Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi

Challenges in biomedical data science: data-driven solutions to clinical questions

by

Samuele Fiorini

Theses Series **DIBRIS-TH-2017-XX**

DIBRIS, Università di Genova
Via Opera Pia, 13 16145 Genova, Italy http://www.dibris.unige.it/

Università degli Studi di Genova

Dipartimento di Informatica, Bioingegneria,

Robotica ed Ingegneria dei Sistemi

Ph.D. Thesis in Computer Science and Systems Engineering
Computer Science Curriculum

Challenges in biomedical data science: data-driven solutions to clinical questions

by

Samuele Fiorini

November, 2017

Dottorato di Ricerca in Informatica ed Ingegneria dei Sistemi
Indirizzo Informatica
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università degli Studi di Genova

DIBRIS, Univ. di Genova
Via Opera Pia, 13
I-16145 Genova, Italy
http://www.dibris.unige.it/

Ph.D. Thesis in Computer Science and Systems Engineering
Computer Science Curriculum
(S.S.D. INF/01)

Submitted by Samuele Fiorini
DIBRIS, Univ. di Genova
. . . .

Date of submission: November 20, 2017

Title: Machine Learning 4 healthcare.

Advisor: Annalisa Barla
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università di Genova
. . .

Ext. Reviewers:
Lo Scopriremo
Lo Scopriremo
Lo Scopriremo

Abstract

Abstract

Contents

1 Introduction

Understanding the underlying mechanisms of biological systems can be a challenging task. Different domains can be involved and their interactions can be unknown.

Nowadays, most of the life science research aims at investigating on the extraction of meaningful information from heterogeneous sources of biological data. Thanks to the remarkable technological progresses of the last decades, the dimensions of modern data collections is continuously increasing.

This PhD thesis is divided in two parts. Part I presents a thorough description of the multidisciplinary prerequisites that are relevant for the comprehension of Part II, which, in turn, presents the original contributions of my work.

Part I is organized as follows: Chapter 3 introduces the concept of *data science* (Section 3.1) and its declination toward life science and biomedical studies. In this chapter, the major challenges of the field are presented (Section 3.2) along with several examples of the most common clinical/biological questions and their translation to data analysis tasks (Section 3.3). Chapter 4 summarizes basic notation and definitions adopted throughout the thesis (Section 2) and presents an overview of the statistical and technological tools that are mostly relevant for this work. In particular, this chapter defines the concept of *machine learning* from a general perspective and provides rigorous description of a selection of supervised and unsupervised learning strategies (Section **??**). This chapter also defines the concept of variable/feature selection (Section 4.3) and introduces the most relevant model selection and evaluation strategies (Section 4.4). At the end of this chapter, hints on the computational requirements and implementation strategies are finally presented (Section 4.5).

Part II describes the main contributions of my PhD work which consisted in the process of translating into data analysis tasks a number of biological questions coming from real-world clinical environments. For each task, this second part shows how the previously introduced tools can be exploited in order to develop statistically sound models that are capable of providing insightful answers to different clinical questions. This part is organized as follows: Chapter 5 introduces ADENINE, an open-source Python framework for large-scale data exploration that I developed during my PhD. Chapter 6 describes a work I conducted in collaboration with *Istituto Giannina Gaslini Children's Hospital* on metabolic age estimation from peripheral blood mononuclear cells samples. Chapter 7 describes a work I conducted in collaboration with the *Italian Multiple Sclerosis Foundation* in which I developed a temporal model that aims at predicting the evolution of multiple sclerosis patients exploiting the use of patient-friendly and inexpensive measures such as patient centered outcomes. Chapter 8 describes a work held in

collaboration with *Ospedale Policlinico San Martino* in which I developed a machine learning time-series forecasting approach for glucose sensor data collected by type I and type II diabetic patients.

My conclusions are finally drawn in Chapter 9.

2 Basic notation and definitions

In this thesis, datasets $\mathcal{D}$ are described as input-output pairs, $X \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^{n \times k}$, respectively. The i-th row of X is a d-dimensional data point $\boldsymbol{x}_i$ belonging to the input space $\mathcal{X} \subseteq \mathbb{R}^d$. The corresponding outputs $\boldsymbol{y}_i$ belong to the output space $\mathcal{Y}$.

The nature of the output space defines the problem as *binary classification* if $\mathcal{Y} = \{a, b\}$ (with $a \neq b$), *multiclass classification* if $\mathcal{Y} = \{\alpha, \beta, \ldots, \omega\}$ (with $\alpha \neq \beta \neq \cdots \neq \omega$)), *regression* if $\mathcal{Y} \subseteq \mathbb{R}$ and *vector-valued regression* if $\mathcal{Y} \subseteq \mathbb{R}^k$. For binary classification problems common choices for the label encoding are $a = 1, b = -1$ or $a = 0, b = 1$. For multiclass classification problems classes are usually encoded as natural numbers, *i.e.* $\alpha, \beta, \ldots, \omega \in \mathbb{N}$.

Predictive models are functions $f : \mathcal{X} \to \mathcal{Y}$. The number of relevant variables is d^*. In feature selection tasks, the number of selected features is $\tilde{d}$.

A kernel function acting on the elements of the input space is defined as $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$, where $\phi(\boldsymbol{x})$ is a *feature map* from $\mathbb{R}^d \to \mathbb{R}^{d'}$. Feature learning algorithms project the data into a p-dimensional space.

Whenever possible, real-valued variables will be indicated with lowercase letters (*e.g.* a), unidimensional vectors with lowercase bold letters (*e.g.* $\boldsymbol{a}$) and matrices, or tensors, with capital letters (*e.g.* A). When the value of some variable/parameter is the result of some data-drive estimation, such variable will be highlighted with a hat (*e.g.* $\hat{a}$). When used in the context of a data matrix, a subscript index will be used to identify a sample whereas a superscript index will refer to a given feature. So, for instance, given a data matrix $X \in \mathbb{R}^{n \times d}$ the j-th feature of the i-th sample is $\boldsymbol{x}_i^j$, with $0 \leq i \leq n$ and $0 \leq j \leq d$.

Part I

3 Background

This chapter defines stuff....

3.1 What is data science and why should we care?

{

- **Data engineering**
- **Data exploration**
- **Machine learning and data understanding**
- **Data visualization**

}

- cross-disciplinary field
- Drew Conway's Data Science Venn Diagram, first published on his blog in September 2010
- data-intensive applications (maybe)

3.2 Challenges in biomedical data science

The process of modeling complex systems often implies collecting large amount of data in the field of life science, where large, multivariate and noisy measurements are typically acquired with the aim of describing multifactorial diseases.

In the era of personalized medicine, biospecimen collection and biological data management is still a challenging and expensive task [Toga and Dinov, 2015]. Only few large-scale research enterprises, such as ENCODE [Consortium et al., 2004], ADNI [Jack et al., 2008], MOPED [Kolker

[1] http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

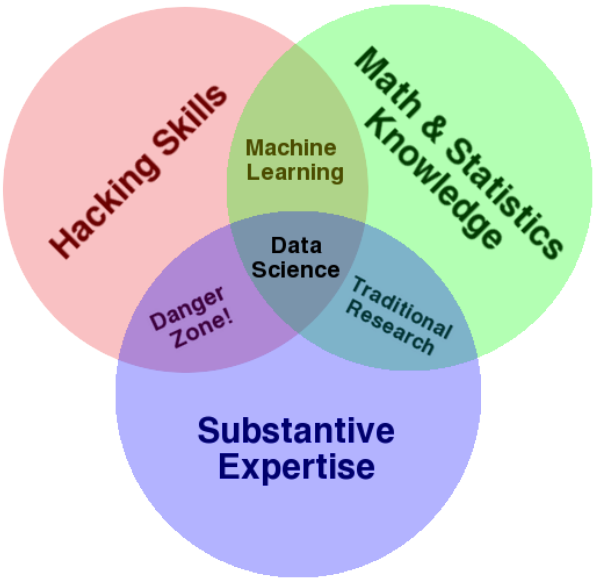


Figure 3.1: Drew Conway's Data Science Venn Diagram[1].

et al., 2012] or TCGA [2] , have sufficient financial and human resources to manage, share and distribute access of heterogeneous types of biological data. To date, many biomedical studies still rely on a small number of collected samples [McNeish and Stapleton, 2016; Button et al., 2013; Yu et al., 2013]. This effect is even worse in case of rare diseases [Garg et al., 2016] or in high-throughput molecular data (*e.g.* genomics and proteomics) where the dimensionality of the problem can be in the order of hundreds of thousands. The setting where the number of measured variables heavily outnumbers the amount of collected samples is usually referred to as *large p small n* scenario, or simply $n \ll p$. In this case, the main goal of the learning step is often to identify a meaningful subset of relevant variables that are the most representative of the observed phenomenon. In machine learning, this is known as variable/feature selection and several techniques addressing this task were presented so far [Guyon et al., 2002]. Variable selection not only increases the prediction power of the learning machine, but it also promotes model interpretability, that is crucial in biology [Altmann et al., 2010]. Regardless [Okser et al., 2014] of the learning machine, regularization can be introduced in several ways and it is of fundamental use in order to achieve the following desired properties:

- identify models with good generalization properties, even with a limited amount of collected samples;
- achieve solutions that are robust to noise;
- learn the data structure when unknown;

[2] https://cancergenome.nih.gov

- exploit prior knowledge on the data structure;
- promote interpretability performing variable selection;
- reduce the feasible set in order to help solving inverse problems.

In this paper we illustrate how regularization impacts in finding robust and meaningful models and we clarify how to choose the most suitable regularization scheme according to the biological context. The remainder of the paper is organized as follows: ....

3.3 From clinical questions to data analysis

In applied life science, the biological question at hand usually drives the data collection and therefore the statistical challenge to be solved. In order to achieve meaningful results, thorough data analysis protocol must be followed (see Section 4.4). In this section, my goal is to illustrate some of the most recurrent biological questions and how they can be translated into machine learning tasks. **{assist next chapter}**

3.3.1 How to predict phenotypes from observed data?

Starting from a collection of input measures that are likely to be related with some known target phenotype, the final goal here is to learn a model that represents the relationship between input and output. Several researches fall in this class, for instance in molecular (*e.g.* lab tests, gene expression, proteomics, sequencing) [Angermueller et al., 2016; Okser et al., 2014; Abraham et al., 2013] or radiomics/imaging studies (*e.g.* MRI, PET/SPECT, microscopy) [Min et al., 2016; Helmstaedter et al., 2013]. Biological questions of this class are usually tackled by *supervised learning* models. In particular, when the observed clinical outcome is expressed as a one-dimensional continuous value, as in survival analysis, a *single-output regression* problem is posed. Moreover, if the outcome is vector-valued, as in the case of multiple genetic trait prediction [He et al., 2016a], the problem can be cast in a *multiple-output regression* framework [Argyriou et al., 2008; Baldassarre et al., 2012]. Biological studies involving categorical outcomes translate into *classification* problems. In particular, if the clinical outcome assumes only two values, as in the *case-control* scenario, the classification problem is said to be *binary*, whilst, if multiple classes are observed, the classification task becomes *multiclass*.

3.3.2 Which variables are the most significant?

In the above case, a complementary question revolves around the interpretability of the predictive model. In particular, if dealing with high-throughput data, the main goal is to identify a relevant subset of meaningful variables for the observed phenomenon. This problem can be cast into a variable/feature selection problem [Guyon et al., 2002].

A machine learning model is said to be *sparse* when it only contains a small number of non-zero parameters, with respect to the number of features that can be measured on the objects this model represents [Hastie et al., 2015; Meier et al., 2008]. This is closely related to feature selection: if these parameters are weights on the features of the model, then only the features with non-zero weights actually enter the model and can be considered *selected*.

3.3.3 How to stratify the data?

Collecting measures from several samples, the final goal here is to divide them in homogeneous groups, according to some *similarity* criterion. In machine learning, this is usually referred to as *clustering* [Hastie et al., 2009].

3.3.4 How to represent the samples?

In order to formulate a model of some natural phenomenon, it is necessary to design and follow a suitable data collection protocol. A natural question that may arise here is whether the raw collected measures are intrinsically representative of the target phenomenon or if some transformation must be applied in order to achieve a data representation that can be successfully exploited by a learning machine. For instance, it may be plausible to assume that the data lie in a low-dimensional embedding or that they can be better represented by a richer polynomial or Gaussian expansion. A common solution, in this case, is to take advantage of *feature engineering* techniques to obtain hand crafted features. However, this process can be very time-consuming and it may require the help of domain experts. The process of automatically identify suitable representations from the data itself is usually referred to as *(un)supervised feature learning* [Angermueller et al., 2016; Mamoshina et al., 2016].

3.3.5 Are there recurring patterns in the data?

Analyzing data coming from complex domains, one may be interested in understanding whether complex observations can be represented by some combination of simpler events. In machine learning this typically translates into *adaptive sparse coding* or *dictionary learning* problems [Masecchia et al., 2015; Alexandrov et al., 2013].

3.3.6 How to deal with missing values?

Applied life science studies must often deal with the issue of missing data. For instance, peaks can be missed in mass-spectrometry [Jung et al., 2014] or gene expression levels can be impossible to measure due to insufficient array resolution or image corruption [Stekhoven and Bühlmann, 2011; Troyanskaya et al., 2001]. Common strategies, such as discarding the samples with missing entries, or replacing the holes with the mean, median or most represented value, fall short when the missing value rate is high or the number of collected samples is relatively small. In machine learning this task usually translates into a *matrix completion* problem [Candès and Recht, 2009].

4 Machine learning state of the art

This chapter starts defining the concept of machine learning in its two major declinations: supervised and unsupervised learning. It continues providing a comprehensive overview of algorithms, models and techniques relevant for the biomedical data science applications described in Part II. At the end of this chapter, an overview on the computational requirements and the most recent machine learning technologies is given.

The term *Machine Learning* (ML) first appeared in the late 50's in the field of computer science and it is now becoming a buzzword used in several contexts spanning from particle physics and astronomy to medicine and social sciences [Service, 2017]. With a simple search on Google Trends[1] it is possible to roughly quantify the pervasiveness of this term on the Internet in the last few years. From Figure 4.1 we can see that the interest toward both the terms *machine learning* and *data science* is growing, with the first consistently superior to the second.

A partial explanation to this phenomenon can be found in a recent article published on Science [Appenzeller, 2017], where the authors observed how the explosion of modern data collection abilities is leading the human kind toward another *scientific revolution*. Biomedical applications are prototypical in this sense. For instance, the volume of the raw data acquired from a genome sequencer for a single DNA has a volume of approximately 140 GB [Marx, 2013]. Another example can be the 3D reconstruction of cardiac MRI acquisition which needs around 20 GB for a single human heart, or the 3D CT scan which has a volume in the order of GB for each patient, and so on. It has been estimated that an average hospital currently stores more than 665 TB of data that needs to be analyzed and understood. Such massive amounts of data have long overwhelmed human analysis and insights potential. This makes ML a key element for clinicians and scientists that try to make sense of large-scale observations.

But, what is *machine learning*? And how does it differ from classical statistics?

A unique answer to this question is not easy to provide. In fact, ML can be defined in different ways and from several standpoints. Let's see three remarkable examples.

1. Kevin P. Murphy in its *Machine Learning - A Probabilistic Perspective* [Murphy, 2012] defines machine learning as follows.

[1] https://trends.google.com

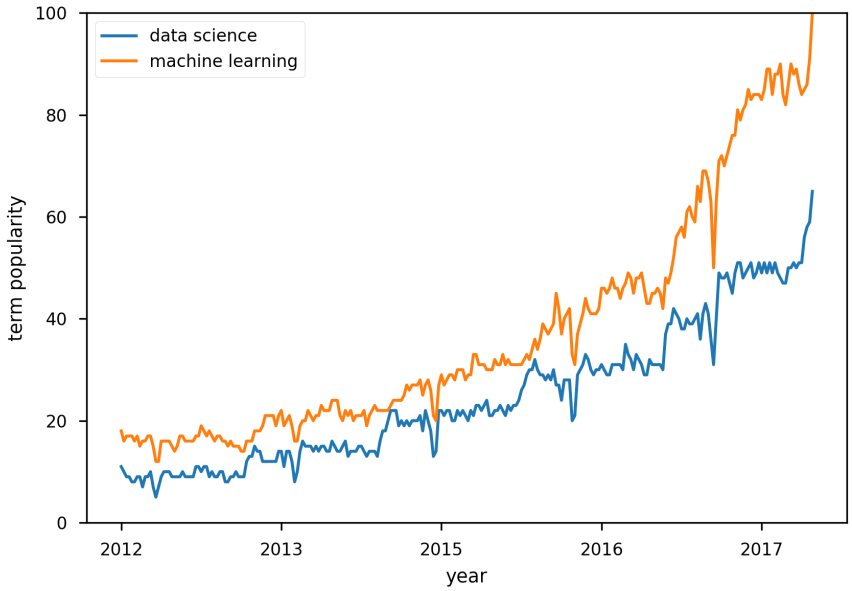


Figure 4.1: The Internet popularity over the past five years of two terms: *data science* and *machine learning*. The vertical axis represents the number of Google searches of an input term normalized with respect to its maximum (source: Google Trends).

> "[. . .] *a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty* [. . .]"

2. Trevor Hastie, a well-known applied statistician, in a famous seminar[2], held in October 2015 at the Stanford University, gave the following three definitions.

> **Machine Learning** *constructs algorithms that can learn from data.*
>
> **Statistical Learning** *is a branch of applied statistics that emerged in response to machine learning, emphasizing statistical models and assessment of uncertainty.*
>
> **Data Science** *is the extraction of knowledge from data, using ideas from mathematics, statistics, machine learning, computer science, engineering...*

3. Carl E. Rasmussen in the preface of its renowned *Gaussian Processes for Machine Learning* [Rasmussen and Williams, 2006] introduces the difference between classical statistics and ML as follows.

[2]part of Data Science @ Stanford Seminar series (source: https://goo.gl/UFgqxU)

> "[...] *in statistics a prime focus is often in understanding the data and relationships in terms of models giving approximate summaries such as linear relations or independencies. In contrast, the goals in machine learning are primarily to make predictions as accurately as possible and to understand the behaviour of learning algorithms* [...]"

It looks like each author, according to his background, expertise and experience, provides a slightly different definition of ML. Trying to summarize these three standpoints, we can say that *ML is an interdisciplinary field that borrows the concept of data-driven model from statistics in order to devise algorithms that can exploit hidden patterns in current data and make accurate predictions on future data*.

As of today ML is the workhorse of data science.

4.1 Supervised learning

Humans are remarkably good at *learning by examples*. When a kid is taught what a pencil looks like, he will be capable of understanding the concept of pencil from a limited number of guided observations. Similarly, when future radiologists are trained to distinguish between healthy tissues from tumors in MRI scans, they will be provided with several annotated biomedical images from which they will be able to generalize. The applied learning paradigm is characterized by the presence of two key objects: *data* and *labels*. In the last example, the MRI scans are the data, and their annotations (*e.g.* tumor *vs* healthy tissue) are the labels.

Supervised learning is the branch of ML in which predictive models are trained on labeled data. In the ML jargon, and in this thesis, one usually refers to *data* as collections of *samples* described by an arbitrarily large number of *predictors* (*features*) that are used as *input* in a training process having labels as *output*.

Input samples throughout this thesis are represented as d-dimensional vectors $\boldsymbol{x}$ belonging to an input space $\mathcal{X}$, where typically $\mathcal{X} \subseteq \mathbb{R}^d$ and labels are represented with the variable y belonging to an output space $\mathcal{Y}$. The nature of $\mathcal{Y}$ defines the learning task as *binary classification* if $\mathcal{Y} = \{-1, +1\}$, *multiclass classification* if $\mathcal{Y} = \{1, 2, \ldots, k\}$, *regression* if $\mathcal{Y} \subseteq \mathbb{R}$ and *vector-valued regression* if $\mathcal{Y} \subseteq \mathbb{R}^k$. The remainder of this section summarizes the methods that are most relevant with the data-driven strategies adopted to tackle the biomedical data science challenges described in the second part of in this thesis.

Given a set of input-output pairs $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n = (X, \boldsymbol{y})$, supervised learning methods aim at finding a function of the inputs $f(\boldsymbol{x})$ that approximates the output y. This translates into the minimization problem defined in Equation (4.1).

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \tag{4.1}$$

The loss function $L(f(\boldsymbol{x}), y)$ can be seen as a measure of *adherence* to the available training data. Several loss functions for regression and classification problems were proposed; Table 4.1 defines the most commonly adopted in biomedical studies while their visual representation is presented in Figure 4.2. Choosing the appropriate loss function for the problem at hand is crucial and there is no trivial solution for this problem. Different choices for $L(f(\boldsymbol{x}), y)$ identifies different learning machines, that are known under different names. The most popular methods will be presented in the next few sections.

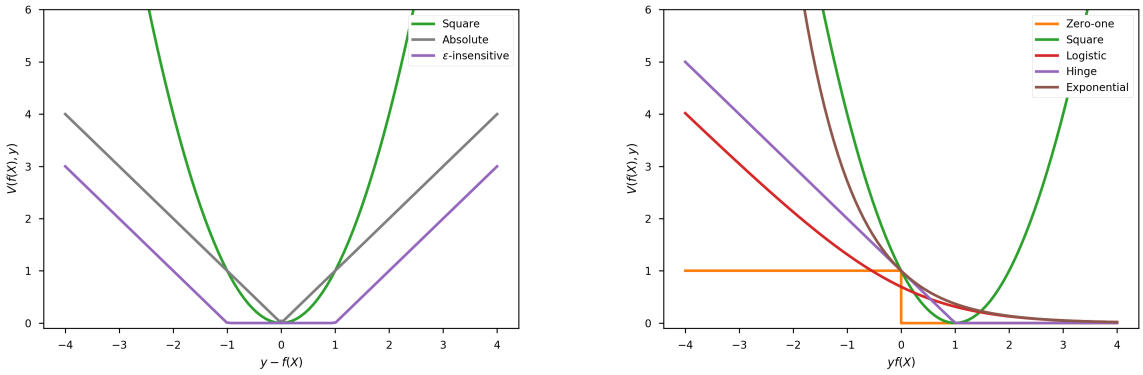
Table 4.1: Definition of the loss functions for regression (top) and classification (bottom) problems represented in Figure 4.2.

