VARIABLE SELECTION ALGORITHMS IN GENERALIZED LINEAR MODELS

Juan Carlos Laria de la Cruz

A DISSERTATION

Submitted in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

(Mathematical Engineering)

Universidad Carlos III de Madrid

Advisors

Rosa E. Lillo
M. Carmen Aguilera-Morillo

Tutor

Rosa E. Lillo

June, 2020

Variable selection algorithms in generalized linear models

A DISSERTATION
June, 2020

By
Juan Carlos Laria de la Cruz

Published by: UC3M, Department of Statistics, Av. Universidad 30, 28912 Leganés, Madrid Spain
www.uc3m.es

A mi madre

Acknowledgements

None of this work would have been possible without the continuous support, expert advice, motivation, constant dedication, and commitment of my thesis advisors and friends, Professors Rosa Lillo and Carmen Aguilera. I am deeply grateful to them.

I would also like to thank my coauthors, and coworkers at the Department of Statistics. Many people have given me relevant suggestions and insights at different points during these years.

Me being here has required so much patience and sacrifice by those involved in my education, that this work is theirs too. Not only my grandmother and my parents but also my school and university teachers authored this thesis.

Special thanks to my family and my friends, who were always there for me, unconditionally, when I needed them.

Contents published and presented

The materials from the following sources are included in the thesis. Their inclusion is not indicated by typographical means or references, since they are fully embedded in each chapter indicated below.

Paper A. Laria Juan C., Aguilera-Morillo M. Carmen and Lillo Rosa E. (2019). "An iterative sparse-group lasso". In: *Journal of Computational and Graphical Statistics*, *28*(3), 722-731.

- url: https://doi.org/10.1080/10618600.2019.1573687
- Included in: Chapter 2 (Full)

Paper B. Laria Juan C., Aguilera-Morillo M. Carmen, Álvarez Enrique, Lillo Rosa E., López-Taruella Sara, del Monte-Millán María, Picornell Antonio C., Martín Miguel and Romo Juan (2020). "Iterative variable selection for high-dimensional data: Prediction of pathological response in triple-negative breast cancer".

- url: http://hdl.handle.net/10016/30572
- Included in: Chapter 3 (Full)

Paper C. Laria Juan C., Aguilera-Morillo M. Carmen and Lillo Rosa E. (2020). "A variable selection and clustering method for generalized linear models"

- Included in: Chapter 4 (Full)
- Note: Submitted for publication

Paper D. Laria Juan C., Clemmensen Line H., Ersbøll Bjarne K. (2020). "A generalized linear joint trained framework for semi-supervised leaning of sparse features". *arXiv preprint arXiv:2006.01671*.

- url: http://arxiv.org/abs/2006.01671
- Included in: Chapter 5 (Full)
- Note: Submitted for publication

Further research achievements

Papers

1. Delgado-Gómez D., Laria J.C. and Ruiz-Hernández, D. (2019). "Computerized adaptive test and decision trees: a unifying approach". *Expert Systems with Applications, 117*, 358-366.
2. Corujo J.M., Valdés J.E. and Laria J.C. (2019). "Stochastic comparisons of two-unit repairable systems". *Communications in Statistics-Theory and Methods, 48(23)*, 5820-5838.

Software

1. *s2net: The Generalized Semi-Supervised Elastic-Net*. Maintainer. https://cran.r-project.org/package=s2net
2. *cat.dt: Computerized Adaptive Testing and Decision Trees*. Coauthor. https://cran.r-project.org/package=cat.dt

Abstract

This thesis has been developed at University Carlos III of Madrid, motivated through a collaboration with the Gregorio Marañón General University Hospital, in Madrid. It is framed within the field of Penalized Linear Models, specifically Variable Selection in Regression, Classification and Survival Models, but it also explores other techniques such as Variable Clustering and Semi-Supervised Learning.

In recent years, variable selection techniques based on penalized models have gained considerable importance. With the advance of technologies in the last decade, it has been possible to collect and process huge volumes of data with algorithms of greater computational complexity. However, although it seemed that models that provided simple and interpretable solutions were going to be definitively displaced by more complex ones, they have still proved to be very useful. Indeed, in a practical sense, a model that is capable of filtering important information, easily extrapolated and interpreted by a human, is often more valuable than a more complex model that is incapable of providing any kind of feedback on the underlying problem, even when the latter offers better predictions.

This thesis focuses on high dimensional problems, in which the number of variables is of the same order or larger than the sample size. In this type of problems, restrictions that eliminate variables from the model often lead to better performance and interpretability of the results. To adjust linear regression in high dimension the Sparse Group Lasso regularization method has proven to be very efficient. However, in order to use the Sparse Group Lasso in practice, there are two critical aspects on which the solution depends: the correct selection of the regularization parameters, and a prior specification of groups of variables. Very little research has focused on algorithms for the selection of the regularization parameters of the Sparse Group Lasso, and none has explored the issue of the grouping and how to relax this restriction that in practice is an obstacle to using this method.

The main objective of this thesis is to propose new methods of variable selection in generalized linear models. This thesis explores the

Sparse Group Lasso regularization method, analyzing in detail the correct selection of the regularization parameters, and finally relaxing the problem of group specification by introducing a new variable clustering algorithm based on the Sparse Group Lasso, but much more flexible and that extends it. In a parallel but related line of research, this thesis reveals a connection between penalized linear models and semi-supervised learning.

This thesis is structured as a compendium of articles, divided into four chapters. Each chapter has a structure and contents independent from the rest, however, all of them follow a common line. First, variable selection methods based on regularization are introduced, describing the optimization problem that appears and a numerical algorithm to approximate its solution when a term of the objective function is not differentiable. The latter occurs naturally when penalties inducing variable selection are added. A contribution of this work is the iterative Sparse Group Lasso, which is an algorithm to obtain the estimation of the coefficients of the Sparse Group Lasso model, without the need to specify the regularization parameters. It uses coordinate descent for the parameters, while approximating the error function in a validation sample. Moreover, with respect to the traditional Sparse Group Lasso, this new proposal considers a more general penalty, where each group has a flexible weight. A separate chapter presents an extension that uses the iterative Sparse Group Lasso to order the variables in the model according to a defined importance index. The introduction of this index is motivated by problems in which there are a large number of variables, only a few of which are directly related to the response variable. This methodology is applied to genetic data, revealing promising results. A further significant contribution of this thesis is the Group Linear Algorithm with Sparse Principal decomposition, which is also motivated by problems in which only a small number of variables influence the response variable. However, unlike other methodologies, in this case the relevant variables are not necessarily among the observed data. This makes it a potentially powerful method, adaptable to multiple scenarios, which is also, as a side effect, a supervised variable clustering algorithm. Moreover, it can be interpreted as an extension of the Sparse Group Lasso that does not require an initial specification of the groups. From a computational

point of view, this paper presents an organized framework for solving problems in which the objective function is a linear combination of a differentiable error term and a penalty. The flexibility of this implementation allows it to be applied to problems in very different contexts, for example, the proposed Generalized Elastic Net for semi-supervised learning.

Regarding its main objective, this thesis offers a framework for the exploration of generalized interpretable models. In the last chapter, in addition to compiling a summary of the contributions of the thesis, future lines of work in the scope of the thesis are included.

Resumen

Esta tesis se ha desarrollado en la Universidad Carlos III de Madrid motivada por una colaboración de investigación con el Hospital General Universitario Gregorio Marañón, en Madrid. Está enmarcada dentro del campo de los Modelos Lineales Penalizados, concretamente Selección de Variables en Modelos de Regresión, Clasificación y Supervivencia, pero también explora otras técnicas como Clustering de Variables y Aprendizaje Semi-Supervisado.

En los últimos años, las técnicas de selección de variables basadas en modelos penalizados han cobrado notable importancia. Con el avance de las tecnologías en la última década, se ha conseguido recopilar y tratar enormes volúmenes de datos con algoritmos de una complejidad computacional superior. Sin embargo, aunque parecía que los modelos que aportaban soluciones sencillas e interpretables iban a ser definitivamente desplazados por otros más complejos, han resultado ser todavía muy útiles. De hecho, en un sentido práctico, muchas veces tiene más valor un modelo que sea capaz de filtrar información importante, fácilmente extrapolable e interpretable por un humano, que otro más complejo incapaz de aportar ningún tipo de retroalimentación al problema de fondo, incluso cuando este último ofrezca mejores predicciones.

Esta tesis se enfoca en problemas de alta dimensión, en los cuales el número de variables es del mismo orden o superior al tamaño muestral. En este tipo de problemas, restricciones que eliminen variables del modelo a menudo conducen a un mejor desempeño e interpretabilidad de los resultados. Para ajustar regresión lineal en alta dimensión el método de regularización Sparse Group Lasso ha demostrado ser muy eficiente. No obstante, para utilizar en la práctica el Sparse Group Lasso, hay que tener en cuenta dos aspectos fundamentales de los cuales depende la solución, que son la correcta selección de los parámetros de regularización, y una especificación previa de grupos de variables. Muy pocas investigaciones se han centrado en algoritmos para la selección de los parámetros de regularización del Sparse Group Lasso, y ninguna ha explorado el tema de la agrupación y cómo relajar esta restricción que en la práctica constituye una barrera para utilizar este método.

El principal objetivo de esta tesis es proponer nuevos métodos de selección de variables en modelos lineales generalizados. Esta tesis explora el método de regularización Sparse Group Lasso, analizando detalladamente la correcta selección de los parámetros de regularización, y finalmente relajando el problema de la especificación de los grupos mediante un nuevo algoritmo de agrupación de variables basado en el Sparse Group Lasso, pero mucho más flexible y que lo extiende. En una línea de investigación paralela, pero relacionada, esta tesis revela una conexión entre los modelos lineales penalizados y el aprendizaje semi-supervisado.

Esta tesis está estructurada en formato por compendio de artículos, dividida en cuatro capítulos. Cada capítulo tiene una estructura y contenidos independiente del resto, sin embargo, siguen todos un eje común. Primeramente, se introducen los métodos de selección de variables basados en regularización, describiendo el problema de optimización que aparece y un algoritmo numérico para aproximar su solución cuando una parte de la función objetivo no es diferenciable. Esto último ocurre de manera natural cuando se añaden penalizaciones que inducen selección de variables. Una de las aportaciones de este trabajo es el iterative Sparse Group Lasso, que es un algoritmo para obtener la estimación de los coeficientes del modelo Sparse Group Lasso, sin la necesidad de especificar los parámetros de regularización. Utiliza descenso por coordenadas para los parámetros, mientras aproxima la función de error en una muestra de validación. Además, con respecto al Sparse Group Lasso clásico, esta nueva propuesta considera una penalización más general, donde cada grupo tiene un peso flexible. En otro capítulo se presenta una extensión que utiliza el iterative Sparse Group Lasso para ordenar las variables del modelo según un índice de importancia definido. La introducción de este índice está motivada por problemas en los cuales hay un número elevado de variables, de las cuales solamente unas pocas están relacionadas directamente con la variable respuesta. Esta metodología es aplicada a unos datos genéticos, mostrando resultados prometedores. Otra importante aportación de esta tesis es el Group Linear Algorithm with Sparse Principal decomposition, que está motivado también por problemas en los cuales solamente un número reducido de variables influye en la variable respuesta. Sin embargo, a dife-

rencia de otras metodologías, en este caso las variables influyentes no necesariamente están entre las características observadas. Esto lo convierte en un método muy potente, adaptable a múltiples escenarios, que además, como efecto secundario, es un algoritmo supervisado de agrupación de variables. En un sentido, puede interpretarse como una extensión del Sparse Group Lasso que no requiere una especificación inicial de los grupos. Desde un punto de vista computacional, este trabajo presenta un enfoque organizado para resolver problemas en los cuales la función objetivo es una combinación lineal de un término de error diferenciable y una penalización. La flexibilidad de esta implementación le permite ser aplicada a problemas en contextos muy diferentes, por ejemplo, el Generalized Elastic Net propuesto para aprendizaje semi-supervisado.

Con relación a su principal objetivo, esta tesis ofrece un marco para la investigación de modelos generalizados interpretables. En el último capítulo, además de recopilarse un resumen de las aportaciones de la tesis, se incluyen líneas de trabajo futuro en el ámbito de la temática de la tesis.

Contents

CHAPTER 1

Introduction

This chapter introduces basic concepts and results, assumed to be known or referenced in the rest of the thesis. This chapter is organized as follows. Section 1.1 introduces regression models from a general perspective, with particular emphasis on their definition as a convex optimization problem, and this approach is subsequently adopted in all chapters. Furthermore, Section 1.1 discusses general techniques for solving this type of problem using gradient descent. After that, Section 1.2 links together regression problems and variable selection methods, explaining the consequences of including regularization terms in the optimization of a regression problem, outlining the Lasso and the Elastic Net variable selection techniques. Computational approaches to address these problems are also covered. Section 1.3 describes the structure of this thesis, providing the links between different chapters.

1.1 Regression models

This section describes the classical regression models, introducing notation and basic concepts. Under the general linear regression framework, there are N observations of p variables, organized in a data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times p}$. The rows of $\boldsymbol{X}$ are denoted by $\boldsymbol{x}_i^\top$, with $1 \leq i \leq N$,

and the columns of $\boldsymbol{X}$ are samples from the random variables X_j, with $1 \leq j \leq p$. Thus, each row $\boldsymbol{x}_i^\top$ is one sample of the random vector $X^\top = (X_1, X_2, \ldots X_p)$. The matrix $\boldsymbol{X}$ is assumed to be already transformed into an appropriate numeric format, where the columns of $\boldsymbol{X}$ may come from different sources, e.g. quantitative inputs, transformations of quantitative inputs (such as logarithm, square-root and square), basis expansions leading to a polynomial representation ($X_j = X_1^j$), interactions between variables, and numeric or dummy coding of the levels of qualitative inputs. Besides, columns might be normalized or scaled (not all methods require a prior standardization of $\boldsymbol{X}$, when that is the case it will be stated explicitly). Working with missing values in $\boldsymbol{X}$ is out of the scope of this thesis, but most of the methodologies discussed in this chapter can handle missing observations as well.

In supervised learning problems, there is also a matrix of labels $\boldsymbol{y} \in \mathbb{R}^{N \times d}$. Usually, $d = 1$ in which case $\boldsymbol{y}$ is a response vector, but set-ups with more than one response column are possible. This thesis assumes $\boldsymbol{y} \in \mathbb{R}^{N \times 1}$, except in the case of survival with right censoring, where the columns of times and censoring indicators are named explicitly. The vector $\boldsymbol{y}$ is a random sample from the response variable Y.

In regression problems one tries to approximate or model $\boldsymbol{y}$ from the linear predictor $\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta}$. Since both $\boldsymbol{X}$ and $\boldsymbol{y}$ are observed, the objective of regression is to estimate the coefficient vector $\boldsymbol{\beta} \in \mathbb{R}^{p \times 1}$ such that $\boldsymbol{X}\boldsymbol{\beta}$ approximates $\boldsymbol{y}$ the best. In generalized linear models, there is a closed expression (the link function) that approximates $\boldsymbol{\eta}$ from $\boldsymbol{y}$, but one can always estimate $\boldsymbol{\beta}$ without explicitly defining such a link.

The quality of the approximation is measured in terms of a risk function,

$$\begin{array}{rccl} L: & \mathbb{R}^{p \times 1} & \to & \mathbb{R}^+ \\ & \boldsymbol{\beta} & \mapsto & L(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{X}) \geq 0. \end{array} \tag{1.1}$$

This function is often derived from the model's likelihood, written as a function of $\boldsymbol{\beta}$. In any case, regression can be expressed as the optimization problem,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p \times 1}}{\operatorname{argmin}} L(\boldsymbol{\beta}), \tag{1.2}$$

which admits a global minimizer when L is differentiable and convex.

1.1.1 Linear regression

Linear regression assumes an underlying model in the form

$$\mathbb{E}(Y|X) = \beta_0 + X^\top \boldsymbol{\beta} + \epsilon, \tag{1.3}$$

where ϵ is an error, independent of Y and normally distributed with mean zero and variance unknown. Thus, this model is suitable when the response is continuous. Maximizing the model's likelihood as a function of $\boldsymbol{\beta}$ leads to the minimization of,

$$L(\boldsymbol{\beta}) = \frac{1}{N} \left\| \boldsymbol{y} - \beta_0 - \boldsymbol{X}\boldsymbol{\beta} \right\|_2^2, \tag{1.4}$$

which is the objective function of linear regression – the mean square error. To simplify notation, from now on the response vector $\boldsymbol{y}$ in linear regression is assumed to be centered at zero, such that β_0 may be omitted from the equations.

The optimization of (1.2) leads to $\hat{\boldsymbol{\beta}}$, an estimation of the linear model's coefficients. With $\hat{\boldsymbol{\beta}}$, computing predictions of $\boldsymbol{y}$ is straightforward,

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}. \tag{1.5}$$

Example 1. *As an illustration, consider the simple linear model* $Y = X\beta + \epsilon$*, with* $X \sim N(0,1), \beta = 1$*, and* $\epsilon \sim N(0, 0.2)$*. Figure 1.1 displays the linear regression on a random sample of size* $N = 10$*. Notice that, for fixed* $\boldsymbol{X}$ *and* $\boldsymbol{y}$*, the function* $L(\beta)$ *has a minimum in the estimated coefficient (*$\hat{\beta} = 0.869$ *in this case). With enough sample size, the minimum of* $L(\boldsymbol{\beta})$ *will be closer to the true generating coefficient* $\beta = 1$*.*

Example 2. *The intuition behind Example 1 extends to higher dimensions. Figure 1.2 describes the geometry of the function* $L(\boldsymbol{\beta})$ *in dimension two. The data is composed of* $N = 100$ *simulated observations from the model* $Y = X_1\beta_1 + X_2\beta_2 + \epsilon$*, with* $X \sim N(0, \boldsymbol{I}_2)$*,* $\boldsymbol{\beta} = (1, 0)$ *and* $\epsilon \sim N(0, 0.2)$*.*

Figure 1.1: Linear regression on $N = 10$ observations generated from the model $Y = X\beta + \epsilon$.

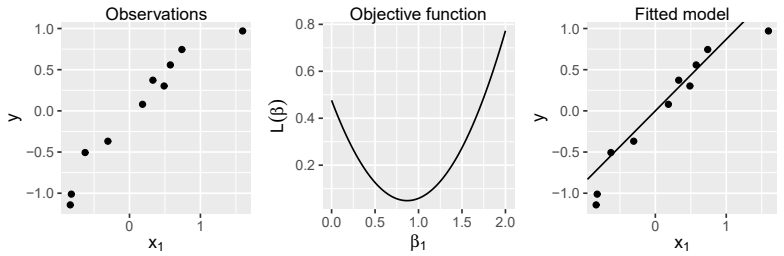
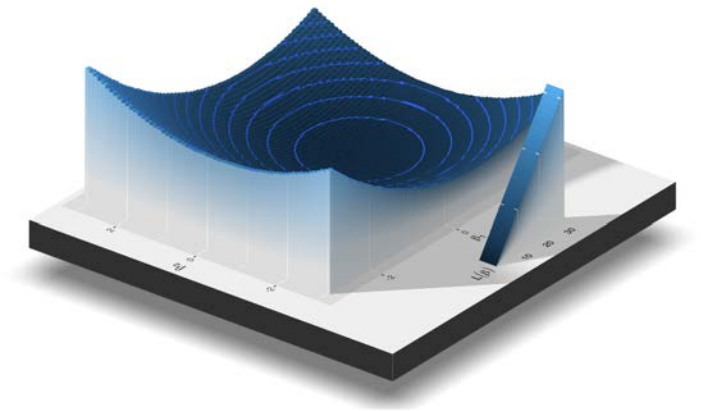


Figure 1.2: Plot of objective function (1.2) with $p = 2$ variables and $N = 100$ observations generated from the model $Y = X^\top \beta + \epsilon$,



1.1.2 Logistic regression

When the response $Y \in \{0, 1\}$, linear regression is not strictly correct, since the normality assumption of $Y|X$ is violated. It is possible to find a solution using linear regression, but logistic regression is more appropriate in this case. The logistic model is formulated as

$$Y|X \sim Ber(p), \text{ where } p = (1 + \exp(-X^\top \boldsymbol{\beta}))^{-1}. \tag{1.6}$$

Maximizing the likelihood as a function of $\boldsymbol{\beta}$ is equivalent to minimizing,

$$L(\boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} \left\{ \log \left[1 + \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}) \right] - \boldsymbol{y}_i \boldsymbol{x}_i^\top \boldsymbol{\beta} \right\}, \tag{1.7}$$

Function (1.7) is commonly called the logistic loss. When $Y \in \{-1, 1\}$, (1.7) has an equivalent expression. This thesis, however, deals with binary classification in the form $\{0, 1\}$ only, but other labels are possible.

In a fitted logistic regression model, making predictions with $\hat{\boldsymbol{\beta}}$ is not as straightforward as in linear regression. There are two possible prediction types one may want from a logistic model, namely class and probability predictions. Suppose $\boldsymbol{x}^\top$ is a new row for which predictions are desired. Then,

$$P(Y = 1|X = \boldsymbol{x}) \approx \hat{p} = (1 + \exp(-\boldsymbol{x}^\top \hat{\boldsymbol{\beta}}))^{-1} \tag{1.8}$$

is the predicted probability of label one. To predict the label, we have to specify a threshold, such that if $\hat{p}$ is above the threshold the predicted class is one and is zero otherwise. Working with predicted probabilities is preferred, since they provide more information than predicted labels.

1.1.3 Proportional hazards model with right-censoring

Under the proportional hazards model framework with right-censoring, the response is a vector of event times $\boldsymbol{t} \in \mathbb{R}^{N\times 1}$, and there is also a vector of event indicator $\boldsymbol{\delta} \in \mathbb{R}^{N\times 1}$ ($\delta_i = 1$ if an event was observed at time t_i, and $\delta_i = 0$ if time t_i is right-censored).

The proportional hazards model assumption states that, for an individual with covariates $\boldsymbol{x}^\top$, their hazard function $h(t)$ is given by

$$h(t) = h_0(t) \exp(\boldsymbol{x}^\top \boldsymbol{\beta}),$$

where $h_0(t)$ is a baseline hazard function. This is a semi-parametric model, because $h_0(t)$ is not assumed to have a particular parametric form.

In the case of right censoring, the objective function L is the negative log-partial likelihood, given by,

$$L(\boldsymbol{\beta}) = \sum_{i\in D} \boldsymbol{x}_i^\top \boldsymbol{\beta} - \sum_{i\in D} \log\left(\sum_{k\in R_i} \exp(\boldsymbol{x}_k^\top \boldsymbol{\beta})\right),$$

where D is the index set of observed events, and R_i is the index set of individuals at risk at time t_i.

The proportional hazards model is usually referred to as Cox regression in honor to David Cox, who observed that $\boldsymbol{\beta}$ could be estimated without any consideration of the baseline hazard function.

Once the model is fitted, to estimate the baseline survival function we can use,

$$S_0(t) = \exp(-H_0(t)), \text{ with } H_0(t) = \sum_{t_i \leq t} h_0(t_i),$$

where

$$h_0(t_i) = \frac{\delta_i}{\sum_{j \in R_i} \exp(\boldsymbol{x}_j^\top \hat{\boldsymbol{\beta}})}.$$

An individual's estimated survival function is given by

$$S(t|\boldsymbol{x}) = S_0(t)^{\exp(\boldsymbol{x}^\top \hat{\boldsymbol{\beta}})}. \tag{1.9}$$

1.1.4 A gradient method

For a continuously differentiable function $R : \mathbb{R}^p \to \mathbb{R}$, consider the general optimization problem,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} R(\boldsymbol{\beta}). \tag{1.10}$$

Suppose there is $M_t(\cdot, \cdot)$ and a sequence $t_1, t_2, \ldots$ such that for every $k \geq 1$ and $\boldsymbol{\beta}_{k-1} \in \mathbb{R}^p$,

$$M_{t_k}(\boldsymbol{\beta}_{k-1}, \boldsymbol{\beta}_{k-1}) = R(\boldsymbol{\beta}_{k-1}) \tag{1.11}$$

and

$$M_{t_k}(\boldsymbol{\beta}_k, \boldsymbol{\beta}_{k-1}) \geq R(\boldsymbol{\beta}_k), \text{ where } \boldsymbol{\beta}_k = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} M_{t_k}(\boldsymbol{\beta}, \boldsymbol{\beta}_{k-1}). \tag{1.12}$$

These two conditions imply that, for each $k \geq 1$

$$R(\boldsymbol{\beta}_k) \leq M_{t_k}(\boldsymbol{\beta}_k, \boldsymbol{\beta}_{k-1}) \leq M_{t_k}(\boldsymbol{\beta}_{k-1}, \boldsymbol{\beta}_{k-1}) = R(\boldsymbol{\beta}_{k-1}), \tag{1.13}$$

and thus $\boldsymbol{\beta}_0, \boldsymbol{\beta}_1, \ldots \boldsymbol{\beta}_{k-1}, \boldsymbol{\beta}_k, \ldots$ is a descent sequence for $R(\boldsymbol{\beta})$.

Consider now the quadratic approximation of $R(\boldsymbol{\beta})$

$$M_t(\boldsymbol{\beta}, \boldsymbol{\beta}_0) = R(\boldsymbol{\beta}_0) + (\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \nabla R(\boldsymbol{\beta}_0) + \frac{1}{2t} \|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_2^2 . \quad (1.14)$$

It is easy to see that M_t satisfies (1.11) and (1.12), if $t < 1/\mathcal{L}(R)$, where $\mathcal{L}(R)$ is a Lipschitz constant of R, i.e.

$$\|\nabla R(\boldsymbol{\beta}) - \nabla R(\boldsymbol{\beta}_0)\|_2 \leq \mathcal{L}(R) \|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_2 . \quad (1.15)$$

After some algebra,

$$\begin{aligned} \boldsymbol{\beta} &= \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} M_t(\boldsymbol{\beta}, \boldsymbol{\beta}_0) \\ &= \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{2t} \|\boldsymbol{\beta} - (\boldsymbol{\beta}_0 - t\nabla R(\boldsymbol{\beta}_0))\|_2^2 \right\} \\ &= \boldsymbol{\beta}_0 - t\nabla R(\boldsymbol{\beta}_0). \end{aligned} \quad (1.16)$$

Equation 1.16 justifies a gradient algorithm to solve (1.10),

$$\boldsymbol{\beta}_0 \to \boldsymbol{\beta}_1 = \boldsymbol{\beta}_0 - t_1 \nabla R(\boldsymbol{\beta}_0) \to \ldots \to \boldsymbol{\beta}_k = \boldsymbol{\beta}_{k-1} - t_k \nabla R(\boldsymbol{\beta}_{k-1}).$$

Notice that this gradient method only requires R to be Lipschitz continuously differentiable and to know both R and ∇R. The computationally intensive part of the algorithm is to find the *step-size* t_k in each step k. In general, the minimum Lipschitz constant is difficult to compute, and one could try to approximate it using backtracking, as discussed later in this thesis.

Regardless of whether this gradient method is the best alternative to solve the regression problem (1.2) – which is not – (1.16) is a very general approach and adapts to the three types of regression introduced earlier (linear, logistic and Cox regression). Moreover, it is the building block of possibly the fastest and most flexible algorithm to this date, dealing with linear combinations of smooth and non-smooth arbitrary functions. The following section discusses this in more detail.

1.2 Variable selection techniques

This section introduces variable selection from the perspective of regularization. To illustrate it, consider again the linear regression problem,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \right\}. \tag{1.17}$$

When $\boldsymbol{X}$ has full column rank, the least squares estimate is simply $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$. In many applications, however, the matrix $\boldsymbol{X}$ does not have full column rank (e.g. when $p > N$) and there are infinite solutions for $\hat{\boldsymbol{\beta}}$, or $\boldsymbol{X}$ is ill-conditioned such that $\hat{\boldsymbol{\beta}}$ has a huge norm, becoming meaningless. One of the most popular regularization methods is Ridge regression (also known as Tikhonov regularization in other contexts),

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right\}. \tag{1.18}$$

Here $\lambda > 0$ is a hyperparameter that controls the tradeoff between adjusting the data $(\boldsymbol{X}, \boldsymbol{y})$ and being robust against noise. This idea extends to arbitrary losses,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ L(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_2^2 \right\}. \tag{1.19}$$

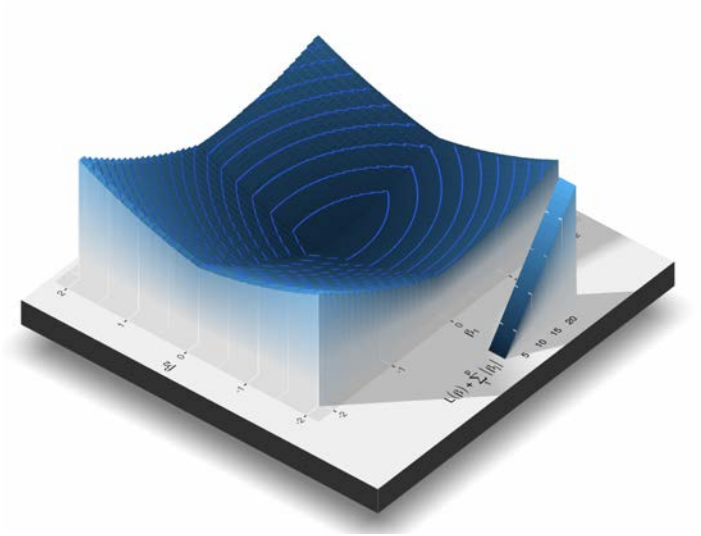
Another popular regularization method is Lasso (Tibshirani, 1996), where the squared norm-2 penalty is substituted by a norm-1 penalty,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ L(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1 \right\}. \tag{1.20}$$

Here the term $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^p |\beta_j|$ induces sparsity in the solution, and thus Lasso naturally performs variable selection. This will be detailed analytically, but Figure 1.3 illustrates intuitively why Lasso tends to force some components of $\boldsymbol{\beta}$ to be exactly zero. The price to pay is that the objective function (1.20) is not differentiable anymore, and thus using gradient descent to find $\hat{\boldsymbol{\beta}}$ is not straightforward.

Elastic Net (Zou and Hastie, 2005) is another popular regularization technique that induces feature selection when there are strong correlations between the columns of the data. It is a linear combination of

Figure 1.3: Plot of Lasso objective (1.20) with $p = 2$ variables, $N = 100$ observations generated from the model $Y = X^\top \beta + \epsilon$, and $\lambda = 2$.



Lasso and Ridge regularization, defined by,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ L(\boldsymbol{\beta}) + \lambda_1 \left\| \boldsymbol{\beta} \right\|_1 + \lambda_2 \left\| \boldsymbol{\beta} \right\|_2^2 \right\}, \quad \lambda_1, \lambda_2 > 0. \quad (1.21)$$

It is often the case in which covariates follow a natural grouped structure. For instance, when some of the variables are qualitative factors, typically they are coded as dummy covariates. In this case, might be of interest to include or exclude from the model all dummy variables associated with the same original factor (perhaps the practical cost of including one of those in the model is the same as if including the whole group). There are also other situations in which grouping covariates is useful (e.g., genes in certain pathways). Solving this problem, Yuan and Lin, 2006 introduced Group Lasso, a regularized regression method, which includes lasso as an extreme case. Several formulations for the group lasso problem can be found in the literature. The formulation from Yuan and Lin, 2006 (the most popular in

literature and implementation), defines the group lasso estimate as,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \hat{L}(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{J} \sqrt{p_j} \|\boldsymbol{\beta}^{(j)}\|_2 \right\}, \quad \lambda > 0, \tag{1.22}$$

where J is the number of groups and $\boldsymbol{\beta}^{(j)} \in \mathbb{R}^{p_j}$ contains the components of $\boldsymbol{\beta}$ corresponding to j-th group, for $j = 1, 2 \dots J$, $p_1 + p_2 + \cdots + p_J = p$ (non overlapping groups).

The Group Lasso method gives a solution corresponding to a sparse set of groups. However, if it includes a group in the model, then all the coefficients in that group will be non-zero, which sometimes is appropriate, depending on the situation. Nevertheless, there are other scenarios in which it would be nice to have both sparsity of groups and within each group (e.g., if the predictors are genes, it would be interesting to identify only particularly important genes in certain pathways). Approaching this problem, Simon et al., 2013 define the Sparse Group Lasso estimate as,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \hat{R}(\boldsymbol{\beta}) + (1-\alpha)\lambda \sum_{j=1}^{J} \sqrt{p_j} \|\boldsymbol{\beta}^{(j)}\|_2 + \alpha\lambda \|\boldsymbol{\beta}\|_1 \right\}, \tag{1.23}$$

where $\alpha \in [0, 1]$. Note that the penalty in (1.23) is a convex combination of the penalties for lasso and group lasso, which is very intuitive, since the objective in this case is finding an equilibrium between both penalties.

This thesis makes important contributions to the Sparse Group Lasso regularization method, including but not limited to, the selection of the regularization parameters and the estimation of the groups.

1.2.1 A fast iterative shrinkage-thresholding algorithm

This section details the fast iterative shrinkage-thresholding algorithm (FISTA) to solve general problems in the form (1.21). Consider the general optimization problem

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}}\, F(\boldsymbol{\beta}) = R(\boldsymbol{\beta}) + \Phi(\boldsymbol{\beta}), \tag{1.24}$$

where

- $R : \mathbb{R}^p \to \mathbb{R}$ is a smooth convex function, continuously differentiable with Lipschitz continuous gradient ∇R (with Lipschitz constant $\mathcal{L}(R)$), such that

$$\|\nabla R(\boldsymbol{\beta}) - \nabla R(\boldsymbol{\beta}_0)\|_2 \leq \mathcal{L}(R) \|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_2 .$$

- $\Phi : \mathbb{R}^{p\times 1} \to \mathbb{R}$ is a continuous convex function which is possibly non-smooth.

The quadratic approximation of $F(\boldsymbol{\beta})$ given by

$$M_t(\boldsymbol{\beta}, \boldsymbol{\beta}_0) = R(\boldsymbol{\beta}_0) + (\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \nabla R(\boldsymbol{\beta}_0) + \frac{1}{2t} \|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_2^2 + \Phi(\boldsymbol{\beta}) \tag{1.25}$$

is such that, for $t < 1/\mathcal{L}(R)$,

$$M_{t_k}(\boldsymbol{\beta}_{k-1}, \boldsymbol{\beta}_{k-1}) = F(\boldsymbol{\beta}_{k-1}) \tag{1.26}$$

and

$$M_{t_k}(\boldsymbol{\beta}_k, \boldsymbol{\beta}_{k-1}) \geq F(\boldsymbol{\beta}_k), \text{ where } \boldsymbol{\beta}_k = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} M_{t_k}(\boldsymbol{\beta}, \boldsymbol{\beta}_{k-1}). \tag{1.27}$$

Therefore, minimizing $M_t(\boldsymbol{\beta}, \boldsymbol{\beta}_0)$ with respect to $\boldsymbol{\beta}$ produces a descent sequence for $F(\boldsymbol{\beta})$. Notice that

$$\begin{aligned} U_t(\boldsymbol{\beta}_0) &= \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} M_t(\boldsymbol{\beta}, \boldsymbol{\beta}_0) \\ &= \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{\beta} - (\boldsymbol{\beta}_0 - t\nabla R(\boldsymbol{\beta}_0))\|_2^2 + t\Phi(\boldsymbol{\beta}) \right\}. \end{aligned} \tag{1.28}$$

An explicit expression for the minimizer $U_t(\boldsymbol{\beta}_0)$ of (1.28) provides a gradient step to go from $\boldsymbol{\beta}_0 \to \boldsymbol{\beta}_1$ and so on. However, to accelerate the global rate of convergence from $1/k$ to $1/k^2$, the FISTA algorithm (Beck and Teboulle, 2009a; Beck and Teboulle, 2009b) updates $\boldsymbol{\beta}_k$ according to

$$\boldsymbol{\beta}_{k+1} \leftarrow U_{t_k}(\boldsymbol{\beta}_k) + \frac{l_k - 1}{l_{k+1}} (U_{t_k}(\boldsymbol{\beta}_k) - U_{t_k - 1}(\boldsymbol{\beta}_{k-1})), \tag{1.29}$$

where $l_{k+1} = (1 + \sqrt{1 + 4l_k^2})/2$, $l_1 = 1$.

