use R! New York: Springer-Verlag. ISBN: 978-0-387-98184-0. DOI: 10.1007/978-0-387-98185-7.

Rasmussen, C. E. and C. K. I. Williams (2005). *Gaussian Processes for Machine Learning*. Cambridge, Mass: The MIT Press. ISBN: 978-0-262-18253-9.

Rätsch, G., T. Onoda, and K.-R. Müller (2001). "Soft Margins for AdaBoost". In: *Machine Learning* 42.3, pp. 287–320. ISSN: 1573-0565. DOI: 10.1023/A:1007618119488.

Reiss, P. T., J. Goldsmith, H. L. Shang, and R. T. Ogden (2017). "Methods for Scalar-on-Function Regression". In: *International statistical review* 85.2, pp. 228–249. ISSN: 0306-7734. DOI: 10.1111/insr.12163.

Rincón, M. and M. D. Ruiz-Medina (2012). "Wavelet-RKHS-based Functional Statistical Classification". In: *Advances in Data Analysis and Classification* 6.3, pp. 201–217. ISSN: 1862-5355. DOI: 10.1007/s11634-012-0112-4.

Rincón Hidalgo, M. M. and M. D. Ruiz-Medina (2012). "Local Wavelet-Vaguelette-Based Functional Classification of Gene Expression Data". In: *Biometrical Journal* 54.1, pp. 75–93. ISSN: 1521-4036. DOI: 10.1002/bimj.201000135.

Rizzo, M. and G. Szekely (2022). *Energy: E-statistics: Multivariate Inference via the Energy of Data*. URL: https://CRAN.R-project.org/package=energy.

Rizzo, M. L. (2009). "New Goodness-of-Fit Tests for Pareto Distributions". In: *ASTIN Bulletin: The Journal of the IAA* 39.2, pp. 691–715. ISSN: 0515-0361, 1783-1350. DOI: 10.2143/AST.39.2.2044654.

Rizzo, M. L. and G. J. Székely (2010). "DISCO Analysis: A Nonparametric Extension of Analysis of Variance". In: *The Annals of Applied Statistics* 4.2, pp. 1034–1055. ISSN: 1932-6157, 1941-7330. DOI: 10.1214/09-AOAS245.

Rizzo, M. L. and G. J. Székely (2016). "Energy distance". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 8.1, pp. 27–38.

Romeo, V. M. and M. V. Marzol Jaén (2014). "Análisis del viento y la niebla en el aeropuerto de Los Rodeos (Tenerife). Cambios y tendencias". In: *Cambio climático y cambio global*. Ed. by S. Fernández Montes and F. Sánchez Rodrigo. Publicaciones de la Asociación Española de Climatología. Serie A. 9. Asociación Española de Climatología, pp. 325–334. ISBN: 978-84-16027-69-9.

Rossi, F. and B. Conan-Guez (2005). "Functional Multi-Layer Perceptron: A Non-Linear Tool for Functional Data Analysis". In: *Neural Networks* 18.1, pp. 45–60. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2004.07.001.

Rossi, F. and N. Villa (2006). "Support Vector Machine for Functional Data Classification". In: *Neurocomputing*. New Issues in Neurocomputing: 13th European Symposium on Artificial Neural Networks 69.7, pp. 730–742. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2005.12.010.

Rudin, W. (1991). *Functional Analysis*. Subsequent edition. New York: Mcgraw-Hill Higher Education. ISBN: 978-0-07-054236-5.

Ruiz-Meana, M., D. Garcia-Dorado, P. Pina, J. Inserte, L. Agulló, and J. Soler-Soler (2003). "Cariporide Preserves Mitochondrial Proton Gradient and Delays ATP Depletion in Cardiomyocytes during Ischemic Conditions". In: *American Journal* *of Physiology. Heart and Circulatory Physiology* 285.3, H999–1006. ISSN: 0363-6135. DOI: 10.1152/ajpheart.00035.2003.

Runge, J. (2022). *Tigramite – Causal Inference and Causal Discovery for Time Series Datasets*. URL: https://github.com/jakobrunge/tigramite (visited on 05/31/2022).

Saez Lab (2021). *Pypath: A Python Module for Molecular Signaling Prior Knowledge Processing*. Saez Lab. URL: https://github.com/saezlab/pypath (visited on 12/08/2021).