Loss function	$L(f(\boldsymbol{x}), y)$	Learning problem
Square	$(y - f(\boldsymbol{x}))^2$	regression
Absolute	$\lvert y - f(\boldsymbol{x})\rvert$	regression
ε-insensitive	$\lvert y - f(\boldsymbol{x})\rvert_\varepsilon$	regression
Zero-one	$\mathbb{1}\{y = f(\boldsymbol{x})\}$	classification
Square	$(1 - yf(\boldsymbol{x}))^2$	classification
Logistic	$\log(1 + e^{-yf(\boldsymbol{x})})$	classification
Hinge	$\lvert 1 - yf(\boldsymbol{x})\rvert_+$	classification
Exponential	$e^{-yf(\boldsymbol{x})}$	classification

Identifying a reliable data-driven model can be a very tricky task. Many unwanted and concurrent factors may be misleading and the solution may have poor predictive power for several reasons. Including:

1. the acquisition devices may introduce random fluctuations in the measures;
2. the amount of collected samples n may be small with respect to the number of observed variables d;
3. a non-negligible number of the measured variables may not be representative of the target phenomenon.

From a modeling standpoint, every combination of the factors above can be seen as *noise* affecting the data. Precautions in the model formulation process must be taken in order to achieve solutions that are insensitive to small changes in the input data and that are, in general, *robust* to the noise effect.



(a) (b)

Figure 4.2: An overview on the most common loss functions for regression (a) and classification (b) problems plotted against the corresponding prediction error.

Considering a ML model ($\hat{f}$) fitted on a collection of data ($\mathcal{D}$), the most desirable property of $\hat{f}$ is that it should be able to achieve good prediction performance not only on $\mathcal{D}$, but also on all the future, therefore unseen, data points $\mathcal{D}'$. In other words, assuming that the samples in $\mathcal{D}$ are affected by some kind of random[3] component, $\hat{f}$ should be a predictive function that does not *follow the noise*, but rather models the true input-output relationship. In ML, a model that fits well $\mathcal{D}$ but performs poorly on $\mathcal{D}'$ is said to be *overfitting*.

In ML literature, *regularization* is the most important countermeasure to overfitting and it is widely adopted, under several forms, to build predictive models out of noisy data.

The original contribution of this PhD thesis mainly relies on the application of data science and ML concepts to noisy domains. Therefore, regularization strategies are of primary interest in this discussion. For each learning algorithm described, particular emphasis will be put on the relevant regularization strategies.

In its broader definition *regularization* can be seen as the process of introducing additional information in order to solve a possibly ill-posed problem. As shown in Equation (4.2), this is typically translated in the use of a regularization penalty $\mathcal{R}(f)$, controlled by a regularization parameter λ [Tikhonov, 1963; Evgeniou et al., 2000].

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) + \lambda \mathcal{R}(f) \tag{4.2}$$

Choosing different $\mathcal{R}(f)$ implies inducing different effects on the solution and it also leads to the

[3]here with *random* means "*uncorrelated with the input-output relationship*"

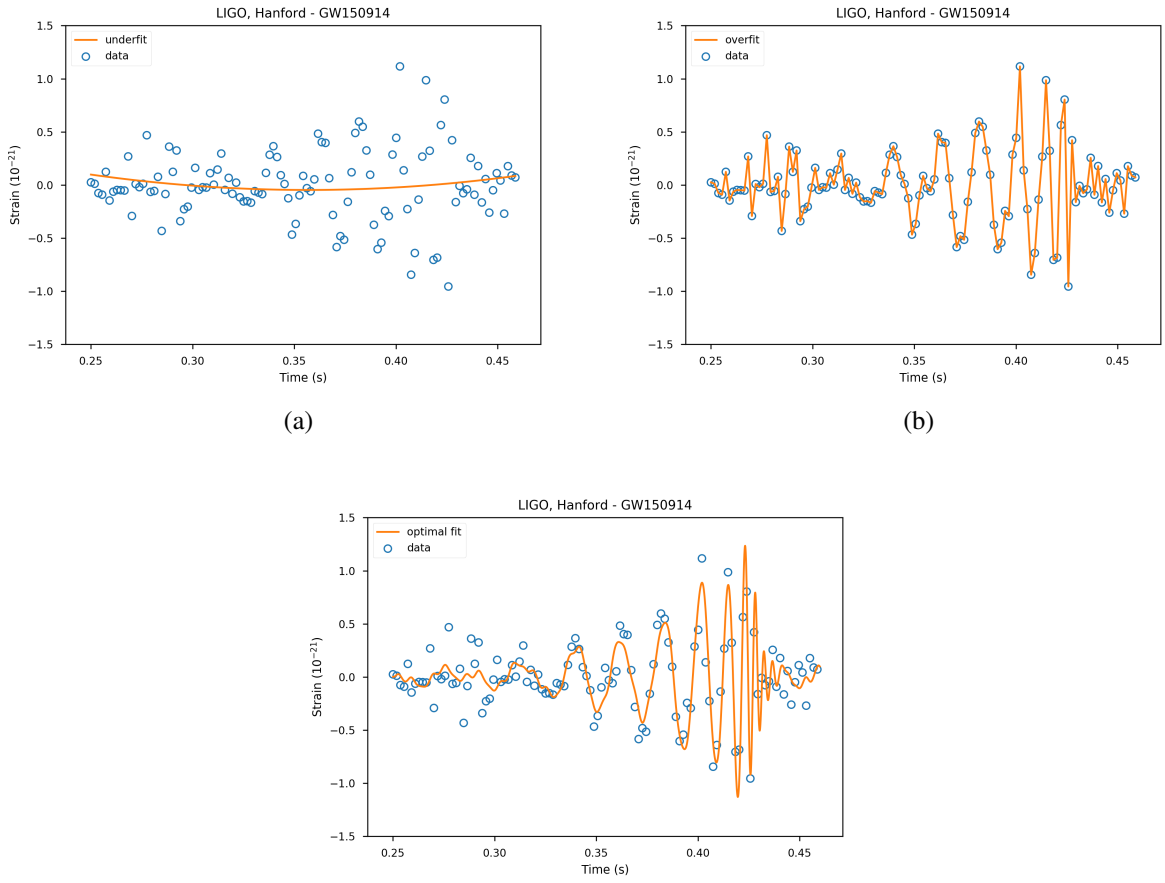


Figure 4.3: An example of underfit (a), overfit (b) and optimal fit (c) for a nonlinear regression problem. The data are a downsampled version ($f_s = 546\ Hz$) of the first observation of gravitational waves from a binary black hole merger detected on September 14$^{\text{th}}$, 2015, 09:50:45 UTC at LIGO Hanford (WA).

definition of different learning machines (see Section 4.1.1). With the regularization parameter λ it is possible to control the trade-off between adherence to the training data and strength of the effect induced by $\mathcal{R}(f)$. As an example, we can think of using a penalty that induces smoothness, such as the ℓ_2-norm, or sparsity, such as the ℓ_1-norm, in the solution. A pictorial representation of a learning machine working in overfitting, underfitting and optimal fitting regime in a regression and a classification case can be seen in Figure 4.3 [4] and Figure 4.4 [5], respectively.

[4] source https://losc.ligo.org/events/GW150914/

[5] source https://archive.ics.uci.edu/ml/datasets/HTRU2

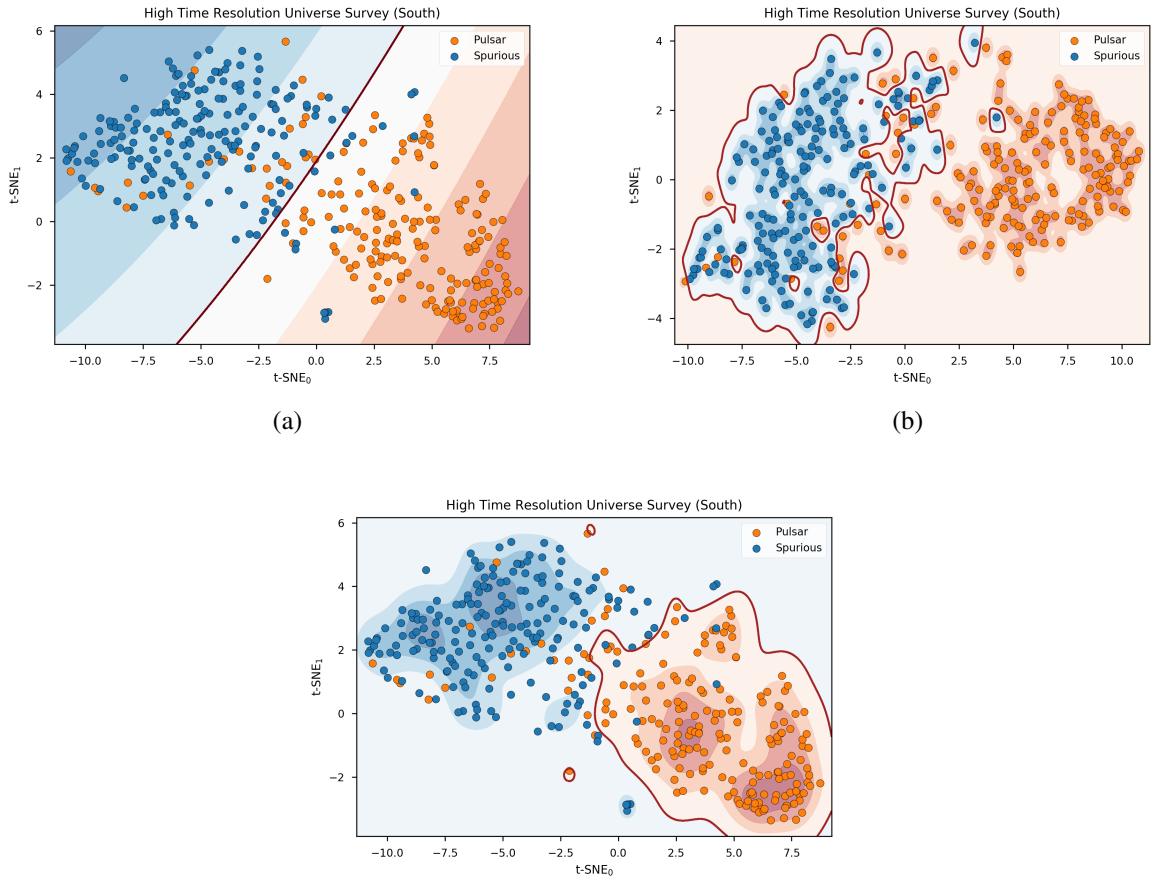


Figure 4.4: An example of underfit (a), overfit (b) and optimal fit (c) for a nonlinear binary classification problem. Each data point is a pulsar candidate randomly sampled from the High Time Resolution Universe Survey (South) dataset. The data are standardized and their dimensionality is reduced by the t-SNE algorithm [Van der Maaten and Hinton, 2008].

Supervised learning machines may rely on very different mathematical backgrounds such as generalized linear models, nonlinear deep neural networks, kernels, trees, ensemble of trees, *etc*. Nevertheless, disregarding their nature, they all share the common structure defined in Equation (4.2). The solution of this problem can be achieved either by Empirical (or Structured) Risk Minimization (ERM) either by Maximum Likelihood/A Posteriori (MLE/MAP) Estimation. See Appendix A for more details on this two strategies, and their connection.

4.1.1 Regularization methods

Regularization methods is a broad class of models that include linear and nonlinear techniques for both regression and classification. The main characteristic of the methods falling in this class, is that they are particularly straightforward to express as in Equation (4.2). In fact, as described in [Evgeniou et al., 2000], they are easily identifiable by the use of one loss function $L(f(\boldsymbol{x}), y)$ and one, or more, regularization penalty $\mathcal{R}(f)$.

In the following sections an overview of the most popular regularization methods is presented.

4.1.1.1 Ordinary least squares

We start this discussion focusing on linear models $\hat{y} = f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{w}$ and taking into account the most popular loss function for regression problems: the square loss $L(\hat{y}, y) = (\boldsymbol{x}^T\boldsymbol{w} - y)^2$. The data fitting problem expressed in Equation (4.3) is known as *Ordinary Least Squares* (OLS), or simply as *linear regression*, and it does not include any regularization term.

$$\hat{\boldsymbol{w}}_{\text{OLS}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)^2 = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\left\|X\boldsymbol{w} - \boldsymbol{y}\right\|_2^2 \tag{4.3}$$

The minimization problem in Equation (4.3) is convex and differentiable, hence its solution can be achieved in closed-form as

$$\hat{\boldsymbol{w}}_{\text{OLS}} = (X^TX)^{-1}X^T\boldsymbol{y}$$

or by iterative minimization routines such as (stochastic) gradient descent-like algorithms [Boyd and Vandenberghe, 2004; Sra et al., 2012]. A pictorial representation of the solution of OLS can be seen in Figure 4.12a.

In case of multiple regression tasks, the OLS approach can be extended to vector-valued regression as well. In this case the least squares problem can be written as

$$\begin{aligned}\hat{W}_{\text{OLS}} = &\underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{k}(\boldsymbol{x}_i^T\boldsymbol{w}^t - y_i^t)^2 \\ &= \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n}\left\|XW - Y\right\|_F^2\end{aligned} \tag{4.4}$$

where $\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{t=1}^{k}|a_i^t|^2}$ is the *Frobenius* norm (also known as *Hilbert-Schmidt* norm) and it can be considered as an extension of the ℓ_2-norm to the matrix case. Lacking of appropriate regularization penalties, solving the problem in Equation (4.4) corresponds to solving k isolated regression problems, one for each task. The vector-valued OLS approach, even if theoretically

legit, is rather uncommon in practical applications and a regularized version of Equation (4.4) is typically preferred (see following sections).

Even though mainly used for regression, the square loss can also be used to solve binary classification problems (see Table 4.1). In this case, it can be rewritten as

$$(\boldsymbol{x}^T\boldsymbol{w} - y)^2 = (1 - y \cdot \boldsymbol{x}^T\boldsymbol{w})^2 \tag{4.5}$$

exploiting the fact that the two classes are encoded with binary labels: $\boldsymbol{y} \in \{+1, -1\}^n$. For multiclass classification problems, strategies such as *One-vs-One* (OVO) or *One-vs-All* (OVA) can be adopted to reduce the problem to multiple binary classifications [Hastie et al., 2009].

OLS is probably the most naïve prediction strategy, nevertheless it is widely adopted in several studies. Let's see what happens when we use the OLS model on a real regression problem.

For this example, and the following ones, we take into account the dataset $\mathcal{D}_{\text{aging}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n=111}$ where each input sample $\boldsymbol{x}_i \in \mathbb{R}^{12}$ presents a set of measures describing the metabolic state of a healthy subject and $y_i \in \mathbb{N}_+$ is its age expressed in years. For the sake of this discussion a thorough description of $\mathcal{D}_{\text{aging}}$ at this point is irrelevant[6], we can simply think as the $d = 12$ variables as predictors of the outcome y and we look for some linear input-output relationship. In order to do that, we randomly split $\mathcal{D}_{\text{aging}}$ in two chunks obtaining a training and a test set of $n_{\text{tr}} = 74$ and $n_{\text{ts}} = 37$ samples respectively. Then, we fit the OLS model on the training set obtaining the weights vector $\hat{\boldsymbol{w}}_{\text{OLS}}$ represented in Figure 4.5. As we can see, in order to achieve a predictive model, OLS can only spread the weights across all the input variables. Evaluating $\hat{\boldsymbol{w}}_{\text{OLS}}$ on the test set, this model has a Mean Absolute Error (MAE) of 10.598 years and explains the 74.29% of the variance.

This result looks promising, but we will see in the next sections whether they can be improved with the use of some regularization penalty.

4.1.1.2 Ridge regression

In its original proposition, *ridge regression* [Hoerl and Kennard, 1970] is defined as a least squares problem penalized by the squared ℓ_2-norm of the regression coefficients, see Equation (4.6).

$$\mathcal{R}_{\ell_2}(\boldsymbol{w}) = \sum_{j=1}^{d} (w_j)^2 = \|\boldsymbol{w}\|_2^2 \tag{4.6}$$

Therefore, the ridge regression minimization problem can be written as in Equation (4.7).

[6] this regression problem is widely described and analyzed in Chapter 6.

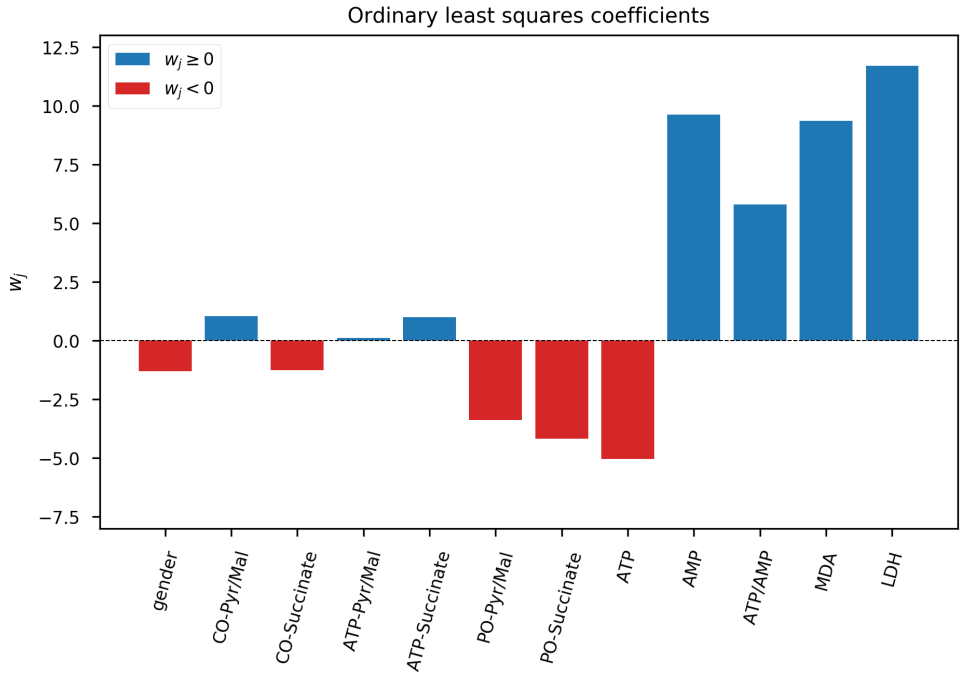


Figure 4.5: A pictorial representation of the vector $\hat{\boldsymbol{w}}_{\text{OLS}}$ obtained fitting an OLS model on 74 randomly selected training samples of $\mathcal{D}_{\text{aging}}$. Variables associated with positive (*i.e.* directly proportional to the output) and a negative (*i.e.* inversely proportional) weight are represented in blue and red, respectively.

$$\hat{\boldsymbol{w}}_{\ell_2} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)^2 + \lambda\sum_{j=1}^{d}(w_j)^2 = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\left\|X\boldsymbol{w} - \boldsymbol{y}\right\|_2^2 + \lambda\left\|\boldsymbol{w}\right\|_2^2 \tag{4.7}$$

This penalty leads to smooth solutions as it shrinks the coefficients toward zero, but it does not achieve a parsimonious representation, as it always keep all the variables in the model. The ridge regression problem of Equation (4.7) is convex and differentiable and a pictorial representation of its solution in a 2D case is depicted in Figure 4.12b. The ridge coefficients $\hat{\boldsymbol{w}}_{\ell_2}$ can be estimated in closed-form as

$$\hat{\boldsymbol{w}}_{\ell_2} = (X^TX + \lambda I)^{-1}X^T\boldsymbol{y}$$

where I is the $d \times d$ identity matrix. An estimate for the ridge coefficients can also be obtained with gradient descent-like optimization routines [Boyd and Vandenberghe, 2004; Sra et al., 2012].

The regularization parameter λ plays the fundamental role of balancing the trade-off between data adherence and smoothness of the solution. Penalizing the ℓ_2-norm of the regression coefficients,

their value is shrunk toward zero. This results in an increased robustness of the solution to the noise affecting the training data.

In case of multiple outputs, the vector-valued ridge regression problem can be written as in Equation (4.8).

$$\begin{aligned}\hat{W}_{\ell_2} &= \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{k}(\boldsymbol{x}_i^T\boldsymbol{w}^t - y_i^t)^2 + \lambda\sum_{j=1}^{d}\sum_{t=1}^{k}|w_j^t|^2 \\ &= \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n}\|XW - Y\|_F^2 + \lambda\|W\|_F^2\end{aligned} \tag{4.8}$$

As already seen for vector-valued OLS, the Frobenius norm penalty does not induce any task coupling, hence solving the problem in Equation (4.8) still corresponds to individually solve k regression tasks.

This method can be applied to binary classification problems by using the margin loss function of Equation (4.5). Nevertheless, for ℓ_2-norm penalized classification problems the use of the *logistic loss* is usually preferred (see Section 4.1.1.5).

In deep learning literature, penalizing the regression coefficients with the ℓ_2-norm is known as *weight decay* [Krogh and Hertz, 1992]. Ridge regression can also be considered a form of *Tikhonov regularization* [Tikhonov, 1963] and of *regularization network* [Evgeniou et al., 2000].

Let's see what happens when this method is applied to a real regression problem. For ease of comparison, we take into account the dataset $\mathcal{D}_{\text{aging}}$, introduced in Section 4.1.1.1. Compared to OLS, ridge regression has the parameter λ that must be fixed before fitting the model. In this example, we estimated the best value $\hat{\lambda}_{\text{cv}}$ according to a standard grid-search cross-validation strategy [Hastie et al., 2009]. This consists in fixing a range of (30) possible values for λ (in a logarithmic scale from 10^{-3} to 10^2) and pick the best value as the one achieving the lowest validation error, estimated via (5-fold) cross-validation. Therefore, once the best value for the regularization parameter is fixed ($\hat{\lambda}_{\text{cv}} = 20.43$), the experimental setup used for OLS is preserved. The ridge coefficients $\hat{\boldsymbol{w}}_{\ell_2}$ are represented in Figure 4.6. Comparing Figure 4.6 and Figure 4.5 we can see that the amplitude of the ridge regression coefficients is, in absolute value, decreased by the use of the ℓ_2 penalty. Indeed, several entries of $\hat{\boldsymbol{w}}_{\ell_2}$ are very small, but none of them is exactly zero. This is the expected behavior of the ℓ_2-norm penalty. Evaluating $\hat{\boldsymbol{w}}_{\ell_2}$ on the test set, this model has MAE $= 8.615$ years and explains the 81.19% of the variance, outperforming OLS.