Example 3. *Consider again the Lasso problem* (1.20)*. In this case,* $F(\boldsymbol{\beta}) = L(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1$*, and thus the update is given by*

$$U_t(\boldsymbol{\beta}_0) = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{\beta} - (\boldsymbol{\beta}_0 - t\nabla L(\boldsymbol{\beta}_0))\|_2^2 + t \|\boldsymbol{\beta}\|_1 \right\}. \quad (1.30)$$

Using subgradient conditions and after some algebra,

$$U_t(\boldsymbol{\beta}_0) = S\left(\boldsymbol{\beta}_0 - t\nabla L(\boldsymbol{\beta}_0),\ \lambda t\right), \quad (1.31)$$

where S *is the coordinate-wise soft threshold operator,*

$$S(\boldsymbol{z}, \lambda)_i = sign(z_i)(|z_i| - \lambda)_+.$$

Equation 1.31 justifies why Lasso is a variable selection method. The operator S *makes some components of* $\boldsymbol{\beta}$ *be exactly zero, and this is due to the non-smoothness of* $\lambda \|\boldsymbol{\beta}\|_1$*.*

1.3 Main contributions

The contributions of this thesis are divided into four distinct chapters. Chapter 2 begins by introducing the Sparse Group Lasso regularization in regression problems. The Sparse Group Lasso has two main issues that are identified and approached in this thesis.

1. The Sparse Group Lasso penalty depends on two regularization hyperparameters. Additionally, it has a relaxed version – the unpooled Sparse Group Lasso – that depends on a fixed but large number of regularization hyperparameters. In any case, Sparse Group Lasso requires a fine tuning of the hyperparameters in order to provide meaningful fitted models.

2. The Sparse Group Lasso requires a prior specification of clusters between the variables. In applications, very few times groups among the variables are known.

To deal with the first, Chapter 2 proposes the Iterative Sparse Group Lasso, a coordinate descent algorithm for hyperparameter selection in the (unpooled) Sparse Group Lasso regularization context. The advantages of the Iterative Sparse Group Lasso are illustrated with a gene-expression dataset. Although a customized hyperparameter

selection algorithm gives better results than no tuning at all, when $p \gg N$ there is not enough sample size to guarantee that the Sparse Group Lasso model will not overfit the data. With this problem in mind, Chapter 3 will attempt to justify model selection, proposing a method to quantify the importance of each variable and determine the best model in terms of out-of-sample prediction.

Chapter 3 partially approaches the second issue, illustrating a naive method to create clusters of variables for the Sparse Group Lasso, based on Principal Component Analysis. However, this solution lacks the theoretical justification to make it a complete approach. Chapter 3 focuses on a variable selection procedure for high-dimensional data, that sorts the features according to a defined importance index.

A rigorous solution to the second issue is given in Chapter 4, that introduces the strongest contribution of this thesis, the Group Linear Algorithm with Sparse Principal decomposition (GLASP). This method on one hand extends the Sparse Group Lasso, because it does not require a prior specification of clusters between the variables. On the other hand, GLASP can be considered as a supervised variable clustering algorithm. Chapter 4 ties together the theoretical as well as algorithmic concepts of this extension. The advantages of GLASP are illustrated using both real and simulated data.

On a parallel study, conducted as part of a research stay at Technical University of Denmark, Chapter 5 proposes a novel solution for semi-supervised learning in the context of generalized linear model estimation: the generalized Semi-Supervised Elastic Net. This method extends the supervised elastic-net, with a general mathematical formulation that covers, but is not limited to, both regression and classification problems. Although the semi-supervised context is different from the supervised one, Chapter 5 shows that the Semi-Supervised Elastic Net can be solved with the same tools used for supervised regression problems with regularization terms.

Finally, Chapter 6 summarizes the results of this thesis, and includes a discussion of the implication of the findings to future research.

Bibliography for Chapter 1

Beck, Amir and Marc Teboulle (2009a). "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1, pp. 183–202.

– (2009b). "Gradient-based algorithms with applications to signal recovery". In: *Convex optimization in signal processing and communications*, pp. 42–88.

Simon, Noah et al. (2013). "A sparse-group lasso". In: *Journal of computational and graphical statistics* 22.2, pp. 231–245.

Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1, pp. 267–288.

Yuan, Ming and Yi Lin (2006). "Model selection and estimation in regression with grouped variables". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.1, pp. 49–67.

Zou, Hui and Trevor Hastie (2005). "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2, pp. 301–320.

CHAPTER 2

An iterative sparse-group lasso

In *Journal of Computational and Graphical Statistics*, Volume 28, 2019:722-731

Juan C. Laria[a,b] M. Carmen Aguilera-Morillo[a,b] Rosa E. Lillo[a,b]

[a] Department of Statistics, University Carlos III of Madrid, Spain
[b] UC3M-BS Institute of Financial Big Data, Madrid, Spain

Abstract

In high-dimensional supervised learning problems, sparsity constraints in the solution often lead to better performance and interpretability of the results. For problems in which covariates are grouped and sparse structure are desired, both on group and within group levels, the sparse-group lasso (SGL) regularization method has proved to be very efficient. Under its simplest formulation, the solution provided by this method depends on two weight parameters that control the penalization on the coefficients. Selecting these weight parameters represents a major challenge. In most of the applications of the SGL, this problem is left aside, and the parameters are either fixed based on a prior information about the data, or chosen to minimize some error function in a grid of possible values. However, an appropriate choice of the parameters deserves more attention, considering that it plays a key role in the structure and interpretation of the solution. In this sense, we present

a gradient-free coordinate descent algorithm that automatically selects the regularization parameters of the SGL. We focus on a more general formulation of this problem, which also includes individual penalizations for each group. The advantages of our approach are illustrated using both real and synthetic data sets.

Keywords: Coordinate descent. Gradient-free. High-dimension. Optimization. Regularization

2.1 Introduction

Regression models with a sparsity constraint on the solution have become very popular in high dimensional problems. After Yuan and Lin, 2006 introduced the group lasso, a considerable literature grew up around the theme of simultaneously selecting individual variables at both the group and within-group levels. For example, Huang et al., 2009 explored this problem with a group bridge perspective. With the same objective in mind, Zhou and Zhu, 2010 introduced the hierarchical lasso (HLasso). An extension of HLasso to quantile regression was latter developed by Zhao, Zhang, and Liu, 2014. Applications of HLasso to pattern recognition were also discussed by Sprechmann et al., 2011.

In recent years, there has been increasing interest in the sparse-group lasso (SGL) problem. SGL is a regularization method introduced by Friedman, Hastie, and Robert Tibshirani, 2010, which generalizes the lasso (Robert Tibshirani, 1996), the group lasso (Yuan and Lin, 2006) and the elastic-net (Zou and Hastie, 2003), by combining lasso and group lasso penalties. The solution provided by the SGL, as in lasso and group lasso, often involves a sparse number of predictor variables, since many coefficients in the solution are exactly zero. SGL has an advantage over lasso when the predictor variables are grouped, as lasso penalizes all the coefficients of the solution equally. SGL, on the other hand, distinguishes among groups and, unlike group lasso, it is able to provide a solution which is also sparse inside each group. It has been shown that SGL can play an important role in addressing the issue of variable selection in genetic models, where genes are grouped

following different pathways.

A common mathematical formulation of the SGL problem is,

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta} \in B} \left\{ \hat{R}(\boldsymbol{\beta}) + (1-\alpha)\lambda\phi_2(\boldsymbol{\beta}) + \alpha\lambda\phi_1(\boldsymbol{\beta}) \right\}, \tag{2.1}$$

where $\hat{R}$ is a risk function depending on $\boldsymbol{\beta}$ and the data, B is a parameter set, ϕ_1 and ϕ_2 are the lasso and group lasso penalties, respectively, $\lambda \geq 0$ and $\alpha \in [0, 1]$ are parameters controlling the regularization terms ϕ_1 and ϕ_2.

Under its simplest formulation given in (2.1), SGL leads to a solution depending on two weight parameters, α and λ, which control the penalization on the coefficient vector $\boldsymbol{\beta}$. In most of the applications of the SGL this problem is left aside, and the parameters are either fixed, based on prior information about the data, or chosen to minimize some error function in a grid of possible values. However, an appropriate choice of the parameters deserves more attention, considering that it plays a key role in the structure and interpretation of the solution provided by the SGL.

Several studies have been published, regarding the computation of the solution of (2.1). Simon et al., 2013 introduced an implementation of the SGL, currently available as an R package. Technical details of their approach can be found in their paper, which includes the SGL for linear, logistic and Cox regression. However, their method failed to consider arbitrary group weights in the group lasso penalty term, as in (3.3). A broader perspective was adopted by Vincent and Hansen, 2014 in their implementation, which allows the user to set different penalty weights for each group. However, no attempt was made to optimally select those parameters in their approach. A more recent technique to solve (3.3), is the CVXPY optimization framework (Diamond and Stephen Boyd, 2016), which is available under python. It integrates with open source solvers such as ECOS (Domahidi, Chu, and S. Boyd, 2013), SCS (O'Donoghue et al., 2016) and CVXOPT, among others. Recently, Feng and Simon, 2017 relied on CVXPY to find the solution of the SGL problem.

Since its introduction, the SGL problem and its applications have been extensively studied. Simon et al., 2013 included a section about the

diagnosis of ulcerative colitis and prediction of recurrence of breast cancer, based on genes from the patients. Chatterjee et al., 2011 explored the use of the SGL to predict climate variables over different land regions, based on measures of those variables in the oceans. They provided a nice interpretation of the sparse solution obtained in this context. Recently, Ndiaye et al., 2016 provided further applications of the SGL to climate data. Xie and Xu, 2014 discussed an application of the SGL to uncertain data. Their approach was illustrated on several real datasets, including financial, biological and signal processing applications. On the other hand, Rao et al., 2016 explored the SGL in classification problems, introducing an extension of the SGL, the sparse overlapping group lasso (SOGlasso), to deal with overlapping groups. Real data applications of the SOGlasso in different areas were exemplified in their paper, including those on text mining, fMRI and computational biology. In this context, Cheng et al., 2017 investigated the use of the similarity-regularized sparse-group lasso (SSGL), an extension of the SGL, in image processing for glaucoma detection.

The number of publications related to the SGL and its applications, evidence the importance of studying its theoretical properties, as well as those details in which the SGL can be improved. Although extensive research has been carried out on the SGL, and the selection the regularization parameters is often mentioned, no single study existed providing a methodology to adequately select these parameters, until recently, when Feng and Simon, 2017 addressed this problem.

Friedman, Hastie, and Robert Tibshirani, 2010 compared their SGL approach with lasso and group lasso using simulations. For the SGL (2.1), they took $\alpha = 0.5$ and λ was optimized along a regularization path. Sprechmann et al., 2011 relied on cross-validation for the choice of α and λ, remarking that the selection of these parameters has an important influence on the sparsity of the obtained solution. As investigating the optimal regularization parameters was beyond the scope of their work, they selected a set of parameters that performed well among a small set of possible parameters. As Chatterjee et al., 2011 noted, the regularization parameters play a key role in variable selection. In order to select the relevant variables in their model, Chatterjee et al. computed the regularization path of the SGL solution, i.e., a plot of the coefficient values for each covariate versus λ, letting $\alpha = 0.5$

fixed. As an illustration, they also plotted the regularization path for one of the relevant variables, on a grid of λ and α values, resulting in a three dimensional surface. However, their approach does not provide a framework to tune the parameters, since it involves the computation of SGL solutions for an entire grid of α and λ values, which may be appropriate in small scale problems, but not for data sets with thousands of variables involved. Xie and Xu, 2014 tried a total of 12 different (α, λ) combinations in their experiments. In the real data applications discussed by Rao et al., 2016, they briefly mentioned that they had trained the models using 4-fold cross-validation to select the regularization parameters, but they did not provide a detailed method. In the experiments illustrated by Ndiaye et al., 2016, they computed the SGL estimator using a sequence of 100 values for the regularization parameter λ, whereas α was chosen among 11 equally spaced values in $[0, 1]$, using 2-fold cross validation.

Overall, these studies highlight the need for a methodology to select the parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$ in (2.9). The aim of this paper is to develop an algorithm that automatically selects all the weight parameters of the SGL. An alternative, but more general formulation of this problem is given by,

$$\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}, \boldsymbol{\gamma}) = \arg\min_{\boldsymbol{\beta} \in B} \left\{ \hat{R}(\boldsymbol{\beta}) + \lambda_2 \sum_{j=1}^{J} \gamma_j \|\boldsymbol{\beta}^{(j)}\|_2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \right\}. \quad (2.2)$$

Here J is the number of groups, and $\boldsymbol{\beta}^{(j)} \in \mathbb{R}^{p_j}$ are vectors with the components of $\boldsymbol{\beta}$ corresponding to j-th group (of size p_j), $j = 1, 2, \ldots, J$. Note that $\boldsymbol{\lambda} \in \mathbb{R}_+^2$ and $\boldsymbol{\gamma} \in \mathbb{R}_+^J$, so that the total number of parameters to be tuned is $J + 2$.

The study presented in this paper is one of the first investigations focused specifically on the selection of the weight parameters in the sparse-group lasso problem, given in (3.3). We present a gradient-free coordinate descent algorithm that automatically selects both parameter vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$. Theoretical and practical advantages of our approach are also illustrated using both real and synthetic data sets.

This paper is organized as follows. Next section ties together the various theoretical concepts that support our approach. Section 2.3 introduces our main contribution, the *iterative sparse-group lasso*. Sec-

tions 2.4 and 2.5 highlight the importance of our study using both synthetic and real datasets, respectively. Finally, Section 2.6 includes a discussion of the implication of our findings to future research.

2.2 Theoretical background

Under the usual regression framework, we consider N observations in the form $\mathcal{Z} = \{y_i, \boldsymbol{x}_i\}_{i=1}^N$. They are a random sample from some population with probability distribution function $\rho(\boldsymbol{x}, y)$, where y is the response and $\boldsymbol{x} = [x_1, x_2, \ldots, x_p]'$ are the predictor variables.

Assuming that a flexible class of functions $\{F(\boldsymbol{x}, \boldsymbol{\beta}), \boldsymbol{\beta} \in B\}$ exists, where B is a parameter set, the objective of regression is finding $\boldsymbol{\beta}^* \in B$ such that $F(\boldsymbol{x}, \boldsymbol{\beta}^*)$ estimates y from $\boldsymbol{x}$ the best among all $\boldsymbol{\beta} \in B$. The quality of an approximation is often measured using a loss function $L(y, F(\boldsymbol{x}, \boldsymbol{\beta}))$. As a common rule, L takes non-negative values, in such a way that very positive numbers correspond to very poor approximations. The empirical risk is defined as,

$$\hat{R}(\boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, F(\boldsymbol{x}_i, \boldsymbol{\beta})). \tag{2.3}$$

and only depends on the available information in $\mathcal{Z}$. Therefore,

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta} \in B} \hat{R}(\boldsymbol{\beta}), \tag{2.4}$$

is a plug-in estimate for $\boldsymbol{\beta}^*$.

In the case of high dimensionality ($p \gg N$), it becomes difficult to work with enough data samples to achieve a high density of points. The penalization (or regularization) approach provides a formalism for controlling the complexity of the approximating functions, to fit available finite data. We are interested in this non-complexity in terms of the sparsity of $\hat{\boldsymbol{\beta}}$, and we focus on linear approximating functions in the form

$$F(\boldsymbol{x}, \boldsymbol{\beta}) = \beta_0 + \sum_{j=1}^{p} \boldsymbol{x}_j \beta_j, \quad \boldsymbol{\beta} \in B = \mathbb{R}^{p+1}. \tag{2.5}$$

The lasso, introduced by Robert Tibshirani in Robert Tibshirani, 1996, is a regularized regression method which solves,

$$\hat{\boldsymbol{\beta}}(\lambda) = \arg\min_{\boldsymbol{\beta}\in B} \left\{ \hat{R}(\boldsymbol{\beta}) + \lambda\phi_1(\boldsymbol{\beta}) \right\}, \tag{2.6}$$

where $\phi_1(\beta) = \|\boldsymbol{\beta}\|_1$ is the lasso penalty. The solution that lasso provides has many components that are exactly zero. Thus, by solving (2.6), we are also doing variable selection. This approach has inspired many important algorithms of the last couple of decades. Yuan and Lin introduced the group lasso (Yuan and Lin, 2006) as an extension of lasso when variables are forming groups, and those groups are involved in the regression. Other extensions involve merging different penalties, like the elastic-net family, proposed by Zou and Hastie in Zou and Hastie, 2003.

The group lasso method gives a solution corresponding to a sparse set of groups. However, if it includes a group in the model, then all the coefficients in that group will be non-zero, which may be appropriate, depending on the situation. Nevertheless, there are other scenarios in which it would be nice to have both sparsity of groups and within each group. Approaching this problem, Friedman, Hastie, and Robert Tibshirani introduced the SGL.

Consider the regression matrix $\boldsymbol{X}$ partitioned as

$$\boldsymbol{X} = [\mathbf{1}\, X^{(1)}\, X^{(2)}\, \cdots\, X^{(J)}], \tag{2.7}$$

where $\mathbf{1} = X^{(0)}$ is a column of ones for the intercept term, J is the number of groups and $X^{(j)}$ is the matrix of observations corresponding to variables in group j, $j = 1, 2, \ldots, J$. Consider the vector of parameters $\boldsymbol{\beta}$ partitioned as,

$$\boldsymbol{\beta} = [\beta^{(0)}, \boldsymbol{\beta}^{(1)\prime}, \boldsymbol{\beta}^{(2)\prime}, \ldots, \boldsymbol{\beta}^{(J)\prime}]', \tag{2.8}$$

where $\beta^{(0)} = \beta_0$ is the intercept term, but here $\boldsymbol{\beta}^{(j)} \in \mathbb{R}^{p_j}$ are also vectors containing the components of $\boldsymbol{\beta}$ corresponding to j-th group, $j = 1, 2, \ldots, J$.
The (classic) sparse-group lasso estimation (from Friedman, Hastie, and Robert Tibshirani, 2010) is defined as,

$$\hat{\boldsymbol{\beta}}(\lambda, \alpha) = \arg\min_{\boldsymbol{\beta}\in B} \left\{ \hat{R}(\boldsymbol{\beta}) + (1-\alpha)\lambda\phi_2(\boldsymbol{\beta}) + \alpha\lambda\phi_1(\boldsymbol{\beta}) \right\}, \tag{2.9}$$

where $\alpha \in [0,1]$ and $\lambda \geq 0$, and the group lasso penalty term is defined as,

$$\phi_2(\boldsymbol{\beta}) = \sum_{j=1}^{J} \sqrt{p_j} \|\boldsymbol{\beta}^{(j)}\|_2, \tag{2.10}$$

with p_j being the size of j-th group. Note that the penalty in (2.9) is a convex combination of the penalties for lasso and group lasso, which is very intuitive, since the objective in this case is finding an equilibrium between both penalties.

2.3 The iterative sparse-group lasso

The purpose of this section is to develop an algorithm that selects the optimal weight parameters of the sparse-group lasso, in the form (3.3). We consider that parameterization (3.3) has an advantage over (2.1), since the group lasso and lasso terms of the penalization are controlled by different regularization parameters that range both in $\mathbb{R}_+$, while allowing further control over individual group weights $\boldsymbol{\gamma}$. Our approach is based on sequential minimizations of the empirical risk, keeping the regularization parameters $\boldsymbol{\lambda}, \boldsymbol{\gamma}$ fixed except for one coordinate, in which an univariate optimization is performed. In the search for the best regularization parameter values, we tried to strike a balance between accuracy and computational efficiency, and that is why we consider a random search algorithm to solve the univariate optimization part.

Traditionally, the data set $\mathcal{Z}$ is partitioned into three disjoint data sets, $\mathcal{Z}_T, \mathcal{Z}_V$, and $\mathcal{Z}_{test}$. The data in $\mathcal{Z}_T$ is used for training the model, i.e., solving (3.3). $\mathcal{Z}_V$ is used for validation, i.e., finding the optimal parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$. The remaining observations in $\mathcal{Z}_{test}$ are used for testing how the model predicts (or describes) future data. Specifically, the selection of the optimal parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$ is based on the minimization of the validation error, defined as,

$$\hat{R}_V(\boldsymbol{\lambda}, \boldsymbol{\gamma}) = \frac{1}{\#\mathcal{Z}_V} \sum_{(y_i, \boldsymbol{x}_i) \in \mathcal{Z}_V} L[y_i, F(\boldsymbol{x}_i, \hat{\boldsymbol{\beta}}_T(\boldsymbol{\lambda}, \boldsymbol{\gamma}))], \tag{2.11}$$

where

$$\hat{\boldsymbol{\beta}}_T(\boldsymbol{\lambda},\boldsymbol{\gamma}) = \arg\min_{\boldsymbol{\beta}\in B}\left\{\hat{R}_T(\boldsymbol{\beta}) + \lambda_2\sum_{j=1}^{J}\gamma_j\|\boldsymbol{\beta}^{(j)}\|_2 + \lambda_1\|\boldsymbol{\beta}\|_1\right\}, \tag{2.12}$$

and

$$\hat{R}_T(\boldsymbol{\beta}) = \frac{1}{\#\mathcal{Z}_T}\sum_{(y_i,\boldsymbol{x}_i)\in\mathcal{Z}_T} L[y_i, F(\boldsymbol{x}_i,\boldsymbol{\beta})], \tag{2.13}$$

with $\#$ denoting the cardinal of a set. Therefore, the problem of finding the optimal parameters $(\boldsymbol{\lambda},\boldsymbol{\gamma})$ can be formulated as,

$$\begin{aligned} &\min_{(\boldsymbol{\lambda},\boldsymbol{\gamma})\in\mathbb{R}_+^2\times\mathbb{R}_+^J} \hat{R}_V(\boldsymbol{\lambda},\boldsymbol{\gamma}) \\ \text{s.t.}\quad &\hat{\boldsymbol{\beta}}_T(\boldsymbol{\lambda},\boldsymbol{\gamma}) = \arg\min_{\boldsymbol{\beta}\in B}\left\{\hat{R}_T(\boldsymbol{\beta}) + \lambda_2\textstyle\sum_{j=1}^{J}\gamma_j\|\boldsymbol{\beta}^{(j)}\|_2 + \lambda_1\|\boldsymbol{\beta}\|_1\right\}. \end{aligned} \tag{2.14}$$

The simplest approach to problems in the form (3.7) is brute-force. However, the dimension of $\boldsymbol{\gamma}$ represents a major challenge. A straightforward solution would be to take $\boldsymbol{\gamma}$ fixed at $\gamma_j = \sqrt{p_j}$ (as formulated by Simon et al., 2013), and solve (3.7) for $\boldsymbol{\lambda}$ only. Under the brute-force approach to this simpler problem, one creates a two dimensional grid for $\boldsymbol{\lambda}$ with feasible solutions of (3.7). The selected pair (λ_1,λ_2) is the one that minimizes (3.4) in the points of the grid.

Figure 3.2 illustrates the search for the optimal penalty weight $\boldsymbol{\lambda}$ in a grid of 100×4. The data set is about Ulcerative colitis (Burczynski et al., 2006). It has been randomly partitioned into 50 and 77 observations for training and validation data, respectively. Under this grid-search/brute-force approach, the internal optimization problem (3.5) has been solved 400 times. As the dimension of the available data increases, the computational burden of this method makes it unfeasible for a practical use.

As an alternative to grid search (Figure 3.2), we present the ITERATIVE SPARSE-GROUP LASSO (iSGL), a gradient-free coordinate descent method to tune the parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$ from the sparse-group lasso (3.3), which performs well under different scenarios while drastically reducing the number of operations required to find optimal penalty weight parameters that minimize the validation error in (3.4). The

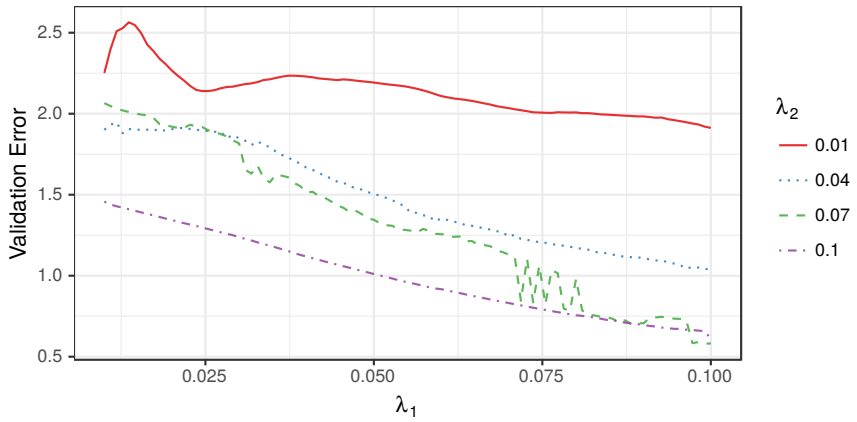


Figure 2.1: Plot of validation error $\hat{R}_V(\boldsymbol{\lambda})$ in a grid of $100\ \lambda_1-$values and $4\ \lambda_2-$values. The data set was studied by Burczynski et al., 2006 and Simon et al., 2013. There were 50 observations in the training set $\mathcal{Z}_T$ and 77 observations in the validation set $\mathcal{Z}_V$. There were considered 7819 genes as explanatory variables, grouped according to the C1 positional geneset.

iSGL iteratively performs a univariate minimization over one of the coordinates of $[\boldsymbol{\lambda}, \boldsymbol{\gamma}]$, while the remaining coordinates are fixed.

There are three important components in our approach, that play key roles in the performance of the iSGL: the initial values of $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$, the optimization of the internal function (3.5), and the univariate optimization of (3.7). These three important steps are explained in the following three sections.

2.3.1 Initial regularization parameters

One major drawback of coordinate descent algorithms is that their performance strongly depends on the initial point, if the function is non-smooth. This is the case of the validation error $\hat{R}_V(\boldsymbol{\lambda}, \boldsymbol{\gamma})$ (3.4) in the sparse-group lasso. It is also important to notice in (3.3) that for some sufficiently large $\boldsymbol{\lambda}, \boldsymbol{\gamma}$, the optimal solution of (3.3) will be $\boldsymbol{\beta} = \mathbf{0}$. If our algorithm iSGL initializes in a zero solution, it will probably get stuck there. That is why it is critical to start the iSGL algorithm with $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$ such that $\boldsymbol{\beta}_T \neq \mathbf{0}$. The following propositions

provide valuable insight into the appropriate initial penalty parameters of our method.

Proposition 1 (Upper bound for λ_1, linear model)**.** *Consider the sparse-group lasso problem in the form* (3.3)*, with*

$$\hat{R}(\boldsymbol{\beta}) = \frac{1}{2N} \left\| \boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}\boldsymbol{\beta} \right\|_2^2, \tag{2.15}$$

being the square error risk. If $\lambda_2 = 0$, then (3.3) *reduces to lasso, and $\boldsymbol{\beta} = \mathbf{0}$ if*

$$\max_{k \leq J,\, i \leq p_k} \left\{ \left| \frac{1}{N} \boldsymbol{X}_i^{(k)\prime} \boldsymbol{r}_{-k,i} \right| \right\} \leq \lambda_1, \tag{2.16}$$

where $\boldsymbol{X}_i^{(k)}$ denotes the i–th column of $\boldsymbol{X}^{(k)}$, and

$$\boldsymbol{r}_{-k,i} = \boldsymbol{y} - \beta_0 \mathbf{1} - \sum_{\substack{j=1 \\ j \neq k}}^{J} \boldsymbol{X}^{(j)} \boldsymbol{\beta}^{(j)} - \sum_{l \neq i}^{p_k} \boldsymbol{X}_l^{(k)} \boldsymbol{\beta}_l^{(k)}. \tag{2.17}$$

Proposition 2 (Upper bound for λ_1, logistic model)**.** *Consider the sparse-group lasso problem in the form* (3.3)*, with*

$$\hat{R}(\boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} \left[\log\left(1 + \exp\{\beta_0 + \boldsymbol{x}_i' \boldsymbol{\beta}\}\right) - y_i(\beta_0 + \boldsymbol{x}_i' \boldsymbol{\beta}) \right], \tag{2.18}$$

being the bernoulli negative log-likelihood (logit) risk. If $\lambda_2 = 0$, then (3.3) *reduces to lasso, and $\boldsymbol{\beta} = \mathbf{0}$ if*

$$\max_{k \leq J,\, i \leq p_k} \left\{ |\boldsymbol{X}_i^{(k)\prime} \boldsymbol{d}_{-k,i}| \right\} \leq \lambda_1, \tag{2.19}$$

where $\boldsymbol{d} \in \mathbb{R}^N$, such that

$$d_l = \frac{1}{N} \left[\left(1 + \exp\{-\boldsymbol{x}_l' \boldsymbol{\beta}\}\right)^{-1} - y_l \right], \tag{2.20}$$

and $\boldsymbol{d}_{-k,i} = \boldsymbol{d}$, taking $\beta_i^{(k)} = 0$.

Proposition 3 (Upper bound for $\lambda_2 \gamma_j$, linear model)**.** *Consider the sparse-group lasso problem in the form* (3.3)*, with $\hat{R}(\boldsymbol{\beta})$ as in* (2.15)*. Then, $\boldsymbol{\beta} = \mathbf{0}$ if*

$$\left\| S\left(\frac{1}{N} \boldsymbol{X}^{(j)\prime} \boldsymbol{r}_{-j}, \lambda_1 \right) \right\|_2 \leq \lambda_2 \gamma_j, \tag{2.21}$$

where $S(\cdot)$ is the coordinate-wise soft thresholding operator,

$$S(\boldsymbol{z}, \lambda_1)_i = \text{sign}(z_i)(|z_i| - \lambda_1)_+, \tag{2.22}$$

and

$$\boldsymbol{r}_{-k} = \boldsymbol{y} - \beta_0 \mathbf{1} - \sum_{\substack{j=1 \\ j \neq k}}^{J} \boldsymbol{X}^{(j)} \boldsymbol{\beta}^{(j)}. \tag{2.23}$$

Proposition 4 (Upper bound for $\lambda_2 \gamma_j$, logistic model)**.** *Consider the sparse-group lasso problem in the form* (3.3)*, with $\hat{R}(\boldsymbol{\beta})$ as in* (2.18)*. Then, $\boldsymbol{\beta} = \mathbf{0}$ if*

$$\left\| S\left(\boldsymbol{X}^{(j)'} \boldsymbol{d}_{-j}, \lambda_1 \right) \right\|_2 \leq \lambda_2 \gamma_j, \tag{2.24}$$

where $\boldsymbol{d}_{-j} = \boldsymbol{d}$, (2.20) *taking $\boldsymbol{\beta}^{(j)} = 0$.*

Note that the upper bounds for λ_1, discussed in Propositions 1 and 2, do not depend on λ_2 nor $\boldsymbol{\gamma}$. Considering $\lambda_2 > 0$, would only decrease the minimum λ_1 giving $\boldsymbol{\beta} = \mathbf{0}$. On the other hand, equations (2.21) and (2.24) from Propositions 3 and 4, respectively, display a dependence between $\lambda_2 \gamma_j$ and λ_1.

Simon et al., 2013 derived from (2.21), the maximum λ such that $\boldsymbol{\beta} \neq \mathbf{0}$, but their parameterization of the sparse-group lasso only considered λ free, and both α and the group weights were fixed. However, the computation of this maximum λ is rather expensive, given that a piecewise quadratic problem has to be solved for each group $j \leq J$. In contrast, the parameterization (3.3) motivates us to compute an upper bound (λ_{1max}) for λ_1 regardless of λ_2 and $\boldsymbol{\gamma}$ (Propositions 1 and 2), choose an appropriate value of λ_1, and then compute an upper bound for $\lambda_2 \gamma_j$ (Propositions 3 and 4) and with γ_j fixed, an upper bound λ_{2max} for λ_2. In our iSGL implementation, we have chosen initial $\lambda_1 = 0.1\lambda_{1max}$, $\gamma_j = \sqrt{p_j},\ j \leq J$, and $\lambda_2 = 0.1\lambda_{2max}$, such that the initial $\boldsymbol{\beta}_T$ is non-zero.

2.3.2 Internal optimization

The internal optimization (3.5) (for fixed parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$) has a significant impact on the performance of the iSGL. Due to the number of computations of the optimal solution (3.5) that our methodology requires, we tried to adopt the fastest approach in practice. We

found the blockwise descent implementation by Simon et al., 2013 to be the most appropriate solution to this problem, and could be easily extended to our parameterization in (3.3). The idea behind the algorithm is based on the separability of the sparse-group lasso penalty between groups. Let $\boldsymbol{\beta}$, the optimum coefficient vector in (3.3), be fixed for all groups, except for group k, $1 \leq k \leq J$. Then, (3.3) becomes

$$R(k,\beta) + \sum_{\substack{j=1 \\ j\neq k}}^{J} (\lambda_2\gamma_j\|\boldsymbol{\beta}^{(j)}\|_2 + \lambda_1\|\boldsymbol{\beta}^{(j)}\|_1) + \lambda_2\gamma_k\|\beta\|_2 + \lambda_1\|\beta\|_1, \tag{2.25}$$

where $R(k,\beta)$ is the empirical risk $\hat{R}(\boldsymbol{\beta})$ with only $\boldsymbol{\beta}^{(k)}$ free, which is denoted by β for simplicity. Removing constant terms, minimizing (2.25) is equivalent to minimizing

$$F(\beta) := R(k,\beta) + \lambda_2\gamma_k\|\beta\|_2 + \lambda_1\|\beta\|_1. \tag{2.26}$$

Under the majorization-minimization scheme (see Lange, Hunter, and Yang, 2000, Nesterov, 2007, Beck and Teboulle, 2009), we consider the surrogate function

$$\begin{aligned} M^*(\beta,\beta_0) = R(k,\beta) &+ (\beta-\beta_0)^T\nabla R(k,\beta_0) \\ &+ \frac{1}{2t}\|\beta-\beta_0\|_2^2 + \lambda_2\gamma_k\|\beta\|_2 + \lambda_1\|\beta\|_1. \end{aligned} \tag{2.27}$$

Notice that, choosing the stepsize t sufficiently small, M^* is such that $M^*(\beta,\beta) = F(\beta)$ and $M^*(\beta,\beta_0) \geq F(\beta)$, which allows to construct a descent scheme for minimizing (2.26). Minimizing (2.27) is equivalent to minimizing

$$M(\beta,\beta_0) = \frac{1}{2t}\|\beta - B_0\|_2^2 + \lambda_2\gamma_k\|\beta\|_2 + \lambda_1\|\beta\|_1, \tag{2.28}$$

where $B_0 = \beta_0 - t\nabla R(k,\beta_0)$. Using the subgradient conditions and after some algebra, an expression for the optimum $\hat{\beta}$ of (2.28) is as follows,

$$\hat{\beta} = \left(1 - \frac{\lambda_2\gamma_k t}{\|S(B_0,\lambda_1 t)\|_2}\right)_+ S(B_0,\lambda_1 t). \tag{2.29}$$

Equation (2.29) suggests an iterative procedure to obtain the minimum of (2.25), which can be applied per group to obtain the overall solution of (3.3) for fixed $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$. Our approach is similar to the implementation in Simon et al., 2013, but adding the individual group weights γ_k.