Sangalli, L. M., P. Secchi, and S. Vantini (2014). "Analysis of AneuRisk65 Data: *k*-Mean Alignment". In: *Electronic Journal of Statistics* 8.2, pp. 1891–1904. ISSN: 1935-7524, 1935-7524. DOI: 10.1214/14-EJS938A.

Sangalli, L. M., P. Secchi, S. Vantini, and A. Veneziani (2009). "A Case Study in Exploratory Functional Data Analysis: Geometrical Features of the Internal Carotid Artery". In: *Journal of the American Statistical Association* 104.485, pp. 37–48. ISSN: 0162-1459. DOI: 10.1198/jasa.2009.0002.

Sangalli, L. M., P. Secchi, S. Vantini, and V. Vitelli (2010). "K-Mean Alignment for Curve Clustering". In: *Computational Statistics & Data Analysis* 54.5, pp. 1219–1233. ISSN: 0167-9473. DOI: 10.1016/j.csda.2009.12.008.

Scheipl, F. (2021). *CRAN Task View: Functional Data Analysis*. URL: https://CRAN.R-project.org/view=FunctionalData (visited on 12/07/2021).

Scheipl, F., J. Gertheiss, and S. Greven (2016). "Generalized Functional Additive Mixed Models". In: *Electronic Journal of Statistics* 10.1, pp. 1455–1492. ISSN: 1935-7524. DOI: 10.1214/16-EJS1145.

Scheipl, F., J. Goldsmith, and J. Wrobel (2020). *tidyfun: Tools for Tidy Functional Data*. Manual. R package version 0.0.83. URL: https://github.com/tidyfun/tidyfun (visited on 05/22/2023).

Schlegel, P., C. Barnes, S. Jagannathan, B. Pedigo, and R. Court (2021). *Navis-Org/Navis: Neuron Analysis and Visualization*. DOI: 10.5281/zenodo.4699382.

Schmutz, A., J. Jacques, C. Bouveyron, L. Chèze, and P. Martin (2020). "Clustering Multivariate Functional Data in Group-Specific Functional Subspaces". In: *Computational Statistics*. ISSN: 1613-9658. DOI: 10.1007/s00180-020-00958-4.

Schwartzman, A., Y. Gavrilov, and R. J. Adler (2011). "Multiple Testing of Local Maxima for Detection of Peaks in 1D". In: *The Annals of Statistics* 39.6, pp. 3290–3319. ISSN: 0090-5364, 2168-8966. DOI: 10.1214/11-AOS943.

Sebastiani, F. (2002). "Machine Learning in Automated Text Categorization". In: *ACM Computing Surveys* 34.1, pp. 1–47. ISSN: 0360-0300. DOI: 10.1145/505282.505283.

Sguera, C., P. Galeano, and R. Lillo (2014). "Spatial Depth-Based Classification for Functional Data". In: *TEST* 23.4, pp. 725–750. ISSN: 1863-8260. DOI: 10.1007/s11749-014-0379-1.

Sguera, C., P. Galeano, and R. E. Lillo (2016). "Functional Outlier Detection by a Local Depth with Application to NO_x Levels". In: *Stochastic Environmental Research and Risk Assessment* 30.4, pp. 1115–1130. ISSN: 1436-3259. DOI: 10.1007/s00477-015-1096-3.

Shepp, L. A. (1966). "Radon-Nikodym Derivatives of Gaussian Measures". In: *The Annals of Mathematical Statistics* 37.2, pp. 321–354. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177699516.

Song, J. J., W. Deng, H.-J. Lee, and D. Kwon (2008). "Optimal Classification for Time-Course Gene Expression Data Using Functional Data Analysis". In: *Computational Biology and Chemistry* 32.6, pp. 426–432. ISSN: 1476-9271. DOI: 10.1016/j.compbiolchem.2008.07.007.

Sørensen, H., J. Goldsmith, and L. M. Sangalli (2013). “An Introduction with Medical Applications to Functional Data Analysis”. In: *Statistics in Medicine* 32.30, pp. 5222–5240. ISSN: 1097-0258. DOI: 10.1002/sim.5989.

Spellman, P. T., G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher (1998). “Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization”. In: *Molecular Biology of the Cell* 9.12, pp. 3273–3297. ISSN: 1059-1524.

Spence, A. (1975). “On the Convergence of the Nyström Method for the Integral Equation Eigenvalue Problem”. In: *Numerische Mathematik* 25.1, pp. 57–66. ISSN: 0945-3245. DOI: 10.1007/BF01419528.