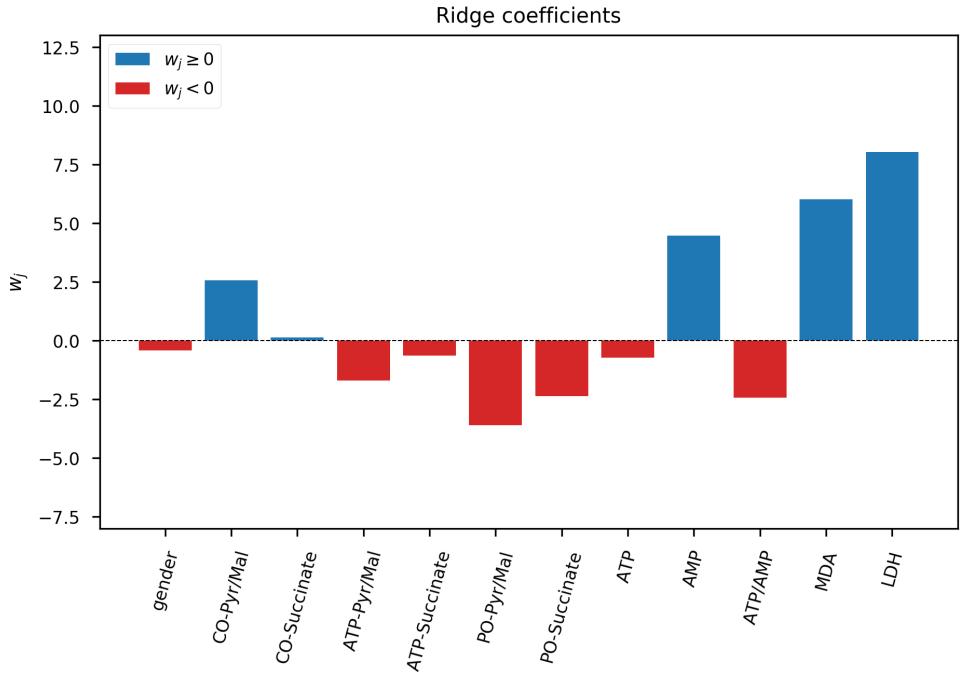


Figure 4.6: A pictorial representation of the vector $\hat{\boldsymbol{w}}_{\ell_2}$ obtained fitting a ridge regression model on 74 randomly selected training samples of $\mathcal{D}_{\text{aging}}$. Variables associated with positive (*i.e.* directly proportional to the output) and a negative (*i.e.* inversely proportional) weight are represented in blue and red, respectively.

4.1.1.3 Lasso

The Lasso [Tibshirani, 1996] can be defined as a least square problem penalized by the ℓ_1-norm of the regression coefficients, see Equation (4.9).

$$\mathcal{R}_{\ell_1}(\boldsymbol{w}) = \sum_{j=1}^{d} |w_j| = |\boldsymbol{w}|_1 \tag{4.9}$$

Therefore, the Lasso minimization problem can be written as in Equation (4.10).

$$\hat{\boldsymbol{w}}_{\ell_1} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i^T \boldsymbol{w} - y_i)^2 + \lambda \sum_{j=1}^{d} |w_j| = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda |\boldsymbol{w}|_1 \tag{4.10}$$

The Lasso can be used to perform linear model fitting and, thanks to its desirable properties, it is a popular choice for embedded variable selection [Guyon and Elisseeff, 2003], see Section 4.3. At first, the ℓ_1-norm enforces sparsity in the solution, hence producing compact and

easily interpretable results. Secondly, the Lasso optimization problem is convex and, although non-differentiable, it is computationally feasible even in very high dimensional scenarios. Popular minimization algorithms for the Lasso problem are, for instance, the *Fast Iterative Shrinkage-Thresholding Algorithm* (FISTA) [Beck and Teboulle, 2009] and the *coordinate descent algorithm* [Wu and Lange, 2008]. A pictorial representation of the Lasso solution can be seen in Figure 4.12c.

A popular application of the Lasso is to perform shrinkage and variable selection in survival analysis for Cox proportional hazard regression [Tang et al., 2017; Gui and Li, 2005; Tibshirani et al., 1997] and additive risk models [Ma and Huang, 2007]. Such ℓ_1-penalized methods are extensively applied in literature to predict survival time from molecular data collected from patients affected by different kinds of tumor.

The Lasso can also be extended to vector-valued regression problems by using the mixed $L_{2,1}$-norm, defined in Equation (4.11), as regularization penalty [Gramfort et al., 2012].

$$\mathcal{R}_{\ell_1}(W) = \sum_{j=1}^{d} \sqrt{\left(\sum_{t=1}^{k} |w_j^t|^2 \right)} = \|W\|_{2,1} \tag{4.11}$$

Therefore, the vector-valued regularization problem can be written as in Equation (4.12)

$$\begin{aligned} \hat{W}_{\ell_1} = & \operatorname*{argmin}_{W \in \mathbb{R}^{d \times k}} \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{k} (\boldsymbol{x}_i^T \boldsymbol{w}^t - y_i^t)^2 + \lambda \sum_{j=1}^{d} \sqrt{\left(\sum_{t=1}^{k} |w_j^t|^2 \right)} \\ & = \operatorname*{argmin}_{W \in \mathbb{R}^{d \times k}} \frac{1}{n} \|XW - Y\|_F^2 + \lambda ||W||_{2,1} \end{aligned} \tag{4.12}$$

and it is known as Multi-task Lasso [Lee et al., 2010]. Such norm enforces a row-structured sparsity in the regression weights, hence preserving the interpretability of the solution.

Originally proposed to solve regression problems, the Lasso can also be adopted in binary classification tasks; although, in this case, *sparse logistic regression* (see Section (4.1.1.5)) is often preferred [Wu et al., 2009].

Let's see what happens when the Lasso model is applied to a real regression problem. Once again, for ease of comparison with OLS and ridge, we take into account the dataset $\mathcal{D}_{\text{aging}}$, introduced in Section 4.1.1.1. The experimental setup in this case is identical to the one previously applied for ridge regression. The best value for the regularization parameter, chosen via grid-search cross-valudation, is $\hat{\lambda}_{\text{cv}} = 1.27$. The Lasso coefficients $\hat{\boldsymbol{w}}_{\ell_1}$ are represented in Figure 4.7. Comparing $\hat{\boldsymbol{w}}_{\ell_1}$ with $\hat{\boldsymbol{w}}_{\ell_2}$ and $\hat{\boldsymbol{w}}_{\text{OLS}}$ (Figure 4.7, Figure 4.6 and Figure 4.5, respectively) we can observe that, for the first time, to only 6 variables, out of 12, a non-negative value is assigned. This is an example of the *sparsity*-enforcing effect of the ℓ_1-norm regularization penalty. The 6 variables with nonzero weight can be considered as *selected* for the prediction problem at hand. Evaluating

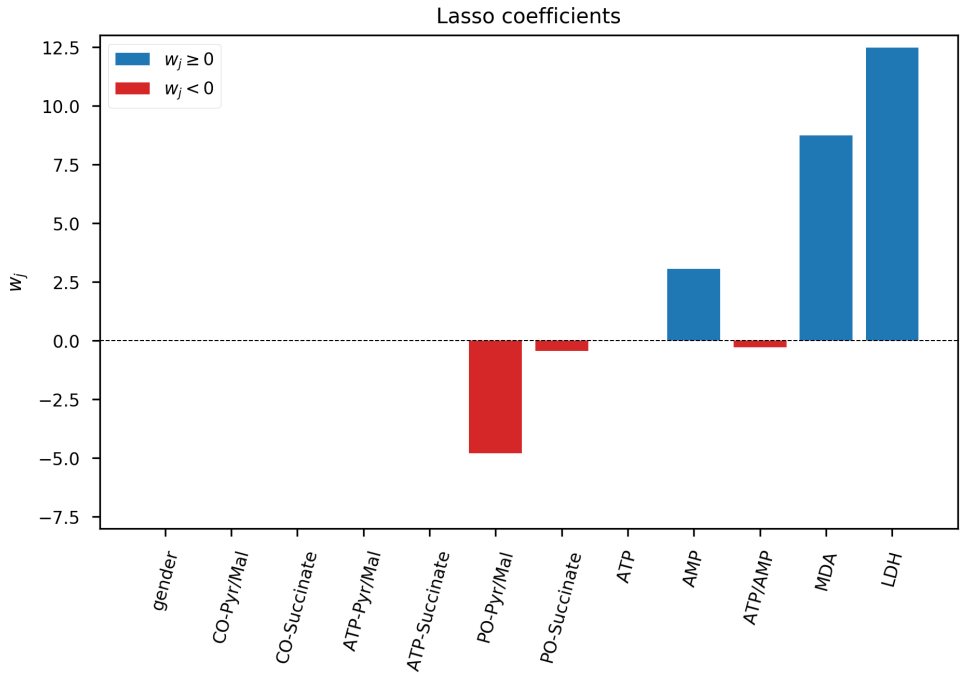


Figure 4.7: A pictorial representation of the vector $\hat{\boldsymbol{w}}_{\ell_1}$ obtained fitting a Ridge model on 74 randomly selected training samples of $\mathcal{D}_{\text{aging}}$. Variables associated with positive (*i.e.* directly proportional to the output) and a negative (*i.e.* inversely proportional) weight are represented in blue and red, respectively.

$\hat{\boldsymbol{w}}_{\ell_1}$ on the test set, the Lasso has MAE $= 8.387$ years and explains the 81.51% of the variance, slightly outperforming ridge.

Using sparsity-enforcing penalties, such as the ℓ_1-norm, an insightful experiment is the analysis of the regularization path. This can be done by iteratively fitting the model with decreasing values of the regularization parameter λ, usually expressed in logarithmic scale. An example of the Lasso path for the aging problem is reported in Figure 4.8. Weights corresponding to features that are more likely to be relevant for the prediction problem should early move away from the horizontal axis. Conversely, as the regularization parameter increases, the model should enforce less sparsity, hence tolerating more and more irrelevant features having nonzero weight. The vertical dashed line in Figure 4.8 corresponds to $\hat{\lambda}_{\text{cv}}$ and it hits the profiles of the weights consistently with what shown in Figure 4.7.

When used for variable selection, the Lasso has two major drawbacks. First, in presence of groups of correlated variables, this method tends to select only one variable per group, ignoring the others. Secondly, the method cannot select more variables than the sample size [Waldmann et al., 2013; De Mol et al., 2009b]. The effect of such drawbacks is dramatic when using the

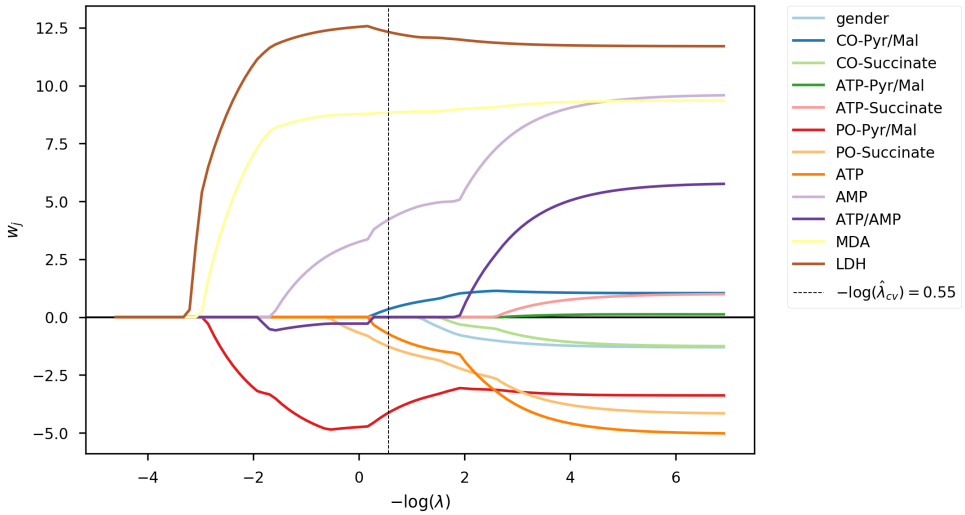


Figure 4.8: Profiles of the Lasso coefficients for the aging problem as λ decreases. The vertical dashed line represents the optimal value $\hat{\lambda}_{\text{cv}}$ estimated by grid-search (5-fold) cross-validation.

Lasso in $n \ll d$ scenarios. In order to ameliorate this issues, several Lasso-inspired models were proposed [Meinshausen and Bühlmann, 2010; Hoggart et al., 2008; Zou, 2006]. In the next section we will describe one of the most popular and straightforward Lasso extensions: the Elastic-Net [Zou and Hastie, 2005].

4.1.1.4 Elastic-Net

The Elastic-Net [Zou and Hastie, 2005; De Mol et al., 2009a] method can be formulated as a least squares problem penalized by a convex combination of Lasso (ℓ_1-norm) and ridge regression (ℓ_2-norm) penalties, as in Equation (4.13).

$$R_{\ell_1\ell_2}(\boldsymbol{w}) = \sum_{j=1}^{d} (\alpha\,|w_j| + (1-\alpha)\,w_j^2) = \alpha\,|\boldsymbol{w}|_1 + (1-\alpha)\;\|\boldsymbol{w}\|_2^2 \tag{4.13}$$

Therefore, the Elastic-Net minimization problem can be written as in Equation (4.14)

$$\begin{aligned} \hat{\boldsymbol{w}}_{\ell_1\ell_2} &= \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)^2 + \lambda\left[\sum_{j=1}^{d}(\alpha\,|w_j| + (1-\alpha)\,w_j^2)\right] \\ &= \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\,\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\left[\alpha\,|\boldsymbol{w}|_1 + (1-\alpha)\;\|\boldsymbol{w}\|_2^2\right] \end{aligned} \tag{4.14}$$

with $0 \leq \alpha \leq 1$, or equivalently as

$$\hat{\boldsymbol{w}}_{\ell_1\ell_2} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \tau \, |\boldsymbol{w}|_1 + \mu \, \|\boldsymbol{w}\|_2^2 \tag{4.15}$$

where $\tau = \lambda\alpha$ and $\mu = \lambda(1-\alpha)$. The first formulation of the problem, Equation (4.14), is more convenient when we want to control the overall amount of regularization with λ, given a certain amount of sparsity α. In fact, it is easy to see that fitting the Elastic-Net model for $\alpha = 0$ or $\alpha = 1$ is equivalent to solve ridge or Lasso regression, respectively. On the other hand, writing the Elastic-Net problem as in Equation (4.15) is more convenient when we want to separately control the ℓ_1- and ℓ_2-norm penalties, as in [De Mol et al., 2009b].

The Elastic-Net[7] model is widely adopted to perform linear model fitting and variable selection. Indeed, the combined presence of the two norms promotes sparse solutions were groups of correlated variables can be simultaneously selected, hence overcoming the variable selection drawbacks of the Lasso and making the Elastic-Net suitable for variable selection in $n \ll d$ scenarios. As already seen for the Lasso, the minimization problem in Equation (4.14) is convex and non-differentiable, due to the ℓ_1-norm, and it can be efficiently solved either by proximal forward-backward splitting strategies (*e.g.* FISTA [Beck and Teboulle, 2009]), or by coordinate descent [Wu and Lange, 2008]. A pictorial representation of the Elastic-Net solution can be seen in Figure 4.12d.

The Elastic-Net method is successfully applied in several biomedical fields, including gene expression [Hughey and Butte, 2015; De Mol et al., 2009b], genome-wide association studies [Waldmann et al., 2013] and other molecular data [Aben et al., 2016; Hughey and Butte, 2015].

The Elastic-Net can also be extended to vector-valued regression problems by using a convex combination of the $L_{2,1}$- and the Frobenius norms. This multiple output regression problem is known as Multi-task Elastic-Net [Chen et al., 2012] can be written as in Equation (4.16).

$$\begin{aligned}\hat{W}_{\ell_1\ell_2} &= \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n}\sum_{t=1}^{k} (\boldsymbol{x}_i^T \boldsymbol{w}^t - y_i^t)^2 + \lambda\left[\alpha \sum_{j=1}^{d} \sqrt{\left(\sum_{t=1}^{k} |w_j^t|^2\right)} + (1-\alpha) \sum_{j=1}^{d}\sum_{t=1}^{k} |w_j^t|^2\right] \\ &= \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n} \|XW - Y\|_F^2 + \lambda[\alpha \, ||W||_{2,1} + (1-\alpha) \, \|W\|_F^2]\end{aligned} \tag{4.16}$$

Even though originally proposed to solve regression problems, as already seen for Lasso and ridge regression, the Elastic-Net can be adopted in binary classification tasks. Nevertheless, for classification problems, the use of the logistic loss is usually preferred.

[7] actually, in the original paper the authors refer to Equations (4.14) and (4.15) as *naïve* Elastic-Net [Zou and Hastie, 2005] because empirical evidence show that the weights $\hat{\boldsymbol{w}}_{\ell_1\ell_2}$ may suffer from over-shrinking when appropriate rescaling strategies, also described in [De Mol et al., 2009b], are not applied.

Let's see what happens when the Elastic-Net model is applied to an actual regression problem. As usually, we tackle the aging task (introduced in Section 4.1.1.1). The experimental setup in this case is the same already adopted for the Lasso in Section 4.1.1.3. The only difference is that the grid-search cross-validation routine looks for the optimal values for the two regularization parameters $(\hat{\lambda}_{\text{cv}}, \hat{\alpha}_{\text{cv}}) = (0.57, 0.48)$ in a 2D grid consisting of 30×30 values (α candidates range from 0 to 1 in a linear scale, whilst λ candidates range from 10^{-3} to 10^{2} in a logarithmic scale). The Elastic-Net coefficients $\hat{\boldsymbol{w}}_{\ell_1\ell_2}$ are represented in Figure 4.9.

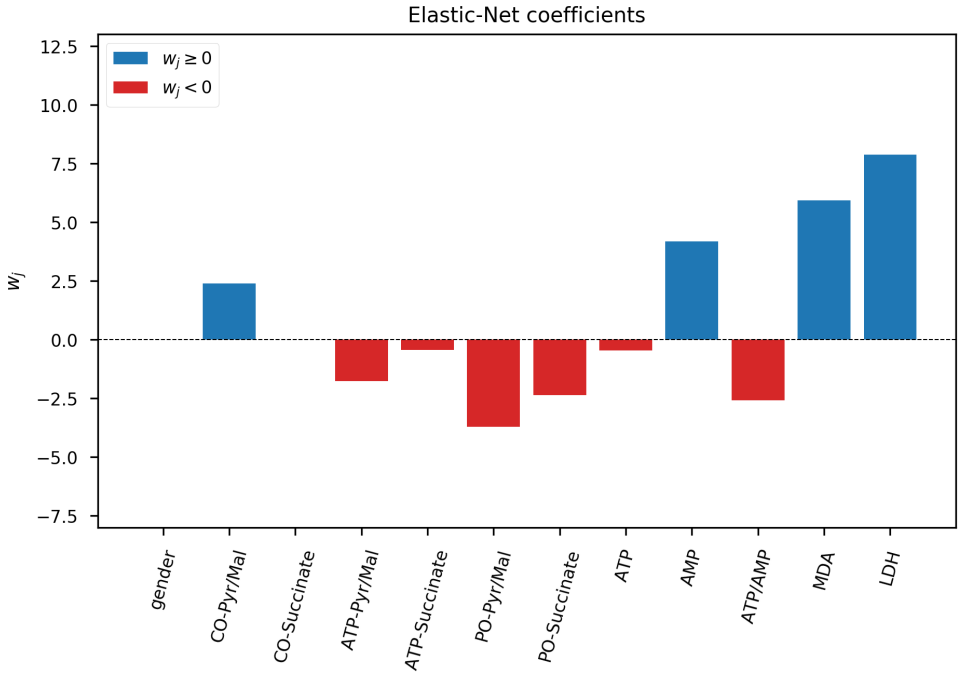


Figure 4.9: A pictorial representation of the vector $\hat{\boldsymbol{w}}_{\ell_1\ell_2}$ obtained fitting a Elastic-Net model on 74 randomly selected training samples of $\mathcal{D}_{\text{aging}}$. Variables associated with positive (*i.e.* directly proportional to the output) and a negative (*i.e.* inversely proportional) weight are represented in blue and red, respectively.

Comparing $\hat{\boldsymbol{w}}_{\ell_1\ell_2}$ in Figure 4.9 with $\hat{\boldsymbol{w}}_{\ell_1}$ in Figure 4.7 we can notice that in the Elastic-Net solution 10 variables have nonzero weight, that is more with respect to the 6 of the Lasso. This is the expected behavior of the added ℓ_2-norm, which helps the model to select groups of collinear variables. Evaluating the Elastic-Net solution on the test set, $\hat{\boldsymbol{w}}_{\ell_1\ell_2}$ achieves MAE $= 8.321$ years explaining the 82.13%, slightly outperforming Lasso, hence ranking first in this little challenge of linear square loss-based regression methods.

As already seen for the Lasso, we can inspect the Elastic-Net weights path, obtained fixing $\alpha = \hat{\alpha}_{\text{cv}}$ for decreasing values of λ, see Figure 4.10. The vertical dashed line in Figure 4.10 corresponds to $\hat{\lambda}_{\text{cv}}$ and it hits the profiles of the weights consistently with the Elastic-Net solution represented in

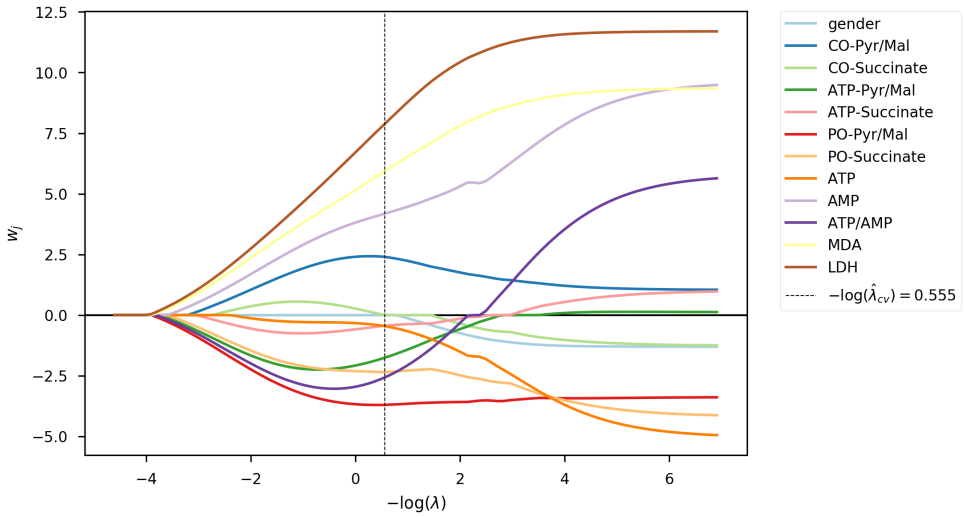


Figure 4.10: Profiles of the Elastic-Net coefficients for the aging problem as λ decreases. The vertical dashed line represents the optimal value $\hat{\lambda}_{\text{cv}}$ estimated by grid-search (5-fold) cross-validation.