2.3.3 Univariate optimization

It is important to remark that the minimization of (3.4) depends on (3.5), which computational burden is very high. This cost should be taken into consideration when choosing an univariate optimization strategy, since many evaluations of (3.4) would require more computational time. Grid search seems like a reasonable approach to tackle the optimization of the validation error in the one-dimensional case. However, as Bergstra and Bengio, 2012 showed using the experiments of Larochelle et al., 2007, randomly chosen trials are more efficient for hyper-parameter optimization than trials on a grid. This motivates our choice of a random search strategy.
Given the regularization parameter vector $[\boldsymbol{\lambda}, \boldsymbol{\gamma}]$ fixed, except for k-th component, say $\lambda_k^{(0)}$, where $1 \leq k \leq J+2$, we set the initial stepsize

$$t_0 = \begin{cases} \lambda_k^{(0)}/10 & \text{if } \lambda_k^{(0)} \neq 0, \\ 0.01 & \text{otherwise.} \end{cases}$$

However, the actual stepsize is randomly chosen between $t_0/10$ and t_0. Therefore, the new value of the regularization parameter $\lambda_{k\rightarrow}^{(1)} \in [\lambda_k^{(0)} + 0.1t_0,\ \lambda_k^{(0)} + t_0]$, in the positive direction. Usually in regularization problems, when there is only one parameter λ to select, the regularization path is constructed from some λ_{max} to λ_{min}, decreasing in the log-scale (see Friedman, Hastie, and Rob Tibshirani, 2010 and Simon et al., 2013). We wanted to use a non-uniform path, adapting the stepsize depending on the smoothness of the validation error (3.4). That is why we introduced a momentum-like term ($\tau \geq 1$) increasing the stepsize. If the new $\lambda_{k\rightarrow}^{(1)}$ improves the evaluation of the objective function, the next value $\lambda_{k\rightarrow}^{(2)}$ is chosen uniformly in the interval $[\lambda_{k\rightarrow}^{(1)} + \tau t_0/10,\ \lambda_{k\rightarrow}^{(1)} + \tau t_0]$. In general, $\lambda_{k\rightarrow}^{(m)} \in [\lambda_{k\rightarrow}^{(m-1)} + \tau^{m-1}t_0/10,\ \lambda_{k\rightarrow}^{(m-1)} + \tau^{m-1}t_0]$, and we stop it once the objective does not improves. Then, our next choice is $\lambda_{k\leftarrow}^{(1)} \in [\lambda_{k\rightarrow}^{(m)} - t_0,\ \lambda_{k\rightarrow}^{(m)} -$

$t_0/10]$, and after that $\lambda^{(2)}_{k\leftarrow} \in [\lambda^{(1)}_{k\leftarrow} - \tau t_0,\ \lambda^{(1)}_{k\leftarrow} - \tau t_0/10]$. We continue updating the step as before, until the objective function does not decrease. We cyclically apply this search in each coordinate of $[\boldsymbol{\lambda}, \boldsymbol{\gamma}]$, and therefore guarantee that each update of the regularization parameters decreases the validation risk.

Algorithm 1 provides an overview of our custom random search minimization for a generic function $f : \mathbb{R}_+ \to \mathbb{R}_+$, which in our case is the validation risk $\hat{R}_V(\boldsymbol{\lambda}, \boldsymbol{\gamma})$ with $[\boldsymbol{\lambda}, \boldsymbol{\gamma}]$ fixed, except for one coordinate.

Regarding the momentum term τ, we tried different configurations for this parameter varying between 1.5 and 3. Although $\tau = 2$ gave slightly better results, we could not find any significant differences in performance and speed between the different configurations for τ.

Algorithm 1: Custom random search

```
/* function to optimize, initial point                    */
Function rs(f, λ):
    t_0, t ← λ/10                                  // Initial stepsize
    dir ← 1                                            // direction →
    while dir ≥ −1 do
        step ← Rand[t/10, t]                  // Compute random step
        if f(λ) > f(λ + dir · step) then
            /* Has the objective improved?                 */
            λ ← λ + dir·step                  // Move to the new point
            t ← τ · t
        else
            dir ← dir − 2                              // Change direction
            t ← t_0                                      // Reset stepsize
    end
    return λ
```

Algorithm 3 summarizes the main idea behind iSGL described in these three sections.

2.4 Simulations

This section critically evaluates the effectiveness of iSGL selecting an appropriate solution $\boldsymbol{\beta}_T$. We emphasize the advantages of this algo-

Algorithm 2: ITERATIVE SPARSE-GROUP LASSO (iSGL)

```
/* Data for training/validation                                    */
```
Function `isgl`$(\mathcal{Z}_T, \mathcal{Z}_V)$**:**
  Initialize $\boldsymbol{\lambda}, \boldsymbol{\gamma}$ $k \leftarrow 1$
  while $\boldsymbol{\lambda}, \boldsymbol{\gamma}$ *not stationary* **do**
    $[\boldsymbol{\lambda}, \boldsymbol{\gamma}]_k \leftarrow \text{RS}(\hat{R}_V, [\boldsymbol{\lambda}, \boldsymbol{\gamma}]_k)$ $k \leftarrow k \bmod (J+2) + 1$;
    `// Next coordinate`
  end
  return $\boldsymbol{\lambda}, \boldsymbol{\gamma}$

rithm over current methodologies.

2.4.1 Linear response

Recently, Feng and Simon, 2017 have addressed the problem of choosing the optimal regularization parameters for a variety of problems with non-smooth penalties, including the sparse-group lasso. Their approach, based on a modified gradient descent algorithm, proves to be superior to grid-search, and other optimization algorithms. In this section, we compare the iSGL with the algorithm introduced by Feng and Simon, 2017 (GD), adapting some of the configurations for the simulations presented in their paper. We have also considered simpler versions of both algorithms (denoted by GD_0 and iSGL_0, respectively), taking $\boldsymbol{\gamma}$ fixed (only $\boldsymbol{\lambda}$ is selected).

Although our simulation studies focused on comparing GD and iSGL, we have also included other methods that are known to perform well in the low dimension: grid-search (GS), random search for hyper-parameter optimization (RS) from Bergstra and Bengio, 2012, and Nelder-Mead (NM), from Nelder and Mead, 1965, taking the group weights $\boldsymbol{\gamma}$ fixed. To implement GS, we use a bi-dimensional square grid with 30^2 points, and values in the interval $[2^{-8},\ 2]$ varying in the log scale. For RS, 30^2 points (λ_1, λ_2) were generated, where $\lambda_1 = R\cos(\theta), \lambda_2 = R\sin(\theta)$, with R varying in the log scale in the interval $[2^{-8}, 2]$ and θ a uniform random variable in $[0, \pi/2]$. For NM, we use the implementation from Feng and Simon, 2017, with 100 iterations.
The matrix $\boldsymbol{X}$ is constructed with i.i.d. Gaussian columns, and the

response is given by,

$$y = \sum_{j=1}^{J^*} \boldsymbol{X}^{(j)} \boldsymbol{\beta}^{(j)} + \sigma\epsilon, \tag{2.30}$$

where $\epsilon \sim N(0, \boldsymbol{I})$, $\boldsymbol{\beta}^{(j)} = [1,\, 2,\, \ldots\, 5,\, 0,\, 0,\, \ldots\, 0]^T$, $J^* \leq J$ are the number of generating groups and the number of groups in the model, respectively, and σ is chosen so that the signal-to-noise ratio is 2.

Table 2.1: Sparse-group lasso parameters tuned using iSGL, GD, RS, GS and NM, over 50 experiments for each configuration. There were considered $90/60/200$ observations in the train/validate/test sample, 4200 predictor variables, and 200 groups of equal size.

	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	$\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}\|_2^2$	fpr_β	fnr_β	fpr_G	fnr_G	time(min.)
1 generating group (5 non-zero coefficients)									
GD	201	**11.02**	20.55	6.82	3.52%	3.2%	13.37%	0%	446.2
GD_0	2	19.71	20.74	7.13	5.2%	2.4%	20.9%	0%	38.3
iSGL	202	12.39	**18.2**	**4.64**	**2.12**%	1.2%	**4.24**%	0%	13.1
iSGL_0	2	19.41	19.9	6.38	2.82%	2%	9.57%	0%	**1.2**
NM	2	19.39	20.24	6.73	7.1%	2%	14.55%	0%	75.2
GS	2	17.94	19.86	6.47	6.75%	2.8%	23.83%	0%	6.61
RS	2	17.89	20.01	6.7	7.23%	2.8%	26.77%	0%	19.34
2 generating groups (10 non-zero coefficients)									
	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	$\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}\|_2^2$	fpr_β	fnr_β	fpr_G	fnr_G	time(min.)
GD	201	**28.27**	48.65	20.49	3.73%	10%	11.55%	0%	466.8
GD_0	2	59.12	63.8	34.69	9.88%	11.2%	32.97%	0%	44.8
iSGL	202	**28.96**	**43.21**	**15.15**	**2.17**%	**5.8**%	**5.42**%	0%	27.1
iSGL_0	2	57.97	61.35	32.14	3.45%	11.2%	11.91%	0%	**1.3**
NM	2	58.63	63.38	34.21	4.09%	11.8%	13.2%	0%	82
GS	2	53.58	59.21	30.35	5.93%	10.6%	21.67%	0%	16.6
RS	2	53.57	59.51	30.69	6.59%	10.8%	21.98%	0%	36.7
3 generating groups (15 non-zero coefficients)									
	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	$\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}\|_2^2$	fpr_β	fnr_β	fpr_G	fnr_G	time(min.)
GD	201	57.09	84.79	45.33	**3.03**%	**13.87**%	**7.62**%	0%	445.3
GD_0	2	132.39	128.24	88.48	13.18%	18.8%	34.78%	0%	43.5
iSGL	202	**49.83**	**77.3**	**36.15**	**2.74**%	**10.53**%	**6.04**%	0.67%	38.2
iSGL_0	2	127.54	121.92	82.04	4.27%	24.27%	14.44%	0.67%	**1.2**
NM	2	131.22	126.87	87.34	5.33%	24.4%	16.44%	0.67%	75.8
GS	2	120.18	119.49	79.1	6.17%	23.2%	20.34%	1.33%	22.6
RS	2	120.27	118.92	78.36	6.09%	22.4%	19.88%	1.33%	46.1

Table 2.1 compares iSGL, GD, NM, GS and RS. Results in Table 2.1 have been averaged over 50 experiments for $J^* = 1, 2, 3$. $\mathtt{R_V}$ and $\mathtt{R_{test}}$ stand for validation error (3.4) and test error (2.13), respectively. The column $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|_2^2$ denotes the euclidean distance between the estimated $\boldsymbol{\beta}$, and the true $\boldsymbol{\beta}$ that generated the model in (2.30).
We have also compared the algorithms in terms of both variable and group selection. fpr_β and fnr_β denote the global false positive and false negative rates, respectively, whereas fpr_G and fnr_G denote those same measures in terms of group selection. We have included a column (`time`) with the mean runtime, aimed to compare the computational complexity between algorithms. However, both RS and GS are easily parallelizable, and thus, their runtime decreases depending on the number of available cores in the system. In these simulations, both RS and GS were run sequentially. We bold those methods that performed significantly better in Table 2.1. For each column, we selected the smallest result and performed paired t-tests of the corresponding method against all the others. We bold both methods if there were no significant differences between the means at level 0.05.

Note that iSGL tunes one extra parameter with respect to GD. This is because Feng and Simon, 2017 considered an equivalent parameterization, taking $\lambda_2 = 1$. In our approach, we believe it is important to consider λ_2 free, so the entire group-lasso regularization term can be weighted against the lasso term with only one step of the algorithm. Although both iSGL and GD appear to be similar in terms of validation and test error minimization, iSGL is considerably faster than GD in runtime. This may be because, as Feng and Simon, 2017 noted, GD needs a very precise optimization of the internal problem (3.5), and that is why they used CVXPY in their implementation. In contrast, iSGL is a coordinate-descent with random search algorithm and therefore, it does not require a very accurate internal solution of (3.5), as long as the validation error is minimized. Our implementation is based on the R package `SGL`, proposed by Simon et al., 2013, but our extension, described in Section 2.3.2, considers different group weights $\boldsymbol{\gamma}$ in the computation of the sparse-group lasso solution.

Figure 2.2 compares iSGL and GD in terms of error minimization, over 50 runs summarized in Table 2.1, with 3 generating groups. Both iSGL and GD perform similar, but the validation and test errors of

GD sometimes exceed those of iSGL. The differences, however, are notable when contrasting the number of parameters to be tuned versus the computational runtime of both algorithms.

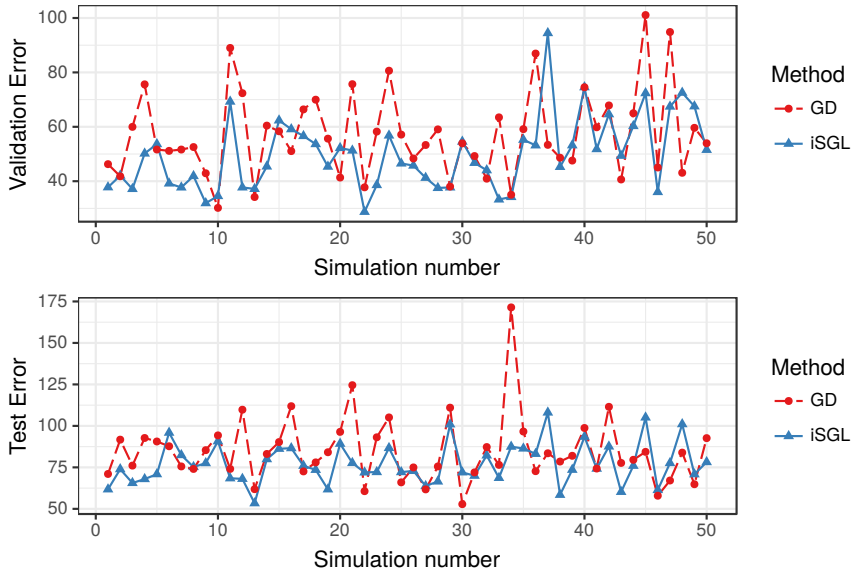


Figure 2.2: Sparse-group lasso parameters tuned using iSGL and Gradient descent from Feng and Simon, 2017 (GD), over 50 experiments, with 3 generating groups (15 non-zero coefficients). Test, training and validation data sizes were $200, 90$, and 60, respectively. There were considered 4200 predictor variables, and 200 groups of equal size.

2.4.2 Binary response

In order to evaluate the effectiveness of the iSGL algorithm on binary data, we have also conducted simulations with logistic response. As reported before, the matrix $\boldsymbol{X}$ is constructed with i.i.d. Gaussian columns, and the linear predictor is given by,

$$\boldsymbol{\eta} = \sum_{j=1}^{J^*} \boldsymbol{X}^{(j)} \boldsymbol{\beta}^{(j)}, \tag{2.31}$$

where $\boldsymbol{\beta}^{(j)} = [1, 2, \ldots 5, 0, 0, \ldots 0]^T$, and $J^* \leq J$ are the number of generating groups and the number of groups in the model, respec-

tively. The binary response is generated as

$$y_i \sim Ber(p_i), \quad p_i = \frac{1}{1 + \exp(-\eta_i)}, \quad i = 1, 2 \dots N. \tag{2.32}$$

Table 2.2 compares both versions of the iSGL. The terms $\texttt{ccr}_V$ and $\texttt{ccr}_T$ denote the correct classification rate of the final model in the validation and test sample sets, respectively. It is important to notice that the version of iSGL that optimizes both $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$ achieves higher $\texttt{ccr}_T$ and smaller error in the estimation of $\boldsymbol{\beta}$, and this difference increases with the number of relevant variables in the model.

Table 2.2: Sparse-group lasso parameters in the logistic regression model, tuned using iSGL over 50 experiments for each configuration. In all the experiments, the test, training and validation data sizes were $200, 90$, and 60, respectively. There were considered 480 predictor variables, and 60 groups of equal size.

	1 generating group (5 non-zero coefficients)								
	$\#\lambda$	$\texttt{ccr}_V$	$\texttt{ccr}_T$	$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|_2^2$	fpr_β	fnr_β	fpr_G	fnr_G	$\texttt{time}$(min.)
iSGL	62	0.95	0.88	17.61	9.89%	2%	12.27%	0%	45.54
iSGL_0	2	0.87	0.87	21.18	23.86%	4%	32.75%	0%	0.35
	2 generating groups (10 non-zero coefficients)								
	$\#\lambda$	$\texttt{ccr}_V$	$\texttt{ccr}_T$	$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|_2^2$	fpr_β	fnr_β	fpr_G	fnr_G	$\texttt{time}$(min.)
iSGL	62	0.96	0.88	26.48	8.86%	2.6%	10.52%	0%	61.41
iSGL_0	2	0.84	0.82	72.29	24.01%	5.4%	31.38%	0%	0.25
	3 generating groups (15 non-zero coefficients)								
	$\#\lambda$	$\texttt{ccr}_V$	$\texttt{ccr}_T$	$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|_2^2$	fpr_β	fnr_β	fpr_G	fnr_G	$\texttt{time}$(min.)
iSGL	62	0.94	0.84	65.12	9.94%	4.53%	10.84%	0%	61.3
iSGL_0	2	0.80	0.79	128.36	24.72%	8.27%	31.47%	0%	0.28

2.5 Application to biomedical data

Simon et al., 2013 explored biological applications of the sparse-group lasso. They compared sparse-group lasso, lasso and group lasso based on their prediction accuracy for a real case-study of gene expressions. The dataset is about *Ulcerative colitis*, previously studied by Burczynski et al., 2006. There are 127 patients in the study, 85 with colitis and 42 healthy, for a total of 8298 gene expressions being measured. In this analysis, variables have been grouped according to the C1 positional gene sets (Subramanian et al., 2005, Liberzon et al., 2015). Those genes not present in the C1 gene set, have been automatically discarded, so in the final model matrix there are included 7819 genes as explanatory variables.

Table 2.3: Result of iSGL on the *Ulcerative colitis* dataset. There were included 127 observations and 7819 genes, grouped into a total of 272 genesets. Standard errors are given in parenthesis.

# $\boldsymbol{\lambda}$	274
# selected genes	33.95 (9.87)
# selected genesets	9.050 (0.96)
`AUC`	0.994 (0.003)
`cutpoint`	0.817 (0.029)
$\texttt{ccr}_V$	0.982 (0.006)
$\texttt{ccr}_T$	0.888 (0.013)

In order to evaluate the power of the iSGL in real data, we have simulated 20 different partitions of the *Ulcerative colitis* dataset into training, validation, and test sample sets, of sizes 49, 31 and 47, respectively. In each run, the training and validation sample sets were passed to the iSGL, as described in Section 2.3. To avoid additional bias, only the training set was considered when fitting the final model. An optimal cut point was also computed, based on the accuracy of the model in the validation sample ($\texttt{ccr}_V$). Using this cut point, we evaluated the accuracy of the model in the test sample set ($\texttt{ccr}_T$). The results of this analysis are summarized in Table 2.3. Note that the accuracy of the model in the validation sample is very high. This is intuitive, considering that both, the regularization parameters in the iSGL and the

cut point, were selected to maximize this measure. In this scenario, a more reliable estimation of the true accuracy of the model is given by the accuracy in the test sample, which is also very good.

On this same data set, Simon et al., 2013 commented that, due to the great number of small groups, $\alpha = 0.05$ (more weight on the group lasso penalty) would be a reasonable selection for the sparse-group lasso penalty parameters in (2.9). We have compared the performance of SGL (Simon et al., 2013) with $\alpha = 0.05$, lasso, group-lasso and the two-parameter version of iSGL, in terms of the correct classification rate (CCR). From the 127 rows in the original dataset, a random sample of 48 observations has been used for the training set, 32 for the validation set, and the remaining 47 for the test set. Figure 2.3 illustrates how each method performed in the validation and test datasets. Since iSGL optimizes the risk in the validation set, it might seem that iSGL has an advantage over the other methods. However, only one solution of iSGL has been computed, whereas SGL, lasso, and group-lasso were fit for a path of $100 - \lambda$ values.

2.6 Discussion

The present study introduces the iterative sparse-group lasso, a novel algorithm to select the optimal regularization parameters of the sparse-group lasso. Being a gradient-free coordinate descent algorithm, one might expect iSGL to have poor performance, compared to other methods that estimate the gradient function. In that sense, this study did not find a significant difference between iSGL and gradient based methods (GD) from Feng and Simon, 2017 with respect to validation and prediction error minimization. The most striking finding, however, is that our approach turned out to be considerable faster. As we mentioned before, a possible explanation for this might be that gradient descent algorithms demand a higher level of accuracy in the optimization of the inner problem (3.5).

Taking into account the results in Table 2.1 and Table 2.2, it is important to highlight that the version of iSGL that optimizes both $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$, achieves the best performance in both the validation and test sample sets. For this reason, in real data studies we recommend to optimize all the regularization parameters, even if it takes more computational

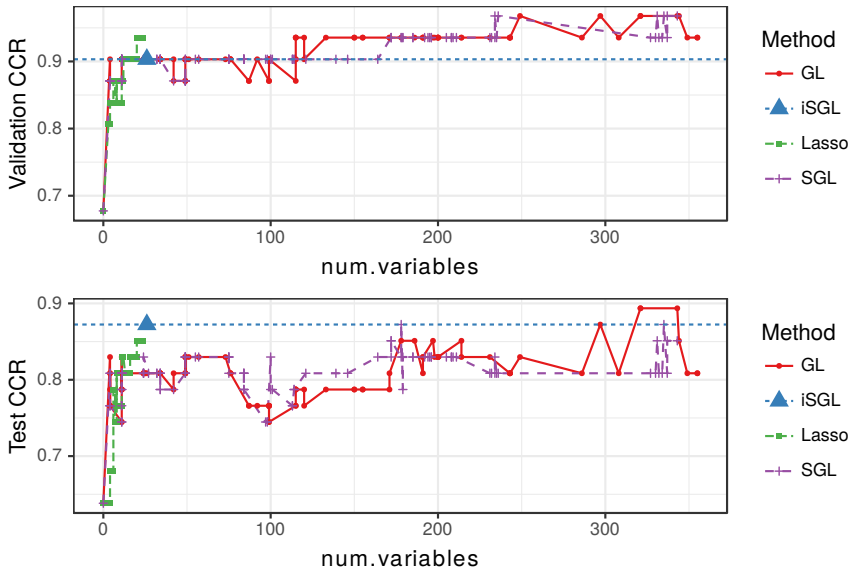


Figure 2.3: Classification accuracy on 32 validation samples (above) and 47 test samples (below) for *Ulcerative colitis* data. Sparse-group lasso (SGL), group-lasso (GL) and Lasso models were fit for a path of $100 - \lambda$ values. For the SGL, $\alpha = 0.05$ was used, according to Simon et al., 2013. The horizontal axis corresponds to the number of non-zero coefficients in the final model.

time.

It is somewhat interesting that, as shown in Table 2.1, when there is exactly one generating group in the model, all the algorithms perform similar in the test set. In contrast, those that select the group parameters $\boldsymbol{\gamma}$ considerably minimize the error in the validation set. It is possible that these results were influenced by an overfitting of the regularization parameters in the validation set. Further studies, which take this overfitting into account, will need to be undertaken.

This research has given rise to many questions in need for further investigation. One of the referees has provided a valuable insight into the extension of our methodology to other regression models. Specifically, we could replace the empirical risk function $\hat{R}(\boldsymbol{\beta})$ by an arbitrary function, which does not even need to be in the form (2.3). In particular, research into extending the sparse group lasso formulation

and the iSGL to Poisson regression is already underway.

The iSGL algorithm presented in this paper can be of broad use to the scientific and biomedical communities. The evidence from simulations and real data analysis suggests that properly selecting the regularization parameters in the sparse-group lasso leads to better results, when it comes to both variable selection and prediction. Further research might explore the extension of Algorithm 3 to other regularization methods which depend on multiple parameters.

2.7 Supplementary files

All the simulations and data analysis in Section 2.4 were run in the same computer, a node with two `Intel(R) Xeon(R) CPU E5-2630 v3` (2.4 GHz, 20 MB Smart Cache) processors with 32Gb of RAM, running CentOS 6.5 Final (Rocks 6.1.1 Sand Boa), R 3.4.2 and python 2.7.14 (Anaconda, Inc.).

The iSGL algorithm is distributed under the `sglfast` package, for installation in R (See R Core Team, 2014). It is available at `https://github.com/jlaria/sglfast/` and can be installed in R via `devtools`. Our `sglfast` package is a fork of the `SGL` package for R. See Simon et al., 2013.

The source code for the simulations and analysis in Sections 2.4 and 2.5 is also available at `https://github.com/jlaria/isgl-paper/`.

Acknowledgments

We gratefully acknowledge the constructive comments of the associate editor and the anonymous referees.

This research has been partially supported by research grants and projects ECO2015-66593-P and MTM2017-88708-P (Ministerio de Economia, Industria y Competitividad).

Appendix

Proof of propositions in Section 2.3

Proposition 1. Let

$$\hat{R}(\boldsymbol{\beta}) = \frac{1}{2N}\left\| y - \beta_0\mathbf{1} - \sum_{j=1}^{J} \boldsymbol{X}^{(j)}\boldsymbol{\beta}^{(j)} \right\|_2^2 .$$

Consider $\boldsymbol{\lambda}, \boldsymbol{\gamma}$ fixed and let $\boldsymbol{\beta}$ be fixed except in components $\boldsymbol{\beta}^{(k)}$. Then (3.3) becomes,

$$\boldsymbol{\beta}^{(k)} = \arg\min_{\beta} \left\{ \frac{1}{2N}\left\| y - \beta_0\mathbf{1} - \sum_{j=0}^{J} \boldsymbol{X}^{(j)}\beta^{(j)} \right\|_2^2 + \lambda_2 \sum_{j=1}^{J} \gamma_j \|\boldsymbol{\beta}^{(j)}\|_2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \right\}. \tag{2.33}$$

The objective function of (2.33) is convex, then the optimal solution is characterized by the subgradient equations,

$$0 \in -\frac{1}{N}\boldsymbol{X}^{(k)\prime}\left(y - \beta_0\mathbf{1} - \sum_{j=1}^{J} \boldsymbol{X}^{(j)}\boldsymbol{\beta}^{(j)} \right) + \lambda_2\gamma_k u + \lambda_1 v, \tag{2.34}$$

where

$$u = \begin{cases} \boldsymbol{\beta}^{(k)}/\|\boldsymbol{\beta}^{(k)}\|_2, & \text{if } \boldsymbol{\beta}^{(k)} \neq 0 \\ \in \{u : \|u\|_2 \leq 1\}, & \text{if } \boldsymbol{\beta}^{(k)} = 0 \end{cases}, \tag{2.35}$$

$$v_i = \begin{cases} \operatorname{sign}(\boldsymbol{\beta}_i^{(k)}), & \text{if } \boldsymbol{\beta}_i^{(k)} \neq 0 \\ \in \{v_i : |v_i| \leq 1\}, & \text{if } \boldsymbol{\beta}_i^{(k)} = 0 \end{cases}. \tag{2.36}$$

If $\lambda_2 = 0$, the subgradient conditions for a particular component $\beta_i^{(k)}$ become,

$$0 \in -\frac{1}{N}\boldsymbol{X}_i^{(k)\prime}\left(y - \beta_0\mathbf{1} - \sum_{j=0}^{J} \boldsymbol{X}^{(j)}\boldsymbol{\beta}^{(j)} \right) + \lambda_1 v_i. \tag{2.37}$$

This is true for $\beta_i^{(k)} = 0$ if

$$0 = -\frac{1}{N}\boldsymbol{X}_i^{(k)'}\boldsymbol{r}_{-k,i} + \lambda_1 v_i.$$

Then,

$$v_i = \frac{1}{\lambda_1 N}\left(\boldsymbol{X}_i^{(k)'}\boldsymbol{r}_{-k,i}\right),$$

which satisfies (2.36) only if

$$\left|\frac{1}{N}\boldsymbol{X}_i^{(k)}\boldsymbol{r}_{-k,i}\right| \leq \lambda_1. \tag{2.38}$$

□

Proposition 3. From (2.34), it follows that $\boldsymbol{\beta}^{(k)} = 0$ is the solution of (2.34), if there is $u \in \{u : \|u\|_2 \leq 1\}$ such that

$$0 = -\frac{1}{N}\boldsymbol{X}^{(k)'}\boldsymbol{r}_{-k} + \lambda_2\gamma_k u + \lambda_1 v, \tag{2.39}$$

Solving (2.39) for u,

$$u = \frac{1}{\lambda_2\gamma_k}\left(\frac{1}{N}\boldsymbol{X}^{(k)'}\boldsymbol{r}_{-k} - \lambda_1 v\right).$$

Since $\|u\|_2 \leq 1$, then the subgradient equation (2.39) is satisfied if for some v,

$$\left\|\frac{1}{N}\boldsymbol{X}^{(k)'}\boldsymbol{r}_{-k} - \lambda_1 v\right\|_2 \leq \lambda_2\gamma_k. \tag{2.40}$$

In particular, $\boldsymbol{\beta}^{(k)} = 0$ if

$$\min_v \left\|\frac{1}{N}\boldsymbol{X}^{(k)'}\boldsymbol{r}_{-k} - \lambda_1 v\right\|_2 \leq \lambda_2\gamma_k, \tag{2.41}$$

subject to $\|v_i\| \leq 1$, for $i = 1, 2 \ldots p_k$. This minimum is attained when

$$v_i^* = \begin{cases} -1, & \text{if } \frac{1}{N}(\boldsymbol{X}^{(k)'}\boldsymbol{r}_{-k})_i < -\lambda_1 \\ 1, & \text{if } \frac{1}{N}(\boldsymbol{X}^{(k)'}\boldsymbol{r}_{-k})_i > \lambda_1 \\ \frac{1}{\lambda_1 N}(\boldsymbol{X}^{(k)'}\boldsymbol{r}_{-k})_i, & \text{if } |\frac{1}{N}(\boldsymbol{X}^{(k)'}\boldsymbol{r}_{-k})_i| \leq \lambda_1 \end{cases}.$$

Substituting v^*, (2.41) becomes,

$$\left\| S\left(\frac{1}{N} \boldsymbol{X}^{(k)'} \boldsymbol{r}_{-k}, \lambda_1 \right) \right\|_2 \leq \lambda_2 \gamma_k,$$

where $S(\cdot, \cdot)$ is the coordinate-wise soft thresholding operator. □

Propositions 2 and 4. The proofs to these propositions are analogous to those of Propositions 1 and 3. Only that, instead of (2.33), we have

$$\begin{aligned} \boldsymbol{\beta}^{(k)} = \arg\min_{\beta} \Bigg\{ \frac{1}{N} \sum_{i=1}^{N} [\log\left(1 + \exp\{\beta_0 + \boldsymbol{x}_i' \boldsymbol{\beta}\}\right) \\ - y_i(\beta_0 + \boldsymbol{x}_i' \boldsymbol{\beta})] + \lambda_2 \sum_{j=1}^{J} \gamma_j \|\boldsymbol{\beta}^{(j)}\|_2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \Bigg\}. \end{aligned} \tag{2.42}$$

□

Further simulations

Tables 2.4-2.6 support the results from Section 2.4.

Table 2.4: Sparse-group lasso parameters tuned using iSGL and Gradient descent from Feng and Simon, 2017 (GD), over 30 experiments for each configuration. Standard errors are given in parenthesis. In all the experiments, the test, training and validation data sizes were $200, 90$, and 60, respectively. There were considered 600 predictor variables, and 30 groups of equal size.

	1 generating groups (5 non-zero coefficients)			
	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	`time`(sec.)
iSGL	2	8.98 (0.39)	9.26 (0.3)	**0.38** (0.02)
iSGL	32	**6.58** (0.33)	8.82 (0.27)	7.16 (2.84)
GD	2	9.02 (0.38)	9.31 (0.3)	140.64 (12.11)
GD	31	**6.27** (0.31)	9.69 (0.28)	1958.53 (139.88)
	2 generating groups (10 non-zero coefficients)			
	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	`time`(sec.)
iSGL	2	23.7 (0.81)	26.43 (0.95)	**0.67** (0.09)
iSGL	32	**14.77** (0.6)	**21.76** (0.57)	11.5 (3.42)
GD	2	24.23 (0.9)	27.11 (1.01)	149.41 (17.77)
GD	31	**15** (0.56)	23.23 (0.65)	1943.54 (134.5)
	3 generating groups (15 non-zero coefficients)			
	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	`time`(sec.)
iSGL	2	45.99 (1.63)	47.9 (1.2)	**0.67** (0.09)
iSGL	32	**26.41** (1.34)	**36.26** (1.42)	16.23 (3.04)
GD	2	46.85 (1.67)	49.65 (1.43)	153.51 (16.1)
GD	31	**26.7** (1.28)	39.15 (1.67)	1636.3 (120.91)

Table 2.5: Sparse-group lasso parameters tuned using iSGL and Gradient descent from Feng and Simon, 2017 (GD), over 30 experiments for each configuration. Standard errors are given in parenthesis. In all the experiments, the test, training and validation data sizes were $200, 90$, and 60, respectively. There were considered 900 predictor variables, and 60 groups of equal size.

	1 generating group (5 non-zero coefficients)			
	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	time(sec.)
iSGL	2	9.73 (0.35)	9.28 (0.34)	**0.66** (0.02)
iSGL	62	7.13 (0.27)	**8.66** (0.26)	12.01 (4.46)
GD	2	9.81 (0.36)	9.4 (0.38)	333.69 (27.64)
GD	61	**6.08** (0.22)	10.14 (0.33)	4004.51 (326.24)
	2 generating groups (10 non-zero coefficients)			
	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	time(sec.)
iSGL	2	25.34 (1.46)	26.1 (0.83)	**0.94** (0.14)
iSGL	62	**13.76** (0.74)	**21.11** (0.71)	53.71 (26.67)
GD	2	25.87 (1.48)	27.15 (0.96)	375.71 (44.24)
GD	61	**13.83** (0.64)	23.43 (0.6)	3912.32 (241.89)
	3 generating groups (15 non-zero coefficients)			
	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	time(sec.)
iSGL	2	43.87 (2.16)	46.63 (1.25)	**1.11** (0.1)
iSGL	62	**22.63** (1.13)	**34.42** (0.81)	79.53 (38.22)
GD	2	44.84 (2.17)	47.61 (1.3)	304.19 (31.29)
GD	61	**23.58** (1.15)	38.87 (1.13)	4281.04 (363.55)

Table 2.6: Sparse-group lasso parameters tuned using iSGL and Gradient descent from Feng and Simon, 2017 (GD), over 30 experiments for each configuration. Standard errors are given in parenthesis. In all the experiments, the test, training and validation data sizes were $200, 90$, and 60, respectively. There were considered 1200 predictor variables, and 100 groups of equal size.

	1 generating group (5 non-zero coefficients)			
	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	`time`(sec.)
iSGL	2	8.76 (0.49)	9.89 (0.31)	**1.16** (0.02)
iSGL	102	**5.82** (0.33)	**9.11** (0.3)	44.35 (20.68)
GD	2	8.8 (0.5)	9.98 (0.33)	594.77 (58.88)
GD	101	**5.2** (0.31)	10.74 (0.33)	6024.1 (770.35)
	2 generating groups (10 non-zero coefficients)			
	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	`time`(sec.)
iSGL	2	22.67 (0.88)	24.52 (0.71)	**1.35** (0.05)
iSGL	102	**12.32** (0.56)	**20.7** (0.63)	47.17 (10.23)
GD	2	23.32 (0.92)	25.31 (0.71)	563.11 (70.08)
GD	101	**12.37** (0.46)	22.54 (0.67)	6919.35 (716.22)
	3 generating groups (15 non-zero coefficients)			
	$\#\lambda$	$\mathtt{R_V}$	$\mathtt{R_{test}}$	`time`(sec.)
iSGL	2	50.62 (2.14)	57.05 (2.24)	**1.41** (0.08)
iSGL	102	**22.16** (1.1)	**38.78** (2.04)	86.28 (20.11)
GD	2	50.33 (2.08)	58.46 (2.35)	443.72 (48.46)
GD	101	**22.16** (1.12)	**40.57** (1.3)	7009.67 (809.94)

Bibliography for Chapter 2

Beck, Amir and Marc Teboulle (2009). "Gradient-based algorithms with applications to signal recovery". In: *Convex optimization in signal processing and communications*, pp. 42–88.

Bergstra, James and Yoshua Bengio (2012). "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* 13.Feb, pp. 281–305.

Burczynski, Michael E et al. (2006). "Molecular classification of Crohn's disease and ulcerative colitis patients using transcriptional profiles in peripheral blood mononuclear cells". In: *The journal of molecular diagnostics* 8.1, pp. 51–61.

Chatterjee, Soumyadeep et al. (2011). "Sparse group lasso for regression on land climate variables". In: *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*. IEEE, pp. 1–8.