Sphinx Development Team (2020). *Sphinx 4.0.0+ Documentation*. URL: https://www.sphinx-doc.org/en/master/ (visited on 06/16/2020).

Srivastava, A. and E. P. Klassen (2016). *Functional and Shape Data Analysis*. Springer Series in Statistics. New York: Springer-Verlag. ISBN: 978-1-4939-4018-9. DOI: 10.1007/978-1-4939-4020-2.

Srivastava, A., W. Wu, S. Kurtek, E. Klassen, and J. S. Marron (2011). *Registration of Functional Data Using Fisher-Rao Metric*. DOI: 10.48550/arXiv.1103.3817.

Sun, Y. and M. G. Genton (2011). “Functional Boxplots”. In: *Journal of Computational and Graphical Statistics* 20.2, pp. 316–334. ISSN: 1061-8600. DOI: 10.1198/jcgs.2011.09224.

Synthesized (2022). *FairLens: Identify Bias and Measure Fairness of Your Data*. Synthesized. URL: https://github.com/synthesized-io/fairlens (visited on 05/31/2022).

Székely, G. and M. L. Rizzo (2004). “Testing for Equal Distributions in High Dimensions”. In: *InterStat* 5.16.10, pp. 1249–1272.

Szekely, G. (2002). *E-Statistics: The Energy of Statistical Samples*. Technical Report 02–16. Department of Mathematics and Statistics: Bowling Green State University. DOI: 10.13140/RG.2.1.5063.9761.

Szekely, G. J. (1989). *Potential and Kinetic Energy in Statistics*. Lecture Notes. Budapest Institute of Technology.

Székely, G. J. and M. L. Rizzo (2005). “A New Test for Multivariate Normality”. In: *Journal of Multivariate Analysis* 93.1, pp. 58–80. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2003.12.002.

Szekely, G. J. and M. L. Rizzo (2005). “Hierarchical Clustering via Joint Between-within Distances: Extending Ward’s Minimum Variance Method”. In: *Journal of Classification* 22.2, pp. 151–183. ISSN: 1432-1343. DOI: 10.1007/s00357-005-0012-9.

Székely, G. J. and M. L. Rizzo (2009). “Brownian Distance Covariance”. In: *The Annals of Applied Statistics* 3.4, pp. 1236–1265. ISSN: 1932-6157, 1941-7330. DOI: 10.1214/09-AOAS312.

Székely, G. J. and M. L. Rizzo (2013a). “Energy Statistics: A Class of Statistics Based on Distances”. In: *Journal of Statistical Planning and Inference* 143.8, pp. 1249–1272. ISSN: 0378-3758. DOI: 10.1016/j.jspi.2013.03.018.

Székely, G. J. and M. L. Rizzo (2013b). “The Distance Correlation T-Test of Independence in High Dimension”. In: *Journal of Multivariate Analysis* 117, pp. 193–213. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2013.02.012.

Székely, G. J. and M. L. Rizzo (2014). “Partial Distance Correlation with Methods for Dissimilarities”. In: *The Annals of Statistics* 42.6, pp. 2382–2412. ISSN: 0090-5364, 2168-8966. DOI: 10.1214/14-AOS1255.

Székely, G. J. and M. L. Rizzo (2017). “The Energy of Data”. In: *Annual Review of Statistics and Its Application* 4.1, pp. 447–479. DOI: 10.1146/annurev-statistics-060116-054026.

Székely, G. J., M. L. Rizzo, and N. K. Bakirov (2007). “Measuring and Testing Dependence by Correlation of Distances”. In: *The Annals of Statistics* 35.6, pp. 2769–2794. ISSN: 0090-5364, 2168-8966. DOI: 10.1214/009053607000000505.

Tan, C. W., C. Bergmeir, F. Petitjean, and G. I. Webb (2020). *Monash University, UEA, UCR Time Series Extrinsic Regression Archive*. DOI: 10.48550/arXiv.2006.10996.

Tan, C. W., C. Bergmeir, F. Petitjean, and G. I. Webb (2021). “Time Series Extrinsic Regression”. In: *Data Mining and Knowledge Discovery* 35.3, pp. 1032–1060. ISSN: 1573-756X. DOI: 10.1007/s10618-021-00745-9.