Figure 4.7. The Elastic-Net, compared to the Lasso, produces smoother regularization paths in which the allegedly correlated variables enter the solution earlier.

Another comparison between the behavior of OLS, ridge, Lasso and Elastic-Net regression is presented in Figure 4.11. The four solutions achieved for the same aging regression problem are represented in a scatter plot where horizontal and vertical axis represents their ℓ_2- and ℓ_1-norm, respectively. As expected,

1. the unpenalized solution $\hat{\boldsymbol{w}}_{\text{OLS}}$ shows the highest values for the two norms and it is placed in the top-right side of the plot,
2. the ridge solution $\hat{\boldsymbol{w}}_{\ell_2}$ has lowest ℓ_2-norm,
3. the Lasso solution $\hat{\boldsymbol{w}}_{\ell_1}$ has lowest ℓ_1-norm and
4. the Elastic-Net solution shows the lowest values for the two norms and it is placed in the bottom-left side of the plot.

This is consistent with the type of regularization imposed to each method. Interestingly, in this case, the method that performs better on the test set (Elastic-Net) has lowest norms.

The Elastic-Net penalty is not the only method in which sparsity is enforced on a group level. For example, (overlapping) group Lasso and graph Lasso penalties can be applied when the variables are naturally partitioned in (overlapping) groups, or when their interaction can be modeled by a

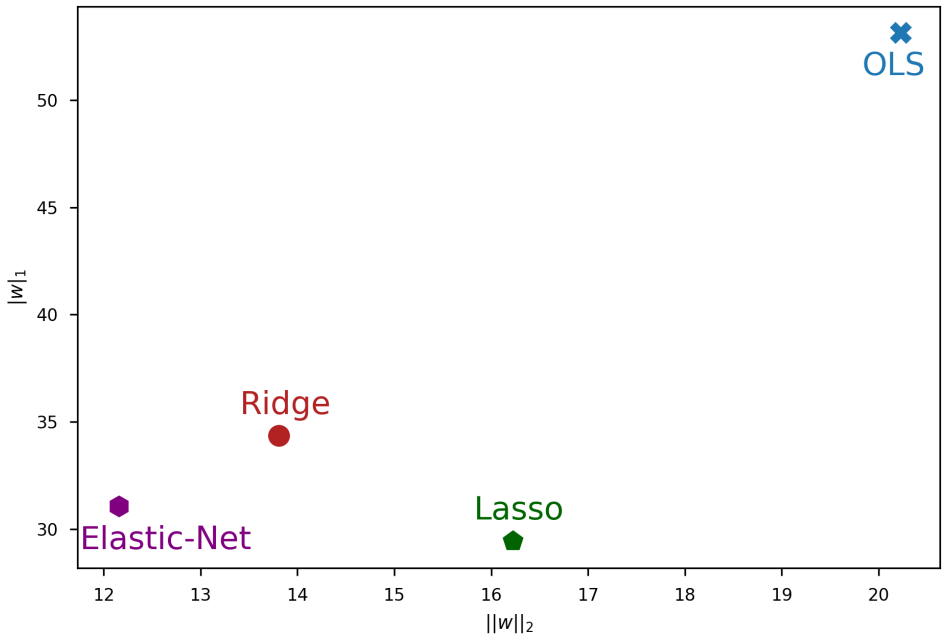


Figure 4.11: A comparison of the value of the ℓ_1 and ℓ_2 norms of the weights obtained by OLS, ridge, Lasso and Elastic-Net.

graph. A detailed description of these methods is beyond the scope of this thesis and we refer to [Jacob et al., 2009; Witten and Tibshirani, 2009] and references therein for their comprehensive description.

4.1.1.5 Logistic Regression

Logistic regression is one of the most popular linear methods for classification problems[8]. In its unpenalized form, it can be simply seen as the problem of minimizing the logistic loss (see Table 4.1) on a given training dataset. Therefore, logistic regression can be written as in Equation (4.17),

$$\hat{\boldsymbol{w}}_{\text{LR}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}}) \tag{4.17}$$

where the labels $y_i \in \{+1, -1\}, \forall i = 1, \ldots, n$. The minimization problem above is convex and differentiable, although as the gradient is nonlinear in $\boldsymbol{w}$, it does not have a closed-form solution. So, unpenalized logistic regression problems are typically solved by (stochastic) gradient descent-like techniques [Boyd and Vandenberghe, 2004; Sra et al., 2012].

[8] as counter-intuitive as it sounds

Unpenalized logistic regression, although theoretically sound, is somewhat uncommon in recent real-world studies. As already seen for the square loss, regularization penalties are typically used. For instance, we can have ℓ_2-regularized logistic regression,

$$\hat{\boldsymbol{w}}_{\ell_2} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i\, \boldsymbol{x}_i^T \boldsymbol{w}}) + \lambda \left\|\boldsymbol{w}\right\|_2^2 \tag{4.18}$$

or various forms of sparse logistic regression such as

$$\hat{\boldsymbol{w}}_{\ell_1} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i\, \boldsymbol{x}_i^T \boldsymbol{w}}) + \lambda |\boldsymbol{w}|_1 \tag{4.19}$$

which uses the Lasso penalty, or

$$\hat{\boldsymbol{w}}_{\ell_1\ell_2} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i\, \boldsymbol{x}_i^T \boldsymbol{w}}) + \lambda[\alpha |\boldsymbol{w}|_1 + (1-\alpha)\left\|\boldsymbol{w}\right\|_2^2] \tag{4.20}$$

which uses the Elastic-Net penalty. The sparse logistic regression minimization problem can be efficiently solved either by proximal forward-backward splitting strategies (*e.g.* FISTA [Beck and Teboulle, 2009]), or by coordinate descent [Wu and Lange, 2008].

Moreover, multi-class classification with logistic regression can be achieved by OVO, AVA approaches, as well as with the multiclass generalization of logistic regression: *softmax* regression [Hastie et al., 2009] (*aka* multinomial logistic regression), which can be expressed as

$$\hat{W}_{\text{LR}} = \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{t=1}^{k} \mathbb{1}\{y_i = t\} \log\left(\frac{e^{\boldsymbol{x}_i^T \boldsymbol{w}^t}}{\sum_{l=1}^{k} e^{\boldsymbol{x}_i^T \boldsymbol{w}^l}}\right)\right] \tag{4.21}$$

where the indicator function $\mathbb{1}\{\cdot\}$ includes in the functional only the correctly classified samples.

4.1.1.6 Support Vector Machine

Support Vector Machine (SVM) is a class of powerful algorithms that can be written as a penalized model and that can used for both regression (SVR) and classification (SVC) problems [Evgeniou et al., 2000]. In the first case, the adopted loss function is Vapnik's ε-insensitive loss:

$$|y - \boldsymbol{x}^T \boldsymbol{w}|_\varepsilon = \begin{cases} 0 & \text{if } |y - \boldsymbol{x}^T \boldsymbol{w}| < \varepsilon \\ |y - \boldsymbol{x}^T \boldsymbol{w}| - \varepsilon & \text{otherwise} \end{cases} \tag{4.22}$$

and in the second case the Hinge loss:

$$|1 - y\, \boldsymbol{x}^T \boldsymbol{w}|_+ = \max[0, 1 - y\, \boldsymbol{x}^T \boldsymbol{w}] \tag{4.23}$$

as reported in Table 4.1 and shown in Figures 4.4c 4.2b. The standard formulation of SVM is penalized with the ℓ_2-norm [Vapnik, 2013]. Therefore, sticking to linear models, the SVR minimization problem can be written as

$$\hat{\boldsymbol{w}}_{\text{SVR}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} |y_i - f(\boldsymbol{x}_i)|_\varepsilon + \lambda \|\boldsymbol{w}\|_2^2 \tag{4.24}$$

while the SVC minimization problem is

$$\hat{\boldsymbol{w}}_{\text{SVC}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} |1 - y_i f(\boldsymbol{x}_i)|_+ + \lambda \|\boldsymbol{w}\|_2^2 \tag{4.25}$$

where, as usually, λ controls the trade-off between data adherence and smoothing of the solution. Equations (4.24) and (4.25) are known as the *primal* SVM problem. However, in order to generalize the SVM to nonlinear cases (see Section 4.1.2), it is often convenient to transform this minimization problem in its *dual* form. Several algorithms to find the solution of the SVM minimization problem in both primal and dual forms were developed, such as Newton's method or coordinate descent algorithm, see [Smola and Schölkopf, 2004; Shawe-Taylor and Sun, 2011] for an exhaustive review.

The standard formulation of SVM does not cope well with high-dimensional data, as no sparsity-enforcing penalty is adopted. Recently, ℓ_1-norm penalized SVM were proposed as well in [Zhu et al., 2004; Peng et al., 2016].

As for the square loss-based methods, let's apply (ℓ_2-penalized) SVR to the aging problem (introduced in Section 4.1.1.1). The experimental setup in this case is the same already adopted for Lasso and ridge regression (see Sections 4.1.1.3 4.1.1.2). The weights $\hat{\boldsymbol{w}}_{\text{SVR}}$ are represented in Figure 4.13. As expected, none of the is exactly zero and they look similar to ridge coefficients in Figure 4.6, as the two model share the same regularization penalty. Evaluating $\hat{\boldsymbol{w}}_{\text{SVR}}$ on the test set, the SVR model has MAE $= 9.280$ and explains the 77.74% of the variance.

4.1.2 The kernel trick

{images and stuff}

4.1.3 Ensemble methods

The key idea behind ensemble methods is to build a prediction model by aggregating a collection of multiple *base learners* that are trained to solve the same problem [Zhou, 2012].

Bagging is a common ensemble method that consists in fitting multiple models $f_b(\boldsymbol{x})$ for $b = 1, \ldots, B$ each one on a *bootstrap* data set $\{X, \boldsymbol{y}\}_b$ obtained from the training set $\{X, \boldsymbol{y}\}$ by

random sampling with replacement [Hastie et al., 2009]. For each sample $\boldsymbol{x}_i$, the bagging estimate $\hat{f}(\boldsymbol{x}_i)$ is obtained by combining the predictions of the base learners $\hat{f}_b(\boldsymbol{x}_i)$. In particular, in case of classification tasks, the bagged model selects the most predicted class casting a vote among the B base learners.

Boosting is another popular ensemble method that, unlike bagging, performs predictions by sequentially fitting a collection of base learner that cast a weighted vote [Hastie et al., 2009]. At each boosting step, the weight corresponding to samples that were misclassified at the previous iterations increases. Therefore, each successive classifier is somewhat forced to learn the relationships between input and output that were previously missed. From a theoretical standpoint, it would be possible to boost any learning machine, nevertheless boosting methods are mainly used with decision trees as base learners.

4.1.3.1 Random Forests

Decision trees are easily interpretable models that recursively partitions the training data into subsets, based on the test of a single feature value at each split (or node). At each iteration, the feature that yields the best split in terms of a pre-selected metric (Gini impurity, information gain or variance reduction) is chosen to create a new node. Decision trees tend to not perform well in practice, which led to the introduction of random forests in 2001 [Breiman, 2001].

Random forests are ensembles of decision trees, each grown on a bootstrap sample from the training data. To increase robustness to noise and diversity among the trees, each node is split using the best split among a subset of features randomly chosen at that node. The final prediction is made by aggregating the prediction of m trees, either by a majority vote in the case of classification problems, or by averaging predictions in the case of regression problems. Random forests are a *bagging* approach, which works on the assumption that the variance of individual decision trees can be reduced by averaging trees built on many uncorrelated subsamples. By contrast, *boosted decision trees* are made by building an iterative collection of decision trees, trained by giving more importance to training examples that were incorrectly classified by the previous trees. **{Add use cases for boosted decision trees. . . . }**

Random forests can provide several measures of *feature importance*, computed by looking at the increase in prediction error when data for a feature is permuted while all other features remain unchanged. Feature selection based on random forests if most often performed using one of these measures. However, several techniques for applying regularization to random forests have been proposed. These techniques broadly fall under two categories: (1) cost-complexity pruning, which consists in limiting tree depth, resulting in less complex models [Ishwaran et al., 2008; Kulkarni and Sinha, 2012]; and (2) Gini index penalization [Deng and Runger, 2013; Liu et al., 2014a]. In addition, [Joly et al., 2012] proposed using an ℓ_1-norm to reduce the space-complexity of random forests.

Random forests naturally handle both numerical and categorical variables, multiple scales, and non-linearities. They also require little parameter tuning. This makes them popular for the analysis of diverse types of biological data, such as gene expression, GWAS data or mass spectrometry [Qi, 2012]. Unfortunately, in practice feature selection schemes that rely on them tend to be very unstable [Kursa, 2014].

4.1.3.2 Gradient Boosting

Gradient boosting [Friedman, 2001] is one of the most widely applied boosting methods in biological problems. The key idea behind gradient boosting is that, under some general hypothesis on the cost function, boosting can be seen as an iterative gradient method for numerical optimization. Gradient boosting has several desirable properties [Mayr et al., 2014], such as its capability to learn nonlinear input/output relationship, its ability to embed a feature importance measure (as random forests [Hastie et al., 2009]) and its stability in case of high-dimensional data [Buehlmann, 2006].

As for any learning machine, boosting methods may suffer of overfitting. The main regularization parameter to control is the number of boosting iterations M, *i.e.* the number of base learners, fitted on the training data. This is typically optimized by cross-validated grid-search, or by information criteria-based heuristics [Tutz and Binder, 2006, 2007] (see Section 4.4).

Regularization in gradient boosting can also be controlled by shrinking the contribution of each base learner by a constant $0 < \nu < 1$ that controls the learning rate of the boosting procedure [Hastie et al., 2009]. In order to achieve comparable predictive power, smaller values of ν imply larger number of M, so there is a tradeoff between them. As usually the base learners are decision trees, another important parameter to tune is their maximum depth [Hastie et al., 2009].

In a recent paper [Lusa et al., 2015], the authors show that in high-dimensional balanced binary classification problems, if the base learner is likely to overfit the training data, the use of *Stochastic gradient boosting* [Friedman, 2002] is preferable. The latter is a modified version of the original method, where each base learner is fitted on a random extraction without resubmission of a fraction η of the training data, where η is de-facto a regularization parameter to choose.

Approaches based on gradient boosting classification are used to detect *de novo* mutations showing an improved specificity and sensitivity with respect to state-of-the-art methods [Liu et al., 2014b]. When combined with stability selection [Meinshausen and Bühlmann, 2010], gradient boosting has demonstrated to be a very resourceful method for variable selection, leading to an effective control of the false discovery rate. This strategy was followed to associate overall survival with single-nucleotide polymorphisms of patients affected by cutaneous melanoma [He et al., 2016b] and to detect differentially expressed amino acid pathways in autism spectrum disorder patients [Hofner et al., 2015].

{consider adding boosting method for cox models...}

4.1.4 Deep learning

Deep Learning (DL) methods are a broad class of machine learning techniques that, starting from raw data, aim at learning a suitable feature representation (see Section **??**) and a prediction function, at the same time [LeCun et al., 2015]. DL methods can be seen as an extension of classical Neural Networks, where the final prediction is achieved by composing several layers of non-linear transformations. DL architectures can be devised to tackle binary/multi-category classification [Angermueller et al., 2016; Leung et al., 2014] as well as single/multiple-output regression [Chen et al., 2016; Ma et al., 2015] tasks.

Many methods fall in this class [LeCun et al., 2015], in order to understand them in general, we sketch here the main ideas behind the most basic one: the Multi-Layer Perceptron (MLP), also known as deep feedforward network. Typically, MLPs are structured as fully connected graphs organized in *layers* that can be of three different types: *input*, *hidden* and *output* (see Figure 4.14). Each node of the graph is called *unit*. The number of units in the input layer matches the dimensionality of the raw data (d), while number and type of output units are related to the learning task. The size of the hidden layer, and their number, can be arbitrarily chosen according to the prediction task and the available computational resources. In MLPs the information flows through the graph from the input to the output. Each layer l transforms its input data $\boldsymbol{x}^{l-1}$ by composing an affine transformation and an activation function $f(\cdot)$ that acts element-wise on its input vector. In other words, defining as p_{l-i} the number of units in the layer $l-i$, the layer l applies the transformation $\boldsymbol{x}^l = f(\boldsymbol{x}^{(l-1)}W_l + b_l)$, where $W_l \in \mathbb{R}^{p_{l-1} \times p_l}$ and $\boldsymbol{b} \in \mathbb{R}^{p_l}$ are the weights of the model that are learned from the data. The function $f(\cdot)$ is known as *activation function* and it can be defined in different ways. In classical neural networks, activation functions are modeled as sigmoids (*e.g.* $f(x) = \tanh(x)$, $f(x) = (1 + e^{-x})^{-1}$) whilst in modern DL architectures the most used activation function is the Rectified Linear Unit (*i.e.* $f(x) = \max(0, x)$) [LeCun et al., 2015] .

Particular attention must be paid when fitting deep models as they can be prone to overfit the training set [Angermueller et al., 2016]. The network topology itself defines the *degrees of freedom* of the model: deeper and wider networks can approximate well very complicated input-output relationship, but also the noise affecting the data. Although, tuning the number of hidden layers and their size is not the recommended strategy to prevent from overfitting, as it may lead to suboptimal solutions.

Regularization in MLPs can be controlled by penalizing the weights of the network (see Section **??**). The most common regularization strategy consists in adding an ℓ_2-norm penalty in the objective function, as in Equation (4.2). In the DL community this procedure is known as weight decay [Krogh and Hertz, 1992].

This strategy is adopted in [Chen et al., 2015] to train a deep architecture on rat cell responses to given stimuli with the final aim to predict human cell responses in the same conditions. Moreover, weight decay is also adopted in [Yuan et al., 2016] to train *DeepGene*, *i.e.* an MLP which is designed to classify the tumor type from a set of somatic point mutations. Furthermore, weight

decay is used in [Fakhry et al., 2016] to train a DL architecture for brain electron microscopy image segmentation. Although less common, the ℓ_1-norm can also be adopted as regularization penalty, as in [Leung et al., 2014].

Training MLPs, and deep networks in general, consists in solving a minimization problem via suitable optimization algorithms [Ruder, 2016]. All these methods iteratively update the weights of the network in order to decrease the training error. A popular regularization strategy, known as *Early stopping* [Prechelt, 1998], consists in interrupting the fitting process as soon as the error on an external validation set increases [Angermueller et al., 2016].

Another common regularization strategy in DL is *Dropout* [Srivastava et al., 2014]. This techniques consists in temporarily deactivating a defined number of randomly chosen units of the network at training phase. This reduces the degrees of freedom of the model and it implicitly allows to achieve an ensemble of several smaller networks whose predictions are combined. The use of dropout alone can improve the generalization properties, as in [Chen et al., 2016], where the authors propose a *D-GEX*, DL regression architecture trained to predict the expression of a number of target genes. Dropout can also be used in combination with weight decay or other forms of regularization, as in [Leung et al., 2014], where the authors propose to use a deep network to achieve splicing pattern prediction.

Table 4.2: Overview of the matrix norms used for multiple-output regression.

Matrix norm	Notation	Definition
Frobenius	$\|A\|_F$	$\sqrt{\text{trace}(A^T A)}$
Nuclear	$\|A\|_*$	$\text{trace}\sqrt{(A^T A)}$
Mixed $L_{2,1}$	$\|A\|_{2,1}$	$\sum_j \|\boldsymbol{a}_j\|_2$

4.2 Unsupervised learning

4.2.1 Manifold learning

4.2.2 Clustering

4.3 Feature selection

4.4 Model selection and evaluation

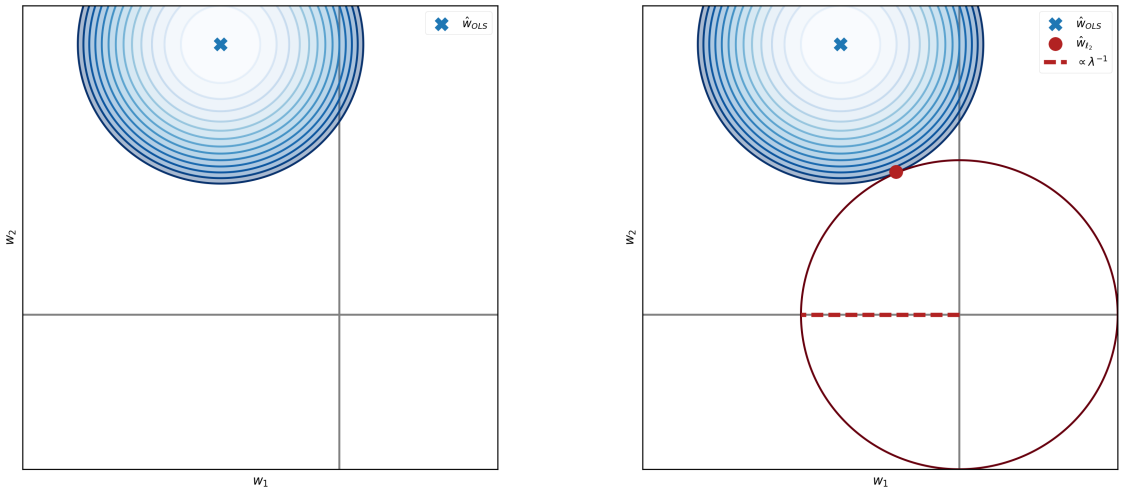
4.4.1 Model selection strategies

4.4.2 Feature selection stability

4.4.3 Performance metrics

4.5 Computational requirements and implementations

- MPI
- GPU and accelerators



(a) (b)

(c) (d)

Figure 4.12: Pictorial representation of the contour lines of the square loss in a 2D regression problem with various penalties: (a) ordinary least squares (no penalty), (b) ridge regression (ℓ_2-norm penalty), (c) the Lasso (ℓ_1-norm penalty) and finally (d) the Elastic-Net (ℓ_1- and ℓ_2-norm penalties).