Cheng, Jun et al. (2017). "Similarity regularized sparse group lasso for cup to disc ratio computation". In: *Biomedical Optics Express* 8.8, pp. 3763–3777.

Diamond, Steven and Stephen Boyd (2016). "CVXPY: A Python-Embedded Modeling Language for Convex Optimization". In: *Journal of Machine Learning Research* 17.83, pp. 1–5.

Domahidi, A., E. Chu, and S. Boyd (2013). *ECOS: An SOCP solver for embedded systems*, pp. 3071–3076.

Feng, Jean and Noah Simon (2017). "Gradient-based Regularization Parameter Selection for Problems with Non-smooth Penalty Functions". In: *Journal of Computational and Graphical Statistics* 0.ja, doi:10.1080/10618600.2017.1390470. DOI: 10.1080/10618600.2017.1390470. eprint: https://doi.org/10.1080/10618600.2017.1390470. URL: https://doi.org/10.1080/10618600.2017.1390470.

Friedman, Jerome, Trevor Hastie, and Rob Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent". In: *Journal of statistical software* 33.1, p. 1.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2010). "A note on the group lasso and a sparse group lasso". In: *arXiv preprint arXiv:1001.0736*.

Huang, Jian et al. (2009). "A group bridge approach for variable selection". In: *Biometrika* 96.2, pp. 339–355.

Lange, Kenneth, David R Hunter, and Ilsoon Yang (2000). "Optimization transfer using surrogate objective functions". In: *Journal of computational and graphical statistics* 9.1, pp. 1–20.

Larochelle, Hugo et al. (2007). "An empirical evaluation of deep architectures on problems with many factors of variation". In: *Proceedings of the 24th international conference on Machine learning*. ACM, pp. 473–480.

Liberzon, Arthur et al. (2015). "The molecular signatures database hallmark gene set collection". In: *Cell systems* 1.6, pp. 417–425.

Ndiaye, Eugene et al. (2016). "Gap safe screening rules for sparse-group lasso". In: *Advances in Neural Information Processing Systems*, pp. 388–396.

Nelder, John A and Roger Mead (1965). "A simplex method for function minimization". In: *The computer journal* 7.4, pp. 308–313.

Nesterov, Yurii (2007). *Gradient methods for minimizing composite objective function*. CORE.

O'Donoghue, B. et al. (Apr. 2016). *SCS: Splitting Conic Solver, version 1.2.7*. https://github.com/cvxgrp/scs.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: http://www.R-project.org/.

Rao, Nikhil et al. (2016). "Classification with the sparse group lasso". In: *IEEE Transactions on Signal Processing* 64.2, pp. 448–463.

Simon, Noah et al. (2013). "A sparse-group lasso". In: *Journal of computational and graphical statistics* 22.2, pp. 231–245.

Sprechmann, Pablo et al. (2011). "C-HiLasso: A collaborative hierarchical sparse modeling framework". In: *IEEE Transactions on Signal Processing* 59.9, pp. 4183–4198.

Subramanian, Aravind et al. (2005). "Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles". In: *Proceedings of the National Academy of Sciences* 102.43, pp. 15545–15550.

Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1, pp. 267–288.

Vincent, Martin and Niels Richard Hansen (2014). "Sparse group lasso and high dimensional multinomial classification". In: *Computational Statistics & Data Analysis* 71, pp. 771–786.

Xie, Zongxia and Yong Xu (2014). “Sparse group LASSO based uncertain feature selection”. In: *International Journal of Machine Learning and Cybernetics* 5.2, pp. 201–210.

Yuan, Ming and Yi Lin (2006). “Model selection and estimation in regression with grouped variables”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.1, pp. 49–67.

Zhao, Weihua, Riquan Zhang, and Jicai Liu (2014). “Sparse group variable selection based on quantile hierarchical Lasso”. In: *Journal of Applied Statistics* 41.8, pp. 1658–1677.

Zhou, Nengfeng and Ji Zhu (2010). “Group variable selection via a hierarchical lasso and its oracle property”. In: *Statistics and Its Interface* 3, pp. 557–574.

Zou, Hui and Trevor Hastie (2003). “Regression shrinkage and selection via the elastic net, with applications to microarrays”. In: *Journal of the Royal Statistical Society: Series B. v67*, pp. 301–320.

CHAPTER 3

Iterative variable selection for high-dimensional data: Prediction of pathological response in triple-negative breast cancer

In *Working paper. Statistics and Econometrics 20-04*. Universidad Carlos III de Madrid. Departamento de Estadística.

Juan C. Laria[a,b] M. C. Aguilera-Morillo[a,b] E. Álvarez[c]
Rosa E. Lillo[a,b] S. López-Taruella[c] M. del Monte-Millán[c]
A. C. Picornell[c] Miguel Martín[c] Juan Romo[a,b]

[a] Department of Statistics, University Carlos III of Madrid, Spain
[b] UC3M-BS Institute of Financial Big Data, Madrid, Spain
[c] Department of Medical Oncology, Hospital General Universitario Gregorio Marañón, Instituto de Investigación Sanitaria Gregorio Marañón, Universidad Complutense, CIBERONC, GEICAM, Madrid, Spain

Abstract

In the last decade, regularized regression methods have offered alternatives for performing multi-marker analysis and feature selection in a whole genome context. The process of defining a list of genes that will characterize an expression profile, remains unclear. This procedure oscillates between select-

ing the genes or transcripts of interest based on previous clinical evidence, or performing a whole transcriptome analysis that rests on advanced statistics. This paper introduces a methodology to deal with the variable selection and model estimation problems in the high-dimensional set-up, which can be particularly useful in the whole genome context. Results are validated using simulated data, and a real dataset from a triple-negative breast cancer study.

Keywords: Variable selection. High-dimension. Regularization. Classification

3.1 Introduction

Breast cancer (BC) is the most frequent cancer among women, representing around 25% of all newly diagnosed cancer in women (Ferlay et al., 2014). One in eight women in developed countries will be diagnosed with BC over the course of a lifetime.

The prognosis of this disease has progressively improved over the past three decades, due to the implementation of population-based screening campaigns and, above all, the introduction of new effective targeted medical therapies, i.e., aromatase inhibitors (effective in hormone receptor-positive tumors) and trastuzumab (effective in HER2-positive tumors). Breast cancer is, however, a heterogeneous disease. The worst outcomes are associated with the so-called triple-negative breast cancer subtype (TNBC), diagnosed in 15-20% of BC patients. TNBC is defined by a lack of immunohistochemistry expression of the estrogen and progesterone receptors and a lack of expression/amplification of HER2 (Dent et al., 2007). The absence of expression of these receptors makes chemotherapy the only available therapy for TNBC.

TNBC is usually diagnosed in an operable (early) stage. Surgery, chemotherapy and radiation therapy are the critical components of the treatment of early TNBC. Many early TNBC patients are treated with upfront chemotherapy (neoadjuvant chemotherapy, NACT) and then operated on and, perhaps, irradiated. The rationale for this sequence is the ability to predict the long-term outcome of patients looking at the pathological response achieved with initial NACT (Cortazar et al.,

2014).

With the currently available neoadjuvant chemotherapy regimens, nearly 50% of TNBC achieve a good pathological response to this therapy, while the remaining patients have an insufficient response. TNBC patients achieving a complete or almost complete disappearance of the tumor in breast and axilla after NACT have an excellent outcome (less than 10% of relapses at five years), in contrast with those with significant residual disease (more than 50% of relapses at five years) (Symmans et al., 2017; Sharma, López-Tarruella, Garcia-Saenz, et al., 2018).

The identification of these two different populations is therefore of the utmost relevance, in order to test new experimental therapies in the population unlikely to achieve a good pathological response.

Several tumor multigene predictors of pathological response of operable BC to NACT have been proposed in the past few years, taking advantage of the recent decreased economic cost of obtaining an individual's full transcriptome (Tabchy et al., 2010; Hatzis et al., 2011; Chang et al., 2003). Most of them have been tested in unselected populations of BC patients and have shown insufficient positive predictive value and sensitivity.

The process of defining a list of genes that will define a characteristic expression profile is still ambiguous. This process oscillates between selecting the genes or transcripts of interest based on the clinical evidence in previous studies or using an agnostic point of view that rests on advanced statistics selection processes in multivariate analysis. RNA-Seq has become one of the most appealing tools of modern whole transcriptome analyses because it combines relatively low cost and a comprehensive approach to transcript quantification. Some approaches to complex disease biomarker discovery already pointed to the need to use a whole genome perspective using joint information in order to predict complex traits instead of a priori selecting individual features (De Los Campos, Gianola, and Allison, 2010; Lupski et al., 2011). This strategy would lead to high predictive accuracy, and there would be no need to know the precise biological associations in the genome background because of the high correlation among the biomarkers (Offit, 2011). This approach is challenging from the statistical point of view because of the large number of biomarkers to be tested along the genome related to the rather small sample sizes

in clinical studies. On the other hand, daily clinical practice scenario requires cheaper and faster quantification platforms than whole-genome RNA-Seq analysis. Thus, it is needed to reduce the number of biomarkers to stick with in order to define a practical gene expression signature for the clinical community.

The regularized regression methods provide alternatives for performing multi-marker analysis and feature selection in a whole genome context (Szymczak et al., 2009). Specifically, we focus on the sparse-group lasso (SGL) regularization method (Simon et al., 2013), which generalizes lasso (Tibshirani, 1996), group lasso (Yuan and Lin, 2006) and elastic-net (Zou and Hastie, 2003), merging lasso and group lasso penalties. The solution provided by SGL, usually involves a small number of predictor variables, given that many coefficients in the solution are exactly zero. It has an advantage over lasso when the predictor variables are grouped, as many groups are entirely zeroed out, but unlike group lasso, the solution is also sparse within those groups that are not completely eliminated from the model. However, as will be explained in next sections, the SGL is not appropriate for the problem we are dealing with, without introducing a broader methodology to control the regularization hyper-parameters, the groups, and the high-dimensionality issue.

From a methodological point of view, this paper provides an original contribution to perform variable selection and model fitting in high-dimensional problems. Furthermore, the results presented in this paper are the first attempt in a Translational Oncology scenario of building a predictive model for the response to treatment, based entirely on the whole genome RNA-Seq data and conventional clinical variables.

This paper is organized as follows. Section 3.2 ties together the various theoretical concepts that support our approach. Section 3.2.1 introduces the mathematical formulation of the SGL, as an optimization problem. Section 3.2.2 discusses the iterative-sparse group lasso, a coordinate descent algorithm to automatically select the regularization parameters of the SGL. Section 3.2.3 describes a clustering strategy for the variables, based on principal component analysis, which makes it possible to work with an arbitrarily large number of variables, without specifying the groups apriori. Section 3.2.5 highlights

our main methodological contributions: the *importance* and the *power indexes*, to weight variables and models, respectively. In Section 3.3, a simulation study is presented, with several synthetic matrix designs, and varying the number of variables from 40 to 4000. Section 3.4 highlights the contributions of our methodology on a TNBC cohort which had undergone neoadjuvant docetaxel/carboplatin chemotherapy. Some conclusions and lines for future work, are drawn in the final section.

3.2 Methodology and algorithms

Consider the usual logistic regression framework, with N observations in the form $\{y^{(i)}, x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}\}_{i=1}^N$, where p is the number of features or predictor variables, and $y^{(i)}$ is the binary response. We assume that the response comes from a random variable with conditional distribution,

$$Y|(X_1 \ldots X_p) \sim Ber(p(X_1 \ldots X_p, \boldsymbol{\beta})),$$

where

$$p(X_1 \ldots X_p, \boldsymbol{\beta}) = (1 + \exp(-\eta))^{-1},$$

and η is the linear predictor,

$$\eta = \beta_0 + \sum_{j=1}^{p} \beta_j X_j, \quad \boldsymbol{\beta} = [\beta_0\, \beta_1\, \ldots\, \beta_p] \in \mathbb{R}^{p+1}.$$

The objective is to predict the response Y for future observations of $X_1 \ldots X_p$, using an estimation of the unknown parameter $\boldsymbol{\beta}$, given by,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p+1}}{\operatorname{argmin}}\, \hat{R}(\boldsymbol{\beta}), \tag{3.1}$$

were

$$\hat{R}(\boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} \left[\log\left(1 + \exp\{\beta_0 + \sum_{j=1}^{p} \beta_j x_j^{(i)}\}\right) - y_i(\beta_0 + \sum_{j=1}^{p} \beta_j x_j^{(i)}) \right]. \tag{3.2}$$

The problem with this approach is that for $N < p$, the minimization (3.1) has infinite optimal solutions. When the features $X_1 \ldots X_p$

represent genetic expressions, this problem of predicting Y becomes more extreme, since we often have N several orders of magnitude smaller than p.

As a solution, variable selection techniques are proposed, in order to tackle the analytical intractability of this problem.

3.2.1 The sparse-group lasso

It has been shown that SGL can play an important role in addressing the issue of variable selection in genetic models, where genes are grouped following different pathways. The mathematical formulation of this problem is,

$$\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\operatorname{argmin}} \left\{ \hat{R}(\boldsymbol{\beta}) + \lambda_2 \sum_{j=1}^{J} \gamma_j \|\boldsymbol{\beta}^{(j)}\|_2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \right\}. \qquad (3.3)$$

Here J is the number of groups, and $\boldsymbol{\beta}^{(j)} \in \mathbb{R}^{p_j}$ are vectors with the components of $\boldsymbol{\beta}$ corresponding to j-th group (of size p_j), and $\gamma_j = \sqrt{p_j}$, $j = 1, 2, \ldots, J$. The regularization parameter is $\boldsymbol{\lambda} = [\lambda_1\ \lambda_2] \in \mathbb{R}^2_+$.

The problem with (3.3) is that vector $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$ of estimated coefficients depends on the selection of a vector of regulation parameters $\boldsymbol{\lambda}$, which must be chosen before estimating $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$. The selection of $\boldsymbol{\lambda}$ is partly an open problem, because although there are several practical strategies for choosing these parameters, there is no established theoretical criterion to follow. In most cases, the regularization parameters are set a priori, based on some additional information about the data, or the characteristics of the desired solution, e.g., greater λ_1 implies more components of $\hat{\boldsymbol{\beta}}$ identically zero. The most commonly used methodology to select $\boldsymbol{\lambda}$ consists of moving the regulation parameters in a fixed grid, usually not very thin. However, this approach has many disadvantages.(Laria, Carmen Aguilera-Morillo, and Lillo, 2019) In contrast, we propose the iterative-sparse group lasso, a coordinate descent algorithm, recently introduced by Laria, Carmen Aguilera-Morillo, and Lillo.

3.2.2 Selection of the optimal regularization parameter

Traditionally, the data set $\mathcal{Z} = \{y^{(i)}, x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}\}_{i=1}^N$ is partitioned into three disjoint data sets, $\mathcal{Z}_T$, $\mathcal{Z}_V$ and $\mathcal{Z}_{test}$. The data in $\mathcal{Z}_T$ is used for training the model, i.e., solving (3.3). $\mathcal{Z}_V$ is used for validation, i.e., finding the optimal parameter $\boldsymbol{\lambda}$. The remaining observations in $\mathcal{Z}_{test}$ are used for testing the prediction ability of the model on future observations. Specifically, the selection of the optimal parameter $\boldsymbol{\lambda}$ is based on the minimization of the validation error, defined as

$$\hat{R}_V(\boldsymbol{\lambda}) = \frac{1}{\#\mathcal{Z}_V} \sum_{(y^{(i)}, \boldsymbol{x}^{(i)}) \in \mathcal{Z}_V} \left[\log(1 + \exp\{\eta(\hat{\boldsymbol{\beta}}_T)\}) - y^{(i)} \eta(\hat{\boldsymbol{\beta}}_T) \right], \tag{3.4}$$

where

$$\hat{\boldsymbol{\beta}}_T(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta} \in B} \left\{ \hat{R}_T(\boldsymbol{\beta}) + \lambda_2 \sum_{j=1}^{J} \gamma_j \|\boldsymbol{\beta}^{(j)}\|_2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \right\}, \tag{3.5}$$

and

$$\hat{R}_T(\boldsymbol{\beta}) = \frac{1}{\#\mathcal{Z}_T} \sum_{(y^{(i)}, \boldsymbol{x}^{(i)}) \in \mathcal{Z}_T} \left[\log(1 + \exp\{\eta(\hat{\boldsymbol{\beta}}_T)\}) - y^{(i)} \eta(\hat{\boldsymbol{\beta}}_T) \right], \tag{3.6}$$

with $\#$ denoting the cardinal of a set. Therefore, the problem of finding the optimal parameter $\boldsymbol{\lambda}$ can be formulated as,

$$\begin{aligned} &\min_{\boldsymbol{\lambda} \in \mathbb{R}_+^2} \hat{R}_V(\boldsymbol{\lambda}) \\ \text{s.t.} \quad &\hat{\boldsymbol{\beta}}_T(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{p+1}} \left\{ \hat{R}_T(\boldsymbol{\beta}) + \lambda_2 \textstyle\sum_{j=1}^{J} \gamma_j \|\boldsymbol{\beta}^{(j)}\|_2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \right\}. \end{aligned} \tag{3.7}$$

Algorithm 3 describes the two-parameter ITERATIVE SPARSE-GROUP LASSO (iSGL$_0$), a gradient-free coordinate descent method to tune the parameter $\boldsymbol{\lambda}$ from the sparse-group lasso (3.3), which performs well under different scenarios while drastically reducing the number of operations required to find optimal penalty weight parameters that minimize the validation error in (3.4). The iSGL$_0$ iteratively performs a univariate minimization over one of the coordinates of $\boldsymbol{\lambda}$, while the other coordinate is fixed.

Algorithm 3: TWO-PARAMETER ITERATIVE SPARSE-GROUP LASSO (iSGL_0)

```
/* Data for training/validation                                    */
```
Function `isgl`$(\mathcal{Z}_T, \mathcal{Z}_V)$**:**
  Initialize $\boldsymbol{\lambda}$ $i \leftarrow 1$
  while $\boldsymbol{\lambda}$ *not stationary* **do**
    $\lambda_i \leftarrow \text{argmin}_{\lambda \in \mathbb{R}_+} \hat{R}_V(\boldsymbol{\lambda} | \lambda_i = \lambda)$; `// minimize over coordinate` i `of` $\boldsymbol{\lambda}$
    $i \leftarrow i \bmod 2 + 1$; `// Next coordinate`
  end
  return $\hat{\boldsymbol{\beta}}_T(\boldsymbol{\lambda})$

Laria, Carmen Aguilera-Morillo, and Lillo, 2019 provide detailed information about Algorithm 3 in their paper. As mentioned before, a very useful property of the sparse-group lasso as a variable selection method, is the ability to remove entire groups from the model (sending to zero the components of the $\hat{\boldsymbol{\beta}}$ vector relative to those groups), as is the case with group lasso. However, this means that a grouping among the variables under consideration must be specified. This does not entail a challenge if there are natural groupings among the variables, for example, if the variables are dummies related to different levels of the same original categorical variable. However, in our study most of the variables are transcriptomes, for which there are no established groupings in the literature. To overcome this problem, we suggest an empirical variable grouping approach, based on the principal component analysis of the data matrix.

3.2.3 Grouping variables using principal component analysis

Principal component analysis (PCA) is a dimension reduction technique, very effective in reducing a large number of variables related to each other to a few latent variables, trying to lose the minimum amount of information. The new latent variables obtained (the principal components), which are a linear transformation of the original variables, are uncorrelated and ordered in such a way that the first components capture most of the variation present in all the original

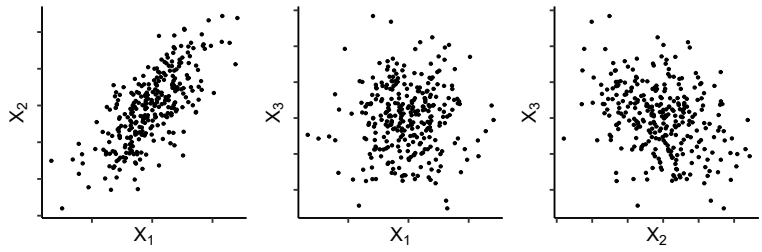
variables.

Given the data matrix $\mathbf{X} \in \mathbb{R}^{N \times p}$, PCA computes the rotation matrix $\mathbf{W} \in \mathbb{R}^{p \times G}$, where $G \leq \min(N, p)$ is the number of principal component to retain. The transformed data matrix (the principal component matrix) is $\mathbf{T} = \mathbf{XW}$. This rotation matrix $\mathbf{W}$ suggest a natural grouping on the columns of $\mathbf{X}$, given by

$$group(X_j) = \arg\max_i |W_{ji}|, \quad j = 1, 2, \ldots, p. \tag{3.8}$$

This strategy will provide at most G groups on the columns of $\mathbf{X}$.

Figure 3.1: Simulated sample from three random variables, that illustrate the grouping based on PCA.



The following example illustrates our approach on a simulated data set. Suppose that we want to cluster variables X_1, X_2 and X_3 using two groups. There are 300 observations (Fig. 3.1) and they are simulated such that corr$(X_1, X_2) = 0.75$, corr$(X_1, X_3) = 0.1$ and corr$(X_2, X_3) = -0.25$. The principal component's rotation matrix $\mathbf{W}$ is given by,

	PC1	PC2
X_1	**-0.67**	0.40
X_2	**-0.70**	-0.08
X_3	0.23	**0.91**

In this example, X_1 and X_2 would be grouped together, whereas X_3 would be in the other group. Apparently, this method is placing highly correlated variables in the same group.

3.2.4 Mining influent variables under a cross-validation approach

In this section, we focus on the problem of variable selection in models where the ratio p/N is in the order of 10^2. In these scenarios, even state-of-the-art methods such as SGL find it hard to select an appropriate set of variables related to the response term. We propose a cross-validation approach to fit and evaluate many different models using only a sample size of N observations initially given.

The solution in terms of $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$ provided by Algorithm 3 strongly depends on the partition $\mathcal{Z}_T, \mathcal{Z}_V$. As a consequence, if we run Algorithm 3 for different partitions $\mathcal{Z}_T, \mathcal{Z}_V$ of the same data $\mathcal{Z}$, it will probably result in different coefficient estimates $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$. Therefore, the indicator function of variable X_j included in the model, $\boldsymbol{I}(\hat{\beta}_j(\boldsymbol{\lambda}) \neq 0)$, will take different values depending on the partition $\mathcal{Z}_T, \mathcal{Z}_V$. In order to avoid this dependency on the sample data partition, we propose Algorithm 4, which computes many different solutions $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$ of Algorithm 3, for different partitions of the original data sample $\mathcal{Z}$. The goal of this algorithm is to be able to fit and evaluate many models using the same data. Since the sample size is small compared to the number of covariates, the variable selection will greatly depend on the train/validate partition. We denote by R the total number of models that will be fitted using different partitions from the original sample. Algorithm 4 stores the information of the fitting $\hat{\boldsymbol{\beta}}$ of each model and the correct classification rate in the validation sample (ccr_V) in each case.

Algorithm 4:

```
/* sample data Z, # of runs R                                    */
```
Function `isgl(`$\mathcal{Z}$`,` R`)`**:**

  for r *in* $1, 2 \ldots R$ **do**

    $\mathcal{Z}_T, \mathcal{Z}_V \leftarrow$ random partition of $\mathcal{Z}$

    $\boldsymbol{\beta}^{(r)} \leftarrow$ ISGL(Z_T, Z_V)

    $ccr_V^{(r)} \leftarrow$ Correct classification rate of $\boldsymbol{\beta}^{(r)}$ in $\mathcal{Z}_V$

  end

  return $\boldsymbol{\beta}$, ccr_V

3.2.5 Selection of the best model

Our objective is to select one of those R models computed in Algorithm 4 to be our final model. We believe that a selection only based on the maximization of ccr_V could lead to overfit in the training sample data $\mathcal{Z}$. To overcome this problem, we define two indexes: the *importance index* of a variable, and the *power* of a model. These indexes are fundamental to choosing a final model that is not overfitting the data.

We consider the *importance index* I_j of variable X_j defined as,

$$I_j = \sum_{r=1}^{R} |\beta_j^{(r)}| \cdot (ccr_V^{(r)} - \delta) / \max_j \left\{ \sum_{r=1}^{R} |\beta_j^{(r)}| \cdot (ccr_V^{(r)} - \delta) \right\}, \quad (3.9)$$

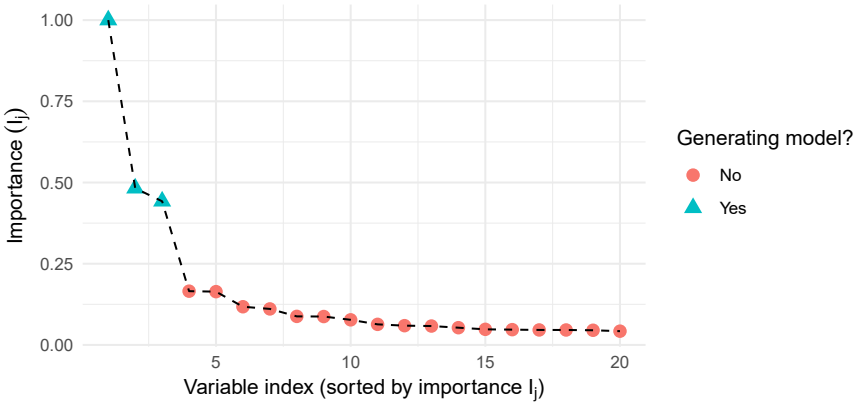
where $\boldsymbol{\beta}^{(r)}$ and $ccr_V^{(r)}$ are those returned by Algorithm 4 on the data $\mathcal{Z}$. With the objective of penalizing those models that had a bad performance on the validation set, the term δ has been introduced, which is the maximum between $\bar{\boldsymbol{y}}$ and $1 - \bar{\boldsymbol{y}}$, i.e., the null model correct classification rate.

The *importance index* weights differently each variable $X_1 \dots X_p$ depending on the correct classification rate of those models in which each variable was present. The larger I_j, the greater the chances of X_j being present in the underlying model that generated the data $\mathcal{Z}$.

Figure 3.2 illustrates the *importance index*, computed on a simulated data set, with $N = 100$ observations and $p = 400$ variables. Notice that the highest three variables in importance are actually in the generating model, and there is a clear gap in Fig. 3.2 between them and the rest of the variables.

Based on the maximization of the *importance index*, an appropriate subset is selected from the original p variables. Although the true number of variables involved in the model is unknown, we can focus our attention on a predefined number of important variables K, which depends only on the sample data $\mathcal{Z}$. We empirically found $K = \lceil \sqrt{N/2} \rceil$ to achieve good results. Using the important index

Figure 3.2: Sorted *importance index* obtained from Algorithm 4, with $R = 150$, and a simulated data sample with $N = 100$ observations and $p = 400$ variables.



of the best K variables, we define the *power* of a model as,

$$P_r = \frac{1}{\sum_{k=1}^{K} I_{(k)}} \sum_{j:I_j \leq I_{(K)}} I_j |\beta_j^{(r)}| / \|\boldsymbol{\beta}^{(r)}\|_1, \quad r = 1, 2 \ldots R, \quad (3.10)$$

where $I_{(k)}$ denotes the $k-$th greatest *importance index*, e.g., $I_{(1)} = \max_j I_j$. The *power index* P weights each model, depending on the *importance* of its included variables.

The selection of the final model is based on the criterion,

$$\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}^{(r^*)}, \quad \text{where} \quad r^* = \max_r \left\{ P_r + ccr_V^{(r)} \right\}. \quad (3.11)$$

Equation (3.11), Algorithm 4, and the framework that supports them, is the main contribution of this paper from a methodological point of view. Equation (3.11) is based on the correct classification rates of R different fitted models, two indexes defined in this paper, and the iterative sparse-group lasso, which is a novel algorithm.

3.3 A simulation study

In this section, we illustrate the performance of Algorithm 4 using synthetic data. To generate observations, we have followed simulation

designs from Simon et al., 2013 (uncorrelated features), Tibshirani, 1996, Zou and Hastie, 2005 and Azevedo Costa et al., 2017 (correlated features). Since our objective was to evaluate Algorithm 4 in binary classification problems, we used a logistic regression model for the response term using the simulated design matrices in each case. We simulated data from the true model,

$$\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta},$$

with logistic response $\boldsymbol{y}$ given by

$$y_i \sim Ber(p_i), \quad p_i = (1 + \exp(-\eta_i))^{-1}, \quad i = 1, 2 \dots N. \quad (3.12)$$

Five scenarios for $\boldsymbol{\beta}$ and $\boldsymbol{X}$ were simulated. In each example, our simulated data consisted of a training set of $N = 100$ observations and p variables, and an independent test set of 5000 observations and p variables. Models were fitted using training data only. Here are the details of the five scenarios.

SFHT_1) This example is adapted from the sparse-group lasso paper (Simon et al., 2013). We set

$$\boldsymbol{\beta} = (1, 2, 3, 4, 5, \underbrace{0, \dots, 0}_{p-5})$$

and X_i are i.i.d $N(0, 1)$, for $1 \leq i \leq p$.

SFHT_2) In this example, $\boldsymbol{\beta}$ is generated as in SFHT_1, but the rows of the model matrix $\boldsymbol{X}$ are i.i.d. generated from a multivariate gaussian distribution with $\text{cov}(X_i, X_j) = 0.5^{|i-j|}, 1 \leq j \leq i \leq p$.

Tibs_1) This example is adapted from the original lasso paper (Tibshirani, 1996), also found in other simulation studies (Zou and Hastie, 2005; Azevedo Costa et al., 2017). We set

$$\boldsymbol{\beta} = (3, 1.5, 0, 0, 2, \underbrace{0, \dots, 0}_{p-5}),$$

and the rows of $\boldsymbol{X}$ are i.i.d. generated from a multivariate gaussian distribution with $\text{cov}(X_i, X_j) = 0.5^{|i-j|}, 1 \leq j \leq i \leq p$.

Tibs_4) This example is also adapted from the original lasso paper (Tibshirani, 1996), and found in other simulation studies as well (Zou and Hastie, 2005; Azevedo Costa et al., 2017). We set

$$\boldsymbol{\beta} = (\underbrace{0,\ldots,0}_{10},\underbrace{2,\ldots,2}_{10},\underbrace{0,\ldots,0}_{10},\underbrace{2,\ldots,2}_{10},\underbrace{0,\ldots,0}_{p-40})$$

and the rows of $\boldsymbol{X}$ are i.i.d. generated from a multivariate gaussian distribution with $\text{cov}(X_i, X_j) = 0.5$, and $\text{var}(X_i) = 1,\ 1 \leq j < i \leq p$.

ZH_d) This example is adapted from the elastic net paper (Zou and Hastie, 2005). We chose

$$\boldsymbol{\beta} = (\underbrace{3,\ldots,3}_{15},\underbrace{0,\ldots,0}_{p-15})$$

and the rows of $\boldsymbol{X}$ were generated as follows,

$$X_i = Z_1 + \epsilon_i^x, \quad Z_1 \sim N(0,1), \quad i = 1,\ldots,5,$$
$$X_i = Z_2 + \epsilon_i^x, \quad Z_2 \sim N(0,1), \quad i = 6,\ldots,10,$$
$$X_i = Z_3 + \epsilon_i^x, \quad Z_3 \sim N(0,1), \quad i = 11,\ldots,15,$$
$$X_i \sim N(0,1),\ X_i \text{ i.i.d. for } i = 16,\ldots,p,$$

where ϵ_i^x are i.i.d. $N(0, 0.01)$, for $1 \leq i \leq 15$.

We aimed to investigate the robustness of our methodology in each example, regarding several measures, as the number of noisy variables (not in the generating model) increased. The criteria we used to evaluate the models in each case were the correct classifications rate in the test sample (ccr), the correct classifications rate in the training sample $\mathbb{E}_t(ccr)$, and the specificity ($spec.$) and sensitivity ($sens.$) concerning variable selection. Let $\hat{\boldsymbol{\beta}}$ be the final estimated coefficient vector and $\boldsymbol{\beta}$ the true generating coefficient vector, then the sensitivity was measured as

$$sens. = \sum_{j=1}^{p} \boldsymbol{I}(\hat{\beta}_j \neq 0) \cdot \boldsymbol{I}(\beta_j \neq 0) / \sum_{j=1}^{p} \boldsymbol{I}(\hat{\beta}_j \neq 0)$$

Analogously, the specificity was defined as

$$spec. = \sum_{j=1}^{p} \boldsymbol{I}(\hat{\beta}_j = 0) \cdot \boldsymbol{I}(\beta_j = 0) / \sum_{j=1}^{p} \boldsymbol{I}(\hat{\beta}_j = 0).$$

Table 3.1 describes the performance of the final model selected under our methodology in the scenarios described above. We have conducted 30 experiments in each case, as we varied the number of variables in the model (p). Standard deviations are given in parenthesis. Table 3.1 reveals that for all the configurations (except, perhaps SFHT_1) the methodology is very robust with respect to an increase in the number of variables p. In fact, for most of them, the ccr does not vary much from $p = 400$ to $p = 4000$. Intuitively, the grouping strategy introduced in Section 3.2.3 places highly correlated variables in the same groups, producing better results when there is correlation between the variables in the model. That is why the simulation scheme SFHT_1 produces the poorest results. In SFHT_1, all the simulated variables are independent and therefore, there is not any clear way to group the variables.

Table 3.1: Average correct classification rate (ccr) of the final model in the test data set (5000 observations), in 30 experiments for each configuration. $E_t(ccr)$ denotes the estimated correct classification rate from the training sample. The mean sensitivity ($sens.$) and the specificity ($spec.$) with respect to variable selection are also given. Standard deviations are given in parenthesis. Algorithm 4 was run with $R = 200$, and $N = 100$ observations in the training sample.

Case		Number of variables in the model (p)				
		40	100	400	1000	4000
SFHT_1	ccr	**0.84** (0.03)	**0.80** (0.04)	**0.76** (0.04)	**0.73** (0.05)	**0.66** (0.06)
	$E_t(ccr)$	0.90 (0.04)	0.87 (0.05)	0.86 (0.05)	0.81 (0.04)	0.80 (0.05)
	$sens.$	0.83 (0.16)	0.71 (0.24)	0.65 (0.18)	0.59 (0.18)	0.47 (0.17)
	$spec.$	0.66 (0.14)	0.83 (0.09)	0.93 (0.04)	0.96 (0.03)	0.98 (0.02)
SFHT_2	ccr	**0.87** (0.02)	**0.86** (0.02)	**0.84** (0.04)	**0.83** (0.04)	**0.82** (0.04)
	$E_t(ccr)$	0.93 (0.04)	0.94 (0.04)	0.92 (0.04)	0.91 (0.04)	0.90 (0.05)
	$sens.$	0.83 (0.18)	0.78 (0.16)	0.74 (0.16)	0.71 (0.16)	0.61 (0.19)
	$spec.$	0.68 (0.17)	0.80 (0.10)	0.92 (0.04)	0.96 (0.03)	0.99 (0.01)
Tibs_1	ccr	**0.82** (0.02)	**0.81** (0.04)	**0.79** (0.04)	**0.77** (0.04)	**0.76** (0.04)
	$E_t(ccr)$	0.90 (0.03)	0.90 (0.04)	0.88 (0.05)	0.87 (0.05)	0.85 (0.05)
	$sens.$	0.99 (0.06)	0.98 (0.08)	0.92 (0.14)	0.90 (0.18)	0.81 (0.19)
	$spec.$	0.68 (0.14)	0.82 (0.08)	0.92 (0.04)	0.96 (0.02)	0.99 (0.01)
Tibs_4	ccr	**0.91** (0.03)	**0.90** (0.02)	**0.89** (0.01)	**0.90** (0.02)	**0.91** (0.01)
	$E_t(ccr)$	0.97 (0.02)	0.96 (0.03)	0.95 (0.03)	0.87 (0.05)	0.97 (0.02)
	$sens.$	0.71 (0.17)	0.43 (0.15)	0.26 (0.11)	0.18 (0.09)	0.17 (0.23)
	$spec.$	0.74 (0.11)	0.77 (0.09)	0.84 (0.04)	0.85 (0.05)	0.83 (0.21)
ZH_d	ccr	**0.91** (0.02)	**0.90** (0.02)	**0.89** (0.03)	**0.88** (0.03)	**0.85** (0.03)
	$E_t(ccr)$	0.98 (0.02)	0.96 (0.02)	0.97 (0.02)	0.97 (0.02)	0.94 (0.03)
	$sens.$	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.01)
	$spec.$	0.67 (0.09)	0.70 (0.08)	0.82 (0.07)	0.84 (0.06)	0.93 (0.03)

3.4 Application to Biomedical Data

In this section, we evaluate the methodology described in Algorithm 4 with the model selection criterion given by (3.11) on a real case study. A sample of TNBC patients from a previously published clinical trial (Sharma, López-Tarruella, García-Saenz, et al., 2016) was used to analyze relations between cancer cells transcriptome and the response of patients to the given medical treatment (docetaxel plus carboplatin). The dataset was composed of 93 observations (patients) and 16616 variables (genetic transcripts and clinical variables).