Tapia, M., D. Heinemann, D. Ballari, and E. Zondervan (2022). “Spatio-Temporal Characterization of Long-Term Solar Resource Using Spatial Functional Data Analysis: Understanding the Variability and Complementarity of Global Horizontal Irradiance in Ecuador”. In: *Renewable Energy* 189, pp. 1176–1193. ISSN: 0960-1481. DOI: 10.1016/j.renene.2022.03.049.

Tokushige, S., H. Yadohisa, and K. Inada (2007). “Crisp and Fuzzy K-Means Clustering Algorithms for Multivariate Functional Data”. In: *Computational Statistics* 22.1, pp. 1–16. ISSN: 1613-9658. DOI: 10.1007/s00180-006-0013-0.

Torrecilla, J. L., C. Ramos-Carreño, M. Sánchez-Montañés, and A. Suárez (2020). “Optimal Classification of Gaussian Processes in Homo- and Heteroscedastic Settings”. In: *Statistics and Computing* 30.4, pp. 1091–1111. ISSN: 1573-1375. DOI: 10.1007/s11222-020-09937-7.

Torrecilla, J. L., C. Ramos-Carreño, and A. Suárez (2023). “Recursive maxima hunting: Variable selection in functional data classification”. (In preparation).

Torrecilla, J. L. and A. Suárez (2016). “Feature Selection in Functional Data Classification with Recursive Maxima Hunting”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., pp. 4835–4843. URL: https://proceedings.neurips.cc/paper/2016/file/28b60a16b55fd531047c0c958ce14b95-Paper.pdf (visited on 09/25/2018).

Tucker, J. D. (2020a). *fdasrsf: Functional Data Analysis Using the Square Root Slope Framework*. Manual. Python package version 2.1.4. URL: https://pypi.org/project/fdasrsf/.

Tucker, J. D. (2020b). *fdasrvf: Elastic Functional Data Analysis*. Manual. R package version 1.9.3. URL: https://CRAN.R-project.org/package=fdasrvf.

Tucker, J. D. (2021). *ElasticFDA: Julia library for elastic functional data analysis*. URL: https://github.com/jdtuck/ElasticFDA.jl (visited on 12/23/2021).

Tuddenham, R. D. and M. M. Snyder (1954). “Physical Growth of California Boys and Girls from Birth to Eighteen Years”. In: *Publications in Child Development. University of California, Berkeley* 1.2, pp. 183–364.

Ullah, S. and C. F. Finch (2013). “Applications of Functional Data Analysis: A Systematic Review”. In: *BMC Medical Research Methodology* 13.1, p. 43. ISSN: 1471-2288. DOI: 10.1186/1471-2288-13-43.

Vallat, R. (2018). “Pingouin: Statistics in Python”. In: *Journal of Open Source Software* 3.31, p. 1026. ISSN: 2475-9066. DOI: 10.21105/joss.01026.

Van Rossum, G., B. Warsaw, and N. Coghlan (2001). *PEP 8: Style Guide for Python Code*. Accessed: 2020-09-28. URL: http://www.python.org/dev/peps/pep-0008/.

Vanschoren, J., J. N. van Rijn, B. Bischl, and L. Torgo (2014). “OpenML: Networked Science in Machine Learning”. In: *ACM SIGKDD Explorations Newsletter* 15.2, pp. 49–60. ISSN: 1931-0145. DOI: 10.1145/2641190.2641198.

Vantini, S. (2012). “On the Definition of Phase and Amplitude Variability in Functional Data Analysis”. In: *TEST* 21.4, pp. 676–696. ISSN: 1863-8260. DOI: 10.1007/s11749-011-0268-9.

Varberg, D. E. (1961). “On Equivalence of Gaussian Measures.” In: *Pacific Journal of Mathematics* 11.2, pp. 751–762. ISSN: 0030-8730. URL: https://projecteuclid.org/journals/pacific-journal-of-mathematics/volume-11/issue-2/On-equivalence-of-Gaussian-measures/pjm/1103037345.full (visited on 01/18/2023).

Vergara, J. R. and P. A. Estévez (2014). “A Review of Feature Selection Methods Based on Mutual Information”. In: *Neural Computing and Applications* 24.1, pp. 175–186. ISSN: 1433-3058. DOI: 10.1007/s00521-013-1368-0.

Vieu, P. (2018). “On Dimension Reduction Models for Functional Data”. In: *Statistics & Probability Letters*. The Role of Statistics in the Era of Big Data 136, pp. 134–138. ISSN: 0167-7152. DOI: 10.1016/j.spl.2018.02.032.