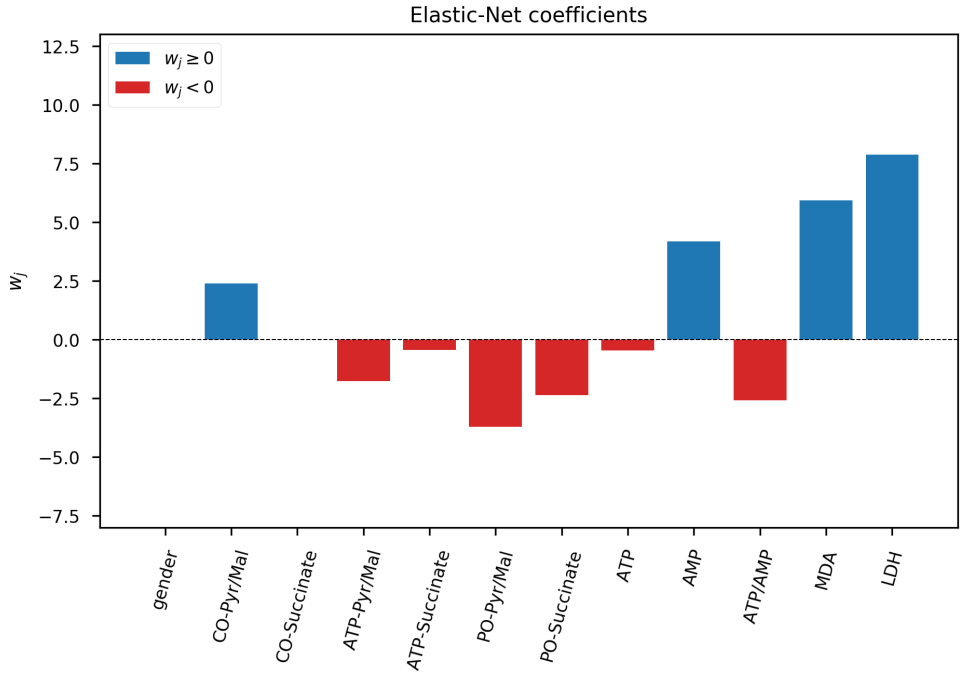
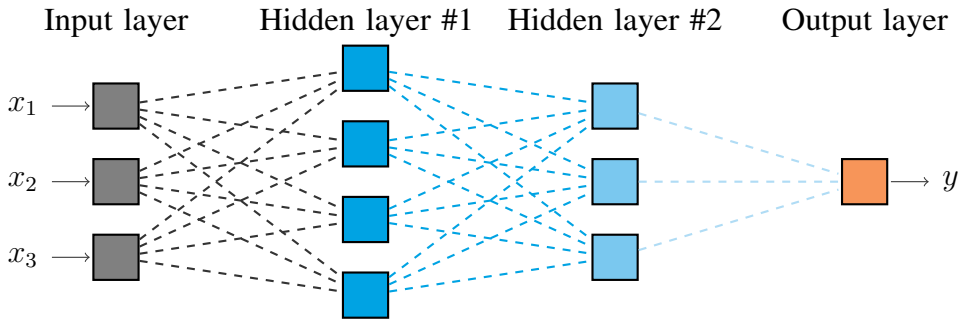


Figure 4.13: A pictorial representation of the vector $\hat{\boldsymbol{w}}_{\text{SVR}}$ obtained fitting a SVR model on 74 randomly selected training samples of $\mathcal{D}_{\text{aging}}$. Variables associated with positive (*i.e.* directly proportional to the output) and a negative (*i.e.* inversely proportional) weight are represented in blue and red, respectively.

Figure 4.14: A pictorial representation of the structure of a Multi-Layer Perceptron example with two hidden layers having four and three hidden units, respectively. According to the nature of the output layer, this network topolgy can be adopted either for regression or binary classification problems starting from raw samples in a two-dimensional space.



Part II

5 ADENINE: a data exploration tool

abstract here.

ADENINE is a machine learning framework designed for biological data exploration and visualization. Its goal is to help bioinformaticians achieving a first and quick overview of the main structures underlying their data. This software tool encompasses state-of-the-art techniques for missing values imputing, data preprocessing, dimensionality reduction and clustering. ADENINE has a scalable architecture which seamlessly work on single workstations as well as on high-performance computing facilities. ADENINE is capable of generating publication-ready plots along with quantitative descriptions of the results. In this paper we provide an example of exploratory analysis on a publicly available gene expression data set of colorectal cancer samples. The software and its documentation are available at `https://github.com/slipguru/adenine` under FreeBSD license.

5.1 What is data exploration?

In biology, as well as in any other scientific domain, exploring and visualizing the collected measures is an insightful starting point for every data analysis process. For instance, the aim of a biomedical study can be detecting groups of patients that respond differently to a given treatment, or inferring possible molecular relationships among all, or a subset, of the measured variables. In both cases, bioinformaticians will be asked to extract meaningful information from collections of complex and high-dimensional measures, such as NGS data.

In these cases, a preliminary Exploratory Data Analysis (EDA) is not only a good practice, but also fundamental before further and deeper investigations can take place. To accomplish this task, several machine learning and data mining techniques were developed over the years. Among those, we recall the combined use of the following classes of methods: (i) missing values imputing, (ii) data preprocessing, (iii) dimensionality reduction and (iv) unsupervised clustering (see Section 5.4).

5.2 Popular data exploration tools

In the last few years, a fair number of data exploration software and libraries were released. Such tools may be grouped in two families: GUI-based and command-line applications. Among the first group we recall Divvy [Lewis et al., 2013], a software tool that performs dimensionality reduction and clustering on input data sets. *Divvy* is a light framework; however, its collection of C/C++ algorithm implementations does not cover common strategies such as kernel principal component analysis [Schölkopf et al., 1997] or hierarchical clustering [Hastie et al., 2009] and it does not offer strategies to perform automatic discovery of the number of clusters. The most notable project that spans between the two families is *Orange* [Demšar et al., 2013], a data mining software suite that offers both visual programming front-end and Python APIs. In the context of data exploration, *Orange* can be successfully employed. However, in order to test different data analysis pipelines, each one must be manually created as it does not support their automatic generation. Moreover, large data sets are difficult to analyze with both *Divvy* and *Orange* as they can run only on a single workstation, lacking of distributed computing support.

5.3 ADENINE overview

In this paper, we present ADENINE, a command-line Python tool for biological data exploration and visualization that, starting from a set of unsupervised algorithms, creates textual and graphical reports of an arbitrary number of pipelines. Missing data imputing, preprocessing, dimensionality reduction and clustering strategies are considered as building blocks for constructing data analysis pipelines. The user is simply required to specify the input data and to select the desired blocks. ADENINE, then, takes care of generating and running the pipelines obtained by all possible combinations of the selected blocks. Every algorithm implementation of the presented software tool is inherited, or extended, from SCIKIT-LEARN [Pedregosa et al., 2011] which is, to the best of our knowledge, the most complete machine learning open source Python library available online.

ADENINE natively supports data integration with the NCBI Gene Expression Omnibus (GEO) archive [Barrett et al., 2013], which data sets can be retrieved specifying their GEO accession number.

Thanks to its scalable architecture, ADENINE pipelines can seamlessly run in parallel as separate Python processes on single workstations or MPI[1] tasks in high-performance computing (HPC) cluster facilities. This remarkable feature allows to explore and visualize massive amounts of data in a reasonable computational time. Moreover, as ADENINE makes large use of NUMPY and SCIPY, it automatically benefits from their bindings with optimized linear algebra libraries (such as OpenBLAS or Intel® MKL).

[1] http://mpi-forum.org/

Step	Algorithm	Reference
Imputing	mean	
	median	
	most frequent	
	k-nearest neighbors	[Troyanskaya et al., 2001]
Preprocessing	recentering	
	standardize	
	normalize	
	min-max	
Dimensionality reduction	principal component analysis (PCA)	[Jolliffe, 2002]
	incremental PCA	[Ross et al., 2008]
	randomized PCA	[Halko et al., 2011]
	kernel PCA	[Schölkopf et al., 1997]
	isomap	[Tenenbaum et al., 2000]
	locally linear embedding	[Roweis and Saul, 2000]
	spectral embedding	[Ng et al., 2002]
	multidimensional scaling	[Borg and Groenen, 2005]
	t-distributed stochastic neighbor embedding	[Van der Maaten and Hinton, 2008]
Clustering	k-means	[Bishop, 2006]
	affinity propagation	[Frey and Dueck, 2007]
	mean shift	[Comaniciu and Meer, 2002]
	spectral	[Shi and Malik, 2000]
	hierarchical	[Hastie et al., 2009]
	DBSCAN	[Ester et al., 1996]

Table 5.1: Pipeline building blocks available in ADENINE.

5.4 ADENINE implementation

ADENINE is developed around the data analysis concept of *pipeline*. A pipeline is a sequence of the following fundamental steps: (i) missing values imputing, (ii) data preprocessing, (iii) dimensionality reduction and (iv) unsupervised clustering. For each task, different off-the-shelf algorithms are available (see Table 5.1).

Data collected in biomedical research studies often present missing values. Devising imputing strategies is a common practice [De Souto et al., 2015] to deal with such issue. ADENINE offers an improved version of the `Imputer` class provided by SCIKIT-LEARN. In addition to the pre-existent feature-wise *mean*, *median* and *most frequent* strategies, this extension presents the k-nearest neighbors imputing method proposed for microarray data in [Troyanskaya et al., 2001].

Collecting data from heterogeneous sources may imply dealing with variables lying in very different numerical ranges and this could have a negative influence on the behavior of data

analysis techniques. To tackle this issue ADENINE offers different strategies to preprocess data, such as recentering, standardizing or rescaling.

The presented software includes a set of linear and nonlinear dimensionality reduction and manifold learning algorithms that are particularly suited for exploration and visualization of high-dimensional data. Such techniques rely on the fact that it is often possible to *decrease* the dimensionality of the problem estimating a low-dimensional embedding in which the data lie.

Besides offering a wide range of clustering techniques, ADENINE implements strategies and heuristics to automatically estimate parameters that yield the most suitable cluster separation. The optimal parameter selection of centroid-based algorithms follows the B-fold cross-validation strategy presented in Algorithm 1, where $\mathcal{S}(X, y)$ is the mean silhouette coefficient [Rousseeuw, 1987] for all input samples.

Algorithm 1 Automatic discovery of the optimal clustering parameter.

for clustering parameter k in $k_1 \dots k_K$ **do**
  for cross-validation split b in $1 \dots B$ **do**
    $X_b^{tr}, X_b^{vld} \leftarrow b$-th training, validation set
    $\hat{m} \leftarrow$ fit model on X_b^{tr}
    $\hat{y} \leftarrow$ predict labels of X_b^{vld} according to $\hat{m}$
    $s_b \leftarrow$ evaluate silhouette score $\mathcal{S}(X_b^{vld}, \hat{y})$
  end for
  $\bar{S}_k = \frac{1}{B} \sum_{i=1}^{B} s_i$
end for
$k_{opt} = \operatorname{argmax} k(\bar{S}_k)$

For affinity propagation [Frey and Dueck, 2007] and k-means [Bishop, 2006] clustering parameters can be automatically defined (*preference* and *number of clusters*, respectively). Mean shift [Comaniciu and Meer, 2002] and DBSCAN [Ester et al., 1996] offer an implicit cluster discovery. For hierarchical [Hastie et al., 2009] and spectral clustering [Shi and Malik, 2000], no automatic discovery of clustering parameters is offered. However, graphical aids are generated to evaluate clustering performance such as dendrogram tree and eigenvalues of the Laplacian of the affinity matrix plots, respectively.

5.5 Usage example

In this section we show how ADENINE can be used to perform two EDAs on a gene expression microarray data set obtained from the GEO repository (accession number GSE87211). This data set was collected in a recent medical study that aimed at understanding the underlying mechanism of colorectal cancer (CRC) as well as identifying molecular biomarkers, fundamental for the disease prognostication. It is composed of 203 colorectal cancer samples and 160 matched mucosa

controls. The adopted platform was the Agilent-026652 Whole Human Genome Microarray, which was used to measure the expression of 34127 probe sets.

ADENINE offers a handy tool to automatically download the data set from the GEO repository given only its accession name. It also let the user select phenotypes and/or probe sets of interest. Given these preferences, ADENINE automatically converts the data set from the *SOFT* format to a comma-separated values text file. To download the remote GEO data set specifying the tissue type as phenotype of interest we used the following command.

```
$ ade_GEO2csv.py GSE87211 --label_field characteristics_ch1.3.tissue
```

This automatically creates `GSE87211_data.csv` and `GSE87211_labels.csv` which contain gene expression levels and tissue type of each sample, respectively.

The first EDA aims at stratifying the samples according to their tissue type (mucosa or rectal tumor) this can be performed by executing the following command.

```
$ ade_run.py ade_config.py
```

Where `ade_config.py` is a configuration file which should look like the snippet below.

{config here}

Each `step` variable refers to a dictionary having the name of the building block as key and a list as value. Each list has a *on**off* trigger in first position followed by a dictionary of keyword arguments for the class implementing the corresponding method. When more than one method is specified in a single step, or more than a single parameter is passed as list, ADENINE generates the pipelines composed of all possible combinations.

The configuration snippet above generates eight pipelines with similar structure. The first and the second halves have recentered and ℓ_2-normalized samples, respectively. Each sample is then projected on a 2D space by isomap or by linear, Gaussian or polynomial kernel PCA. k-means clustering with automatic cluster discovery is eventually performed on each dimensionality-reduced data set, as in Algorithm 1. Results of such pipelines are all stored in a single output folder. Once this process is completed, plots and reports can be automatically generated running the following command.

```
$ ade_analysis.py results/ade_output_folder_YYYY-MM-DD_hh:mm:ss
```

The aim of the second EDA is to uncover the relationships among a set of genes known from the literature to be strongly associated with CRC. Specifically this signature is composed of the following genes: APC, KRAS, CTNNB1, TP53, MSH2, MLH1, PMS2, PTEN, SMAD4, STK11, GSK3B and AXIN2 [Schulz, 2005]. We also considered probe sets measuring expression level of the same gene, and we labelled them with a progressive number. Three partially overlapping sublists compose this signature.

S1) Genes fundamental for the progression of CRC (*i.e.* APC, KRAS, CTNNB1, TP53).

S2) Genes relevant in the *Wnt signaling pathway*, which is strongly activated in the first phases of CRC (*i.e.* APC, CTNNB1, GSK3B, AXIN2).

S3) Genes involved in hereditary syndromes which predispose to CRC (*i.e.* APC, MSH2, MLH1, PMS2, PTEN, SMAD4, STK11) [Schulz, 2005].

A reduced version of the GEO data set that comprises only such genes can be easily created calling `ade_GEO2csv.py` with the option `--signature GENE_1,GENE_2,...,GENE_N`. On the same line, the option `--phenotypes P_1,P_2,...,P_M` can be used to keep only mucosa or rectal tumor samples. To run such experiment, one simply needs to select and activate the hierarchical clustering building block and to follow the same steps presented above.

For ADENINE installation instructions and for a comprehensive description of all the options available in the configuration file we refer to the online documentation and tutorials[2].

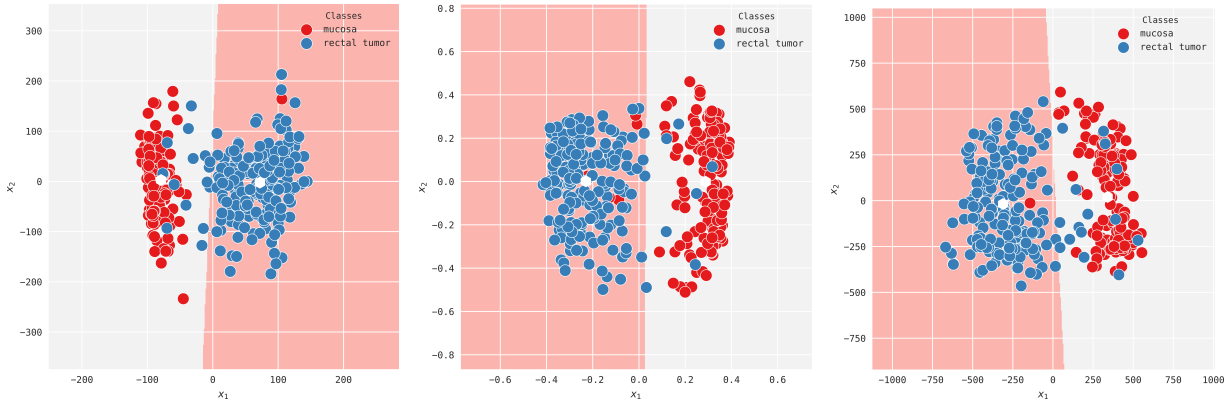
5.6 Results

In the first EDA, we compared the clustering performance achieved by the eight ADENINE pipelines and we reported in Figure 5.1 an intuitive visualization of the results achieved by the top three, evaluated in terms of silhouette score [Rousseeuw, 1987]. As expected, the top performing pipelines show a clear separation between the two sample groups, as the k-means algorithm devises a domain partitioning that is consistent with the actual tissue types.

For the second EDA, the relationships among the probe sets corresponding to the genes of the signature are separately explored learning a different hierarchical clustering [Hastie et al., 2009] tree for mucosa (Figures 5.2a) and CRC samples (Figure 5.2b), separately. The two trees are learned from different tissues, nevertheless they show some remarkable similarities. For instance, the pairs TP53-TP53.1 and MSH2-PMS2.1 always share a common parent. Interestingly, the first is a relationship between probe sets of the same gene, and the second is confirmed in literature, as MSH2 and PMS2 are both involved in hereditary non-polyposis CRC, a syndrome that predisposes for CRC. Moreover, two probe sets of the the genes of *S1*, namely APC and CTNNB1, are consistently close to the root of the two trees. This suggest that the expression level of these two genes highly differs from the others. Two interesting differences between the two trees can also be noticed. First, most of the elements of the sublist *S3*, which contains genes that enhance the risk of developing CRC, tend to be grouped together in Figure 5.2b, while the same observation cannot be done for Figure 5.2a. Secondly, probe sets of the genes belonging to sublists *S2* and *S3* tend more to more closely connected in Figure 5.2b than in Figure 5.2a.

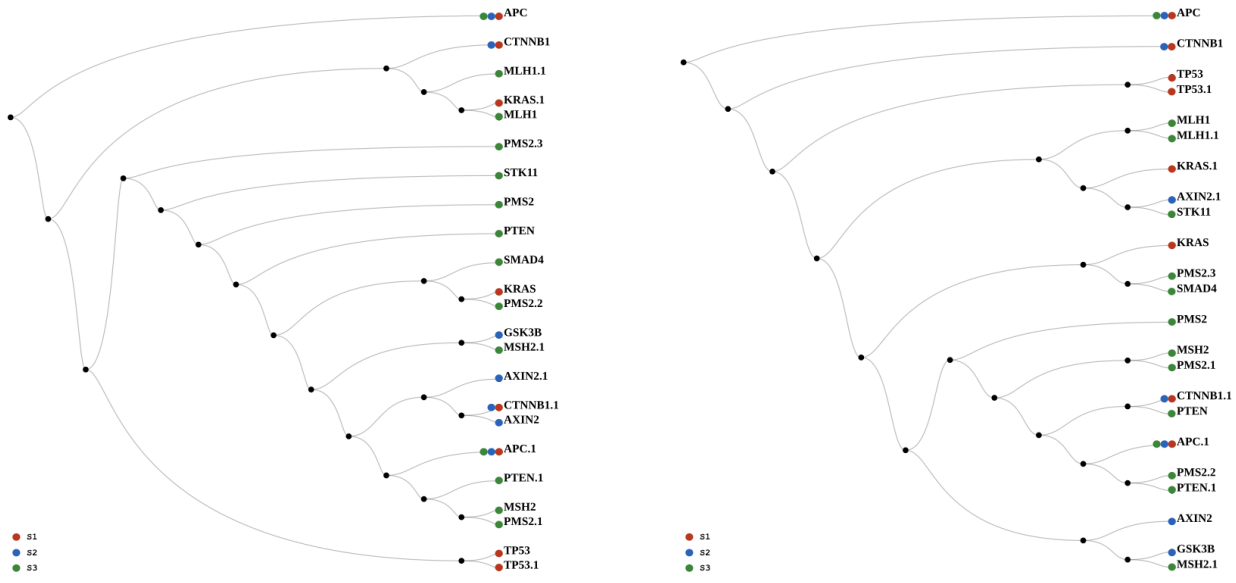
{...} In this paper we presented ADENINE, a biomedical data exploration and visualization tool that can seamlessly run on single workstations as well as on HPC clusters. Thanks to its scalable

[2] http://slipguru.github.io/adenine



(a) (b) (c)

Figure 5.1: Three different 2D projections of the samples of the GEO gene expression data set used in this work. Projections on the left (a), middle (b) and right (c) panes are obtained via linear PCA, Gaussian PCA and isomap, respectively. The color of each point corresponds to the actual tissue type, while the background color is automatically learned by the k-means clustering algorithm. White hexagons correspond to cluster centroids.



(a) (b)

Figure 5.2: An example of hierarchical trees visualization learned by two ADENINE pipelines on mucosa (a) and CRC (b) samples. Each probe set is color coded according to the corresponding sublist. This visualization provides insights on the underlying structure of the measured gene expression level.

architecture, ADENINE is suitable for the analysis of large and high-dimensional data collections, that are nowadays ubiquitous in biomedicine.

ADENINE natively supports the integration with the GEO repository. Therefore, a user provided with the accession number of the data set of interest can select target phenotypes and genotypes and ADENINE takes care of automatically downloading the data and plugging them into the computational framework. ADENINE offers a wide range of missing values imputing, data preprocessing, dimensionality reduction and clustering techniques that can be easily selected and applied to any input data.

In this paper we showed ADENINE capabilities performing two EDAs on a CRC gene expression data set. From the obtained results we can observe that a clear discrimination between CRC and control samples can be achieved by unsupervised data analysis pipeline. Moreover, a meaningful description of the relationships among the group of genes strongly associated with CRC can be represented as hierarchical trees.

6 Model for metabolic age prediction

abstract here.

7 Temporal model for multiple sclerosis evolution

abstract here.