Figure 3.3: Sorted *Importance indexes*, according to the criterion given in (3.9), and after running Algorithm 4 with $R = 200$. The cutoff value was set to $K = \lceil \sqrt{N/2} \rceil = 7$, as described in Section 3.2.5.

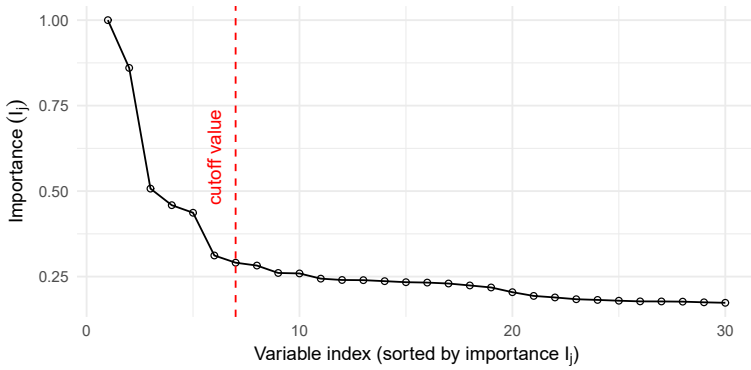


Figure 3.3 shows the highest 30 *importance indexes* out of a total of 16616 variables. The criterion to measure the importance of the variables is given in (3.9). Algorithm 4 was run with $R = 200$, and the cutoff value was set to $K = \lceil \sqrt{N/2} \rceil = 7$, as described in Section 3.2.5. With this importance index, the power of each model was computed using (3.10) and the best model was chosen according to (3.11), as highlighted in Fig. 3.4.

The selected model included 843 out of 16616 variables. The grouping strategy commented in Section 3.2.3 found a total of 82 groups, from which 18 were included in the final model.

Figure 3.5 displays the distribution of the number of non-zero coef-

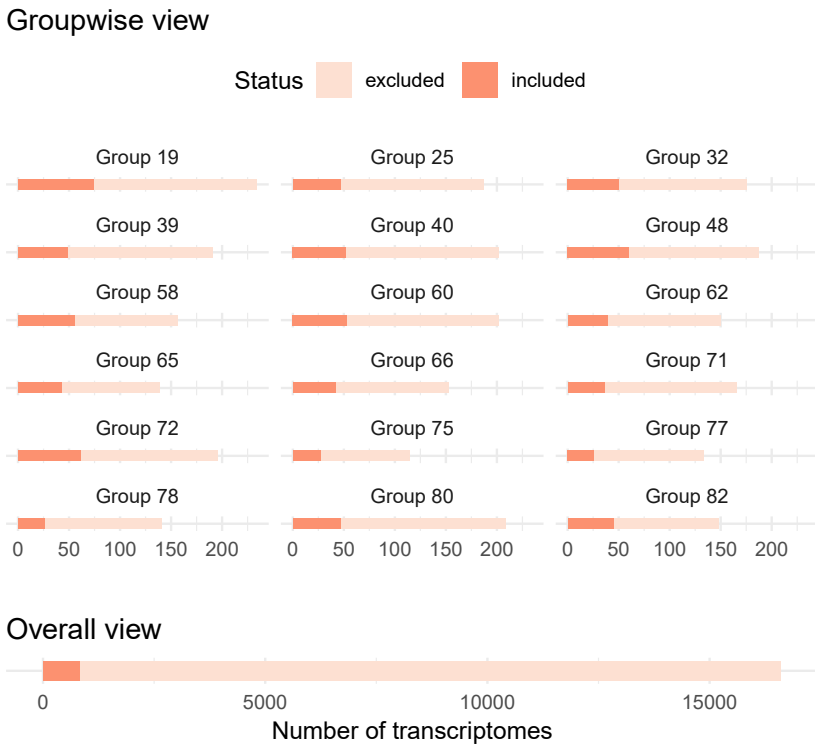
Figure 3.4: Power index (3.10), measured in $R = 200$ models, in decreasing order, with the corresponding correct classification rate (ccr) of each model in the validation sample.

ficients for each group that was included in the final model, which is revealing in several ways. Firstly, it indicates that PCA finds groups of similar lengths, and secondly, the selected model is sparse at both the group and the variable levels.

In an attempt to discover the biological and genetic meaning in the model selected by our methodology, we ran DAVID (Huang, Sherman, and Lempicki, 2008b; Huang, Sherman, and Lempicki, 2008a) to detect enriched functional-related gene groups. The clustering and functional annotation was performed using the default analysis options, and the role of the potential multiple testing effect was considered using the false discovery rate (FDR).

We observed just two remarkable families of pathways after the gene enrichment analysis: the homeobox-related and the oxidative phosphorylation pathways. They are both involved in the mechanism of action of docetaxel and carboplatin in response to the provided treatment.

Figure 3.5: Number of included variables in the final model, by groups (top) and total (bottom). There were included 18 out of 82 groups.



The homeobox genes have been proposed to be involved in mechanisms of resistance to taxane-based oncologic treatments in ovarian and prostate cancer (J. Li et al., 2014; Hanrahan et al., 2017; Marín-Aguilera et al., 2014; Puhr et al., 2012). Docetaxel hyper-stabilizes the microtube structure, irreversibly blocking the cytoskeleton function in the mitotic process and intracellular transport. In addition, this drug induces programmed cell death (Wishart et al., 2017).

On the other hand, carboplatin attaches alkyl groups to DNA bases resulting in fragmentation by repair enzymes when trying to repair it. It also inducts to mutations due to nucleotide despairing and generates DNA cross-links that affects the transcription process (Wishart et al., 2017). The development of resistance to platinum-based schemes of chemotherapy is a common feature. Several studies demonstrate

that dysfunctions in mitochondrial processes, in conjunction with the mentioned mechanism of action, can contribute to develop the phenotypes associated with resistance (Matassa et al., 2016; Dai et al., 2010; Chappell et al., 2012; Marrache, Pathak, and Dhar, 2014; Belotte et al., 2014; McAdam, Brem, and Karran, 2016).

3.5 Conclusions

The present study introduces a methodology to deal with the variable selection problem in the high dimensional set-up. It can be seen as an extension of the sparse-group lasso regularization method, without the dependencies on both the hyper-parameters and the groups. There are several critical components in this approach,

- A clustering on the variables, based on PCA, makes it possible to work with an arbitrarily large number of variables, without specifying groups apriori.
- The iterative sparse group lasso removes the dependence on the hyper-parameters of the sparse group lasso, but it is sensible to the train/validate sample partitions. This problem has been solved running the algorithm for a large number of different train/validate sample partitions (Algorithm 4).
- The correct classification rate of each model in its respective validation sample is stored. Notice that this is an overestimation of the true correct classification rate on future observations, and the highest validation rate does not imply the best model.
- The *importance index* weights the variables, based on the correct classification rate of the models that include them.
- The *power index* weights the models, based on the *importance* of the variables they include.

This methodology was tested on a sample of TNBC patients, trying to reveal the genetic profile associated with resistance to the treatment of interest. The literature studies mentioned in Section 3.4 provide a rationale supporting the potential predictive value of the two gene pathways identified in our study (the homeobox-related and the oxidative phosphorylation pathways). In order to validate these results,

we are testing the model in a new cohort of TNBC patients from the same clinical trial.

Future studies should examine other strategies to group the variables, as discussed in Section 3.2.3, based on supervised algorithms as well as unsupervised ones.

Acknowledgements

Simulations in Sections 3.3 and 3.4 have been carried out in Uranus, a supercomputer cluster located at Universidad Carlos III de Madrid and funded jointly by EU-FEDER funds and by the Spanish Government via the National Projects No. UNC313-4E-2361, No. ENE2009-12213- C03-03, No. ENE2012-33219 and No. ENE2015-68265-P.

Bibliography for Chapter 3

Azevedo Costa, Marcelo et al. (2017). "Sequential selection of variables using short permutation procedures and multiple adjustments: An application to genomic data". In: *Statistical methods in medical research* 26.2, pp. 997–1020.

Belotte, Jimmy et al. (2014). "The role of oxidative stress in the development of cisplatin resistance in epithelial ovarian cancer". In: *Reproductive sciences* 21.4, pp. 503–508.

Chang, Jenny C et al. (2003). "Gene expression profiling for the prediction of therapeutic response to docetaxel in patients with breast cancer". In: *The Lancet* 362.9381, pp. 362–369.

Chappell, Nicole P et al. (2012). "Mitochondrial proteomic analysis of cisplatin resistance in ovarian cancer". In: *Journal of proteome research* 11.9, pp. 4605–4614.

Cortazar, Patricia et al. (2014). "Pathological complete response and long-term clinical benefit in breast cancer: the CTNeoBC pooled analysis". In: *The Lancet* 384.9938, pp. 164–172.

Dai, Zhiqin et al. (2010). "Mitochondrial comparative proteomics of human ovarian cancer cells and their platinum-resistant sublines". In: *Proteomics* 10.21, pp. 3789–3799.

De Los Campos, Gustavo, Daniel Gianola, and David B Allison (2010). "Predicting genetic predisposition in humans: the promise of whole-genome markers". In: *Nature Reviews Genetics* 11.12, p. 880.

Dent, Rebecca et al. (2007). "Triple-negative breast cancer: clinical features and patterns of recurrence". In: *Clinical cancer research* 13.15, pp. 4429–4434.

Ferlay, J et al. (2014). *GLOBOCAN 2012 v1. 0, Cancer Incidence and Mortality Worldwide: IARC CancerBase No. 11. Lyon, France: International Agency for Research on Cancer; 2013*.

Hanrahan, Karen et al. (2017). "The role of epithelial–mesenchymal transition drivers ZEB1 and ZEB2 in mediating docetaxel-resistant prostate cancer". In: *Molecular oncology* 11.3, pp. 251–265.

Hatzis, Christos et al. (2011). "A genomic predictor of response and survival following taxane-anthracycline chemotherapy for invasive breast cancer". In: *Jama* 305.18, pp. 1873–1881.

Huang, Da Wei, Brad T Sherman, and Richard A Lempicki (2008a). "Bioinformatics enrichment tools: paths toward the comprehensive

functional analysis of large gene lists". In: *Nucleic acids research* 37.1, pp. 1–13.
– (2008b). "Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources". In: *Nature protocols* 4.1, p. 44.
Laria, Juan C, M Carmen Aguilera-Morillo, and Rosa E Lillo (2019). "An iterative sparse-group lasso". In: *Journal of Computational and Graphical Statistics*, pp. 1–10.
Li, Jianchao et al. (2014). "Downregulation of HNF1 homeobox B is associated with drug resistance in ovarian cancer". In: *Oncology reports* 32.3, pp. 979–988.
Lupski, James R et al. (2011). "Clan genomics and the complex architecture of human disease". In: *Cell* 147.1, pp. 32–43.
Marín-Aguilera, Mercedes et al. (2014). "Epithelial-to-mesenchymal transition mediates docetaxel resistance and high risk of relapse in prostate cancer". In: *Molecular cancer therapeutics*.
Marrache, Sean, Rakesh K Pathak, and Shanta Dhar (2014). "Detouring of cisplatin to access mitochondrial genome for overcoming resistance". In: *Proceedings of the National Academy of Sciences*, p. 201405244.
Matassa, DS et al. (2016). "Oxidative metabolism drives inflammation-induced platinum resistance in human ovarian cancer". In: *Cell death and differentiation* 23.9, p. 1542.
McAdam, Elizabeth, Reto Brem, and Peter Karran (2016). "Oxidative Stress–Induced Protein Damage Inhibits DNA Repair and Determines Mutation Risk and Therapeutic Efficacy". In: *Molecular Cancer Research*.
Offit, Kenneth (2011). "Personalized medicine: new genomics, old lessons". In: *Human genetics* 130.1, pp. 3–14.
Puhr, Martin et al. (2012). "Epithelial-to-mesenchymal transition leads to docetaxel resistance in prostate cancer and is mediated by reduced expression of miR-200c and miR-205". In: *The American journal of pathology* 181.6, pp. 2188–2201.
Sharma, Priyanka, Sara López-Tarruella, Jose A Garcia-Saenz, et al. (2018). "Pathological response and survival in triple-negative breast cancer following neoadjuvant carboplatin plus docetaxel". In: *Clinical Cancer Research*, clincanres–0585.

Sharma, Priyanka, Sara López-Tarruella, Jose Angel García-Saenz, et al. (2016). "Efficacy of neoadjuvant carboplatin plus docetaxel in triple negative breast cancer: Combined analysis of two cohorts". In: *Clinical Cancer Research*, clincanres–0162.

Simon, Noah et al. (2013). "A sparse-group lasso". In: *Journal of computational and graphical statistics* 22.2, pp. 231–245.

Symmans, W Fraser et al. (2017). "Long-Term Prognostic Risk After Neoadjuvant Chemotherapy Associated With Residual Cancer Burden and Breast Cancer Subtype." In: *Journal of clinical oncology: official journal of the American Society of Clinical Oncology* 35.10, pp. 1049–1060.

Szymczak, Silke et al. (2009). "Machine learning in genome-wide association studies". In: *Genetic epidemiology* 33.S1.

Tabchy, Adel et al. (2010). "Evaluation of a 30-gene paclitaxel, fluorouracil, doxorubicin and cyclophosphamide chemotherapy response predictor in a multicenter randomized trial in breast cancer". In: *Clinical Cancer Research*, clincanres–1265.

Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1, pp. 267–288.

Wishart, David S et al. (2017). "DrugBank 5.0: a major update to the DrugBank database for 2018". In: *Nucleic acids research* 46.D1, pp. D1074–D1082.

Yuan, Ming and Yi Lin (2006). "Model selection and estimation in regression with grouped variables". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.1, pp. 49–67.

Zou, Hui and Trevor Hastie (2003). "Regression shrinkage and selection via the elastic net, with applications to microarrays". In: *Journal of the Royal Statistical Society: Series B. v67*, pp. 301–320.

– (2005). "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2, pp. 301–320.

CHAPTER 4

A variable selection and clustering method for generalized linear models

Juan C. Laria[a,b] M. Carmen Aguilera-Morillo[a,b] Rosa E. Lillo[a,b]

[a] Department of Statistics, University Carlos III of Madrid, Spain
[b] UC3M-BS Institute of Financial Big Data, Madrid, Spain

Abstract

This paper introduces the Group Linear Algorithm with Sparse Principal decomposition, an algorithm for supervised variable selection and clustering. Our approach extends the Sparse-Group Lasso regularization to calculate clusters as part of the model fit. Therefore, unlike Sparse-Group Lasso, our idea does not require prior specification of clusters between variables. To determine the clusters, we solve a particular case of sparse Singular Value Decomposition, with a regularization term that follows naturally from the Group Lasso penalty. Moreover, this paper proposes a unified implementation to deal with, but not limited to, linear regression, logistic regression, and proportional hazards models with right-censoring. Our methodology is evaluated using synthetic data, and details of the implementation in R and hyperparameter search are discussed.

Keywords: Regression, Classification and Clustering, Statistical Computing

4.1 Introduction

In recent years, penalized regression problems for variable selection have become very popular. Since the introduction of Lasso (Robert Tibshirani, 1996) as a regularization term for linear models, many extensions have dealt with variable selection by penalizing the loss function. Most of these extensions are limited to linear regression, but some of them, such as Elastic-Net (Zou and Hastie, 2005), Group Lasso (Zhou and Zhu, 2010) or recently Sparse Group Lasso (Simon et al., 2013) have been also extended to generalized linear models (GLMs).

In this paper, we focus on the Sparse Group Lasso as a variable selection method in high-dimensional problems. The hypothesis of the existence of previously known clusters among the variables poses a significant practical difficulty for this method to be applied to every supervised problem. Besides, the Sparse Group Lasso penalty function is the linear combination of a Lasso penalty (ℓ_1 norm) and a Group Lasso penalty (ℓ_2 norm), so there are at least two regularization hyperparameters (plus one for each group, usually fixed). In most of the applications of the Sparse Group Lasso, the parameters are either fixed based on a prior information about the data, or chosen to minimize some error function in a grid of possible values. In that sense, Laria, Carmen Aguilera-Morillo, and Lillo, 2019 proposed a gradient-free coordinate descent algorithm, which allows the automatic selection of the regularization parameters in the SGL. However, the problem of grouping the variables was not solved. In genetic or financial applications, there is a growing demand not only for building predictive models but also for clustering the variables.

The main methodological contribution of this article is the formal definition of GLASP, a Group Linear Algorithm with Sparse Principal decomposition. GLASP is an extension of the Sparse Group Lasso, that, not only avoids the need for a specification of clusters among the variables, but also computes such clusters during the model fitting process. Therefore, apart from a predictive model, GLASP can be considered as a supervised variable clustering algorithm.

The GLASP specification is motivated by the Cluster Elastic Net (CEN)

(Witten, Shojaie, and Zhang, 2014), where the authors extend the elastic-net to obtain groups between variables using k-means, besides variable selection and model fitting. Recently, some extensions have considered multivariate response CEN, for example, Price and Sherwood, 2017 and Ren, Kang, and Lu, 2020. A minor disadvantage of CEN is that the number of clusters between variables has to be specified initially. Unlike CEN, our GLASP algorithm can obtain a smaller number of groups than initially specified.

From an algorithmic point of view, GLASP has two parts. The first is an accelerated block gradient descent algorithm to adjust the Sparse Group Lasso with an arbitrary and flexible error function, which is a linear combination of the model loss function and a differentiable regularization term. The second is a particular type of regularized Singular Value Decomposition, with a penalty function adapted to this specific problem, in order to find the groups.

The method proposed in this paper is, to the best of our knowledge, the first extension of the Sparse Group Lasso that computes groups automatically. The internal supervised variable clustering algorithm is also an original contribution and integrates naturally within the Group Lasso penalty. Moreover, our implementation provides the flexibility to change the risk function and address any regression problem.

This paper is organized as follows. Section 4.2 introduces GLASP as the solution to a problem involving sparsity, clustering, and structure assumptions on the variables. Section 4.3 describes in detail the solutions of both sub-problems addressed, with particular emphasis on the internal optimization algorithms. Later, Section 4.4 compares our approach with other linear regression methods that perform variable selection and clustering. Although a general notation is adopted from the beginning to refer to the loss function, the main differences when a linear, logistic or Cox survival model with right-censoring is adjusted with GLASP are explained in Section 4.5. For the latter, additional details related to prediction are presented, as well as a simulation study on survival data. Moreover, relevant details and practical examples related to the implementation of GLASP in R language are illustrated in Section 4.6, with special emphasis on its tidy interface, and the optimization of hyper-parameters. Section 4.7 illustrates an

application of GLASP to gene clustering and survival prediction with right-censored data.

Finally, Section 4.8 discusses the implications of our work and future directions of research.

4.2 Formulation of GLASP

Under the penalized general linear regression framework, we have a data matrix $\boldsymbol{X} \in \mathbb{R}^{N\times p}$, a response vector $\boldsymbol{y} \in \mathbb{R}^{N\times 1}$, and we are interested in finding $\boldsymbol{\beta} \in \mathbb{R}^{p\times 1}$ such that $L(\boldsymbol{\beta}) + \phi(\boldsymbol{\beta})$ is minimum. Here $\phi : \mathbb{R}^{p\times 1} \to \mathbb{R}^{+}$ is some penalty, and $L : \mathbb{R}^{p\times 1} \to \mathbb{R}$ is an empirical risk function that measures how good can we approximate $\boldsymbol{y}$ knowing $\boldsymbol{X}\boldsymbol{\beta}$. Throughout this paper, and without loss of generality, we will assume that the data matrix $\boldsymbol{X}$ is standardized to have mean 0 and variance 1 in each column (i.e. $\bar{\boldsymbol{X}}_j = 0$ and $\boldsymbol{X}_j^{\top}\boldsymbol{X}_j = 1$ for every $j = 1, 2 \dots p$). This is important for the computations in next sections. We will make the following extra assumptions:

1. (Sparsity) There is a small number of columns of $\boldsymbol{X}$ that are actually related to $\boldsymbol{y}$, and therefore many components of $\boldsymbol{\beta}$ are exactly zero.

2. (Clustering) There is a (possible unknown) number K of unknown groups, or clusters, among the variables of $\boldsymbol{X}$.

3. (Structure) For every group, there is associated a latent variable that summarizes the information provided by all the variables in that cluster. In linear models, information is measured in terms of linear predictors. A variable $\boldsymbol{X}_j$ provides information to the model through $\boldsymbol{X}_j\boldsymbol{\beta}_j$. Knowing those groups will improve the estimation of $\boldsymbol{\beta}$, and knowing $\boldsymbol{\beta}$ will give us insight into the groups.

These assumptions are aligned with those of Witten, Shojaie, and Zhang, 2014. However, we want to remark that often, the number K is unknown. In addition, we do not want to assume beforehand that $\boldsymbol{X}_j\beta_j$ and $\boldsymbol{X}_l\beta_l$ are close in the squared euclidean distance, for $\boldsymbol{X}_j$ and $\boldsymbol{X}_l$ in the same group.

Solving the sparse regression problem and, at the same time, finding

the clusters in the columns of $\boldsymbol{X}$, motivates the GLASP optimization problem,

$$\min_{\boldsymbol{\beta},\boldsymbol{W},\boldsymbol{T}} \left\{ L(\boldsymbol{\beta}) + \lambda_1 \left\|\boldsymbol{\beta}\right\|_1 + \lambda_2 \sum_{k=1}^{K} \left\|\boldsymbol{J}_k\boldsymbol{\beta}\right\|_2 + \frac{\lambda_3}{2} \left\| \boldsymbol{X} \sum_{j=1}^{p} (e_j e_j^\top)\beta_j - \boldsymbol{T}\boldsymbol{W}^\top \right\|_F^2 \right\}, \tag{4.1}$$

where

- $\|\cdot\|_F^2$ is the squared Frobenius norm, given by $\|\boldsymbol{M}\|_F^2 = Tr(\boldsymbol{M}\boldsymbol{M}^\top)$.
- $\boldsymbol{W} \in \mathbb{R}^{p\times K}$ is an orthogonal matrix with cluster information, $\boldsymbol{W}^\top\boldsymbol{W}$ diagonal.
- $\boldsymbol{T} \in \mathbb{R}^{N\times K}$ (latent groups) is a low-rank unitary representation of the linear predictors, $\boldsymbol{T}^\top\boldsymbol{T} = \boldsymbol{I}_K$.
- e_j is the j–th vector in the canonical basis of $\mathbb{R}^{p\times 1}$.
- $\boldsymbol{X}\sum_{j=1}^{p}(e_j e_j^\top)\beta_j \in \mathbb{R}^{N\times p}$ is the matrix of linear predictors.
- $\boldsymbol{J}_k = \left\|\boldsymbol{W}_k\right\|_0 \sum_{j=1}^{p}(e_j e_j^\top)\mathbb{1}(W_{jk} \neq 0)$ is a diagonal projection matrix such that $\left\|\boldsymbol{J}_k\boldsymbol{\beta}\right\|_2$ is the euclidean norm of the vector of coefficients associated with group k, penalized by the size of the group. Here $\left\|\boldsymbol{W}_k\right\|_0$ denotes the number of elements in column k–th of $\boldsymbol{W}$ that are non-zero, which is the size of group k.
- $\lambda_1, \lambda_2, \lambda_3$ are regularization hyperparameters.

Problem (4.1) is a non-convex optimization problem, and finding the global optimum would require to search for orthogonal matrices $\boldsymbol{T}$, rotation matrices $\boldsymbol{W}$ and coefficient vectors $\boldsymbol{\beta}$ that minimize (4.1). This is impractical, and we propose a two-step iterative approach to find a local minimum of (4.1). See, for example Witten, Shojaie, and

Zhang, 2014. This problem can be separated in two optimization subproblems.

$$\min_{\boldsymbol{\beta}} \left\{ L(\boldsymbol{\beta}) + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{k=1}^{K} \|\boldsymbol{J}_k \boldsymbol{\beta}\|_2 + \frac{\lambda_3}{2} \left\| \boldsymbol{X} \sum_{j=1}^{p} (e_j e_j^\top) \beta_j - \boldsymbol{T}\boldsymbol{W}^\top \right\|_F^2 \right\}, \tag{4.2}$$

and

$$\min_{\boldsymbol{W},\boldsymbol{T}} \left\{ \frac{\lambda_3}{2} \left\| \boldsymbol{X} \sum_{j=1}^{p} (e_j e_j^\top) \beta_j - \boldsymbol{T}\boldsymbol{W}^\top \right\|_F^2 + \lambda_2 \sum_{k=1}^{K} \|\boldsymbol{J}_k \boldsymbol{\beta}\|_2 \right\}, \tag{4.3}$$

Remark 1. *If $\lambda_3 = 0$, problem* (4.2) *is the Sparse Group Lasso.*

It is easy to see that, when $\lambda_3 = 0$, (4.2) is equivalent to the formulation of the Sparse Group Lasso (Simon et al., 2013), given by,

$$\min_{\boldsymbol{\beta}} \left\{ L(\boldsymbol{\beta}) + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{k=1}^{K} \sqrt{p_k} \left\|\boldsymbol{\beta}^{(k)}\right\|_2 \right\},$$

with $\boldsymbol{\beta}^{(k)}$ and p_k the coefficients and size of group k, respectively.

Remark 2. *In general, for $\boldsymbol{T}, \boldsymbol{W}$ fixed, the penalization $\varphi(\boldsymbol{\beta}) = \lambda_3/2 \left\| \boldsymbol{X} \sum_{j=1}^{p} (e_j e_j^\top) \beta_j - \boldsymbol{T}\boldsymbol{W}^\top \right\|_F^2$ does not shrink $\boldsymbol{\beta}$ towards zero, and therefore, the regularization function in* (4.2) *may not shrink to zero, but to some other vector.*

After some algebra, $\varphi(\boldsymbol{\beta})$ can be written in the form

$$\varphi(\boldsymbol{\beta}) = (\beta_1 - c_1)^2/a_1 + (\beta_2 - c_2)^2/a_2 + \cdots + (\beta_p - c_p)^2/a_p - k^2,$$

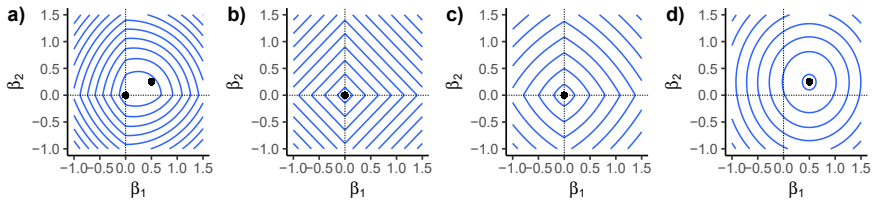
where a_j, c_j, k are values depending on $\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{T}$. The contour levels of $\varphi(\boldsymbol{\beta})$ correspond to ellipsoids in $\mathbb{R}^p$, centered at $(c_1, \ldots c_p)$. To see this, we will plot the penalty as a function of $\boldsymbol{\beta}$. As a toy example, consider a data matrix $\boldsymbol{X} \in \mathbb{R}^{100\times 3}$, with $N(0,1)$ columns, such that $cov(X_1, X_2) = 0$, $cov(X_2, X_3) = 0$ and $cov(X_1, X_3) = 0.5$. Let $\tilde{\boldsymbol{c}} = (0.5, 0.25, 0.1)^\top$, and $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ the singular value decomposition of $\boldsymbol{X}\tilde{\boldsymbol{c}}$. We choose $\boldsymbol{T} = \boldsymbol{U}$ and $\boldsymbol{W}$ such that $\boldsymbol{W}^\top\boldsymbol{W}$ diagonal, but

close to $\boldsymbol{V\Sigma}$. Then, we have $\boldsymbol{T}^\top\boldsymbol{T} = \boldsymbol{I}$ and $\boldsymbol{W}$ is approximately given by,

$$\boldsymbol{W} = \begin{pmatrix} 4.97 & 0 \\ 0 & 2.46 \\ 0.37 & 0 \end{pmatrix},$$

such that β_1 and β_2 are in different groups. The contour plot for the GLASP penalty is shown in Figure 5.6a, compared with Lasso (Figure 5.6b), Sparse Group Lasso (Figure 5.6c), and GLASP with $\lambda_1 = \lambda_2 = 0$ (Figure 5.6d).

Figure 4.1: Contour plots for the GLASP(a), the Lasso (b), the Sparse Group Lasso (c) and GLASP with $\lambda_1 = \lambda_2 = 0$ (d).



Remark 3. *For a fixed $\boldsymbol{\beta}$, problem* (4.3) *can be written in the form*

$$\min_{\boldsymbol{W},\boldsymbol{T}} \left\{ \left\| \boldsymbol{M} - \boldsymbol{T}\boldsymbol{W}^\top \right\|_F^2 + \gamma P(\boldsymbol{W}) \right\}. \tag{4.4}$$

This is a penalized low-rank approximation problem, and in this case, P is a sparsity penalty.

This problem written in the general form (4.4) is very similar to those investigated by Shen and Huang, 2008. The first part is a low rank approximation problem, which is known to be solved by the singular value decomposition. In our case, the challenging part is the function P, which is non-differentiable and non-convex.

$$P(\boldsymbol{W}) = \sum_{k=1}^{K} \left(\sum_{j=1}^{p} \beta_j^2 \mathbb{1}(\boldsymbol{W}_{jk} \neq 0) \left\| \boldsymbol{W}_k \right\|_0 \right)^{1/2}. \tag{4.5}$$

However, it is clear that P is a sparsity penalty, and therefore, (4.4) will force sparsity in $\boldsymbol{W}$. We propose a solution based to the *sparse PCA via regularized SVD* of Shen and Huang, 2008, but considering our function P as penalty for $\boldsymbol{W}$, instead of common choices.

4.3 Algorithms

In this section, we detail the computations to solve the GLASP problem, separated into two sub-problems, defined in (4.2) and (4.3), respectively.

4.3.1 Internal optimization by groups

Consider (4.2) for $\boldsymbol{W}, \boldsymbol{T}$ fixed. The final algorithm is a block gradient descent method. We found this solution to be very fast to solve convex optimization problems in a general context, where there is a differentiable loss and a sub-differentiable penalty. Recent papers dealing with the Sparse Group Lasso and extensions have also adopted similar approaches (Simon et al., 2013; Ren, Kang, and Lu, 2020; Laria, Carmen Aguilera-Morillo, and Lillo, 2019).

Problem (4.2) can be minimized using a cyclic group-wise gradient descent. Assume that vector $\boldsymbol{\beta}$ is fixed for all groups but k–th, and without loss of generality, assume that the coefficients in group k are $\beta_1, \beta_2 \dots \beta_{p_k}$. To avoid difficult notation, throughout this section $\boldsymbol{\beta} = (\beta_1, \beta_2 \dots \beta_{p_k})^\top$ will denote the coefficient vector for group k. Since the remaining groups are fixed, and using the definition of the Frobenius norm, (4.2) becomes

$$\min_{\boldsymbol{\beta}} \left\{ L(\boldsymbol{\beta}) + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \sqrt{p_k} \|\boldsymbol{\beta}\|_2 + \frac{\lambda_3}{2} \sum_{j=1}^{p_k} \left\| \boldsymbol{X}_j \beta_j - \boldsymbol{T} \boldsymbol{W}_{j\cdot}^\top \right\|_2^2 \right\}. \tag{4.6}$$

To solve (4.6) we will use the *fast iterative shrinkage-thresholding algorithm* (FISTA) (Beck and Teboulle, 2009).

Consider the general optimization problem

$$\min_{\boldsymbol{\beta}} \{F(\boldsymbol{\beta}) := R(\boldsymbol{\beta}) + \Phi(\boldsymbol{\beta})\}, \tag{4.7}$$

where

- $R : \mathbb{R}^{p\times 1} \to \mathbb{R}$ is a smooth convex function, continuously differentiable with Lipschitz continuous gradient ∇R (with Lipschitz constant $\mathcal{L}(R)$), such that

$$\|\nabla R(\boldsymbol{\beta}) - \nabla R(\boldsymbol{\beta}_0)\|_2 \leq \mathcal{L}(R) \|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_2 .$$

- $\Phi : \mathbb{R}^{p\times 1} \to \mathbb{R}$ is a continuous convex function which is possibly non-smooth.

Consider (4.6) in the form (4.7), taking

$$R(\boldsymbol{\beta}) = L(\boldsymbol{\beta}) + \frac{\lambda_3}{2}\sum_{j=1}^{p_k} \left\| \boldsymbol{X}_j\beta_j - \boldsymbol{T}\boldsymbol{W}_{j\cdot}^\top \right\|_2^2,$$

and

$$\phi(\boldsymbol{\beta}) = \lambda_1 \left\|\boldsymbol{\beta}\right\|_1 + \lambda_2\sqrt{p_k}\left\|\boldsymbol{\beta}\right\|_2.$$

The core of the FISTA algorithm is to consider, for any $t > 0$, the quadratic approximation of $F(\boldsymbol{\beta})$ at a given point $\boldsymbol{\beta}_0$, given by,

$$M_t(\boldsymbol{\beta}, \boldsymbol{\beta}_0) = R(\boldsymbol{\beta}_0) + (\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \nabla R(\boldsymbol{\beta}_0) + \frac{1}{2t}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}_0\right\|_2^2 + \Phi(\boldsymbol{\beta}), \tag{4.8}$$

which admits a unique minimizer (that we refer to as *update function*)

$$\begin{aligned} U_t(\boldsymbol{\beta}_0) &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\{M_t(\boldsymbol{\beta}, \boldsymbol{\beta}_0)\} \\ &= \operatorname{argmin}_{\boldsymbol{\beta}}\left\{M_t^*(\boldsymbol{\beta}, \boldsymbol{\beta}_0) = \tfrac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{B_0}\right\|_2^2 + t\Phi(\boldsymbol{\beta})\right\}, \end{aligned} \tag{4.9}$$

where $\boldsymbol{B_0} = \boldsymbol{\beta}_0 - t\nabla R(\boldsymbol{\beta}_0)$. The idea of the *iterative shrinkage-thresholding algorithm* (ISTA) algorithm is to produce a descent sequence for F via $\boldsymbol{\beta}_{(k+1)} \leftarrow U_t(\boldsymbol{\beta}_{(k)})$, choosing t carefully such that $t < 1/\mathcal{L}(R)$. If the Lipschitz constant $\mathcal{L}(R)$ is unknown, t_k is found in each step using a backtracking stepsize rule, to be the maximum $t > 0$ such that,

$$F(U_t(\boldsymbol{\beta}_{(k)})) \leq M_t(U_t(\boldsymbol{\beta}_{(k)}), \boldsymbol{\beta}_{(k)}). \tag{4.10}$$

To accelerate the global rate of convergence from $1/k$ (ISTA) to $1/k^2$, the FISTA algorithm updates $\boldsymbol{\beta}_{(k)}$ according to

$$\boldsymbol{\beta}_{(k+1)} \leftarrow U_{t_k}(\boldsymbol{\beta}_{(k)}) + \frac{l_k - 1}{l_{k+1}}(U_{t_k}(\boldsymbol{\beta}_{(k)}) - U_{t_k-1}(\boldsymbol{\beta}_{(k-1)})), \tag{4.11}$$

where $l_{k+1} = (1 + \sqrt{1 + 4l_k^2})/2$, $l_1 = 1$.