Virtanen, P., R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey,
bibinitperiodI. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, and P. van Mulbregt (2020). “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17.3, pp. 261–272. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2.

Wagner-Muns, I. M., I. G. Guardiola, V. A. Samaranayke, and W. I. Kayani (2018). “A Functional Data Analysis Approach to Traffic Volume Forecasting”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.3, pp. 878–888. ISSN: 1558-0016. DOI: 10.1109/TITS.2017.2706143.

Wahl, P. W. and R. A. Kronmal (1977). “Discriminant Functions When Covariances Are Unequal and Sample Sizes Are Moderate”. In: *Biometrics* 33.3, pp. 479–484. ISSN: 0006-341X. DOI: 10.2307/2529362.

Wang, J.-L., J.-M. Chiou, and H.-G. Müller (2016). “Functional Data Analysis”. In: *Annual Review of Statistics and Its Application* 3.1, pp. 257–295. ISSN: 2326-8298. DOI: 10.1146/annurev-statistics-041715-033624.

Wasserman, L. (2006). *All of Nonparametric Statistics*. Springer Texts in Statistics. New York: Springer-Verlag. ISBN: 978-0-387-25145-5.

Wei, L. and E. Keogh (2006). “Semi-Supervised Time Series Classification”. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’06. New York, NY, USA: ACM, pp. 748–753. ISBN: 978-1-59593-339-3. DOI: 10.1145/1150402.1150498.

Weihs, C., D. Jannach, I. Vatolkin, and G. Rudolph, eds. (2016). *Music Data Analysis: Foundations and Applications*. New York: Chapman and Hall/CRC. ISBN: 978-1-315-37099-6. DOI: 10.1201/9781315370996.

Wickham, H., M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani (2019). “Welcome to the Tidyverse”. In: *Journal of Open Source Software* 4.43, p. 1686. DOI: 10.21105/joss.01686.

Winkler, R., F. Klawonn, and R. Kruse (2010). “Fuzzy C-Means in High Dimensional Spaces”. In: *International Journal of Fuzzy System Applications* 11. DOI: 10.4018/IJFSA.2011010101.

Wrobel, J., S. Y. Park, A. M. Staicu, and J. Goldsmith (2016). "Interactive Graphics for Functional Data Analyses". In: *Stat* 5.1, pp. 108–118. ISSN: 2049-1573. DOI: 10.1002/sta4.109.

Yang, G. (2012). "The Energy Goodness-of-Fit Test for Univariate Stable Distributions". PhD thesis. Bowling Green State University. URL: https://etd.ohiolink.edu/apexprod/rws_olink/r/1501/10?p10_etd_subid=49947&clear=10 (visited on 10/11/2022).

Yao, F., H.-G. Müller, and J.-L. Wang (2015). *PACE Package for Functional Data Analysis and Empirical Dynamics (MATLAB)*. URL: http://www.stat.ucdavis.edu/PACE/.

Yenigün, C. D. and M. L. Rizzo (2015). "Variable Selection in Regression Using Maximal Correlation and Distance Correlation". In: *Journal of Statistical Computation and Simulation* 85.8, pp. 1692–1705. ISSN: 0094-9655. DOI: 10.1080/00949655.2014.895354.

Yu, W., S. Wade, H. D. Bondell, and L. Azizi (2022). "Nonstationary Gaussian Process Discriminant Analysis With Variable Selection for High-Dimensional Functional Data". In: *Journal of Computational and Graphical Statistics*, pp. 1–13. ISSN: 1061-8600. DOI: 10.1080/10618600.2022.2098136.

Zhang, X., B. Podobnik, D. Y. Kenett, and H. Eugene Stanley (2014). "Systemic Risk and Causality Dynamics of the World International Shipping Market". In: *Physica A: Statistical Mechanics and its Applications* 415, pp. 43–53. ISSN: 0378-4371. DOI: 10.1016/j.physa.2014.07.068.

Zhelezniak, V., A. Shen, D. Busbridge, A. Savkov, and N. Hammerla (2019). "Correlations between Word Vector Sets". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 77–87. DOI: 10.18653/v1/D19-1008.

Zhu, H., P. J. Brown, and J. S. Morris (2012). "Robust Classification of Functional and Quantitative Image Data Using Functional Mixed Models". In: *Biometrics* 68.4, pp. 1260–1268. ISSN: 1541-0420. DOI: 10.1111/j.1541-0420.2012.01765.x.