Multiple Sclerosis (MS) is a neurodegenerative and chronic disease of the central nervous system characterized by damages to the myelin sheaths, resulting in a wide range of symptoms, such as fatigue, numbness, visual disturbances, bladder problems, mobility issues and cognitive deficits.

People with MS (PwMS) are mainly classified according to their disease course: relapsing-remitting (*RR*), secondary-progressive (*SP*), primary-progressive (*PP*) and progressive-relapsing (*PR*) [Giovannoni et al., 2016]. Neurological disability in *RR* patients is mainly due to the development of multifocal inflammatory lesions and it results in relapses, that are attacks of neurological worsening, followed by partial or complete recovery. Disability accrues predominantly in progressive courses (*SP*, *PP*, *PR*) that are more characterized from diffuse immune mechanisms and neurodegeneration. An estimated 15% of PwMS have a *PP* or *PR* course at the onset, the remaining 85% is diagnosed with a *RR* course. About 80% of *RR* patients develop *SP* course within 15–20 years if untreated, or if the adopted pharmacological and rehabilitative protocols are not continuously adjusted according to the evolution of the disease [Scalfari et al., 2014].

For this reason, the prediction of the transition from *RR* to *SP* is one of the most important methodological gaps that MS researchers are currently addressing. The availability of a statistical model able to predict disease worsening is one of the major unmet needs that could significantly improve timeliness, personalization and, consequently, the efficacy of the treatments. Nowadays, there are no clear clinical, imaging, immunologic or pathologic criteria to foresee the transition from *RR* to *SP* [Lublin et al., 2014]. Several clinical factors relating to possible *SP* course predictors have been identified [Bergamaschi et al., 2015; Dickens et al., 2014]. However, as showed by [Vukusic and Confavreux, 2003], studies investigating on prognostic factors for MS course evolution generally suffer from two shortcomings: they report a high proportion of *RR* patients not monitored enough to reach progressive course and they lead, to some extent, to contradictory results. Currently, MS research mainly focuses on developing and assessing drugs and rehabilitative protocols for *RR* patients disregarding progressive courses.

In the recent past, researchers explored the potential role of Patient-Centered Outcomes (*PCO*) to follow the progression of neurodegenerative diseases and to take timely healthcare decisions [Black, 2013]. PCOs comprise self- and physician-administered tests, questionnaires and clinical scales consisting of either ordinal or categorical scaled answers. As opposed to stressful, not frequently repeatable and expensive clinical exams, like magnetic resonance imaging or blood tests, PCOs are patient-friendly and low-cost measures that could allow to investigate the indi-

vidual changes and disease impact on several aspects such as physical, cognitive, psychological, social and well-being domains [Fiorini et al., 2015]. To date, PCOs are extensively used to assess general health status, to support diagnosis and monitor progress of disease and to quantify the patients' perception of the effectiveness of a given therapy or procedure [Nelson et al., 2015]. Nevertheless, it is still unclear which are the most informative PCOs and, contextually, whether they can be used as *predictors* for disease evolution.

In our study, we propose a machine learning approach that, leveraging on *PCO* data, aims at predicting the temporal evolution of MS disease course providing insights on the most appropriate use of PCOs. We resort to a vast category of predictive models, ranging from sparse regularization to ensemble and deep learning methods. These models are widely adopted in the biomedical context as they benefit from good generalization properties as well as they allow to address regression and classification problems within the same statistical and computational framework [LeCun et al., 2015; Qi, 2012; Nowak et al., 2011; Teramoto et al., 2009; Zou and Hastie, 2005].

7.1 PCOs data set description

The predictive model presented in this work is based on a *PCO* data set acquired from a cohort of PwMS progressively enrolled within an ongoing funded project. Ethical review committee approval *023REG2014* was obtained for this work.

Each patient is evaluated every four months through the items of the PCOs reported in Table 7.1 which cover physical, cognitive and psychosocial domains. *PCO* data are intrinsically noisy due to the subjectivity of self-reported measures provided by the patients that can be influenced by personal feelings and opinions. In order to ameliorate this issue, 4 questionnaires out of 10 are administered by medical staff which is trained to keep a homogeneous level of evaluation.

In our analysis we considered all the PCOs reported in Table 7.1 except EDSS. Such scale is based on a neurological examination and, although usually adopted as an index of the disability level, it focuses mainly on deambulation disability without taking into account other aspects that could impact patient disability, such as upper limb or cognitive functions [Meyer-Moock et al., 2014; Uitdehaag, 2014].

The collected *PCO* data set comprises additional information such as: i) number of relapses in the last four months (NR), ii) educational level expressed in terms of total years of education (EDU), iii) height (H) and iv) weight (W). Each sample of the data set is represented by a vector of $d = 165$ predictors. Moreover, a neurologist assigns to each patient the corresponding disease course. The global distribution of MS types across time points is depicted in Figure 7.1b.

In this work we focus on predicting MS course evolution of *RR* and *SP* patients, hence the subjects with *PR* and *PP* forms will not be taken into account. We considered all the patients with a minimum of 1 time point (the most recently enrolled) up to $T = 8$ time points for a total of 2699

Acronym	Full name	Reference
$MFIS$	Modified fatigue impact scale	[Flachenecker et al., 2002]
$HADS$	Hospital anxiety and depression scale	[Honarmand and Feinstein, 2009]
$LIFE$	Life satisfaction index	[Franchignoni et al., 1999]
OAB	Overactive bladder questionnaire	[Cardozo et al., 2014]
$EDINB$	Edinburgh handedness inventory	[Oldfield, 1971]
$ABILH$	Hand ability index	[Arnould et al., 2012]
FIM	Functional independence measure	[Granger et al., 1990]
$MOCA$	Montreal cognitive assessment	[Dagenais et al., 2013]
$PASAT$	Paced auditory serial addition task	[Aupperle et al., 2002]
$SDMT$	Symbol digit modality test	[Parmenter et al., 2007]
$EDSS$	Expanded disability status scale	[Kurtzke, 1983]

Table 7.1: The set of available PCOs. The first 6 are self-reported, while the last 5 are administered by trained medical staff. In our analysis all PCOs were used, with the exception of $EDSS$.

samples, of which 1220 ***RR*** and 1579 ***SP*** (see Figure 7.1a). As this is an ongoing project, the number of PwMS decreases with time. We expect to fill the gap of samples between *Exam* 1 and *Exam* 8 by the end of the funded study.

In this work we analyze PCOs data acquired every four months from a cohort of MS patients enrolled in a funded study. Currently, we have collected data for eight examinations and, as patients enrollment is still ongoing, the number of individuals for each time point is successively decreasing

The number of considered samples across examinations is hence 2699, of which 1220 ***RR*** and 1579 ***SP***.

7.2 Problem description

Predicting the MS course evolution can be split in three different related tasks: Current Course Assignment (CCA), PCOs Evolution Prediction (PEP) and Future Course Assignment (FCA). In particular, given the 165-dimensional representation of a patient at a fixed time point $\boldsymbol{x}_i^t$, CCA consists in assigning the corresponding disease course y_i^t. Given the historical representation of a patient $\boldsymbol{x}_i^t$ for $t = 1, \ldots, \tau$, PEP consists in predicting the patient representation $\boldsymbol{x}_i^{\tau+1}$. Finally, FCA consists in foreseeing the MS disease course $y_i^{\tau+1}$ from $\boldsymbol{x}_i^t$ for $t = 1, \ldots, \tau$.

Here, we developed a predictive model that solves these tasks assuming the temporal structure outlined in Figure 7.2. The CCA problem is translated into a binary classification task and we address it by learning a discriminative function $f(\boldsymbol{x}_i^t) = y_i^t$. The PEP problem is modeled as $g(\boldsymbol{x}_i^t) = \boldsymbol{x}_i^{t+1}$, where $g(\boldsymbol{x})$ is a multiple-output regression function. Once $\hat{f}(\boldsymbol{x})$ and $\hat{g}(\boldsymbol{x})$ are

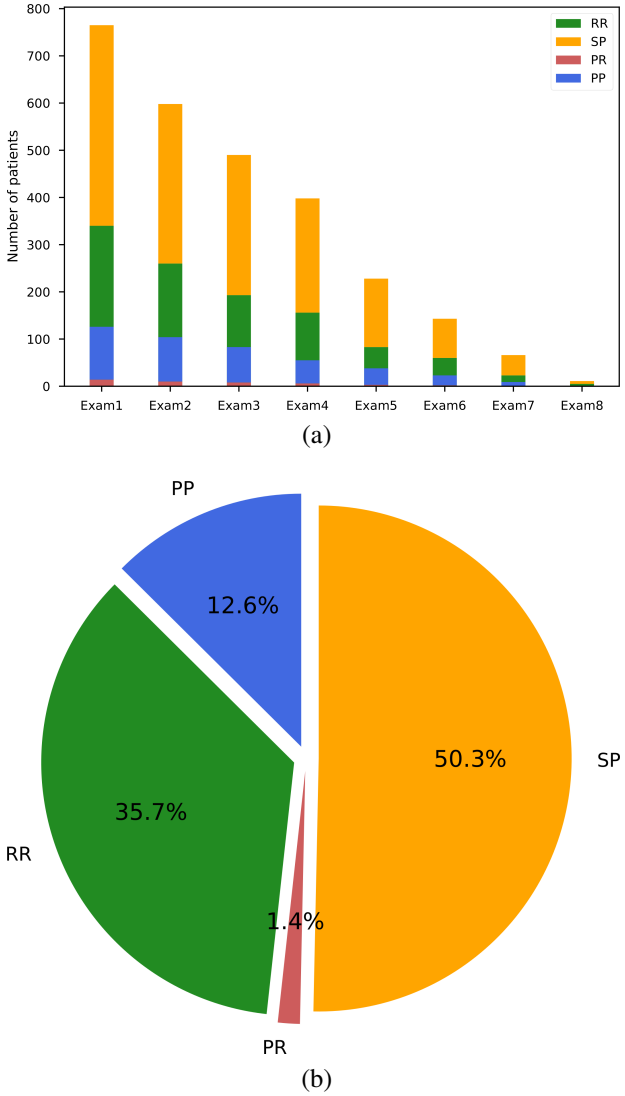


Figure 7.1: An overview of the *PCO* data set used in this study. The left panel (a) shows a bar chart of the number of MS patients in each disease form at different examinations. The right panel (b) presents a representation of the distribution of the total amount of acquisitions (3137), divided according to the disease form. **{break figure in two}**

learned by training on historical *PCO* data, the FCA problem is finally solved by the temporal model $\hat{f} \circ \hat{g}(\boldsymbol{x}_i^t) = y_i^{t+1}$. In time-series data analysis, this is known as *one-step-ahead forecast*. Notably, the FCA model allows to foresee if the patient at the next time point is going to experience a transition from *RR* to *SP*, or not.

7.2.1 Data preprocessing

Analyzing *PCO* data is challenging from several respects. First, items belonging to different questionnaires are encoded with numerical values in different ranges. To tackle this issue, we opted for a $[0-1]$ scaling of the ordinal answers and a binary one-hot-encoding of the categorical ones. Secondly, as the missing data amount to 1.52% of the entire data set, we resort to the K-nearest neighbor data imputing strategy proposed in [Troyanskaya et al., 2001]. To ensure unbiasedness of the results, this preprocessing phase is not performed on the entire data collection, but it is separately evaluated prior to each model fitting process on its cross-validation portion of the training set, as described in the next section.

7.2.2 Experimental design

We shall discuss separately the experimental designs used to learn $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$.

The CCA model $f(\boldsymbol{x})$ solves a binary classification problem: to each input $\boldsymbol{x}_i^t$ is associated an output y_i^t that encodes the corresponding MS disease course (*RR* or *SP*) with a binary label. We split the data set in three temporal chunks, namely *training*, *validation* and *test* sets, consisting of all samples collected at time points $t = 1, 2, 3$, $t = 4$ and $t = 5, 6, 7, 8$, respectively. Accordingly, we used 1853 samples for training $f(\boldsymbol{x})$, 398 for validation leaving the remaining 448 for test. Five candidate models for $f(\boldsymbol{x})$ are fitted on 20 Monte Carlo (MC) random sampling of the training set each time keeping $\frac{1}{4}$ of the samples aside [Molinaro et al., 2005]. For each MC sampling the fitting procedure is performed on the remaining $\frac{3}{4}$ of the samples and it includes an inner parameter optimization via grid-search cross-validation [Hastie et al., 2009]. In particular, we require the MS course prediction to be based on a reduced number of variables (see Section 7.2.3), therefore we enforce sparsity in each candidate model. Leveraging on the MC strategy, we rank the variables according to their selection frequency [Barbieri et al., 2016; Meinshausen and Bühlmann, 2010]. Once a variable ranking is achieved for each candidate model, the list

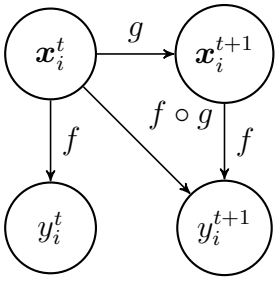


Figure 7.2: A visual representation of the temporal structure assumed in the collected data. When the two functions f (CCA) and g (PEP) are learned, the FCA model $f \circ g$ is able to predict the evolution of the disease course for future time points y_i^{t+1}.

of selected variables is identified by thresholding the corresponding ranking with the threshold that maximizes the accuracy on the validation set. Finally, the last training step consists in fitting each candidate model on the union of training and validation sets taking only into account the corresponding reduced subset of selected variables. The final CCA model $\hat{f}(\boldsymbol{x})$ is chosen as the one that performs better on the previously unseen test set in terms of accuracy, precision, recall and F_1 score.

On the other hand, learning the PEP model $g(\boldsymbol{x})$ implies solving a multiple-output regression problem and each input $\boldsymbol{x}_i^t$ is associated with the output vector $\boldsymbol{x}_i^{t+1}$. Therefore, we can only consider samples at time point t with an available follow-up at the next time point $t+1$, which reduces the overall number of available samples. The data set splitting is consistent with the one followed for learning $f(\boldsymbol{x})$, although there is no need for a separate validation set, as learning $g(\boldsymbol{x})$ does not require any variable selection process. We used the samples collected at time points $t = 1, 2, 3, 4$ for training and those at $t = 5, 6, 7, 8$ for test, resulting in 1737 and 254 samples, respectively. The fitting procedure includes an inner parameter optimization via grid-search cross-validation. Each candidate model is a function $g : \mathbb{R}^{165} \rightarrow \mathbb{R}^k$ where k is the number of variables selected by the best CCA model. The final PEP model $\hat{g}(\boldsymbol{x})$ is chosen as the candidate model that performs better on the previously unseen test set in terms of mean absolute error (MAE).

The predictive capability of the FCA model $\hat{f} \circ \hat{g}(\boldsymbol{x})$ is finally evaluated on the test set. The CCA model $\hat{f}(\boldsymbol{x}_i^t)$ predicts the MS course $\hat{y}_i^t$ from the *PCO* data vector $\hat{\boldsymbol{x}}_i^t$ that, in turn, is predicted by the PEP model $\hat{g}(\boldsymbol{x}_i^{t-1})$. We shall notice here that the predictions $\hat{f} \circ \hat{g}(\boldsymbol{x}_i^t) = y_i^{t+1}$ for $t = 8$ are foreseeing possible *RR* to *SP* transitions that are beyond our data observation, hence predictions at the last time point cannot be used to assess the FCA model performance. Therefore, its performance is evaluated only on 220 test samples.

7.2.3 Learning $f(x)$

We imposed $f(\boldsymbol{x})$ to be sparse. This requirement is helpful from two distinct respects: a) the performance of the predictive model may increase thanks to a reduced effect the course of dimensionality [Hastie et al., 2015] and b) the identification of a reduced subset of meaningful PCOs provides interpretability of the results for the clinicians. In order to achieve such sparse model, we take advantage of two main variable selection strategies: embedded and wrapper methods [Guyon and Elisseeff, 2003]. When using embedded methods, we exploited the sparsity inducing penalties of EN to take into account possible correlation between *PCO* variables and of SLR to benefit from the renowned classification capability of the logistic loss function. We applied the RFE wrapper method to two tree-based learning machines (RF and GB) that are capable of capturing nonlinear relationship between input and output and are intrinsically well-suited to deal with categorical/ordinal variables. We also explored the use of RFE with SVM, as in [Guyon et al., 2002].

7.2.4 Learning $g(x)$

As no prior information on the relationship between PCOs evaluated at different time points was available, to learn $g(\boldsymbol{x})$ we investigated on the use of both linear and nonlinear models.

Concerning the linear models, we explored two different solutions: NNM and MTEN. The first imposes a low-rank prior on the result. The second is a natural multiple-output extension of EN, hence it induces a row-structured sparsity pattern on the solution where collinear variables are more likely to be included in the model together. For nonlinear prediction, we resorted to the state-of-the-art MLP approach.

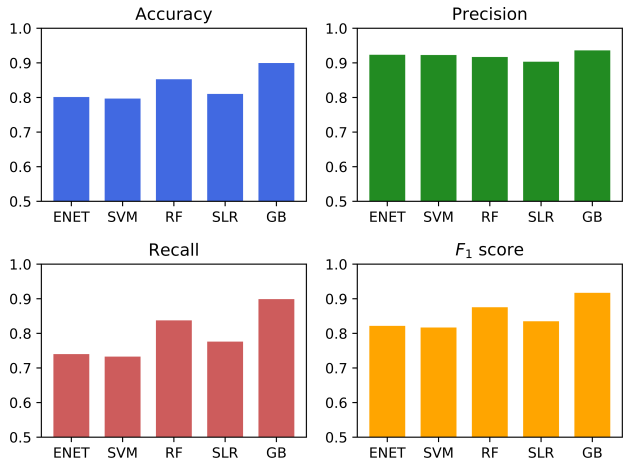
7.3 Results

We shall discuss separately the results achieved in terms of CCA, PEP and FCA models.

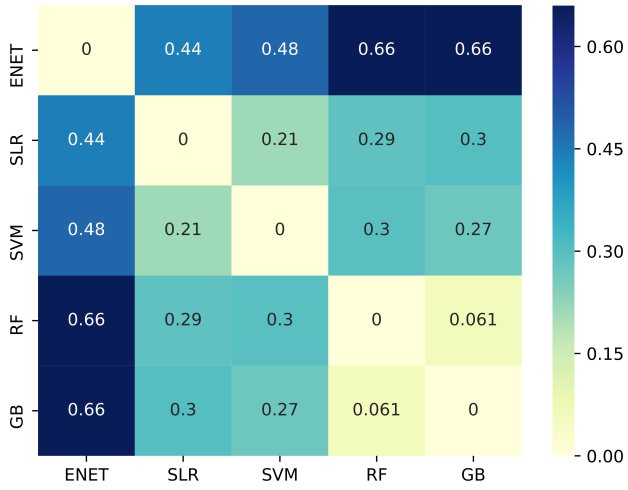
Regarding CCA, the GB method outperforms the other candidate models reaching accuracy 0.900, precision 0.936, recall 0.899 and F_1 score 0.917, as shown in Figure 7.3a. Therefore we chose it as CCA model $\hat{f}(\boldsymbol{x})$. Insights on the use of PCOs for MS assessment are provided by the sparsity of the CCA model induced by the RFE schema. The 31 selected variables are reported in Table **??**. Comparing the full list of *PCO* questionnaires of Table 7.1 with Table **??**, we observe that each *PCO* used in this study is represented at least once, except EDINB, and the most represented is FIM. We also see that, whenever possible, the model tends to select aggregate scores (total and subtotal) rather that single items. This is consistent with the clinical practice, where neurologists are more likely to assess patient's health status by using the aggregate scores, rather than the single questions. Quite surprisingly, the recent number of relapses is the only additional information not selected by the model. Finally, we note that all the domains that are known to be affected by the disease are well covered: mobility (upper and lower limbs), cognition, emotional, fatigue, bladder and psychosocial. The heatmap in Figure 7.3b shows the Hamming distance estimated across the list of variables selected by the five CCA candidate models. Interestingly, tree-based methods are more prone to select similar variables with respect to linear methods. As expected, the sparsity induced by the ℓ_1-norm of SLR allows the method to achieve a list of variables similar to the one obtained by SVM-RFE, while the list obtained by ENET includes collinear variables and it is significantly different from the others.

Regarding PEP, MTEN outperforms the other candidate models in terms of MAE ($\text{MAE}_{\text{MTEN}} = 0.095$, $\text{MAE}_{\text{NNM}} = 0.102$, $\text{MAE}_{\text{MLP}} = 0.105$), hence we select it as our PEP model $\hat{g}(\boldsymbol{x})$.

Finally, the FCA model $\hat{f} \circ \hat{g}(\boldsymbol{x})$, obtained by combining MTEN and GB achieves the following performance scores on the 220 test samples: accuracy 0.841, precision 0.900, recall 0.824 and F_1 score 0.860.



(a)



(b)

Figure 7.3: A visual representation of the results obtained from the CCA model. On the left panel (a) we show the classification performance achieved on the test set by the candidate models. Precision, recall and F_1 score are estimated considering *SP* as the positive class. As GB outperforms the other methods on each performance metric, it is chosen as CCA model. On the right panel (b) a heatmap displays the distance between the lists of variables selected by each model in terms of their hamming distance. **{break figure in two}**

In this work we proposed a novel temporal model based on patient-centered outcomes and machine learning for disease form prediction in multiple sclerosis. In particular, we address the tasks of current course assignment, PCOs evolution prediction and future course assignment. The model is built on a collection of PCOs acquired on a cohort of individuals enrolled in an ongoing funded study (*DETECT-MS PRO*). PCOs data are typically used to corroborate evidence provided

by quantitative exams, in our case the absence of clear MS disease form predictors makes the information extracted from PCOs data the only available resource. The proposed temporal model was able to correctly assign the current MS form and to foresee future ones with accuracy of 90.0% and 84.1%, respectively. This demonstrates that PCOs can effectively be used as MS disease course predictor. In the next future, we plan to further investigate on the predictive capabilities of the proposed model with longer temporal horizons and to compare it with different approaches, such as probabilistic graphical models. Given the achieved promising results, the proposed model is soon going to be validated in clinical practice, where it will assist the clinicians involved in this study to foresee possible disease course transition and to take important decisions concerning treatment and therapies that can substantially improve the quality of life of their patients. In the context of neurodegenerative diseases, clinicians typically use PCOs data to corroborate evidences coming from standard quantitative exams [Black, 2013]. Interestingly, in our case the absence of clear *SP* predictors makes the information extracted from PCOs data the only available resource. In the era of precision medicine, the problem of predicting MS course evolution still relies on stressful exams and clinical judgement. To the best of our knowledge, this is the first attempt to solve this delicate task leveraging on patient-friendly measures and machine learning.