The most difficult part to formulate in our algorithm to solve (4.6) using FISTA, is to minimize M_t^* as a function of $\boldsymbol{\beta}$, which in our case is given by,

$$M_t^*(\boldsymbol{\beta}) = \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{B_0}\right\|_2^2 + t\lambda_1\left\|\boldsymbol{\beta}\right\|_1 + t\lambda_2\sqrt{p_k}\left\|\boldsymbol{\beta}\right\|_2 \tag{4.12}$$

Next proposition provides the update function (4.9) corresponding to M_t^* in (4.12).

Proposition 1. *The update function of problem* (4.6) *is given by,*

$$U_t(\boldsymbol{\beta}_0) = \left(1 - \frac{t\lambda_2\sqrt{p_k}}{\left\|S(\boldsymbol{B_0}, t\lambda_1)\right\|_2}\right)_+ S(\boldsymbol{B_0}, t\lambda_1),$$

where S is the coordinate-wise soft threshold operator,

$$S(\boldsymbol{z}, \lambda)_i = sign(z_i)(|z_i| - \lambda)_+$$

The following proposition provides conditions for $\boldsymbol{\beta} = \mathbf{0}$ to be the minimizer of (4.6). If we know, after a simple computation, that $\boldsymbol{\beta} = \mathbf{0}$, then we can skip the FISTA optimization for the coefficients in that group. Moreover, these conditions are also upper bounds for the maximum values of the hyper-parameters, such that $\boldsymbol{\beta} \neq 0$.

Proposition 2. $\boldsymbol{\beta} = \mathbf{0}$ *is the minimizer of* (4.6) *if*

$$\left\|S(\nabla R(\mathbf{0}), \lambda_1)\right\|_2 \leq \lambda_2\sqrt{p_k}. \tag{4.13}$$

In particular, it is also true if,

$$\max_j |\nabla_j R(\mathbf{0})| \leq \lambda_1. \tag{4.14}$$

The proofs of Propositions 1 and 2 can be found in the Appendix.

4.3.2 Group optimization

This section describes the solution that we propose for sub-problem (4.3). In addition, we will assume that there are no overlapping groups.

As stated in Remark 3, when $\boldsymbol{\beta}$ is fixed, assuming that $\lambda_3 > 0$ and ignoring constant terms, (4.3) can be written as (4.4), where M is the matrix of linear predictors,

$$\boldsymbol{M} = \boldsymbol{X} \sum_{j=1}^{p} (e_j e_j^\top) \beta_j,$$

and P is a sparsity penalty on W, given in (4.5). Furthermore, to assume that there are not overlapping groups (and each variable belongs to exactly one group) can be written as a constraint in $\boldsymbol{W}$, $\|\boldsymbol{W}_{j\cdot}\|_0 = 1$, for every $j = 1, 2 \dots p$. We will deal with this constraint later, but first let's tackle problem (4.4).

Problem (4.4) is a special type of *regularized Singular Value Decomposition*, where the penalty term can be separated into a sum of penalties on the columns of $\boldsymbol{W}$. An efficient way of dealing with this problem, is solving regularized one-rank approximation problems to construct $\boldsymbol{W}$ and $\boldsymbol{T}$ column-wise. An example of such an algorithm is the *sPCA-rSVD* from Shen and Huang, 2008, Algorithm 1. Our approach here is very similar to theirs, except for the penalty term.

Consider the simpler problem,

$$\min_{\boldsymbol{u},\boldsymbol{v}} \left\{ \left\| \boldsymbol{M} - \boldsymbol{u}\boldsymbol{v}^\top \right\|_F^2 + \gamma \left(\sum_{j=1}^{p} \beta_j^2 \mathbb{1}(\boldsymbol{v}_j \neq 0) \left\| \boldsymbol{v} \right\|_0 \right)^{1/2} \right\}. \quad (4.15)$$

Although the regularization in (4.15) is discontinuous, an iterative solution is possible, and it is shown in Preposition 3.

Proposition 3. *The optimal* $\boldsymbol{v}$ *in* (4.15) *is such that, for* $l = 1, 2 \dots p$,

$$\begin{aligned} \boldsymbol{v}_l = (\boldsymbol{M}^\top \boldsymbol{u})_l \mathbb{1} \Bigg(& (\boldsymbol{M}^\top \boldsymbol{u})_l^2 \\ & > \gamma \left(C_{\beta,v}^{(-l)} + \beta_l^2 \right)^{1/2} \left(C_v^{(-l)} + 1 \right)^{1/2} - \gamma \left(C_{\beta,v}^{(-l)} C_v^{(-l)} \right)^{1/2} \Bigg), \end{aligned} \quad (4.16)$$

where

$$C_{\beta,v}^{(-l)} = \sum_{\substack{j=1 \\ j\neq l}}^{p} \beta_j^2 \mathbb{1}(\boldsymbol{v}_j \neq 0), \quad C_v^{(-l)} = \sum_{\substack{j=1 \\ j\neq l}}^{p} \mathbb{1}(\boldsymbol{v}_j \neq 0).$$

The update function for $\boldsymbol{v}$ in Proposition 3 can not be applied in one step, because the expression for each component $\boldsymbol{v}_l$ can not be separated from the whole vector $\boldsymbol{v}$. To tackle this, we propose to iterate through $\boldsymbol{v}$, updating each $\boldsymbol{v}_l$ with (4.16) until convergence. Algorithm 5 describes the iterative optimization to solve (4.15), which is a special case of one-rank regularized singular value decomposition. The whole process to find all the columns of $\boldsymbol{W}$ and $\boldsymbol{T}$ is explained in Algorithm 6.

Algorithm 5: One-rank regularized singular value decomposition (*1rSVD*).

Result: $\boldsymbol{u}, \boldsymbol{v}$ that minimize (4.15)
Input: $\boldsymbol{M}, \boldsymbol{\beta}$
Compute $\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, s$ that minimize $\left\|\boldsymbol{M} - \hat{\boldsymbol{u}} s \hat{\boldsymbol{v}}^\top\right\|_F^2$ (one-rank SVD).
Initialize $\boldsymbol{u} \leftarrow \hat{\boldsymbol{u}};\ \boldsymbol{v} \leftarrow s\hat{\boldsymbol{v}}$
while $\boldsymbol{v}$ *not stationary* **do**
    Update $\boldsymbol{v}$ with (4.16), cyclically iterating component-wise until convergence.
    Update $\boldsymbol{u} \leftarrow \boldsymbol{M}\boldsymbol{v}/\|\boldsymbol{M}\boldsymbol{v}\|_2$
end

Algorithm 6: Regularized singular value decomposition.

Result: $\boldsymbol{W}, \boldsymbol{T}$ that minimize (4.4)
Input: $\boldsymbol{M}, \boldsymbol{\beta}, K$
for $k = 1 \dots K$ **do**
    $\boldsymbol{u}, \boldsymbol{v} \leftarrow$ 1rSVD$(\boldsymbol{M}, \boldsymbol{\beta})$ (Solve the one-rank SVD problem)
    Set $\boldsymbol{T}_k \leftarrow \boldsymbol{u};\ \boldsymbol{W}_k \leftarrow \boldsymbol{v}$
    $\boldsymbol{M} \leftarrow \boldsymbol{M} - \boldsymbol{u}\boldsymbol{v}^\top$ (update $\boldsymbol{M}$ with the residuals)
end

Finally, we have to deal with the non-overlapping groups restriction $\|\boldsymbol{W}_{j\cdot}\|_0 = 1$. We propose a greedy approach to force $\boldsymbol{W}$ to have the

desired structure. The idea is to update $\boldsymbol{W}$ by,

$$\boldsymbol{W}_{jk} \leftarrow \boldsymbol{W}_{jk}\mathbb{1}\left(|\boldsymbol{W}_{jk}| = \max_i |\boldsymbol{W}_{ji}|\right), \text{ for all } j,k \tag{4.17}$$

For sufficiently large values of the penalization hyper-parameter γ, most of the components of $\boldsymbol{W}$ will be zero, so the effect of update (4.17) will be negligible as γ increases. Update (4.17) guarantees that $\|\boldsymbol{W}_{j\cdot}\|_0 \leq 1$. To get the equality, we will append $\boldsymbol{W}$ a column $\boldsymbol{W}_{K+1}$ such that $\boldsymbol{W}_{j\,K+1} = \prod_{k=1}^{K} \mathbb{1}(W_{jk} = 0)$, and $\boldsymbol{T}$ a null column $\boldsymbol{T}_{K+1}$.

4.4 Simulations

This simulation set-up is described in Witten, Shojaie, and Zhang, 2014. The data is simulated according to the linear model $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, with $p = 1000$ features, and ϵ_i i.i.d. from a $N(0, 2.5^2)$ distribution ($1 \leq i \leq n$). The data matrix $\boldsymbol{X}$ is simulated from a multivariate $N(\mathbf{0}, \boldsymbol{\Sigma})$ distribution, where $\boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$ is block diagonal, given by

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{\rho}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\boldsymbol{\rho}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}_{1000\times 1000},$$

with $\boldsymbol{\Sigma}_{\boldsymbol{\rho}} \in \mathbb{R}^{50\times 50}$ such that

$$\boldsymbol{\Sigma}_{\boldsymbol{\rho}}(i,j) = \begin{cases} 1 & i = j \\ \rho & i \neq j \end{cases}.$$

The parameter ρ is varied from 0 to 0.8, exploring different scenarios for the correlation inside groups. The true coefficient vector $\boldsymbol{\beta} \in \mathbb{R}^p$ is random, given by,

$$\boldsymbol{\beta} = [\beta_1\ \beta_2\ \ldots\ \beta_{25}\ \underbrace{0\ \ldots\ 0}_{25}\ \beta_{51}\ \beta_{52}\ \ldots\ \beta_{75}\ \underbrace{0\ \ldots\ 0}_{925}],$$

where

$$\beta_j \sim \begin{cases} U[0.9, 1.1], & 1 \leq j \leq 25 \\ U[-1.1, -0.9], & 51 \leq j \leq 75 \end{cases}.$$

The data matrix is composed of two groups of 50 variables, correlated within each group and independent between groups. Only 25 columns

within each group are significant. Additionally, there are another 900 variables that are independent of each other and have no impact on the response. This simulation scheme, as Witten, Shojaie, and Zhang mention, is motivated by gene pathways, where genes within the same pathway have correlated levels of expression, but only a fraction of these are associated with the response of interest.

Table 4.1 reports the results of the different methods in these simulations. We compared Lasso (Robert Tibshirani, 1996), Ridge, Elastic Net (EN) (Friedman, Hastie, and Rob Tibshirani, 2010), Elastic Net Cluster (CEN) (Witten, Shojaie, and Zhang, 2014), CEN with known groups, Cluster Group Lasso (Bühlmann et al., 2013), Group Lasso with known groups (Friedman, Hastie, and Robert Tibshirani, 2010), and our approach the GLASP. CEN and Group Lasso with known groups have been included for baseline comparisons since groups are, in general, unknown. A training data set composed of 200 observations was used to compare the different algorithms, whereas the hyperparameters were chosen using a validation sample also of size 200. The experiments were repeated 30 times in order to obtain more relevant results, calculated on an independent test sample of 800 observations. The different algorithms have been compared in terms of root mean squared error (RMSE) of the linear predictor, i.e,

$$\text{RMSE} = \left\| \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{X}\hat{\boldsymbol{\beta}} \right\|_2 .$$

We have also studied the accuracy of the variable selection (Correct Zeros), the number of coefficients different from zero (notice that 50 is the correct number of coefficients different from zero in the generating model), as well as the Rand Index (RI) (Rand, 1971), which measures the agreement between the actual and estimated clusters with each algorithm. This index varies between 0 and 1 (from low to high agreement). In the case of Lasso, Ridge, and EN, the reported groups are found by the k-means algorithm applied to the linear predictor matrix after estimating $\hat{\boldsymbol{\beta}}$. The values reported in Table 4.1 correspond to the means in 30 repetitions. The standard errors of the mean are shown in parentheses.

The results in Table 4.1 show that GLASP is superior to the other methods (except for the baseline methods with known groups) in terms

of RMSE and Correct Zeros, sometimes by a large margin, when correlations within groups are moderate (0.1, 0.2, 0.5). Furthermore, in general, the number of non-zero coefficients selected by GLASP is the lowest among the different methods, resulting in more parsimonious models. Concerning the Rand Index, GLASP is usually lower than other approaches, but we believe this is because GLASP builds the groups by balancing both criteria, the correlations between predictors and the relationship between predictors and the response variable. Therefore, GLASP does not produce either of the two groupings that are trivial in this case: two groups of 50 and one group of 900 (correlation), or two groups of 25 and one group of 950 (prediction). Groups found by GLASP are closely related to those groups that one can compute from the singular value decomposition (or, equivalently, the principal components) of the matrix of linear predictors.

Table 4.1: Average results of GLASP and other methods on a test set (800 observations) over 30 simulations. Standard errors are given in parenthesis. Models were fit on a training set (200 observations) with the hyperparameters that led to optimal RMSE on a validation set (200 observations). CEN and Group Lasso with known groups have been included for baseline comparisons (shaded rows).

$\rho = 0.0$				
Method	RMSE	Correct Zeros	Num. Non-Zeros	RI
Lasso + Kmeans	166.662(1.532)	0.885(0.005)	133.967(6.35)	0.909(0.001)
Ridge + Kmeans	182.004(0.97)	0.05(0)	1000(0)	0.897(0.004)
EN + Kmeans	165.491(1.264)	0.831(0.013)	196.767(13.684)	0.909(0.001)
CEN	167.013(1.463)	0.777(0.032)	253.933(32.999)	0.908(0)
CEN Known Groups	162.681(1.282)	0.807(0.008)	224.767(9.708)	1(0)
Cluster Group Lasso	183.714(0.871)	0.05(0)	1000(0)	0.366(0)
Group Lasso Known Groups	56.759(1.277)	0.113(0.044)	936.667(44.005)	1(0)
GLASP	172.771(1.414)	0.67(0.047)	358.633(49.427)	0.774(0.013)
$\rho = 0.1$				
Method	RMSE	Correct Zeros	Num. Non-Zeros	RI
Lasso + Kmeans	93.876(2.304)	0.911(0.004)	138.533(4.245)	0.951(0.003)
Ridge + Kmeans	199.147(1.619)	0.05(0)	1000(0)	0.949(0.002)
EN + Kmeans	93.635(2.283)	0.906(0.004)	143.5(4.127)	0.953(0.003)
CEN	93.954(2.357)	0.892(0.011)	157.433(11.147)	0.953(0.003)
CEN Known Groups	91.809(2.117)	0.881(0.011)	169.167(10.995)	1(0)
Cluster Group Lasso	166.879(2.137)	0.154(0.032)	895.433(32.323)	0.395(0.004)

Group Lasso Known Groups	39.468(0.866)	0.335(0.081)	715(80.841)	1(0)
GLASP	90.545(2.669)	0.956(0.009)	87(9.184)	0.914(0.003)
		$\rho = 0.2$		
Method	RMSE	Correct Zeros	Num. Non-Zeros	RI
Lasso + Kmeans	77.449(1.807)	0.933(0.004)	116.4(3.753)	0.98(0.001)
Ridge + Kmeans	185.387(1.779)	0.05(0)	1000(0)	0.936(0.001)
EN + Kmeans	77.166(1.768)	0.931(0.004)	118.467(3.907)	0.981(0.001)
CEN	73.654(1.306)	0.744(0.026)	306.267(25.934)	0.983(0.002)
CEN Known Groups	74.051(1.5)	0.829(0.017)	221.4(16.967)	1(0)
Cluster Group Lasso	77.141(3.553)	0.104(0.037)	946.5(37.447)	0.839(0.021)
Group Lasso Known Groups	35.551(0.775)	0.43(0.086)	620(86.423)	1(0)
GLASP	62.03(1.517)	0.97(0.005)	79.333(5.137)	0.943(0.002)
		$\rho = 0.5$		
Method	RMSE	Correct Zeros	Num. Non-Zeros	RI
Lasso + Kmeans	64.632(1.583)	0.958(0.002)	91.933(2.505)	0.982(0.001)
Ridge + Kmeans	149.217(1.627)	0.05(0)	1000(0)	0.91(0)
EN + Kmeans	63.369(1.407)	0.946(0.003)	103.7(3.077)	0.984(0.001)
CEN	61.36(1.52)	0.789(0.049)	260.567(49.496)	0.988(0.001)
CEN Known Groups	52.998(1.119)	0.813(0.022)	236.667(22.255)	1(0)
Cluster Group Lasso	59.377(0.888)	0.2(0.062)	850(62.284)	0.906(0)
Group Lasso Known Groups	29.144(0.691)	0.905(0.053)	145(52.923)	1(0)
GLASP	58.516(1.757)	0.968(0.002)	82.3(1.675)	0.963(0.003)

$\rho = 0.8$				
Method	RMSE	Correct Zeros	Num. Non-Zeros	RI
Lasso + Kmeans	60.031(1.348)	0.955(0.002)	89.7(2.476)	0.964(0.002)
Ridge + Kmeans	114.86(1.454)	0.05(0)	1000(0)	0.906(0)
EN + Kmeans	52.864(0.94)	0.935(0.003)	114.033(2.943)	0.969(0.001)
CEN	42.833(0.879)	0.732(0.043)	317.9(43.026)	0.993(0.001)
CEN Known Groups	32.346(0.834)	0.753(0.048)	296.867(48.34)	1(0)
Cluster Group Lasso	48.994(0.493)	0.17(0.057)	880(56.812)	0.906(0)
Group Lasso Known Groups	20.968(0.692)	0.905(0.053)	145(52.923)	1(0)
GLASP	48.291(0.892)	0.954(0.001)	96.333(0.946)	0.987(0.002)

4.5 Extension to other models

In Section 4.4, the choice of the function $L(\boldsymbol{\beta})$ corresponds to classical linear models. However, one strength of our methodology is that it can easily extend other risk functions, such as logistic regression or Cox models.

We have implemented the following three types of problems: linear and logistic regression and Cox proportional hazard models with right-censoring. In the first two cases, the function L is given by,

- *Linear regression*

$$L(\boldsymbol{\beta}) = \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{\eta}\|_2^2 ,$$

  where $\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta}$ is the linear predictor.

- *Logistic regression*

$$L(\boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} \log\left(1 + e^{\eta_i}\right) - \boldsymbol{y}_i \eta_i .$$

Our implementation requires to determine ∇L, which is given in each case by,

- *Linear regression*

$$\nabla L(\boldsymbol{\beta}) = -\frac{1}{N} \boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{\eta}).$$

- *Logistic regression*

$$\nabla L(\boldsymbol{\beta}) = \frac{1}{N} \boldsymbol{X}^\top \left(\frac{1}{1 + e^{\boldsymbol{\eta}}} - \boldsymbol{y} \right)$$

There are lots of numerical details to consider, especially in the case of logistic regression. For example, the function $\log(1 + e^{\eta})$ is unstable when $|\eta| > 30$. However, it can be substituted by a more stable approximation, given by

$$\hat{\log}(1+e^{\eta}) = \begin{cases} \eta, & \eta > 33.3 \\ \eta + e^{-\eta}, & 18 < \eta < 33.3 \\ \log(1+e^{\eta}), & -37 < \eta < 18 \\ e^{\eta}, & \eta < -37 \end{cases} \tag{4.18}$$

Similarly, its derivative can be replaced by

$$\frac{d}{d\eta}\hat{\log}(1+e^{\eta}) = \begin{cases} (1+e^{-\eta}, & \eta > -30 \\ e^{\eta}, & \eta < -30 \end{cases} \tag{4.19}$$

Although our implementation can address logistic regression, we will now focus on the Cox model, which has been less addressed in the literature from the perspective of variable selection.

4.5.1 Proportional hazards model with right-censoring

Under the proportional hazards model framework with right-censoring, we assume we have a covariate matrix $\boldsymbol{X} \in \mathbb{R}^{N\times p}$, a vector of event times $\boldsymbol{t} \in \mathbb{R}^{N\times 1}$ and a vector of event indicator $\boldsymbol{\delta} \in \mathbb{R}^{N\times 1}$ ($\delta_i = 1$ if an event was observed at time t_i, and $\delta_i = 0$ if time t_i is right-censored).

The proportional hazards model assumption states that, for an individual with covariates $\boldsymbol{x}^{\top} \in \mathbb{R}^{1\times p}$, their hazard function $h(t)$ is given by

$$h(t) = h_0(t)\exp(\boldsymbol{x}^{\top}\boldsymbol{\beta}),$$

where $h_0(t)$ is a baseline hazard function. This is a semi-parametric model, because $h_0(t)$ is not assumed to have a particular parametric form. More details can be found in Moore, 2016.

In the case of right censoring, our function L is the negative log-partial likelihood and it is given by,

$$L(\boldsymbol{\beta}) = \sum_{i\in D} \boldsymbol{x}_i^{\top}\boldsymbol{\beta} - \sum_{i\in D} \log\left(\sum_{k\in R_i} \exp(\boldsymbol{x}_k^{\top}\boldsymbol{\beta})\right),$$

where D is the index set of observed events, and R_i is the index set of individuals at risk at time t_i. Furthermore, the first derivative of L

has the expression,

$$\frac{\partial}{\partial \beta_j} L(\boldsymbol{\beta}) = \sum_{i \in D} \left(\boldsymbol{x}_{ij} - \frac{\sum_{k \in R_i} \boldsymbol{x}_{kj} \exp(\boldsymbol{x}_k^\top \boldsymbol{\beta})}{\sum_{k \in R_i} \exp(\boldsymbol{x}_k^\top \boldsymbol{\beta})} \right).$$

Once the model is fitted, with coefficient vector $\hat{\boldsymbol{\beta}}$, to estimate the baseline survival function we use,

$$S_0(t) = \exp(-H_0(t)), \text{ with } H_0(t) = \sum_{t_i \leq t} h_0(t_i),$$

where

$$h_0(t_i) = \frac{\delta_i}{\sum_{j \in R_i} \exp(\boldsymbol{x}_j^\top \hat{\boldsymbol{\beta}})}.$$

An individual's estimated survival function is given by

$$S(t|\boldsymbol{x}) = S_0(t)^{\exp(\boldsymbol{x}^\top \hat{\boldsymbol{\beta}})}. \tag{4.20}$$

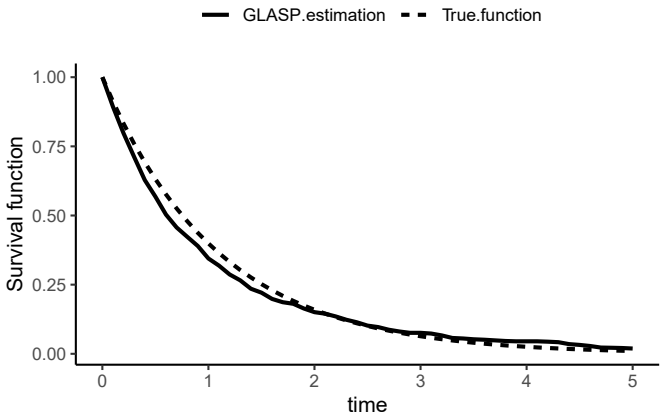
Example 1. *For illustrative purposes, we simulated a survival data set. The data matrix* $\boldsymbol{X} \in \mathbb{R}^{1000 \times 10}$ *has i.i.d.* $N(0,1)$ *columns and* $\boldsymbol{\beta}_j \sim N(0, 1/9)$ *for* $j \leq 5$ *and* 0 *otherwise. The underlying survival time* t_i^* *for a row* $\boldsymbol{x}_i^\top$ *is simulated exponential with parameter* $\lambda = \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})$*. The censoring time* s_i *distributes exponential with parameter* $\lambda = \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})/2$*. The observed time is the minimum between* t_i^* *and* s_i*.*

Figure 4.2 displays the estimation of the survival function given by (4.20)*, for an individual outside the training sample. In this case, the estimated function is remarkably close to the true survival function of this individual, according to the simulated model.*

4.5.2 Simulation studies: right-censored survival data

We consider an adaptation of the simulation set-ups described in Section 4.4. This time the response variable $\boldsymbol{t}$ and the event indicator $\boldsymbol{\delta}$ are simulated, for every $i = 1, 2 \ldots N$, according to the scheme,

Figure 4.2: Survival functions $S(t|\boldsymbol{x})$ estimated and real for an individual with simulated covariates. The GLASP model has been fitted on simulated data with the same $\boldsymbol{x}$ distribution.



$$
\begin{aligned}
h_i &= \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}),\\
t_i^* &\sim Exp(\lambda = h_i),\\
s_i &\sim Exp(\lambda = h_i/2),\\
t_i &= \min(t_i^*, s_i),\\
\delta_i &= \mathbb{1}(t_i = t_i^*).
\end{aligned}
$$

The data matrix $\boldsymbol{X}$ is simulated from a multivariate $N(\mathbf{0}, \boldsymbol{\Sigma})$ distribution, where $\boldsymbol{\Sigma}$ is block diagonal, given by

$$
\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma_\rho} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma_\rho} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}_{20\times 20},
$$

with $\boldsymbol{\Sigma_\rho} \in \mathbb{R}^{5\times 5}$ such that

$$
\boldsymbol{\Sigma_\rho}(i,j) = \begin{cases} 1 & i = j \\ 0.5 & i \neq j \end{cases}.
$$

The true coefficient vector $\boldsymbol{\beta} \in \mathbb{R}^p$ is random, given by,

$$
\boldsymbol{\beta} = [\beta_1\ \beta_2\ 0\ 0\ 0\ \ \beta_6\ \beta_7\ \underbrace{0\ \ldots\ 0}_{13}],
$$

where

$$\beta_j \sim \begin{cases} U[0.9, 1.1], & 1 \leq j \leq 2 \\ U[-1.1, -0.9], & 6 \leq j \leq 7 \end{cases}.$$

In this case, the X matrix has two significant groups of 5 variables, and only 2 variables within each group have an actual impact on the generating model. We have simulated 50 observations for training and 50 for testing. Furthermore, to obtain relevant results, the simulations were repeated 30 times, and the results averaged.

The models studied in Section 4.4 no longer apply, as they are not studied for Cox regression, or do not have an effective method for the selection of the regularization hyperparameters in the case of survival data. We have compared GLASP and the function `coxph` of the R package `survival` (Therneau, 2015; Terry M. Therneau and Patricia M. Grambsch, 2000). To calculate the groups given by `coxph`, we have used k-means, applied to the matrix of linear predictors once the model was adjusted, as we did in Section 4.4 for the algorithms that would not directly compute variable clusters.

Table 4.2 highlights the results of the simulations for survival data. The metric $\boldsymbol{\beta}$ WMSE (weighted mean squared error) refers to the $\boldsymbol{\beta}$ estimation error, given by $(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\top \Sigma (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})$, as described in Zhao et al., 2019. The rates $\boldsymbol{\beta}$ TPR (true positive rate) and TNR (true negative rate) refer to the correct identification of the variables that enter the model. Moreover, we included in Table 4.2 the mean estimation error of the survival curve $S(t|\boldsymbol{x})$ for the individuals in the test sample, measured as the integral of the absolute difference of the estimated and actual curves for each individual.

One can see from Table 4.2 that the estimation of GLASP is superior to the classical estimation of `coxph` in almost every aspect. We believe that this difference is, apart from the algorithm itself, also accentuated by the regularization hyperparameter selection approach that we have integrated with GLASP, described in the next section.

Table 4.2: Average results of GLASP and the Cox proportional hazards model.

Method	$\boldsymbol{\beta}$ WMSE	Correct Zeros	$\boldsymbol{\beta}$ TPR	$\boldsymbol{\beta}$ TNR	Num. Non-Zeros	$S(t\|x)$ error	RI
coxph	9.11 (1.54)	0.2 (0)	1 (0)	0 (0)	20 (0)	15.88 (0.35)	0.53 (0.01)
GLASP	3.27 (0.61)	0.37 (0.04)	0.95 (0.03)	0.23 (0.06)	16.03 (1.01)	13.86 (0.43)	0.58 (0.03)

4.6 Implementation details

In this section, we will describe some details of the implementation of GLASP, with emphasis on its R interface, and the selection of hyperparameters.

4.6.1 Interface

Recently, Kuhn and Vaughan have developed the R `parsnip` package (Kuhn and Vaughan, 2020), which provides a standard and organized interface for creating modelling packages in R. A critical advantage of this approach is that it integrates very well with other `tidymodels` packages. We have implemented our GLASP algorithm in an R package called `glasp`[1], created with the vision to integrate with `parsnip`, as well as the rest of `tidymodels` packages. This offers numerous advantages, highlighting, for example, the optimization of hyperparameters, which is always a concern in penalized models.

All the internal optimization of the `glasp` library has been implemented in C++, and integrated in R through `RcppArmadillo` (Eddelbuettel and François, 2011) and `Rcpp` (Eddelbuettel and François, 2011).

Currently, the `parsnip` library considers two types of objectives for the models: regression and classification. Taking this into account, we have created a model for `parsnip`, called `glasp_model`, which supports both regression and classification. For example, one way to fit a GLASP model for linear regression would be as follows.

```
glasp_model() %>%
set_mode("regression") %>%
set_engine("glasp") %>%
fit(y~., data)
```

To adjust logistic regression is analogous, changing `"regression"` to `"classification"`. However, it is challenging to match this tidy approach to the specification of a survival model. One path we have decided to take is to consider Cox regression as a particular case of classification, so that the response variable is δ (indicating whether the

[1] https://github.com/jlaria/glasp

event has been observed at each instant of time), and the covariates are the columns of $\boldsymbol{X}$ and the instants of time $\boldsymbol{t}$. Thus, a GLASP model for Cox regression with right-censoring would fit as follows.

```
model <- glasp_model() %>%
set_mode("classification") %>%
set_engine("glasp") %>%
fit(event ~ time + ., data)
```

Let $\hat{\boldsymbol{\beta}}$ be the coefficient estimation obtained by GLASP. Since (4.20) provides an approximation of the survival function, then the probability of having observed the event at a time prior to t for an individual with covariates $\boldsymbol{x}$, can be estimated as $p(t, \boldsymbol{x}) = 1 - S(t|\boldsymbol{x})$. One would expect $p(t_i, \boldsymbol{x}_i)$ to be high if $\delta_i = 1$ and low if $\delta_i = 0$, therefore, in a very practical predictive context, a survival problem can be considered as a classification problem, where $p(T, X) \approx P(\delta = 1)$. An important advantage of considering it this way, is that all predictive error metrics associated to classification problems (accuracy, sensitivity, specificity, F1-score, etc) can be calculated in survival problems, which provides a general approach to optimize hyperparameters.

4.6.2 Hyperparameter selection

Since `glasp_model` is a `parsnip` model, it integrates with the `tune` and `dials` libraries (Kuhn, 2020) to deal with the optimization of the hyperparameters λ_1, λ_2, λ_3, and K, from a very general approach. Package `tune` offers implementations of the three most popular types of hyperparameter search: grid search, random search (Bergstra and Bengio, 2012) and Bayesian Optimization (Snoek, Larochelle, and Adams, 2012).

For example, the following code in R finds the optimal combination of hyperparameters that minimizes the area under the ROC curve for a GLASP model in simulated survival data, using Bayesian Optimization, and 4-fold cross validation.

```
data <- simulate_dummy_surv_data()
model <- glasp_model(l1 = tune(), l2 = tune(),
                     frob = tune(), num_comp = tune()) %>%
         set_mode("classification") %>% set_engine("glasp")
```

```
data_rs <- vfold_cv(data, v = 4)
hist <- tune_bayes(model, event~.,
                   resamples = data_rs,
                   metrics = metric_set(roc_auc),
                   iter = 100)
show_best(hist, metric = "roc_auc")
```

In the simulation studies of Sections 4.4 and 4.5, the GLASP hyperparameters were optimized using random search.

4.7 Application to right-censored survival data

In this section we present an application of GLASP to real data from a study of patients with diffuse large-B-cell lymphoma (DLBCL). The data is available as right-censored survival sample data in the BioNet packages. For more information see Dittrich et al., 2008, Beisser et al., 2010, and Alizadeh et al., 2000.

The study of gene-expression profiles as predictors for survival of patients with DLBCL is motivated by the large variation in survival times after treatment of this disease, even for patients with similar clinical features. Several authors have studied patients with DLBCL, trying to predict the survival of individuals receiving treatment based on high-dimensional microarray gene expression data. Among these, we can find the works of Rosenwald et al., 2002, Bair et al., 2006 and Chen et al., 2011. To pre-process the data, we have followed an approach similar to that of Chen et al., 2011. We selected the genes for which individual Cox scores, obtained after fitting univariate Cox regression models, were more significant than a certain threshold. After removing missing values, the data were composed of 190 observations, 78 genetic features, and one clinical variable, which is a factor variable with several levels.

The objective of using the GLASP methodology with this dataset is twofold. Firstly, to build a survival model that includes only relevant genetic and clinical characteristics. Secondly, to find clusters among those relevant features, as GLASP can reveal hidden biological interrelations between gene expressions associated with this particular disease.

Figure 4.3 depicts the resulting coefficient estimation and feature clustering from GLASP in the DLBCL survival data described above. The output in Figure 4.3 includes only those variables with associated nonzero coefficients. According to the model, there are 11 groups, with varying sizes. From a biological perspective, the resulting clustering could give insight into possible genetic interactions. For example, Cluster 1 includes BCL2 and CASP10. Both genes are associated with cell apoptosis. BCL2 blocks the apoptotic death of some cells such as lymphocytes[2], whereas CASP10 plays a central role in the execution-phase of cell apoptosis[3]. This not only explains that they are in the same group, but also that their associated coefficients have opposite sign. As another illustration, Cluster 3, is formed by BMP6 and SRP72. BMP6 induces cartilage and bone formation[4], and mutations of SRP72 are associated with familial bone marrow failure[5].

[2] https://www.genecards.org/cgi-bin/carddisp.pl?gene=BCL2

[3] https://www.genecards.org/cgi-bin/carddisp.pl?gene=CASP10

[4] https://www.genecards.org/cgi-bin/carddisp.pl?gene=BMP6&keywords=BMP6

[5] https://www.genecards.org/cgi-bin/carddisp.pl?gene=SRP72&keywords=SRP72

Figure 4.3: GLASP model estimation and gene-clustering for the diffuse large-B-cell lymphoma dataset. Each row represents a cluster, with squares describing each variable that was included in the final model and its associated coefficient estimation in the Cox model.

4.8 Conclusions

The main contribution of this paper is the formulation of GLASP, a supervised variable clustering method, very competitive not only as a clustering method but also as a predictive model. Multiple models have been unified under a joint implementation, which also integrates with the latest algorithms for hyperparameter search in R. The methodologies are rarely so flexible that they allow adjusting classification problems, regression, and Cox survival models with the same algorithm. Moreover, Section 4.7 showcased an application of GLASP to biological survival data. From a methodological point of view, this paper has also introduced a particular case of sparse Singular Value Decomposition, with a penalty term appearing naturally from the Group Lasso penalty. Its solution and implementation using coordinate-descend was demonstrated in detail.

In the simulation studies in sections 4.4 and 4.5, it is observed that GLASP is substantially advantageous in terms of predictive ability and variable selection, apart from providing the simplest models. In the simulations of Section 4.4 we noticed that GLASP is the preferred alternative when the correlations between variables of the same group are moderate. If the dependencies are low, all the methods have similar performance, whereas if the correlations are high, Cluster Elastic Net has better performance.

Regarding possible extensions, we propose to explore possible safe-rules that would rule out multiple predictors from the very beginning, in order to reduce the dimension of the problem, as Ndiaye et al., 2016; Robert Tibshirani et al., 2012 do. Moreover, this would also allow finding bounds for the regularization hyperparameters, and thus accelerate the search for the best combinations.

Acknowledgments

We gratefully acknowledge the help provided by Prof. Daniela Witten, who gave us access to the source code of CEN and the simulation setups compared in Section 4.4.