8 Temporal model for glucose predictions

abstract here.

9 Conclusion

A Appendix

As already pointed out at the beginning of Chapter 4, ML is a cross-disciplinary field and the statistical tools used in literature to describe models and algorithms heavily depend on the academic background of the author. This can make the approach to ML fascinating and somewhat cumbersome at the same time.

The goal of this appendix is to shed light on some of the statistical tools and definitions that are typically left unsaid, or given for granted, by most of the authors. In particular, in the following sections insightful statistical details on the formulation of the supervised learning problem expressed in Equation (4.2) will be provided.

A.1 Useful theorems and definitions

This first section lists the theorems and the definitions that are useful for the comprehension of the following sections.

Theorem 1 (Law of the unconscious statistician) *Given two continuous random variables $(a, b) \in A \times B$ with joint probability distribution $p(a, b)$, the expected value of the function $g(a, b)$ can be stated as follows.*

$$\mathbb{E}[g(a,b)] = \iint_{A\times B} g(a,b)\, p(a,b)\, dadb$$

Theorem 2 (Bayes rule) *Given A and B two events with probability $P(A)$ and $P(B) \neq 0$, the conditional probability of observing A given that B is true is*

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

where $P(B|A)$ is the probability of observing B given that A is true.

Definition 1 (Well-posed problem) *A problem is **well-posed** if its solution: (i) exists, (ii) is unique, (iii) depends continuously on the data (e.g. it is stable).*

Definition 2 (Ill-posed problem) *A problem is **ill-posed** if it is not well-posed.*

Definition 3 (Likelihood function) *Let $\boldsymbol{a}$ be a continuous random variable with probability distribution $p(\boldsymbol{a}, \boldsymbol{\phi})$ depending on the parameter $\boldsymbol{\phi}$; then the function $\mathcal{L}(\boldsymbol{\phi}|\bar{\boldsymbol{a}}) = p(\bar{\boldsymbol{a}}, \boldsymbol{\phi})$ is the likelihood function of $\boldsymbol{\phi}$ given that $\bar{\boldsymbol{a}}$ is the outcome of $\boldsymbol{a}$.*

A.2 Empirical risk minimization

In Section 4.1 we introduced the concept of supervised learning as the branch of ML in which predictive models are trained on labeled data. The final goal of supervised learning is to find a function of the input variables $f : \mathcal{X} \to \mathcal{Y}$ that provides a *good* approximation of the output y. In order to measure the adherence between predictions $\hat{y} = f(\boldsymbol{x})$ and actual output y, we introduced the concept of *loss function* $L(\hat{y}, y)$, see Table 4.1. For a fixed choice of the loss, the ideal estimator, also known as the *target* function, $f^*(\boldsymbol{x})$ is the minimizer of the (true) expected risk $\mathcal{E}(f)$ in a rather *large* class of functions $\mathcal{F}$.

$$f^*(\boldsymbol{x}) = \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \mathcal{E}(f) \tag{A.1}$$

Applying the law of the unconscious statistician, stated in Theorem 1, the expected risk $\mathcal{E}(f)$ can be written as in Equation (A.2), where $(\boldsymbol{x}, y)$ are two random variables with joint probability distribution $p(\boldsymbol{x}, y)$.

$$\mathcal{E}(f) = \mathbb{E}[L(f(\boldsymbol{x}), y)] = \iint_{\mathcal{X} \times \mathcal{Y}} L(f(\boldsymbol{x}), y)\, p(\boldsymbol{x}, y)\, d\boldsymbol{x} dy \tag{A.2}$$

In real situations, a direct computation of $\mathcal{E}(f)$ is unfeasible as the joint probability distribution $p(\boldsymbol{x}, y)$ is unknown. Although, we assume to be provided with a collection of input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ that are supposed to be sampled *i.i.d.* from $\mathcal{X} \times \mathcal{Y}$ according to $p(\boldsymbol{x}, y)$. In statistical learning theory, as introduced by Vapnik [Vapnik, 2013], the dataset $\mathcal{D}$ can be used to build a stochastic approximation of $\mathcal{E}(f)$ called *empirical risk* $\mathcal{E}(f_{\mathcal{D}})$ and defined in Equation (A.3).

$$\mathcal{E}(f_{\mathcal{D}}) = \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \tag{A.3}$$

As $\mathcal{D}$ is drawn according to the probability distribution $p(\boldsymbol{x}, y)$, our hope is that the empirical risk can be used as a proxy for the expected risk, hence $\mathcal{E}(f_{\mathcal{D}}) \approx \mathcal{E}(f)$. The solution of the supervised learning problem is then found by *Empirical Risk Minimization* (ERM), defined in Equation (A.4).

$$\hat{f}(\boldsymbol{x}) = \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \mathcal{E}(f_{\mathcal{D}}) = \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \tag{A.4}$$

In practice, minimizing $\mathcal{E}(f_{\mathcal{D}})$ instead of $\mathcal{E}(f)$ comes at a price. The central problem is whether the first is a good approximation of the second. For instance, when $p(\boldsymbol{x}, y)$ is too *complex*, the number of examples is too small and/or the class of functions $\mathcal{F}$ is *too large*, $\hat{f}(\boldsymbol{x})$ will be far from the target function $f^*(\boldsymbol{x})$, even when its empirical error is 0. In real circumstances, it is impossible to control the true probability distribution and it is often extremely difficult to collect a very large number of examples. The only element we can control is the class of functions $\mathcal{F}$ and, in particular, its *size*. Since Tikhonov [Tikhonov, 1963] it is known that, for an arbitrary function space $\mathcal{F}$, the ERM problem is ill-posed (see Definition 2). A possible way to ensure well-posedness is to impose a constrain that restricts the function space. Hence, the constrained ERM problem assumes the form in Equation (A.5), where $\lambda \neq 0$.

$$\begin{aligned} &\underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \\ &\text{subject to } \mathcal{R}(f) < \frac{1}{\lambda} \end{aligned} \tag{A.5}$$

Applying the *Lagrange multipliers technique*[1] Equation (A.5) can be finally written as Equation (A.6).

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) + \lambda \mathcal{R}(f) \tag{A.6}$$

The penalty term $\mathcal{R}(f)$ acts as regularizer [2] and, according to its definition, it can ensure well-posedness of the problem and it can enforce different interesting properties on the achieved solution, see Section 4.1.1. It can also be shown that the use of appropriate regularizes promotes generalization, hence increases our chance to find a $\hat{f}(\boldsymbol{x})$ *close* to $f^*(\boldsymbol{x})$.

According to the choice made for $L(\cdot)$ and $\mathcal{R}(\cdot)$, the minimization problem posed in Equation (A.6) can have very different properties; it can be convex or non-convex, it can include differentiable as well as non-differentiable terms. A rigorous review of the most common optimization methods for ML is beyond the scope of this thesis and can be found here [Boyd and Vandenberghe, 2004; Bach et al., 2012; Sra et al., 2012; Nesterov, 2013].

[1] for a thorough description of this technique, see Appendix E of Bishop's book [Bishop, 2006]

[2] $\mathcal{R}(f)$, in general, can be thought as $\mathcal{R}(f) = \|f\|_K^2$ where $\|\cdot\|_K^2$ is the norm defined by the kernel K in a *Reproducing Kernel Hilbert Space* $\mathcal{H}$ [Evgeniou et al., 2000]

A.3 Maximum likelihood estimation

In this section we will see a different approach to tackle the supervised learning problem. Once again, let the training data be made of input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, with $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $\forall\, i = 1, \ldots, n$. This approach relies on the expression of the uncertainty over the value of y with a probability distribution $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta} \in \Theta$. Applying Definition 3, and assuming that the samples are drawn *i.i.d.*, we can write the likelihood function for $\boldsymbol{\theta}$ as in Equation (A.7).

$$\mathcal{L}(y|\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) \tag{A.7}$$

Equation (A.7) can be considered as the probability of observing the output y_i, given the input $\boldsymbol{x}_i$ and the parameters $\boldsymbol{\theta}$ ($\forall i = 1, \ldots, n$). This statistical setup suggests a strategy to obtain an estimate for $\boldsymbol{\theta}$ known as *Maximum Likelihood Estimation* (MLE), see Equation (A.8).

$$\hat{\boldsymbol{\theta}}_{MLE} = \underset{\theta \,\in\, \Theta}{\operatorname{argmax}}\, \mathcal{L}(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \underset{\theta \,\in\, \Theta}{\operatorname{argmax}} \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) \tag{A.8}$$

Instead of maximizing the likelihood it is often convenient to minimize the *negative log-likelihood*[3]. Equation (A.8) can then be rewritten as Equation (A.9).

$$\hat{\boldsymbol{\theta}}_{MLE} = \underset{\theta \,\in\, \Theta}{\operatorname{argmin}} - \sum_{i=1}^{n} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) \tag{A.9}$$

Moreover, if some prior knowledge on $\boldsymbol{\theta}$ is available, it is possible to incorporate it in the form of a prior distribution $p(\boldsymbol{\theta})$. Applying the Bayes rule (Theorem 2) it is possible to write the *posterior distribution* $p(\boldsymbol{\theta}|y, \boldsymbol{x})$ as in Equation (A.10).

$$p(\boldsymbol{\theta}|y, \boldsymbol{x}) = \frac{p(y|\boldsymbol{x}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})}{p(y|\boldsymbol{x})} \tag{A.10}$$

The normalizing constant in Equation (A.10) $p(y|\boldsymbol{x})$ is independent from $\boldsymbol{\theta}$. It is known as the marginal likelihood and it can be estimated as in Equation (A.11).

$$p(y|\boldsymbol{x}) = \int p(y|\boldsymbol{x}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})\, d\boldsymbol{\theta} \tag{A.11}$$

[3] approaching information theory or deep learning literature, the negative log-likelihood is often referred to as *cross-entropy*

Equation (A.10) suggest a new strategy to achieve an estimate for $\boldsymbol{\theta}$ that takes into account the prior distribution. This criterion is stated in Equation (A.12) and it is known as *Maximum A Posteriori* (MAP).

$$\hat{\boldsymbol{\theta}}_{MAP} = \underset{\theta \in \Theta}{\operatorname{argmax}}\, p(\boldsymbol{\theta}|y, \boldsymbol{x}) = \underset{\theta \in \Theta}{\operatorname{argmax}}\, p(y|\boldsymbol{x}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}) \tag{A.12}$$

Finally, given that $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ is the likelihood of $\boldsymbol{\theta}$ (see Definition 3), we can assume *i.i.d.* samples and apply the negative log-likelihood trick to rewrite Equation (A.12) as in Equation (A.13).

$$\hat{\boldsymbol{\theta}}_{MAP} = \underset{\theta \in \Theta}{\operatorname{argmin}} - \left[\sum_{i=1}^{n} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \right] \tag{A.13}$$

Fixing the two distributions, the predictive model can be achieved solving the minimization problem in Equation (A.13). For the solution of this minimization problem the same observations provided at the end of the last section for Equation (A.6) hold.

A.4 ERM vs MLE/MAP

The goal of this last section is to show that the approaches described in Section A.2 and in Section A.3 look very different, but they actually are two sides of the same coin.

Assuming that we have the usual collection of *i.i.d.* samples $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, where $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\ \forall\ i = 1, \ldots, n$, our aim here is to learn a good input-output relationship $f : \mathcal{X} \to \mathcal{Y}$. The function f may depend from some parameters that, for ease of writing, will be temporary omitted. If we decide to proceed by MLE, we can find $\hat{f}_{MLE}$ solving the optimization problem in Equation (A.14).

$$\hat{f}_{MLE} = \underset{f \in \mathcal{F}}{\operatorname{argmax}} \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, f) \tag{A.14}$$

Applying the negative log-likelihood trick, Equation (A.14) can be rewritten as Equation (A.15)[4].

$$\hat{f}_{MLE} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} - \frac{1}{n} \sum_{i=1}^{n} \log p(y_i|\boldsymbol{x}_i, f) \tag{A.15}$$

[4]the two minimization problems $\min_x f(x)$ and $\min_x \alpha f(x)$ have the same solution if α is a constant that does not depend from x

As we can see here, we are naming *negative log-likelihood* what in Section A.2 was called *loss function*. In fact, using $L(f(\boldsymbol{x}), y) = -\log p(y|\boldsymbol{x}, f)$ Equation (A.15) can be rewritten as in Equation (A.16), which is the ERM problem.

$$\hat{f}_{ERM} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \tag{A.16}$$

In Section A.2 we have seen that introducing $\mathcal{R}(f)$ reduces the space of functions $\mathcal{F}$ and prevents the achieved solution from overfitting. Intuitively, the same effect can be achieved by introducing a prior $p(f)$ as in the MAP estimate. Following Equation (A.13) we can write Equation (A.17).

$$\hat{f}_{MAP} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} -\frac{1}{n} \left[\sum_{i=1}^{n} \log p(y_i|\boldsymbol{x}_i, f) + \lambda \log p(f) \right] \tag{A.17}$$

Finally, as for Equation (A.16), we can express the penalty as $R(f) = -\frac{\lambda}{n} \log p(f)$ and Equation (A.17) becomes Equation (A.18), which is the classical $Loss + Penalty$ formulation of the ERM problem.

$$\hat{f}_{ERM} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) + \lambda R(f) \tag{A.18}$$

In this section an intuitive explanation of the connection between two popular supervised learning approaches is provided. For a more rigorous overview on ERM and MLE/MAP we refer to [Hastie et al., 2009] and to [Rasmussen and Williams, 2006], respectively.

A.4.1 Linear regression revisited

To clarify the connection between ERM and MLE/MAP we can revisit the simple linear regression problem.

Once again, we have a collection of *i.i.d.* samples $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n} = (X, \boldsymbol{y})$, where $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} \; \forall \; i = 1, \ldots, n$ and our aim is to learn an input-output relationship $f : \mathcal{X} \to \mathcal{Y}$. Moreover, we assume that the outputs y_i is affected by additive Gaussian noise, hence $y_i = f(\boldsymbol{x}_i) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$. Interestingly, this corresponds to the assumption that f is modeling the mean of the outputs y_i, while its standard deviation σ_n remains unknown, therefore $y_i \sim \mathcal{N}(f(\boldsymbol{x}_i), \sigma_n^2)$ ($\forall \; i = 1, \ldots, n$). In Section A.2 we have seen that the ERM solution can be estimated as in Equation A.16. For the sake of simplicity we can restrict to the Ridge Regression case (see Section 4.1.1), *i.e.* we look for a model that can be written as $\hat{y} = f(\boldsymbol{x}) = \boldsymbol{x}^T \hat{\boldsymbol{w}}$ minimizing the

square loss $L(\hat{y}, y) = \frac{1}{n} \|y - X\boldsymbol{w}\|_2^2$ penalized by the ℓ_2-norm $\mathcal{R}(\boldsymbol{w}) = \|w\|_2^2$. Therefore, the minimization problem is stated in Equation (A.19).

$$\hat{\boldsymbol{w}}_{\ell_2} = \underset{\hat{\boldsymbol{w}} \in \mathbb{R}^d}{\operatorname{argmin}} J(\boldsymbol{y}, X, \boldsymbol{w}) = \underset{\hat{\boldsymbol{w}} \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2n} \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \tag{A.19}$$

In Section A.4 we have seen that the regularized minimization problem corresponds to a MAP estimate with an appropriate choice for negative log-likelihood and prior distribution (on $\boldsymbol{w}$) which correspond to loss function and regularization penalty, respectively. So, considering $J(\boldsymbol{y}, X, \boldsymbol{w})$ as a negative log-posterior and factoring out λ we can write

$$\exp\left[- J(\boldsymbol{y}, X, \boldsymbol{w})\right] \propto \exp\left[- \frac{1}{2n\lambda} \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2 \right] \cdot \exp\left[- \frac{1}{2} \|w\|_2^2 \right]$$

which can be seen as

$$p(\boldsymbol{w}|\boldsymbol{y}, X) = p(\boldsymbol{y}|X, \boldsymbol{w}) \cdot p(\boldsymbol{w})$$

where $p(\boldsymbol{y}|X, \boldsymbol{w}) = \mathcal{N}(X\boldsymbol{w}, n\lambda I)$ and $p(\boldsymbol{w}) = \mathcal{N}(0, I)$. So, in this probabilistic interpretation, the variance of the noise affecting the output $\boldsymbol{y}$ plays the role of the regularization parameter $\lambda \approx \sigma_n^2$.

A.4.2 Logistic regression revisited

In this section we revisit binary classification via logistic regression from a probabilistic perspective.

In binary classification problems we are provided with a collection of input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n = (X, \boldsymbol{y})$, where $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \{+1, -1\}$, $\forall i = 1, \ldots, n$. Once again we are looking for a model $f : \mathcal{X} \to \mathcal{Y}$ that associates each input sample with its corresponding class. For the sake of simplicity we restrict to the case of linear functions $\hat{y} = f(\boldsymbol{x}) = \boldsymbol{x}^T \hat{\boldsymbol{w}}$.

The main idea behind logistic regression is to use a loss function having a $[0, 1]$ range to estimate the probability that a sample $\boldsymbol{x}_i$ belongs to one of the two classes. As suggested by its name, the function of choice is the logistic $\sigma(z) = [1 + \exp(-z)]^{-1}$.

In this context, we model our outputs y_i as Bernoulli random variables, which implies that

$$P(y = 1|\boldsymbol{x}) = \sigma(-f(\boldsymbol{x})) = \frac{1}{1 + \exp(-f(\boldsymbol{x}))}$$

and

$$P(y = -1|\boldsymbol{x}) = \sigma(f(\boldsymbol{x})) = \frac{1}{1 + \exp(f(\boldsymbol{x}))}$$

consequently, we can write the general form as in Equation (A.20).

$$P(y = \pm 1|\boldsymbol{x}) = \frac{1}{1 + \exp\left(-yf(\boldsymbol{x})\right)} \tag{A.20}$$

Therefore, assuming a Gaussian prior on the weights $\boldsymbol{w} \sim \mathcal{N}(0, I)$, we can perform a MAP estimate of $\hat{f}$ solving the problem in Equation (A.21).

$$\hat{\boldsymbol{w}}_{\text{LR}} = \underset{\hat{\boldsymbol{w}} \in \mathbb{R}^d}{\operatorname{argmax}} \prod_{i=1}^{n} p(\boldsymbol{w}|y_i, \boldsymbol{x}_i) = \underset{\hat{\boldsymbol{w}} \in \mathbb{R}^d}{\operatorname{argmax}} \prod_{i=1}^{n} \frac{1}{1 + \exp\left(-y_i\, \boldsymbol{x}_i^T \boldsymbol{w}\right)} \cdot \exp\left(-\frac{1}{2} \left\|\boldsymbol{w}\right\|_2^2\right) \tag{A.21}$$

Log-transforming Equation (A.21), and applying some elementary linear algebra, we can write Equation (A.22).

$$\hat{\boldsymbol{w}}_{\text{LR}} = \underset{\hat{\boldsymbol{w}} \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2n\lambda} \sum_{i=1}^{n} \log\left[1 + \exp\left(-y_i \boldsymbol{x}_i^T \boldsymbol{w}\right)\right] + \frac{\lambda}{2} \left\|\boldsymbol{w}\right\|_2^2 \tag{A.22}$$

The minimization problem expressed in Equation (A.22) is known as Regularized Logistic Regression. It can be casted in the regularization framework of Equation (A.18) where the logistic loss function $L(f(\boldsymbol{x}), y) = \log\left[1 + \exp\left(-yf(\boldsymbol{x})\right)\right]$ is penalized by the ℓ_2-norm.

List of Figures

List of Tables

List of Abbreviations

API Application Program Interface

CT Computerized Tomography

DNA Deoxyribonucleic Acid

EDA Exploratory-Data Analysis

ERM Empirical Risk Minimization

FISTA Fast Iterative Shrinkage-Thresholding Algorithm

GUI Graphical User Interface

i.i.d. Independent and Identically Distributed

MAE Mean Absolute Error

MAP Maximum A Posteriori

MLE Maximum Likelihood Estimation

MRI Magnetic Resonance Imaging

NGS Next-Generation Sequencing

OLS Ordinary Least Squares

OVA *One-vs-All*

OVO *One-vs-One*

PCO Patient Centered Outcomes

PET Positron Emission Tomography

PP Primary-Progressive Multiple Sclerosis

PR Progressive-Relapsing Multiple Sclerosis

RR Relapsing-Remitting Multiple Sclerosis

SP Secondary-Progressive Multiple Sclerosis

SPECT Single-Photon Emission Computed Tomography

SVC Support Vector Classification

SVM Support Vector Machine

SVR Support Vector Regression

Bibliography

Aben, N., Vis, D. J., Michaut, M., and Wessels, L. F. (2016). Tandem: a two-stage approach to maximize interpretability of drug response models based on multiple molecular data types. *Bioinformatics*, 32(17):i413–i420. [Cited on page 25.]

Abraham, G., Kowalczyk, A., Zobel, J., and Inouye, M. (2013). Performance and robustness of penalized and unpenalized methods for genetic prediction of complex human disease. *Genetic Epidemiology*, 37(2):184–195. [Cited on page 7.]

Alexandrov, L. B., Nik-Zainal, S., Wedge, D. C., Aparicio, S. A., Behjati, S., Biankin, A. V., Bignell, G. R., Bolli, N., Borg, A., Børresen-Dale, A.-L., et al. (2013). Signatures of mutational processes in human cancer. *Nature*, 500(7463):415–421. [Cited on page 8.]

Altmann, A., Toloşi, L., Sander, O., and Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347. [Cited on page 6.]

Angermueller, C., Pärnamaa, T., Parts, L., and Stegle, O. (2016). Deep learning for computational biology. *Molecular systems biology*, 12(7):878. [Cited on pages 7, 8, 33, and 34.]

Appenzeller, T. (2017). The ai revolution in science. *Science*. [Cited on page 10.]

Argyriou, A., Evgeniou, T., and Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, 73(3):243–272. [Cited on page 7.]

Arnould, C., Vandervelde, L., Batcho, C. S., Penta, M., and Thonnard, J.-L. (2012). Can manual ability be measured with a generic abilhand scale? a cross-sectional study conducted on six diagnostic groups. *BMJ open*, 2(6):e001807. [Cited on page 50.]