SUPPLEMENTARY MATERIAL

Appendix: Proof of Propositions 2, 3 and 1. (PDF)

R-package glasp: R-package glasp containing code of the method described in the article. (GNU zipped tar file)

Source code: R scripts to generate Figures 5.6, 4.2, and 4.3, as well as Tables 4.1 and 4.2. The data set studied in Section 4.7 is included. These files are also available at `https://github.com/jlaria/glasp-code`. (Zip archive)

Dockerfile: The Dockerfile to build the docker image `jlaria/glasp:0.0.1` (`https://hub.docker.com/repository/docker/jlaria/glasp`) that includes the dependencies to run the experiments in this paper. (Plain text file)

Appendix

Proof of Proposition 1. To find the minimizer of (12), we use the sub-gradient conditions, since the penalization part is sub-differentiable. We have,

$$\partial_j \left(\frac{1}{2} \left\| \boldsymbol{\beta} - \boldsymbol{B_0} \right\|_2^2 \right) = \beta_j - (B_0)_j$$

$$\partial_j \left(\left\| \boldsymbol{\beta} \right\|_1 \right) = v_j, \quad \text{where} \quad v_j = \begin{cases} \operatorname{sign}(\beta_j) & \beta_j \neq 0 \\ \in [-1, 1] & \beta_j = 0 \end{cases}$$

$$\partial_j \left(\left\| \boldsymbol{\beta} \right\|_2 \right) = u_j, \quad \text{where} \quad \boldsymbol{u} = \begin{cases} \boldsymbol{\beta} / \left\| \boldsymbol{\beta} \right\|_2 & \boldsymbol{\beta} \neq \mathbf{0} \\ \in \{ \boldsymbol{u} : \left\| \boldsymbol{u} \right\|_2 \leq 1 \} & \boldsymbol{\beta} = \mathbf{0} \end{cases}$$

Consider $\boldsymbol{\beta} \neq \mathbf{0}$. If $\boldsymbol{\beta}$ is a minimizer of (12), then for every $j = 1, 2 \dots p_k$,

$$0 = \beta_j - (B_0)_j + t\lambda_1 v_j + t\lambda_2 \sqrt{p_k} \beta_j / \left\| \boldsymbol{\beta} \right\|_2 .$$

To take β_j out of this equation, we have to separate by cases and find v_j.

- Case $\beta_j > 0$ ($v_j = 1$)

$$\beta_j (1 + t\lambda_2 \sqrt{p_k} / \left\| \boldsymbol{\beta} \right\|_2) = (B_0)_j - t\lambda_1$$

- Case $\beta_j < 0$ ($v_j = -1$)

$$\beta_j (1 + t\lambda_2 \sqrt{p_k} / \left\| \boldsymbol{\beta} \right\|_2) = (B_0)_j + t\lambda_1$$

- Case $\beta_j = 0$ ($|v_j| \leq 1$)

$$0 = -(B_0)_j + t\lambda_1 v_j$$

Taking v_j out and since $|v_j| \leq 1$,

$$1 \geq \left| \frac{(B_0)_j}{t\lambda_1} \right|$$

which means that $\beta_j = 0$ if $-t\lambda_1 \leq (B_0)_j \leq t\lambda_1$.

Summarizing these three cases, we obtain,

$$\boldsymbol{\beta} \left(1 + \frac{t\lambda_2\sqrt{p_k}}{\|\boldsymbol{\beta}\|_2} \right) = S(\boldsymbol{B_0}, t\lambda_1).$$

Taking $\|\cdot\|_2$,

$$\|\boldsymbol{\beta}\|_2 = \|S(\boldsymbol{B_0}, t\lambda_1)\|_2 - t\lambda_2\sqrt{p_k}$$

and substituting $\|\boldsymbol{\beta}\|_2$ in the expression above,

$$\boldsymbol{\beta} = \left(1 - \frac{t\lambda_2\sqrt{p_k}}{\|S(\boldsymbol{B_0}, t\lambda_1)\|_2} \right) S(\boldsymbol{B_0}, t\lambda_1).$$

The case $\boldsymbol{\beta} = 0$ is analogous to Proposition 2 and leads to

$$\|S(\boldsymbol{B_0}, t\lambda_1)\|_2 \leq t\lambda_2\sqrt{p_k}.$$

The expression for the update function follows. □

Proof of Proposition 2. The subgradient condition for $\boldsymbol{\beta} = \mathbf{0}$ to be the minimizer of (6) is that $\mathbf{0} \in \partial_{\boldsymbol{\beta}} F(\mathbf{0})$, i.e., there is $\boldsymbol{u}$ with $\|\boldsymbol{u}\|_2 \leq 1$ and $\boldsymbol{v} \in [0,1]^p$ such that,

$$0 = \nabla R(\mathbf{0}) + \lambda_1 \boldsymbol{v} + \lambda_2\sqrt{p_k}\boldsymbol{u}.$$

Taking $\boldsymbol{u}$ out,

$$\boldsymbol{u} = -\frac{1}{\lambda_2\sqrt{p_k}} \left(\nabla R(\mathbf{0}) + \lambda_1 \boldsymbol{v} \right)$$

and using $\|u\|_2 \leq 1$,

$$\|\nabla R(\mathbf{0}) + \lambda_1 \boldsymbol{v}\|_2 \leq \lambda_2 \sqrt{p_k}.$$

This is true if and only if

$$\min\{\|\nabla R(\mathbf{0}) + \lambda_1 \boldsymbol{v}\|_2 : \boldsymbol{v} \in [0,1]^p\} \leq \lambda_2 \sqrt{p_k},$$

and that minimum is attained when

$$v_j = \begin{cases} -1, & \nabla_j R(\mathbf{0}) > \lambda_1 \\ -(\nabla_j R(\mathbf{0}))/\lambda_1, & |\nabla_j R(\mathbf{0})| \leq \lambda_1 \\ 1 & \nabla_j R(\mathbf{0}) < -\lambda_1 \end{cases}.$$

Substituting $\boldsymbol{v}$ in the expression above, the condition becomes,

$$\|S(\nabla R(\mathbf{0}), \lambda_1)\|_2 \leq \lambda_2 \sqrt{p_k}.$$

In particular, $\boldsymbol{\beta} = \mathbf{0}$ is also optimal if

$$0 = \|S(\nabla R(\mathbf{0}), \lambda_1)\|_2 \leq \lambda_2 \sqrt{p_k},$$

and by the definition of the coordinate-wise soft thresholding operator, that means,

$$\max_j \{|\nabla R(\mathbf{0})|\} \leq \lambda_1.$$

□

Proof of Proposition 3. Firstly, notice that the differentiable part of (15) can be written as,

$$\begin{aligned} \left\|\boldsymbol{M} - \boldsymbol{u}\boldsymbol{v}^\top\right\|_F^2 &= \sum_{i=1}^{N}\sum_{j=1}^{p}(\boldsymbol{M}_{ij} - \boldsymbol{u}_i \boldsymbol{v}_j)^2 \\ &= \sum_{i=1}^{N}\sum_{j=1}^{p}(\boldsymbol{M}_{ij}^2 - 2\boldsymbol{M}_{ij}\boldsymbol{u}_i\boldsymbol{v}_j + \boldsymbol{u}_i^2\boldsymbol{v}_j^2) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{p}\boldsymbol{M}_{ij}^2 - \sum_{i=1}^{N}\sum_{j=1}^{p}2\boldsymbol{M}_{ij}\boldsymbol{u}_i\boldsymbol{v}_j + \sum_{i=1}^{N}\sum_{j=1}^{p}\boldsymbol{u}_i^2\boldsymbol{v}_j^2 \\ &= \|\boldsymbol{M}\|_F^2 - \sum_{j=1}^{p}\boldsymbol{v}_j\sum_{i=1}^{N}\boldsymbol{M}_{ij}\boldsymbol{u}_i + \sum_{j=1}^{p}\boldsymbol{v}_j^2\sum_{i=1}^{N}\boldsymbol{u}_i^2. \end{aligned}$$

Since $\boldsymbol{u}^\top \boldsymbol{u} = 1$, we can write,

$$\left\|\boldsymbol{M} - \boldsymbol{u}\boldsymbol{v}^\top\right\|_F^2 = \|\boldsymbol{M}\|_F^2 + \sum_{j=1}^{p} \boldsymbol{v}_j^2 - \sum_{j=1}^{p} \boldsymbol{v}_j \left(\boldsymbol{M}^\top \boldsymbol{u}\right)_j,$$

and after removing constant terms,

$$\min_{\boldsymbol{u},\boldsymbol{v}} \left\|\boldsymbol{M} - \boldsymbol{u}\boldsymbol{v}^\top\right\|_F^2 = \min_{\boldsymbol{u},\boldsymbol{v}} \sum_{j=1}^{p} \left(\boldsymbol{v}_j^2 - 2(\boldsymbol{M}^\top \boldsymbol{u})_j \boldsymbol{v}_j\right).$$

Assuming $\boldsymbol{u}$ fixed, we want to minimize the function,

$$F(\boldsymbol{v}) := \sum_{j=1}^{p} \left(\boldsymbol{v}_j^2 - 2(\boldsymbol{M}^\top \boldsymbol{u})_j \boldsymbol{v}_j\right) + \gamma \left(\sum_{j=1}^{p} \beta_j^2 \mathbb{1}(\boldsymbol{v}_j \neq 0)\right)^{1/2} \left(\sum_{j=1}^{p} \mathbb{1}(\boldsymbol{v}_j \neq 0)\right)^{1/2}.$$

For a particular component $\boldsymbol{v}_l$, the optimality conditions follow after separating by cases.

- Case $\boldsymbol{v}_l \neq 0$. $F(\boldsymbol{v}_l \neq 0)$ becomes

$$\begin{aligned} &\sum_{\substack{j=1 \\ j\neq l}}^{p} \left(\boldsymbol{v}_j^2 - 2(\boldsymbol{M}^\top \boldsymbol{u})_j \boldsymbol{v}_j\right) \\ &\quad + \left(\boldsymbol{v}_l^2 - 2(\boldsymbol{M}^\top \boldsymbol{u})_l \boldsymbol{v}_l\right) \\ &\quad + \gamma \left(C_{\boldsymbol{\beta},\boldsymbol{v}}^{(-l)} + \beta_l^2\right)^{1/2} \left(C_{\boldsymbol{v}}^{(-l)} + 1\right)^{1/2}, \end{aligned} \tag{4.21}$$

  whose minimum is $\boldsymbol{v}_l^* = (\boldsymbol{M}^\top \boldsymbol{u})_l$.

- Case $\boldsymbol{v}_l = 0$. $F(\boldsymbol{v}_l = 0)$ becomes,

$$\sum_{\substack{j=1 \\ j\neq l}}^{p} \left(\boldsymbol{v}_j^2 - 2(\boldsymbol{M}^\top \boldsymbol{u})_j \boldsymbol{v}_j\right) + \gamma \left(C_{\boldsymbol{\beta},\boldsymbol{v}}^{(-l)} C_{\boldsymbol{v}}^{(-l)}\right)^{1/2}.$$

  For $\boldsymbol{v}_l^* = 0$ to be the optimum of $F(\boldsymbol{v}_l)$ it must hold that $F(0) \leq F((\boldsymbol{M}^\top \boldsymbol{u})_j)$. That is,

$$(\boldsymbol{M}^\top \boldsymbol{u})_l^2 \leq \gamma \left(C_{\boldsymbol{\beta},\boldsymbol{v}}^{(-l)} + \beta_l^2\right)^{1/2} \left(C_{\boldsymbol{v}}^{(-l)} + 1\right)^{1/2} - \gamma \left(C_{\boldsymbol{\beta},\boldsymbol{v}}^{(-l)} C_{\boldsymbol{v}}^{(-l)}\right)^{1/2}.$$

  The formula for $\boldsymbol{v}_l^*$ follows.

□

Bibliography for Chapter 4

Alizadeh, Ash A et al. (2000). "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling". In: *Nature* 403.6769, pp. 503–511.

Bair, Eric et al. (2006). "Prediction by supervised principal components". In: *Journal of the American Statistical Association* 101.473, pp. 119–137.

Beck, Amir and Marc Teboulle (2009). "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1, pp. 183–202.

Beisser, Daniela et al. (2010). "BioNet: an R-Package for the functional analysis of biological networks". In: *Bioinformatics* 26.8, pp. 1129–1130.

Bergstra, James and Yoshua Bengio (2012). "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* 13.Feb, pp. 281–305.

Bühlmann, Peter et al. (2013). "Correlated variables in regression: clustering and sparse estimation". In: *Journal of Statistical Planning and Inference* 143.11, pp. 1835–1858.

Chen, Kun et al. (2011). "Stringing high-dimensional data for functional analysis". In: *Journal of the American Statistical Association* 106.493, pp. 275–284.

Dittrich, Marcus T et al. (2008). "Identifying functional modules in protein–protein interaction networks: an integrated exact approach". In: *Bioinformatics* 24.13, pp. i223–i231.

Eddelbuettel, Dirk and Romain François (2011). "Rcpp: Seamless R and C++ Integration". In: *Journal of Statistical Software* 40.8, pp. 1–18. DOI: 10.18637/jss.v040.i08. URL: http://www.jstatsoft.org/v40/i08/.

Friedman, Jerome, Trevor Hastie, and Rob Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent". In: *Journal of statistical software* 33.1, p. 1.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2010). "A note on the group lasso and a sparse group lasso". In: *arXiv preprint arXiv:1001.0736*.

Kuhn, Max (2020). *tune: Tidy Tuning Tools*. R package version 0.1.0. URL: https://CRAN.R-project.org/package=tune.

Kuhn, Max and Davis Vaughan (2020). *parsnip: A Common API to Modeling and Analysis Functions*. R package version 0.0.5. URL: https://CRAN.R-project.org/package=parsnip.

Laria, Juan C, M Carmen Aguilera-Morillo, and Rosa E Lillo (2019). "An iterative sparse-group lasso". In: *Journal of Computational and Graphical Statistics*, pp. 1–10.

Moore, Dirk F (2016). *Applied survival analysis using R*. Springer.

Ndiaye, Eugene et al. (2016). "Gap safe screening rules for sparse-group lasso". In: *Advances in Neural Information Processing Systems*, pp. 388–396.

Price, Bradley S and Ben Sherwood (2017). "A cluster elastic net for multivariate regression". In: *The Journal of Machine Learning Research* 18.1, pp. 8685–8723.

Rand, William M (1971). "Objective criteria for the evaluation of clustering methods". In: *Journal of the American Statistical association* 66.336, pp. 846–850.

Ren, Sheng, Emily L Kang, and Jason L Lu (2020). "MCEN: a method of simultaneous variable selection and clustering for high-dimensional multinomial regression". In: *Statistics and Computing* 30.2, pp. 291–304.

Rosenwald, Andreas et al. (2002). "The use of molecular profiling to predict survival after chemotherapy for diffuse large-B-cell lymphoma". In: *New England Journal of Medicine* 346.25, pp. 1937–1947.

Shen, Haipeng and Jianhua Z Huang (2008). "Sparse principal component analysis via regularized low rank matrix approximation". In: *Journal of multivariate analysis* 99.6, pp. 1015–1034.

Simon, Noah et al. (2013). "A sparse-group lasso". In: *Journal of computational and graphical statistics* 22.2, pp. 231–245.

Snoek, Jasper, Hugo Larochelle, and Ryan P Adams (2012). "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems*, pp. 2951–2959.

Terry M. Therneau and Patricia M. Grambsch (2000). *Modeling Survival Data: Extending the Cox Model*. New York: Springer. ISBN: 0-387-98784-3.

Therneau, Terry M (2015). *A Package for Survival Analysis in S*. version 2.38. URL: https://CRAN.R-project.org/package=survival.

Tibshirani, Robert (1996). “Regression Shrinkage and Selection via the Lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1, pp. 267–288.

Tibshirani, Robert et al. (2012). “Strong rules for discarding predictors in lasso-type problems”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 74.2, pp. 245–266.

Witten, Daniela M, Ali Shojaie, and Fan Zhang (2014). “The cluster elastic net for high-dimensional regression with unknown variable grouping”. In: *Technometrics* 56.1, pp. 112–122.

Zhao, Hui et al. (2019). “Simultaneous estimation and variable selection for interval-censored data with broken adaptive ridge regression”. In: *Journal of the American Statistical Association*, pp. 1–13.

Zhou, Nengfeng and Ji Zhu (2010). “Group variable selection via a hierarchical lasso and its oracle property”. In: *Statistics and Its Interface* 3, pp. 557–574.

Zou, Hui and Trevor Hastie (2005). “Regularization and variable selection via the elastic net”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2, pp. 301–320.

CHAPTER 5

A Generalized Elastic-Net for Semi-Supervised Learning of Sparse Features

In *arXiv preprint arXiv:2006.01671* (2020)

Juan C. Laria[a,b] Line H. Clemmensen[c] Bjarne K. Ersbøll[c]

[a] Department of Statistics, University Carlos III of Madrid, Spain
[b] UC3M-BS Institute of Financial Big Data, Madrid, Spain
[c] DTU Compute, Technical University of Denmark, Copenhagen

Abstract

The elastic-net is among the most widely used types of regularization algorithms, commonly associated with the problem of supervised generalized linear model estimation via penalized maximum likelihood. Its nice properties originate from a combination of ℓ_1 and ℓ_2 norms, which endow this method with the ability to select variables taking into account the correlations between them. In the last few years, semi-supervised approaches, that use both labeled and unlabeled data, have become an important component in the statistical research. Despite this interest, however, few researches have investigated semi-supervised elastic-net extensions. This paper introduces a novel solution for semi-supervised learning of sparse features in the con-

text of generalized linear model estimation: the generalized semi-supervised elastic-net (s²net), which extends the supervised elastic-net method, with a general mathematical formulation that covers, but is not limited to, both regression and classification problems. We develop a flexible and fast implementation for s²net in R, and its advantages are illustrated using both real and synthetic data sets.

Keywords: Semi Supervised. Elastic Net. GLM. Variable selection. Linear models. R package

5.1 Introduction

In this paper, we propose a simple, but novel solution for extending the elastic-net to semi-supervised generalized linear models. Semi-supervised statistical methods are attracting increasing interest due to their ability to learn from both labeled and unlabeled data. They represent a remarkable alternative to supervised methods, that only use labeled observations in their learning process. There are many practical problems in which a semi-supervised framework arises naturally. For instance, when we fit a predictive model, often some part of the "future" data (with unknown labels) that we want to predict, is already available. This data represents information that can be exploited to improve the performance of the trained model.

In the history of statistical learning, the focus has often been on supervised methods, possibly due to their ability to predict labels when new observations are given, which also make their evaluation and benchmark straightforward. Recent developments in distributed computing and data storage technologies, have contributed to boost the research on statistical models. In this new context, semi-supervised approaches are likely to become an important component in the statistical research, as demonstrated by the active investigations on artificial neural networks, deep-learning and image classification in the semi-supervised context (Ji, Henriques, and Vedaldi, 2019; Genkin, Sengupta, and Chklovskii, 2019; Oliver et al., 2018).

Despite this interest, as far as we know, few researchers have investigated semi-supervised elastic-net extensions, from the perspective

of penalized linear models. Among the few, we find the work of Tan, Zhang, and Wang, 2011, where the authors propose a novel elastic-net approach to deal with sequential data for pedestrian counting. However, their context is very different from the problem set-up that we investigate, which bears a close resemblance to the one explored by Ryan and M. V. Culp, 2015; M. Culp, 2013, where very detailed theoretical results and proofs of the advantages of the joint trained linear framework (JT) in the semi-supervised framework are provided. The JT simultaneously shrinks the linear estimator and de-correlates the data (as the supervised elastic-net does), but using the existing unlabeled observations to more accurately define the correlations in the data, introduced as an additional regularization term. From a computational point of view, JT is not a novel algorithm. Its solution is computed using the supervised elastic-net (specifically, the **glmnet** package for R), but it can exploit properties of that elastic-net implementation, such as the regularization paths (Friedman, Hastie, and Rob Tibshirani, 2010), and the safe rules (Robert Tibshirani et al., 2012). Regarding this, our method could be interesting because the loss function is more general, and it does not rely on other implementations. Recently, Larsen et al., 2020 introduced the extended linear joint trained framework (ExtJT), where the shift in mean value and the covariance structure are modelled explicitly, resulting in a more flexible framework. Larsen et al., 2020 focused on semi-supervised regression with a penalized least squares error loss to transfer a model from a labeled source domain to an unlabeled target domain. Although the ExtJT approach is interesting, it does not allow for automatic variable selection via elastic-net, since the authors use partial least squares to solve the supervised least squares part. Moreover, to date, the joint trained methodology is only applicable to linear regression problems. Our s²net integrates the core ideas of ExtJT, adding the elastic-net regularization to deal with high dimensional data, and a generalization to both regression and classification problems. Thus, our framework also provides semi-supervised logistic regression models with elastic-net penalizations.

Regarding classification with unlabeled data, early extensions of logistic models to handle unlabeled observations are found in the work by Amini and Gallinari, 2002, from a maximum likelihood approach.

More details on the semi-supervised literature are provided by Chapelle, Schölkopf, and Zien, 2010. More recent approaches to deal with classification in this context, but not from an elastic-net regularization perspective, are described by M. V. Culp and Ryan, 2018 and Krijthe and Loog, 2015.

This paper outlines a new approach to semi-supervised learning: the Generalized semi-supervised elastic-net (s²net), including the following contributions.

- Our method extends the supervised elastic-net problem, and thus it is a practical solution to the problem of feature selection in semi-supervised contexts.

- Its mathematical formulation is presented from a general perspective, covering a wide range of models. We focus on linear and logistic responses, but the implementation could be easily extended to other losses in generalized linear models.

- We develop a flexible and fast implementation for s²net in R, written in C++ using **RcppArmadillo** and integrated into R via **Rcpp** modules (R Core Team, 2019; Eddelbuettel and François, 2011; Eddelbuettel and Balamuta, 2017; Eddelbuettel and Sanderson, 2014; Sanderson and Curtin, 2016; Sanderson and Curtin, 2019). The software is available in the **s2net** package.

This paper is organized as follows. Section 5.2 provides the mathematical framework of our methodology. Details regarding the algorithm and its implementation are discussed in Sections 5.3 and 5.4. Sections 5.5 and 5.6 explore its properties using synthetic and real data sets, respectively. Some conclusions are drawn in the final section.

5.2 Methodology

Given labeled data $\boldsymbol{X}_L \in \mathbb{R}^{n_L \times p}$, with labels $\boldsymbol{y}_L \in \mathbb{R}^{n_L}$ and unlabeled data $\boldsymbol{X}_U \in \mathbb{R}^{n_U \times p}$, the Extended Linear Joint Trained Framework

(ExtJT) optimization problem from Larsen et al., 2020 is given as

$$\boldsymbol{\beta} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \|\boldsymbol{y}_L - \boldsymbol{X}_L\boldsymbol{\beta}\|_2^2 + \gamma_1 \|\boldsymbol{T}_1(\gamma_2)\boldsymbol{\beta}\|_2^2 + \gamma_3 \frac{n_L n_U}{n_L + n_U} \|\boldsymbol{T}_2\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2 \right\}, \tag{5.1}$$

where $\lambda_1, \lambda_2, \gamma_1, \gamma_2, \gamma_3$ are regularization hyper-parameters, $\boldsymbol{T}_2 = \boldsymbol{\mu}^\top \in \mathbb{R}^{1\times p}$ is the vector of column-means of $\boldsymbol{X}_U$, and

$$\boldsymbol{T}_1(\gamma_2) = \sqrt{\gamma_2}(\boldsymbol{\Sigma}^2 + \gamma_2 \boldsymbol{I})^{-1/2}\boldsymbol{\Sigma}\boldsymbol{V}^\top,$$

with $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ the singular value decomposition of the centered unlabeled data $\boldsymbol{X}_U - \mathbf{1}\boldsymbol{\mu}^\top$. To simplify computations and notation, we assume that the labeled data $\boldsymbol{X}_L$ is column-centered ($\boldsymbol{X}_L^\top \mathbf{1} = \mathbf{0}_p$).

Here we have included the elastic-net regularization term $\lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2$. In their methodology, Larsen et al. solve (5.1) using partial least squares regression, and thus avoid the need of the elastic-net regularization to solve the least squares objective in the high-dimensional setting. However, this has two downsides: the number of PLS components is a hyper-parameter that has to be selected, and the coefficient vector $\boldsymbol{\beta}$ produced by the PLS regression model is not sparse. We instead prefer to set (5.1) as our initial framework.

The objective function in (5.1) has three important parts, namely

- The error function for the labeled data, $\|\boldsymbol{y}_L - \boldsymbol{X}_L\boldsymbol{\beta}\|_2^2$.
- The elastic-net regularization on the coefficients, $\lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2$.
- A regularization part that only depends on the unlabeled data,

$$\gamma_1 \|\boldsymbol{T}_1(\gamma_2)\boldsymbol{\beta}\|_2^2 + \gamma_3 \frac{n_L n_U}{n_L + n_U} \|\boldsymbol{T}_2\boldsymbol{\beta}\|_2^2. \tag{5.2}$$

Using a reparameterization of γ_1, γ_2 and γ_3, one can show that (5.2) is equivalent to $\gamma_1 \|\boldsymbol{T}(\gamma_2, \gamma_3)\boldsymbol{\beta}\|_2^2$, where $\boldsymbol{T}(\gamma_2, \gamma_3)$ is a transformation

of the unlabeled data that captures both the covariance structure and the shift with respect to the labeled data, given by,

$$\boldsymbol{T}(\gamma_2, \gamma_3) = \sqrt{\gamma_2}\boldsymbol{U}(\boldsymbol{\Sigma}^2 + \gamma_2\boldsymbol{I})^{-1/2}\boldsymbol{\Sigma}\boldsymbol{V}^\top + \gamma_3\mathbf{1}\boldsymbol{\mu}^\top. \tag{5.3}$$

Furthermore, to obtain (5.1), Larsen et al. assume that the labels $\boldsymbol{y}_L$ are centered. If they are not centered, (5.1) can be rewritten as,

$$\begin{aligned}\boldsymbol{\beta} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \big\{ \|\boldsymbol{y}_L - \boldsymbol{X}_L\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2 \\ + \gamma_1 \|\bar{\boldsymbol{y}}_L\mathbf{1} - \boldsymbol{T}(\gamma_2, \gamma_3)\boldsymbol{\beta}\|_2^2 \big\}.\end{aligned} \tag{5.4}$$

The intuition behind (5.4) is that we are adding information about the unlabeled data to the model through a transformation of this data, and we want predictions on those points to be close to $\bar{\boldsymbol{y}}_L$, which is the mean response we expect *a-priori* on future unknown data.

Figure 5.1 provides insights into the intuition behind $\boldsymbol{T}(\gamma_2, \gamma_3)$, when the hyper-parameters γ_2 and γ_3 are changed. We can see that γ_2 regulates the covariance structure, whereas γ_3 controls the shift between the center of the labeled data and the center of the unlabeled data.

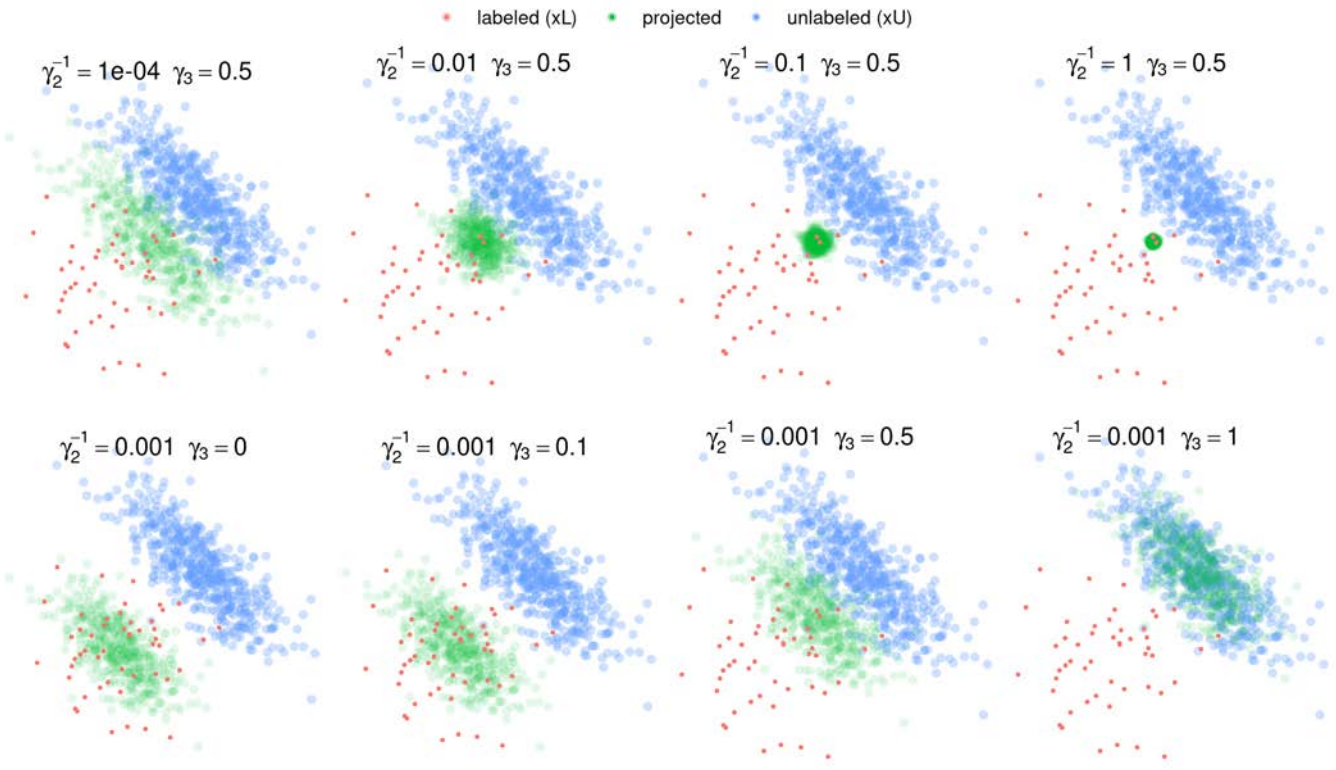


Figure 5.1: Simulated 2D-data that illustrates how varying the parameters γ_2 and γ_3 affects the projected "null" data $\boldsymbol{T}(\gamma_2, \gamma_3)$.

We now turn our attention to an extension of (5.4). The choice of square error norm for the error term $\|\boldsymbol{y}_L - \boldsymbol{X}_L\boldsymbol{\beta}\|_2^2$ is justified when the underlying model is linear. However, in other scenarios (for instance, binary response) it makes more sense to use other risk functions. With that in mind, we propose to write (5.4) in a more general form, letting $R(\cdot \mid \boldsymbol{y}, \boldsymbol{X}) : \mathbb{R}^p \to \mathbb{R}$ be any (continuously differentiable and convex) risk function.

$$\boldsymbol{\beta} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ R(\boldsymbol{\beta} \mid \boldsymbol{y}_L, \boldsymbol{X}_L) + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2 + \gamma_1 R(\boldsymbol{\beta} \mid \bar{\boldsymbol{y}}_L, \boldsymbol{T}(\gamma_2, \gamma_3)) \right\}. \tag{5.5}$$

Notice that both the input data matrices and the hyper-parameters are fixed, and therefore, (without loss of generality) problem (5.5) can be reparameterized as (s²net)

$$\boldsymbol{\beta} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ L(\boldsymbol{\beta}) + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2 \right\}, \tag{5.6}$$

where $L(\boldsymbol{\beta} \mid \boldsymbol{y}_L, \boldsymbol{X}_L, \boldsymbol{X}_U, \gamma_1, \gamma_2, \gamma_3)$ is given by

$$L(\boldsymbol{\beta}) = R(\boldsymbol{\beta} \mid \boldsymbol{y}_L, \boldsymbol{X}_L) + \gamma_1 R(\boldsymbol{\beta} \mid \bar{\boldsymbol{y}}_L, \boldsymbol{T}(\gamma_2, \gamma_3)). \tag{5.7}$$

Remark 1. *Problem* (5.6) *is a generalized elastic-net problem with a custom loss function. If* $\gamma_1 = 0$*, then* (5.6) *is the (naive) supervised elastic-net problem (Zou and Hastie, 2003).*

Remark 2. *If we let* $\boldsymbol{T}(\gamma_2) = \sqrt{\gamma_2}\boldsymbol{U}(\boldsymbol{\Sigma}^2 + \gamma_2\boldsymbol{I})^{-1/2}\boldsymbol{U}^\top\boldsymbol{X}_U$*, with* $\boldsymbol{X}_U = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ *the singular value decomposition of* $\boldsymbol{X}_U$ *(without centering), and* $R(\cdot \mid \boldsymbol{y}, \boldsymbol{X})$ *the norm-2 squared error , then* (5.6) *is the Linear Joint Trained Framework (JT) (M. Culp, 2013).*

Remark 3. *Letting* $\gamma_2 = 0$ *and* $R(\cdot \mid \boldsymbol{y}, \boldsymbol{X})$ *the norm-2 squared error,* (5.6) *is the NARE formulation from Andries, Kalivas, and Gurung, 2019.*

Previous remarks highlight that s²net generalizes other approaches and therefore, with a strong algorithm to optimize the objective function and an appropriate selection of the hyperparameters, s²net can outperform (or at least emulate) other popular methods' results.

5.3 Algorithm

Remark 1 suggests that the solution of (5.6) can be found solving an elastic-net problem with a general error term. To solve it, we prefer the *fast iterative shrinkage-thresholding algorithm* (FISTA) (Beck and Teboulle, 2009), which is an accelerated gradient descent approach with backtracking. In each step, given an initial $\boldsymbol{\beta}_0 \in \mathbb{R}^p$, we minimize the surrogate function

$$M_t(\boldsymbol{\beta}) = \frac{1}{2t} \left\| \boldsymbol{\beta} - \boldsymbol{\beta}_0 + t\nabla L(\boldsymbol{\beta}_0) \right\|_2^2 + \lambda_1 \left\| \boldsymbol{\beta} \right\|_1 + \lambda_2 \left\| \boldsymbol{\beta} \right\|_2^2, \quad (5.8)$$

where $t > 0$ is some step-size (chosen using backtracking).

Proposition 1.

$$U_t(\boldsymbol{\beta}) := \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \{ M_t(\boldsymbol{\beta}) \} = \underbrace{(1 + 2t\lambda_2)^{-1}}_{\textit{ridge}} \underbrace{S\left(\boldsymbol{\beta}_0 - t\nabla L(\boldsymbol{\beta}_0),\, t\lambda_1\right)}_{\textit{lasso shrinkage}}, \quad (5.9)$$

where S is the coordinate-wise soft-thresholding operator,

$$S(\boldsymbol{z}, \lambda)_i = sign(z_i)(|z_i| - \lambda)_+.$$

Proposition 1 suggests a gradient descent procedure to minimize (5.8). In addition, after each iteration k, we apply the FISTA update, given by

$$\boldsymbol{\beta}_{(k+1)} \leftarrow U_{t_k}(\boldsymbol{\beta}_{(k)}) + \frac{l_k - 1}{l_{k+1}} (U_{t_k}(\boldsymbol{\beta}_{(k)}) - U_{t_{k-1}}(\boldsymbol{\beta}_{(k-1)})), \quad (5.10)$$

where $l_{k+1} = (1 + \sqrt{1 + 4l_k^2})/2,\ l_1 = 1$.