Aupperle, R. L., Beatty, W. W., Shelton, F. d. N., and Gontkovsky, S. T. (2002). Three screening batteries to detect cognitive impairment in multiple sclerosis. *Multiple Sclerosis*, 8(5):382–389. [Cited on page 50.]

Bach, F., Jenatton, R., Mairal, J., Obozinski, G., et al. (2012). Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106. [Cited on page 61.]

Baldassarre, L., Rosasco, L., Barla, A., and Verri, A. (2012). Multi-output learning via spectral filtering. *Machine learning*, 87(3):259–301. [Cited on page 7.]

Barbieri, M., Fiorini, S., Tomasi, F., and Barla, A. (2016). PALLADIO: a parallel framework for robust variable selection in high-dimensional data. *PyHPC2016 conference, IEEE proceedings*. [Cited on page 52.]

Barrett, T., Wilhite, S. E., Ledoux, P., Evangelista, C., Kim, I. F., Tomashevsky, M., Marshall, K. A., Phillippy, K. H., Sherman, P. M., Holko, M., et al. (2013). Ncbi geo: archive for functional genomics data sets—update. *Nucleic acids research*, 41(D1):D991–D995. [Cited on page 40.]

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202. [Cited on pages 22, 25, and 29.]

Bergamaschi, R., Montomoli, C., Mallucci, G., Lugaresi, A., Izquierdo, G., Grand'Maison, F., Duquette, P., Shaygannejad, V., Alroughani, R., Grammond, P., et al. (2015). Bremso: A simple score to predict early the natural course of multiple sclerosis. *European journal of neurology*, 22(6):981–989. [Cited on page 48.]

Bishop, C. M. (2006). Pattern recognition. *Machine Learning*. [Cited on pages 41, 42, and 61.]

Black, N. (2013). Patient reported outcome measures could help transform healthcare. *BMJ (Clinical research ed)*, 346:f167. [Cited on pages 48 and 56.]

Borg, I. and Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media. [Cited on page 41.]

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press. [Cited on pages 17, 19, 28, and 61.]

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32. [Cited on page 31.]

Buehlmann, P. (2006). Boosting for high-dimensional linear models. *The Annals of Statistics*, pages 559–583. [Cited on page 32.]

Button, K. S., Ioannidis, J. P., Mokrysz, C., Nosek, B. A., Flint, J., Robinson, E. S., and Munafò, M. R. (2013). Power failure: why small sample size undermines the reliability of neuroscience. *Nature Reviews Neuroscience*, 14(5):365–376. [Cited on page 6.]

Candès, E. J. and Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717. [Cited on page 9.]

Cardozo, L., Staskin, D., Currie, B., Wiklund, I., Globe, D., Signori, M., Dmochowski, R., MacDiarmid, S., Nitti, V. W., and Noblett, K. (2014). Validation of a bladder symptom screening tool in women with incontinence due to overactive bladder. *International urogynecology journal*, 25(12):1655–1663. [Cited on page 50.]

Chen, L., Cai, C., Chen, V., and Lu, X. (2015). Trans-species learning of cellular signaling systems with bimodal deep belief networks. *Bioinformatics*, 31 18:3008–15. [Cited on page 33.]

Chen, X., He, J., Lawrence, R., and Carbonell, J. G. (2012). Adaptive multi-task sparse learning with an application to fmri study. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 212–223. SIAM. [Cited on page 25.]

Chen, Y., Li, Y., Narayan, R., Subramanian, A., and Xie, X. (2016). Gene expression inference with deep learning. *Bioinformatics*, 32 12:1832–9. [Cited on pages 33 and 34.]

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619. [Cited on pages 41 and 42.]

Consortium, E. P. et al. (2004). The encode (encyclopedia of dna elements) project. *Science*, 306(5696):636–640. [Cited on page 5.]

Dagenais, E., Rouleau, I., Demers, M., Jobin, C., Roger, É., Chamelian, L., and Duquette, P. (2013). Value of the moca test as a screening instrument in multiple sclerosis. *The Canadian Journal of Neurological Sciences*, 40(03):410–415. [Cited on page 50.]

De Mol, C., De Vito, E., and Rosasco, L. (2009a). Elastic-net regularization in learning theory. *Journal of Complexity*, 25(2):201–230. [Cited on page 24.]

De Mol, C., Mosci, S., Traskine, M., and Verri, A. (2009b). A regularized method for selecting nested groups of relevant genes from microarray data. *Journal of Computational Biology*, 16(5):677–690. [Cited on pages 23 and 25.]

De Souto, M. C., Jaskowiak, P. A., and Costa, I. G. (2015). Impact of missing data imputation methods on gene expression clustering and classification. *BMC bioinformatics*, 16(1):64. [Cited on page 41.]

Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., et al. (2013). Orange: data mining toolbox in python. *The Journal of Machine Learning Research*, 14(1):2349–2353. [Cited on page 40.]

Deng, H. and Runger, G. (2013). Gene selection with guided regularized random forest. *Pattern Recognition*, 46(12):3483–3489. [Cited on page 31.]

Dickens, A. M., Larkin, J. R., Griffin, J. L., Cavey, A., Matthews, L., Turner, M. R., Wilcock, G. K., Davis, B. G., Claridge, T. D., Palace, J., et al. (2014). A type 2 biomarker separates relapsing-remitting from secondary progressive multiple sclerosis. *Neurology*, 83(17):1492–1499. [Cited on page 48.]

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231. [Cited on pages 41 and 42.]

Evgeniou, T., Pontil, M., and Poggio, T. (2000). Regularization networks and support vector machines. *Advances in computational mathematics*, 13(1):1–50. [Cited on pages 14, 17, 20, 29, and 61.]

Fakhry, A., Peng, H., and Ji, S. (2016). Deep models for brain em image segmentation: novel insights and improved performance. *Bioinformatics*, 32 15:2352–8. [Cited on page 34.]

Fiorini, S., Verri, A., Tacchino, A., Ponzio, M., Brichetto, G., and Barla, A. (2015). A machine learning pipeline for multiple sclerosis course detection from clinical scales and patient reported outcomes. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pages 4443–4446. IEEE. [Cited on page 49.]

Flachenecker, P., Kümpfel, T., Kallmann, B., Gottschalk, M., Grauer, O., Rieckmann, P., Trenkwalder, C., and Toyka, K. (2002). Fatigue in multiple sclerosis: a comparison of different rating scales and correlation to clinical parameters. *Multiple sclerosis*, 8(6):523–526. [Cited on page 50.]

Franchignoni, F., Tesio, L., Ottonello, M., and Benevolo, E. (1999). Life satisfaction index: Italian version and validation of a short form1. *American journal of physical medicine & rehabilitation*, 78(6):509–515. [Cited on page 50.]

Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *science*, 315(5814):972–976. [Cited on pages 41 and 42.]

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232. [Cited on page 32.]

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378. [Cited on page 32.]

Garg, R. P., Dong, S., Shah, S. J., and Jonnalagadda, S. R. (2016). A bootstrap machine learning approach to identify rare disease patients from electronic health records. *CoRR*, abs/1609.01586. [Cited on page 6.]

Giovannoni, G., Butzkueven, H., Dhib-Jalbut, S., Hobart, J., Kobelt, G., Pepper, G., Sormani, M. P., Thalheim, C., Traboulsee, A., and Vollmer, T. (2016). Brain health: time matters in multiple sclerosis. *Multiple Sclerosis and Related Disorders*, 9:S5–S48. [Cited on page 48.]

Gramfort, A., Kowalski, M., and Hämäläinen, M. (2012). Mixed-norm estimates for the m/eeg inverse problem using accelerated gradient methods. *Physics in medicine and biology*, 57(7):1937. [Cited on page 22.]

Granger, C., Cotter, A., Hamilton, B., Fiedler, R., and Hens, M. (1990). Functional assessment scales: a study of persons with multiple sclerosis. *Archives of physical medicine and rehabilitation*, 71(11):870–875. [Cited on page 50.]

Gui, J. and Li, H. (2005). Penalized cox regression analysis in the high-dimensional and low-sample size settings, with applications to microarray gene expression data. *Bioinformatics*, 21(13):3001–3008. [Cited on page 22.]

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182. [Cited on pages 21 and 53.]

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422. [Cited on pages 6, 8, and 53.]

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288. [Cited on page 41.]

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning*, volume 2. Springer. [Cited on pages 8, 18, 20, 29, 31, 32, 40, 41, 42, 44, 52, and 64.]

Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical learning with sparsity: the lasso and generalizations*. CRC Press. [Cited on pages 8 and 53.]

He, D., Kuhn, D., and Parida, L. (2016a). Novel applications of multitask learning and multiple output regression to multiple genetic trait prediction. *Bioinformatics*, 32(12):i37–i43. [Cited on page 7.]

He, K., Li, Y., Zhu, J., Liu, H., Lee, J. E., Amos, C. I., Hyslop, T., Jin, J., Lin, H., Wei, Q., et al. (2016b). Component-wise gradient boosting and false discovery control in survival analysis with high-dimensional covariates. *Bioinformatics*, 32(1):50–57. [Cited on page 32.]

Helmstaedter, M., Briggman, K. L., Turaga, S. C., Jain, V., Seung, H. S., and Denk, W. (2013). Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168–174. [Cited on page 7.]

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67. [Cited on page 18.]

Hofner, B., Boccuto, L., and Göker, M. (2015). Controlling false discoveries in high-dimensional situations: boosting with stability selection. *BMC bioinformatics*, 16(1):144. [Cited on page 32.]

Hoggart, C. J., Whittaker, J. C., De Iorio, M., and Balding, D. J. (2008). Simultaneous analysis of all snps in genome-wide and re-sequencing association studies. *PLoS genetics*, 4(7):e1000130. [Cited on page 24.]

Honarmand, K. and Feinstein, A. (2009). Validation of the hospital anxiety and depression scale for use with multiple sclerosis patients. *Multiple Sclerosis*. [Cited on page 50.]

Hughey, J. J. and Butte, A. J. (2015). Robust meta-analysis of gene expression using the elastic net. *Nucleic Acids Research*, 43(12):e79. [Cited on page 25.]

Ishwaran, H., Kogalur, U. B., Blackstone, E. H., and Lauer, M. S. (2008). Random Survival Forests. *The Annals of Applied Statistics*, 2(3):841–860. [Cited on page 31.]

Jack, C. R., Bernstein, M. A., Fox, N. C., Thompson, P., Alexander, G., Harvey, D., Borowski, B., Britson, P. J., L Whitwell, J., Ward, C., et al. (2008). The alzheimer's disease neuroimaging initiative (adni): Mri methods. *Journal of magnetic resonance imaging*, 27(4):685–691. [Cited on page 5.]

Jacob, L., Obozinski, G., and Vert, J.-P. (2009). Group lasso with overlap and graph lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 433–440, New York, NY, USA. ACM. [Cited on page 28.]

Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library. [Cited on page 41.]

Joly, A., Schnitzler, F., Geurts, P., and Wehenkel, L. (2012). L1-based compression of random forest models. In *20th European Symposium on Artificial Neural Networks*. [Cited on page 31.]

Jung, K., Dihazi, H., Bibi, A., Dihazi, G. H., and Beißbarth, T. (2014). Adaption of the global test idea to proteomics data with missing values. *Bioinformatics*, 30(10):1424–1430. [Cited on page 9.]

Kolker, E., Higdon, R., Haynes, W., Welch, D., Broomall, W., Lancet, D., Stanberry, L., and Kolker, N. (2012). Moped: model organism protein expression database. *Nucleic acids research*, 40(D1):D1093–D1099. [Cited on page 5.]

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957. [Cited on pages 20 and 33.]

Kulkarni, V. Y. and Sinha, P. K. (2012). Pruning of Random Forest classifiers: A survey and future directions. In *2012 International Conference on Data Science Engineering (ICDSE)*, pages 64–68. [Cited on page 31.]

Kursa, M. B. (2014). Robustness of Random Forest-based gene selection methods. *BMC Bioinformatics*, 15:8. [Cited on page 32.]

Kurtzke, J. F. (1983). Rating neurologic impairment in multiple sclerosis an expanded disability status scale (edss). *Neurology*, 33(11):1444–1444. [Cited on page 50.]

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444. [Cited on pages 33 and 49.]

Lee, S., Zhu, J., and Xing, E. P. (2010). Adaptive multi-task lasso: with application to eqtl detection. In *Advances in neural information processing systems*, pages 1306–1314. [Cited on page 22.]

Leung, M. K., Xiong, H. Y., Lee, L. J., and Frey, B. J. (2014). Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129. [Cited on pages 33 and 34.]

Lewis, J. M., De Sa, V. R., and Van Der Maaten, L. (2013). Divvy: fast and intuitive exploratory data analysis. *The Journal of Machine Learning Research*, 14(1):3159–3163. [Cited on page 40.]

Liu, S., Dissanayake, S., Patel, S., Dang, X., Mlsna, T., Chen, Y., and Wilkins, D. (2014a). Learning accurate and interpretable models based on regularized random forests regression. *BMC Systems Biology*, 8(3):S5. [Cited on page 31.]

Liu, Y., Li, B., Tan, R., Zhu, X., and Wang, Y. (2014b). A gradient boosting approach for filtering de novo mutations in parent-offspring trios. *Bioinformatics*, page btu141. [Cited on page 32.]

Lublin, F. D., Reingold, S. C., Cohen, J. A., Cutter, G. R., Sørensen, P. S., Thompson, A. J., Wolinsky, J. S., Balcer, L. J., Banwell, B., Barkhof, F., et al. (2014). Defining the clinical course of multiple sclerosis the 2013 revisions. *Neurology*, 83(3):278–286. [Cited on page 48.]

Lusa, L. et al. (2015). Boosting for high-dimensional two-class prediction. *BMC bioinformatics*, 16(1):300. [Cited on page 32.]

Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E., and Svetnik, V. (2015). Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274. [Cited on page 33.]

Ma, S. and Huang, J. (2007). Additive risk survival model with microarray data. *BMC bioinformatics*, 8(1):192. [Cited on page 22.]

Mamoshina, P., Vieira, A., Putin, E., and Zhavoronkov, A. (2016). Applications of deep learning in biomedicine. *Molecular pharmaceutics*, 13(5):1445–1454. [Cited on page 8.]

Marx, V. (2013). Biology: The big challenges of big data. *Nature*, 498(7453):255–260. [Cited on page 10.]

Masecchia, S., Coco, S., Barla, A., Verri, A., and Tonini, G. P. (2015). Genome iny model of metastatic neuroblastoma tumorigenesis by a dictionary learning algorithm. *BMC medical genomics*, 8(1):57. [Cited on page 8.]

Mayr, A., Binder, H., Gefeller, O., Schmid, M., et al. (2014). The evolution of boosting algorithms. *Methods of Information in Medicine*, 53(6):419–427. [Cited on page 32.]

McNeish, D. M. and Stapleton, L. M. (2016). The effect of small sample size on two-level model estimates: A review and illustration. *Educational Psychology Review*, 28(2):295–314. [Cited on page 6.]

Meier, L., Van De Geer, S., and Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71. [Cited on page 8.]

Meinshausen, N. and Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473. [Cited on pages 24, 32, and 52.]

Meyer-Moock, S., Feng, Y.-S., Maeurer, M., Dippel, F.-W., and Kohlmann, T. (2014). Systematic literature review and validity evaluation of the expanded disability status scale (edss) and the multiple sclerosis functional composite (msfc) in patients with multiple sclerosis. *BMC neurology*, 14(1):58. [Cited on page 49.]

Min, S., Lee, B., and Yoon, S. (2016). Deep learning in bioinformatics. *arXiv preprint arXiv:1603.06430*. [Cited on page 7.]

Molinaro, A. M., Simon, R., and Pfeiffer, R. M. (2005). Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307. [Cited on page 52.]

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press. [Cited on page 10.]

Nelson, E. C., Eftimovska, E., Lind, C., Hager, A., Wasson, J. H., and Lindblad, S. (2015). Patient reported outcome measures in practice. *Bmj*, 350:g7818. [Cited on page 49.]

Nesterov, Y. (2013). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media. [Cited on page 61.]

Ng, A. Y., Jordan, M. I., Weiss, Y., et al. (2002). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856. [Cited on page 41.]

Nowak, G., Hastie, T., Pollack, J. R., and Tibshirani, R. (2011). A fused lasso latent feature model for analyzing multi-sample acgh data. *Biostatistics*, page kxr012. [Cited on page 49.]

Okser, S., Pahikkala, T., Airola, A., Salakoski, T., Ripatti, S., and Aittokallio, T. (2014). Regularized machine learning in the genetic prediction of complex traits. *PLoS Genet*, 10(11):e1004754. [Cited on pages 6 and 7.]

Oldfield, R. C. (1971). The assessment and analysis of handedness: the edinburgh inventory. *Neuropsychologia*, 9(1):97–113. [Cited on page 50.]

Parmenter, B., Weinstock-Guttman, B., Garg, N., Munschauer, F., and Benedict, R. H. (2007). Screening for cognitive impairment in multiple sclerosis using the symbol digit modalities test. *Multiple Sclerosis*, 13(1):52–57. [Cited on page 50.]

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. [Cited on page 40.]

Peng, B., Wang, L., and Wu, Y. (2016). An error bound for l1-norm support vector machine coefficients in ultra-high dimension. *The Journal of Machine Learning Research*, 17(1):8279–8304. [Cited on page 30.]

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer. [Cited on page 34.]

Qi, Y. (2012). Random forest for bioinformatics. In *Ensemble machine learning*, pages 307–323. Springer. [Cited on pages 32 and 49.]

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, volume 1. MIT press Cambridge. [Cited on pages 11 and 64.]

Ross, D. A., Lim, J., Lin, R.-S., and Yang, M.-H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141. [Cited on page 41.]

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65. [Cited on pages 42 and 44.]

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326. [Cited on page 41.]

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*. [Cited on page 34.]

Scalfari, A., Neuhaus, A., Daumer, M., Muraro, P. A., and Ebers, G. C. (2014). Onset of secondary progressive phase and long-term evolution of multiple sclerosis. *Journal of Neurology, Neurosurgery & Psychiatry*, 85(1):67–75. [Cited on page 48.]

Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel principal component analysis. In *Artificial Neural Networks—ICANN'97*, pages 583–588. Springer. [Cited on pages 40 and 41.]

Schulz, W. (2005). *Molecular biology of human cancers: an advanced student's textbook*. Springer Science & Business Media. [Cited on pages 43 and 44.]

Service, R. F. (2017). Ai is changing how we do science. get a glimpse. *Science*. [Cited on page 10.]

Shawe-Taylor, J. and Sun, S. (2011). A review of optimization methodologies in support vector machines. *Neurocomputing*, 74(17):3609–3618. [Cited on page 30.]

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905. [Cited on pages 41 and 42.]

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222. [Cited on page 30.]

Sra, S., Nowozin, S., and Wright, S. J. (2012). *Optimization for machine learning*. Mit Press. [Cited on pages 17, 19, 28, and 61.]

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958. [Cited on page 34.]

Stekhoven, D. J. and Bühlmann, P. (2011). Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118. [Cited on page 9.]

Tang, Z., Shen, Y., Zhang, X., and Yi, N. (2017). The spike-and-slab lasso cox model for survival prediction and associated genes detection. *Bioinformatics*, page btx300. [Cited on page 22.]

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323. [Cited on page 41.]

Teramoto, R. et al. (2009). Balanced gradient boosting from imbalanced data for clinical outcome prediction. *Statistical applications in genetics and molecular biology*, 8(1):1–19. [Cited on page 49.]

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288. [Cited on page 21.]

Tibshirani, R. et al. (1997). The lasso method for variable selection in the cox model. *Statistics in medicine*, 16(4):385–395. [Cited on page 22.]

Tikhonov, A. (1963). Solution of incorrectly formulated problems and the regularization method. In *Soviet Math. Dokl.*, volume 5, pages 1035–1038. [Cited on pages 14, 20, and 61.]

Toga, A. W. and Dinov, I. D. (2015). Sharing big biomedical data. *Journal of big data*, 2(1):7. [Cited on page 5.]

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., and Altman, R. B. (2001). Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525. [Cited on pages 9, 41, and 52.]

Tutz, G. and Binder, H. (2006). Generalized additive modeling with implicit variable selection by likelihood-based boosting. *Biometrics*, 62(4):961–971. [Cited on page 32.]

Tutz, G. and Binder, H. (2007). Boosting ridge regression. *Computational Statistics & Data Analysis*, 51(12):6044–6059. [Cited on page 32.]

Uitdehaag, B. (2014). Clinical outcome measures in multiple sclerosis. *Handb Clin Neurol*, 122:393–404. [Cited on page 49.]

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85. [Cited on pages 16, 41, and .]

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media. [Cited on pages 30 and 60.]

Vukusic, S. and Confavreux, C. (2003). Prognostic factors for progression of disability in the secondary progressive phase of multiple sclerosis. *Journal of the neurological sciences*, 206(2):135–137. [Cited on page 48.]

Waldmann, P., Mészáros, G., Gredler, B., Fuerst, C., and Sölkner, J. (2013). Evaluation of the lasso and the elastic net in genome-wide association studies. *Frontiers in genetics*, 4:270. [Cited on pages 23 and 25.]

Witten, D. M. and Tibshirani, R. (2009). Covariance-regularized regression and classification for high dimensional problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):615–636. [Cited on page 28.]

Wu, T. T., Chen, Y. F., Hastie, T., Sobel, E., and Lange, K. (2009). Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics*, 25(6):714–721. [Cited on page 22.]

Wu, T. T. and Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, pages 224–244. [Cited on pages 22, 25, and 29.]

Yu, D., Huber, W., and Vitek, O. (2013). Shrinkage estimation of dispersion in negative binomial models for rna-seq experiments with small sample size. *Bioinformatics*, 29(10):1275–1282. [Cited on page 6.]

Yuan, Y., Shi, Y., Li, C., Kim, J., Cai, T. W., Han, Z., and Feng, D. D. (2016). Deepgene: an advanced cancer type classifier based on deep learning and somatic point mutations. In *BMC Bioinformatics*. [Cited on page 33.]

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC press. [Cited on page 30.]

Zhu, J., Rosset, S., Tibshirani, R., and Hastie, T. J. (2004). 1-norm support vector machines. In *Advances in neural information processing systems*, pages 49–56. [Cited on page 30.]

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429. [Cited on page 24.]

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320. [Cited on pages 24, 25, and 49.]