The choice for the function R in (5.7) depends on the type of response variable. For instance, if the response is continuous (linear regression) then $R(\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$ is probably the best choice. However, if the response is binary (logistic regression) then the logit loss is more appropriate,

$$R(\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X}) = \sum_{i=1}^{n} \left(\log(1 + \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})) - y_i \boldsymbol{x}_i^\top \boldsymbol{\beta} \right) \quad (5.11)$$

Here we want to emphasize that the function $\log(1+e^{\eta})$ is computationally problematic when, roughly, $|\eta|> 30$. In our implementation we substitute it by a more stable approximation – see Mächler, 2012; Pedregosa and Merrienboer, 2019,

$$\hat{\log}(1+e^{\eta}) = \begin{cases} \eta, & \eta > 33.3 \\ \eta + e^{-\eta}, & 18 < \eta < 33.3 \\ \log(1+e^{\eta}), & -37 < \eta < 18 \\ e^{\eta}, & \eta < -37 \end{cases} \tag{5.12}$$

5.3.1 Removing the shift in the unlabeled data

When the direction of the mean shift of the unlabeled data $\boldsymbol{X}_U$ with respect to the labeled data $\boldsymbol{X}_L$ is in the same direction as $\boldsymbol{\beta}$ (or close), then $\mathbb{E}y_L \neq \mathbb{E}y_U$. This, as Larsen et al. noticed, forces the optimal hyper-parameter γ_3 to be zero. One strategy that they propose is to remove the effect of $\boldsymbol{\beta}$ in $\boldsymbol{\mu}$ (which is the mean shift of $\boldsymbol{X}_U$ with respect to $\boldsymbol{X}_L$) by updating $\boldsymbol{X}_U$ with

$$\tilde{\boldsymbol{X}}_U = X_U - \mathbf{1}\boldsymbol{\mu}^{\top}\boldsymbol{p}\boldsymbol{p}^{\top}, \tag{5.13}$$

where

$$\boldsymbol{p} = \frac{\boldsymbol{X}_L^{\top}\boldsymbol{y}_L}{\left\|\boldsymbol{X}_L^{\top}\boldsymbol{y}_L\right\|_2}. \tag{5.14}$$

We instead propose to use

$$\boldsymbol{p} = -\frac{\nabla R(\mathbf{0} \mid \boldsymbol{y}_L, \boldsymbol{X}_L)}{\|\nabla R(\mathbf{0} \mid \boldsymbol{y}_L, \boldsymbol{X}_L)\|_2} \tag{5.15}$$

thus extending this idea to a general loss functions. However, the update in (5.13) is not necessary (and may introduce unwanted noise) if the angle between $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ is too big (Larsen et al.). In our implementation, we have set the threshold to $\pi/4$, but the user can choose whether to apply this update or not. Figure 5.2 illustrates update (5.13) with a 2D example. The unlabeled data $\boldsymbol{X}_U$ (blue) is shifted (green) towards the center of $\boldsymbol{X}_L$ (red) in the direction of $\nabla R(\mathbf{0})$, after evaluating if $|\cos(\theta)|< 1/\sqrt{2}$.

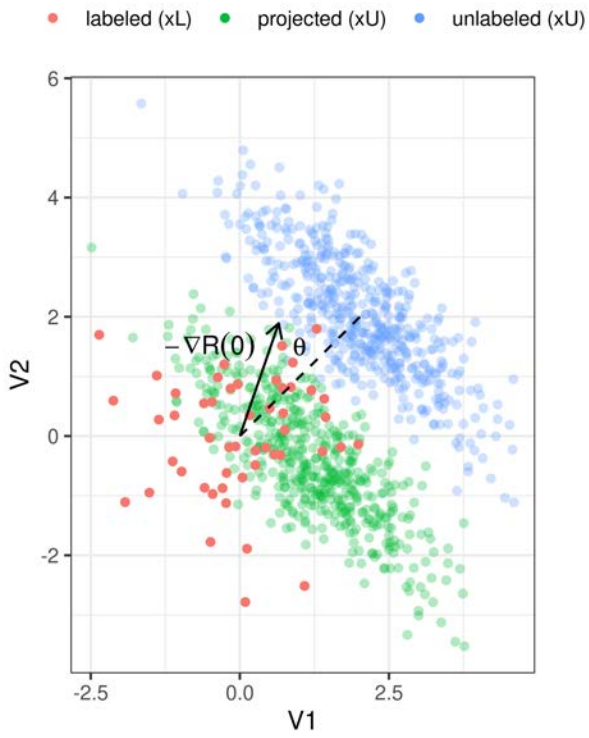


Figure 5.2: Example update of the unlabeled data in the direction of $-\nabla R(0)$ prior to computing the s²net solution.

5.4 The s2net package

This section describes the implementation and usage of R package **s2net**. Figure 5.3 summarizes the most important exported S3 and S4 classes. Method `fit` of S4 class `s2net` features the main functionality of this package, estimating the regression coefficients $\boldsymbol{\beta}$ as described in Section 5.3.

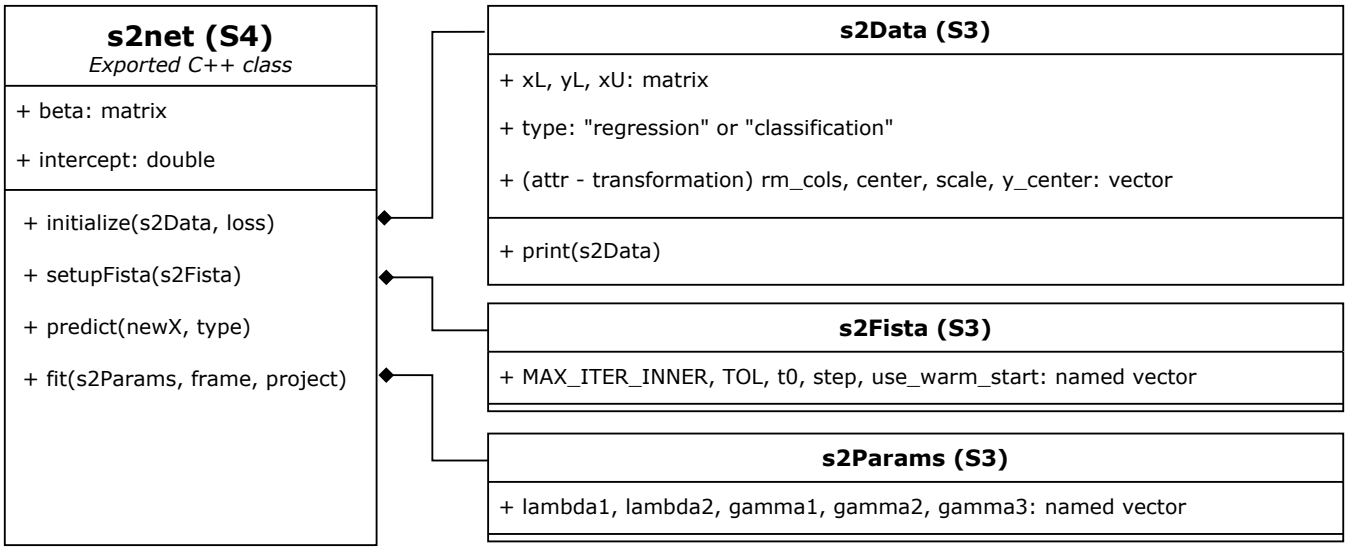


Figure 5.3: S4 and S3 classes in package **s2net**.

The S3 class `s2Data` contains the data to fit the model. Such data is supposed to be fixed for each model, and therefore `s2Data` is an independent class, that handles all the pre-processing and cross-validation set-up. The `"auto_mpg"` dataset Dua and Graff, 2017; Quinlan, 1993 is included for benchmark, with two semi-supervised set-ups described in Section 5.6. A typical usage would be the following.

```
R> library("s2net")
R> data("auto_mpg")
```

Function `s2Data` transforms the data for the semi-supervised framework. Using `model.matrix` from **stats**, factor variables are expanded to dummies, and additionally, constant columns are removed. This function also handles input errors, and impossible situations that might trigger errors, such as missing data or non-matching dimensions.

```
R> train = s2Data(auto_mpg$P2$xL, auto_mpg$P2$yL,
                  auto_mpg$P2$xU)
```

A nice feature of `s2Data` is that is can receive as input another `s2Data` object and process the new data according to the same transformation.

```
R> valid = s2Data(auto_mpg$P2$xU, auto_mpg$P2$yU,
                  preprocess = train)
```

S3 classes `s2Params` and `s2Fista` are simple wrappers for the model's hyper-parameters and the FISTA optimization set-up, respectively. There are two ways to fit a semi-supervised elastic-net using **s2net**, one is trough the function `s2netR`.

```
R> model = s2netR(train, params = s2Params(0.01, 0.01,
                                  0.01, 100, 0.1))
```

Alternatively, if we are fitting the semi-supervised elastic-net many times, using the same `train` data (for example, searching for the best hyper-parameters), then it is faster to use the S4 class `s2net` instead.

```
R> obj = new(s2net, train, 0)
R> obj$fit(s2Params(0.01, 0.01, 0.01, 100, 0.1), 0, 2)
R> obj$beta
```

```
               [,1]
[1,] -0.28700933
```

```
[2,]  0.04228791
[3,] -3.02580178
[4,]  0.61559052
[5,]  3.65723926
[6,]  0.71451133
[7,]  0.43040118
```

Depending on the choice to fit the model, there are several ways to predict the labels for new observations. The prediction type (linear predictor, probability, class) may be specified, otherwise it is automatically inferred from the input data. All of the following yield the same result.

```
R> ypred = predict(model, valid$xL)
R> ypred = obj$predict(valid$xL, 0)
R> ypred = predict(obj, valid$xL)
```

5.5 Simulations

In this section, we will investigate our proposed method s²net as a semi-supervised alternative to the elastic-net, when the underlying model is linear and sparse. The simulation designs discussed in this section are available as functions `simulate_groups` and `simulate_extra` exported from **s2net**.

To introduce the simulations and analysis in the rest of the paper, we make the following assumptions on the problem.

1. There are labeled samples $\boldsymbol{X}_L^s, \boldsymbol{y}_L^s$ from a source domain (e.g., measurements taken with an old instrument).
2. There are (some) labeled samples $\boldsymbol{X}_L^t, \boldsymbol{y}_L^t$ from a target domain (e.g., measurements taken with a new instrument or with different raw materials going into the production).
3. There are unlabeled samples $\boldsymbol{X}_U^t$ from a target domain (e.g., measurements taken with a new instrument, which are very expensive to label).
4. The objective is to construct a model that predicts the labels from the target domain.

In a recent paper, Oliver et al., 2018 establish some guidelines for comparing semi-supervised deep-based methods. Some of them, can be adapted to our framework of study as follows.

- *High quality supervised baseline*. The goal is to obtain better performance using $\boldsymbol{X}_U^t$ and $\boldsymbol{X}_L^s$ than what would be obtained using $\boldsymbol{X}_L^s$ alone. In our case, a natural baseline to compare against is s²net with $\gamma_1 = 0$ (as mentioned in Remark 1). We denote this supervised method as baseline. In addition, we also include the elastic-net (glmnet) from the R package **glmnet** (Friedman, Hastie, and Rob Tibshirani, 2010), to compare the naive estimation of baseline with the actual elastic-net solution. The hyper-parameters of each method were selected using random search, which has been shown to be superior to grid search (Bergstra and Bengio, 2012), with a total of 1000 random points. The hyper-parameters that minimized the loss in the validation data set, were selected as the best combination.

- *Varying the amount of labeled and unlabeled data*. To cover different scenarios in the simulations, we vary the number of unlabeled target samples n^t, in addition to the number of variables p.

- *Realistically small validation dataset*. This is related to the assumption 2 above, which is very important in order to have validation data. Without it, there is no clear and realistic way to select the hyper-parameters of the methods. It is possible to select the hyper-parameters using test data, but this would contradict the fact that in a real semi-supervised scenario, these labels are unknown. To make it feasible, we assume that the number of available samples for validation is small (in the rest of the simulations and data analyses, we fix it at 20).

Additionally, the following semi-supervised methods were included in the simulations: the safe semi-supervised semi-parametric model (s4pm) and fast anchor graph approximation (agraph) from M. V. Culp and Ryan, 2018, available in the R package **SemiSupervised**, the implicitly constrained semi-supervised least squares classifier (ICLS) (Krijthe and Loog, 2015), available in the R package **RSSL**, and the joint trained linear framework (JT) from M. Culp, 2013.

5.5.1 Two-group design

The simulation design is the following. Let

$$\boldsymbol{\Sigma}_{\rho}^{\sigma^2} = \begin{bmatrix} \sigma^2 & \rho & \dots & \rho \\ \rho & \sigma^2 & \dots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \dots & \sigma^2 \end{bmatrix}_{p/2 \times p/2}, \quad \boldsymbol{\Sigma}_{\rho_1, \rho_2}^{\sigma_1^2, \sigma_2^2} = \begin{bmatrix} \boldsymbol{\Sigma}_{\rho_1}^{\sigma_1^2} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\rho_2}^{\sigma_2^2} \end{bmatrix}_{p \times p}.$$

The source and target data rows are i.i.d., given by,

$$\boldsymbol{x}^s \sim N\left(\mathbf{0},\, \boldsymbol{\Sigma}_{.8,\,.01}^{1,\,.05}\right), \quad \boldsymbol{x}^t \sim N\left(\mathbf{0},\, \boldsymbol{\Sigma}_{.01,\,.5}^{.1,\,1}\right). \tag{5.16}$$

Figure 5.4 illustrates this simulation design using an example data set, with $p = 200$ variables, and $50, 200$ source and target observations, respectively.

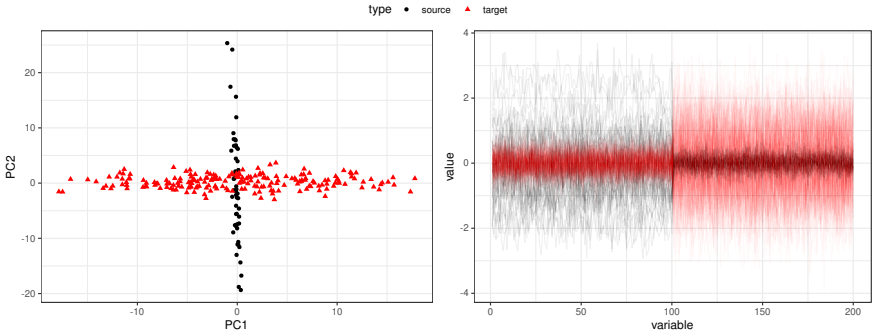


Figure 5.4: Example of simulated source/target data structure. Left panel shows the projected data on the first two principal components. Right panel compares the rows of $\boldsymbol{X}^s$ (black) and $\boldsymbol{X}^t$ (red).

To generate the responses for the source data $\boldsymbol{X}^s$, we have used a sparse coefficient vector, given by

$$\boldsymbol{\beta}_j = \begin{cases} 0 & j \notin I \\ 1 & j \in I \end{cases},$$

where I is the included variables' index set, that contains 5 random indexes between 1 and $p/2 - 1$ and 5 random indexes between $p/2$

and p. Therefore, there are 10 out of p "true" variables in the model. The target model's coefficients, however, are given by

$$\boldsymbol{\beta}_j^t = U_j \boldsymbol{\beta}_j, \text{ where } U_j \sim U[0.9,\ 1.1] \text{ for } j = 1, 2 \ldots p. \tag{5.17}$$

This introduces additional uncertainty in the target data, and models the case of a small change in the underlying coefficient vector for the new data.

The training set consists of labeled source data $\boldsymbol{X}^s_{train}, \boldsymbol{y}^s_{train}$ ($n^s = 50$ rows) and unlabeled target data $\boldsymbol{X}^t_{train}$ (n^t rows), whereas the validation set consists of labeled target samples $\boldsymbol{X}^t_{valid}, \boldsymbol{y}^t_{valid}$ (20 rows). A test data set $\boldsymbol{X}^t_{test}$, $\boldsymbol{y}^t_{test}$ (800 rows) was used to evaluate the performance of both methods, for each of 100 repetitions.

Linear response

In the regression case, the source labels were simulated as $\boldsymbol{y}^s = \boldsymbol{X}^s\boldsymbol{\beta} + \boldsymbol{\epsilon}^s$, where $\boldsymbol{\epsilon}^s \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I})$, with σ^2 such that the signal-to-noise ratio was 4. Analogously, $\boldsymbol{y}^t = \boldsymbol{X}^t\boldsymbol{\beta}^t + \boldsymbol{\epsilon}^t$.

Logistic response

For the classification case, to simulate the source data labels $\boldsymbol{y}^s$, we used a logistic model,

$$y^s|\boldsymbol{x}^s \sim \text{Ber}(p), \text{ with } p = \left(1 + \exp(-\boldsymbol{\beta}^\top \boldsymbol{x}^s)\right)^{-1}. \tag{5.18}$$

The target labels $\boldsymbol{y}^t$ were generated analogously, but using $\boldsymbol{\beta}^t$ instead – the noisy version of $\boldsymbol{\beta}$ given in (5.17).

Table 5.1 and 5.2 summarize the simulation results for linear and logistic responses, respectively. To evaluate the statistical significance of the difference between each method and baseline, we performed a Friedman rank test, followed by paired post-hoc tests (Pohlert, 2019). Significant improvements ($\alpha = 0.05$) with respect to baseline are shown in bold font. In these simulations, s²net achieves the best result in every scenario. In addition, the semi-supervised s4pm and JT are also superior to glmnet and baseline in some cases.

5.5.2 Extrapolation design

This simulation design is based on the one described in Ryan and M. V. Culp, 2015, but we varied the number of variables and unla-

	$n^t = 50$			$n^t = 250$		
	$p = 50$	$p = 100$	$p = 200$	$p = 50$	$p = 100$	$p = 200$
baseline	.59	.58	.69	.56	.53	.64
glmnet	.61	.60	.71	.58	.56	.66
s²net	**.55**	**.54**	**.65**	**.53**	**.51**	**.62**
s4pm	.71	.71	.75	.64	.57	.65
agraph	.86	.88	.99	.77	.76	.91
JT	.62	.61	.72	.56	**.53**	**.63**

Table 5.1: Average test MSE of the different methods (two-group design, linear response), over 100 simulations for each scenario. Significant improvements ($\alpha = 0.05$) with respect to baseline are shown in bold font.

	$n^t = 50$			$n^t = 250$		
	$p = 50$	$p = 100$	$p = 200$	$p = 50$	$p = 100$	$p = 200$
baseline	75.3	70.2	78.4	74.8	73.7	72.1
glmnet	75.9	71.8	78.3	73.6	74.9	71.7
s²net	**79.4**	**73.8**	**79.4**	**78.6**	**75.8**	**76.6**
s4pm	71.1	68.5	77.0	**75.0**	**74.8**	**75.8**
agraph	68.7	65.3	73.5	68.8	67.0	70.8
ICLS	60.4	54.2	57.6	60.4	55.8	53.6

Table 5.2: Average test area under the ROC curve (AUC, %) of the different methods (two-group design, logistic response), over 100 simulations for each scenario. Significant improvements ($\alpha = 0.05$) with respect to baseline are shown in bold font.

beled target samples, the shift, and included the logistic response case. The source data are simulated with i.i.d. rows given by,

$$\boldsymbol{x}^s \sim N(\mathbf{0}, 0.4\boldsymbol{I}) \tag{5.19}$$

Two possible coefficient patterns are considered,

$$\boldsymbol{\beta}^{(lucky)} = (\underbrace{1 \ \dots \ 1}_{5} \ \underbrace{-1 \ \dots \ -1}_{5} \ \underbrace{0 \ \dots \ 0}_{p-10})$$

and

$$\boldsymbol{\beta}^{(unlucky)} = (\underbrace{1 \ \dots \ 1}_{10} \ \underbrace{0 \ \dots \ 0}_{p-10})$$

There are three scenarios for the target data,

same $\boldsymbol{x}^t \sim N(\mathbf{0}, 0.4\boldsymbol{I})$ and $\boldsymbol{\beta} = 5/\sqrt{10}\boldsymbol{\beta}^{(lucky)}$

lucky $\boldsymbol{x}^t \sim N(\delta\boldsymbol{\beta}^{(unlucky)}, 0.4\boldsymbol{I})$, and $\boldsymbol{\beta} = 5/\sqrt{10}\boldsymbol{\beta}^{(lucky)}$

unlucky $\boldsymbol{x}^t \sim N(\delta\boldsymbol{\beta}^{(unlucky)}, 0.4\boldsymbol{I})$, and $\boldsymbol{\beta} = 5/\sqrt{10}\boldsymbol{\beta}^{(unlucky)}$

with δ the shift of the target with respect to the source domain. Figure 5.5 displays the three possible configurations for the data, projected in X_1 and X_6. In the "same" scenario, the source and target data follow the same distribution, and thus the direction of $\boldsymbol{\beta}$ is not important. In the "lucky" case, $\boldsymbol{\beta}$ is orthogonal to the shift (the source and target domains are different, but the response is less affected by the shift). In the "unlucky" case, however, $\boldsymbol{\beta}$ is parallel to the shift, and thus we expect the responses to be shifted as well. This "unlucky" scenario is more challenging, specially in the linear response case, where the bias in the estimation of $\boldsymbol{\beta}$ will impact the extrapolation.

For each repetition, the training data consist of $n^s = 50$ rows of labeled $\boldsymbol{X}^s_{train}, \boldsymbol{y}^s_{train}$, and varying n^t rows of unlabeled target data $\boldsymbol{X}^t_{train}$. The validation and test sets consist of 20 and 100 observations, respectively, from the target domain.

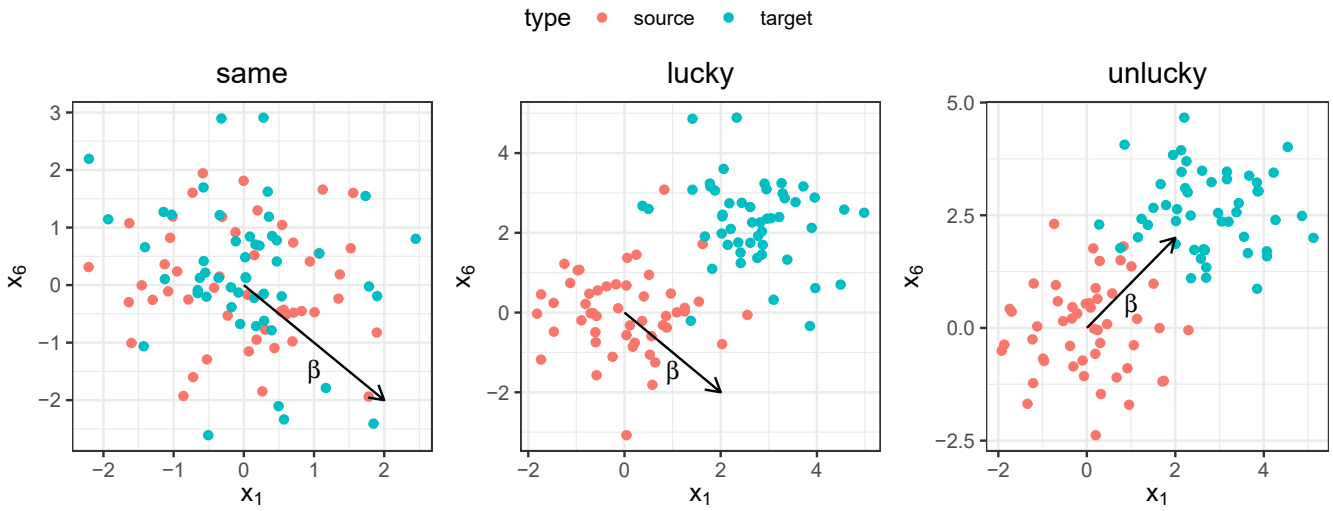


Figure 5.5: Simulated source/target data structure: Extrapolation design.

Linear response

The labels (for the source and target data, respectively) were simulated as $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, with $\epsilon_i \sim N(0, 2.5)$, for $i = 1, 2 \ldots n$. The number of features $p = 100$ and the shift $\delta = 1$.

Logistic response

The labels (source and target) are generated following a logistic response model,

$$y|\boldsymbol{x} \sim \mathsf{Ber}(p), \text{ with } p = \left(1 + \exp(-\boldsymbol{\beta}^\top \boldsymbol{x})\right)^{-1}. \tag{5.20}$$

The number of features $p = 20$ and the shift $\delta = 0.1$.

Tables 5.3 and 5.4 compare the simulations for linear and logistic responses, respectively. Table 5.4 displays a better performance for baseline, and s²net, suggesting that there is improvement when choosing the semi-supervised elastic-net framework. However, in the "unlucky" scenario of Table 5.3 (where the shift δ is in a direction parallel to the response direction of the labeled data), glmnet outperforms the other alternatives by a weak margin. The implementation of JT estimates the coefficients using glmnet, so they are expected to yield similar estimations when the supervised model prevails. However, glmnet and baseline are (in theory) solving the same optimization problem. We believe such differences are due to the way coefficients are actually estimated: baseline uses a block gradient descent optimization with soft-threshold, whereas glmnet is optimized using coordinate-gradient descent, with rules to discard predictors (Robert Tibshirani et al., 2012), and a correction factor in the $\boldsymbol{\beta}$ estimations. A detailed description of the differences between the naive and the elastic-net solution can be found in Bühlmann and Van De Geer, 2011. Nevertheless, the relative improvement of glmnet over s²net is less than 5% in this "unlucky" case, which is approximately the relative improvement of s²net over glmnet in the "same" and "lucky" scenarios.

5.6 Application to real data

The purpose of this section is to evaluate the performance of s²net in real data - based examples, and compare it with glmnet, s4pm, agraph,

	"same"		"lucky"		"unlucky"	
	$n^t = 50$	$n^t = 250$	$n^t = 50$	$n^t = 250$	$n^t = 50$	$n^t = 250$
baseline	5.58	5.71	5.85	5.74	61.6	48.0
glmnet	5.66	5.82	6.03	5.97	**56.5**	**46.1**
s²net	**5.56**	5.70	**5.75**	5.73	62.1	48.1
s4pm	6.23	6.21	**5.76**	5.81	120	86.7
agraph	6.21	6.39	6.09	6.06	**56.6**	71.6
JT	5.79	5.74	**5.58**	**5.69**	**59.1**	**47.7**

Table 5.3: Average test MSE of the different methods (extrapolation design, linear response), over 100 simulations for each scenario. Significant improvements ($\alpha = 0.05$) with respect to baseline are shown in bold font.

JT, ICLS, and the baseline (s²net with $\gamma_1 = 0$) in regression and classification tasks. An overview of the datasets used in this section is given in Table 5.5.

5.6.1 IDRC 2002 ``Shootout'' data

This data set was published in the International Diffuse Reflectance Conference in 2002, and it is currently available online[1]. It consists of the spectra from 655 pharmaceutical tablets measured with two spectrometers. The response variable is the proportion of active ingredient. As shown in Figure 5.6, there are differences in both instruments' measures ranging from $0.6 - 0.7$ μm and $1.7 - 1.8$ μm.

[1] http://eigenvector.com/data/tablets last access: 21-Oct-2019

	"same"		"lucky"		"unlucky"	
	$n^t = 50$	$n^t = 250$	$n^t = 50$	$n^t = 250$	$n^t = 50$	$n^t = 250$
baseline	74.7	74.9	76.2	74.0	77.5	75.5
glmnet	**74.7**	75.1	76.2	74.0	77.3	75.5
s²net	**76.3**	74.9	**76.3**	**74.1**	77.5	**75.6**
s4pm	74.2	74.4	74.6	74.1	73.6	74.2
agraph	74.4	73.0	74.3	72.8	75.7	72.9
ICLS	69.0	68.1	68.3	68.1	68.2	67.0

Table 5.4: Average test area under the ROC curve (AUC, %) of the different methods (extrapolation design, logistic response), over 100 simulations for each scenario. Significant improvements ($\alpha = 0.05$) with respect to baseline are shown in bold font.

Dataset	Lab. n^s (train)	Unlab. n^t (train)	Reg.	Class.	p
shootout	50	50	✓		575
auto-mpg (P1)	149	100	✓		9
auto-mpg (P2)	208	100	✓		7
spambase	100	500		✓	52

Table 5.5: Description of the data used in the analysis.

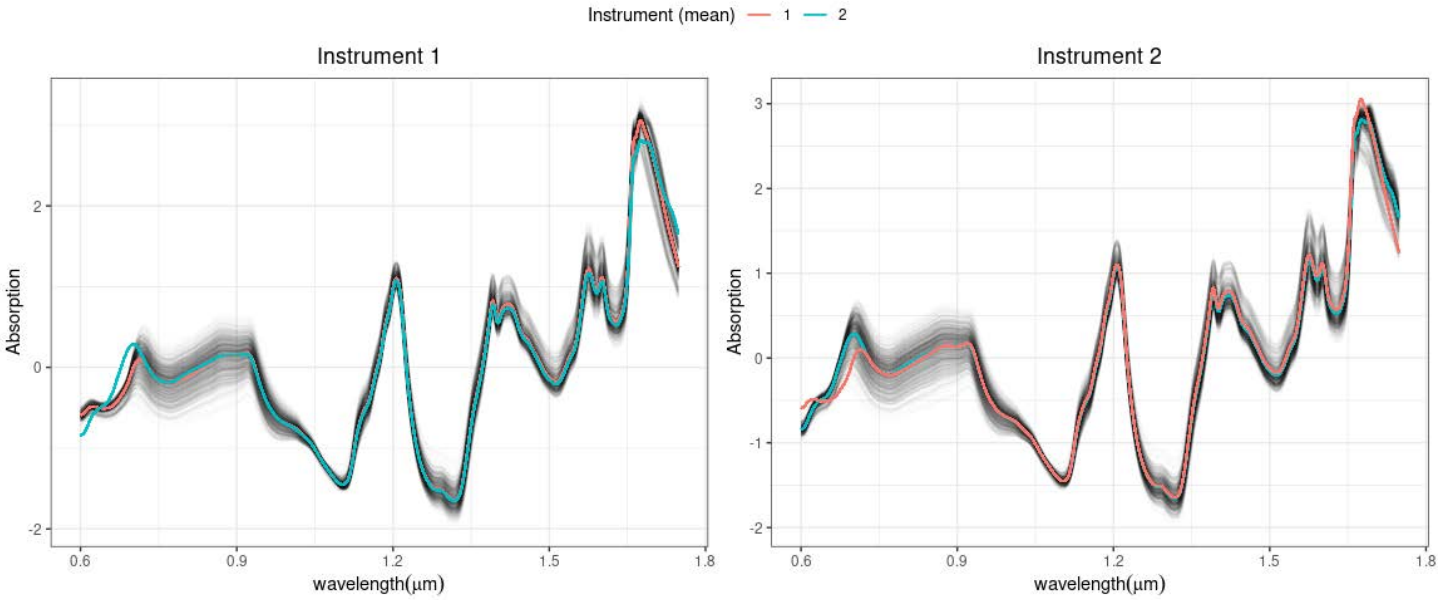


Figure 5.6: Spectra from 655 tablets (IDRC 2002 “Shootout” data) measured with two different instruments (left-right).

To illustrate the s²net methodology, we will assume that labels associated with measures from Instrument 1 are known, and we will investigate how predictions are affected when labels are predicted using measures from Instrument 2. For this purpose, the original data is randomly divided up into training, validation and test data sets, and this process is repeated 100 times. A total of 50 tablets are used as training labeled samples from Instrument 1 (source), whereas 50 measures from Instrument 2 (target) are used as training unlabeled samples. To select the best hyper-parameters for the methods, we separated a sample of 20 labeled measurements from Instrument 2 (target). The remaining tablets (unknown during the training process) are used as test samples from Instrument 2, in addition to the (already known) 50 measures used as training unlabeled samples. The response variable in the test data is used to compute prediction errors.

Figure 5.7 compares the distributions of the MSE obtained by the different algorithms in the test data set, for 100 repetitions. Notice that s²net is the one that achieves the smallest error mean and variance, but all the methods are very similar.

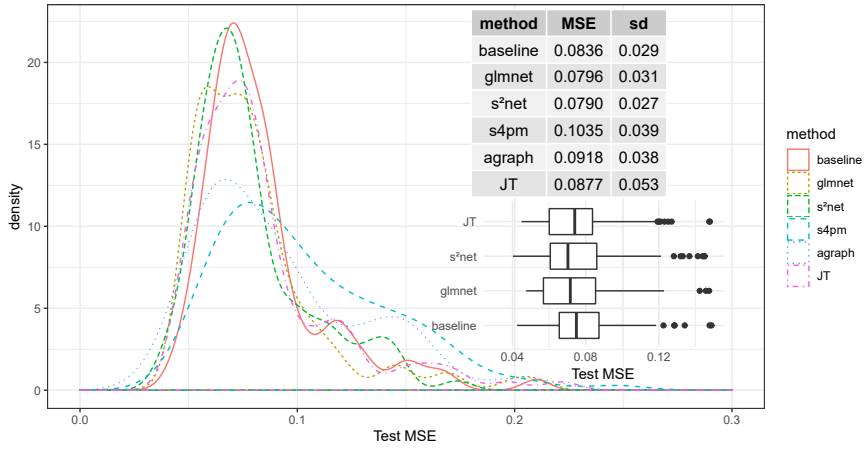


Figure 5.7: Density estimation of the (test) MSE of each method for 100 repetitions (*shootout* data).

5.6.2 Auto MPG dataset

This data set is available in the UCI repositories (Dua and Graff, 2017), and the original data was published by Quinlan, 1993. We have processed this data for the semi-supervised setting following the paper by Ryan and M. V. Culp, 2015. The first set-up (P1) separates source and target domains by variable `Domestic`, whereas the second set-up (P2) splits the data by variable `Cylinder <= 4`.

Figure 5.8 and 5.9 display the results for 100 repetitions (varying the validation and training target samples). As indicated by the distribution of the test error, and its mean in Figure 5.8, s²net clearly outperforms the other methods in the *auto-mpg (P1)* data. However, for the *auto-mpg (P2)* setting, the supervised glmnet is the one minimizing the test error. Apparently in this last case, the supervised methods have an advantage, and semi-supervised alternatives do a poor job (although, in theory, s²net and JT should *always* be better than baseline and glmnet, respectively – with the appropriate choice of hyper-parameters).

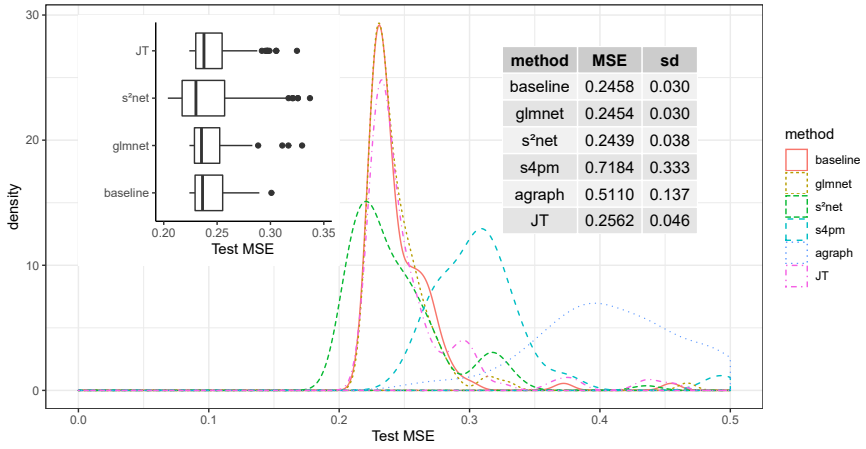


Figure 5.8: Density estimation of the (test) MSE of each method for 100 repetitions (*auto-mpg-P1* data).

5.6.3 Spambase data

This data set was collected by Hewlett-Packard Labs, and it is available at the UCI Repository of Machine Learning Databases (Dua and

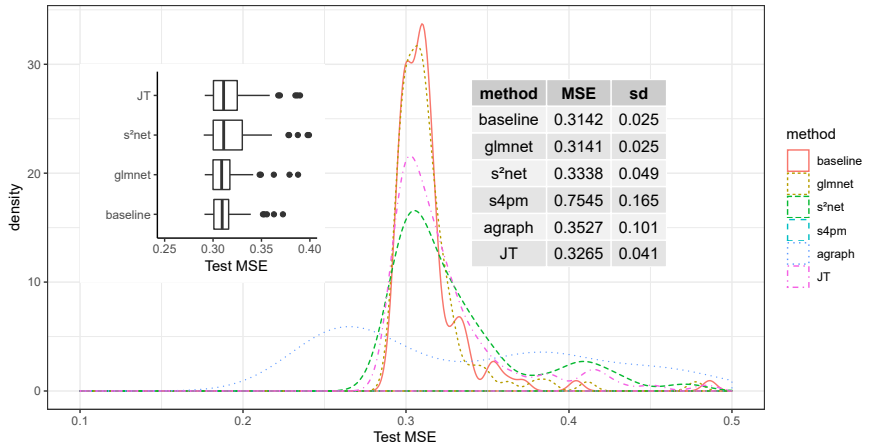
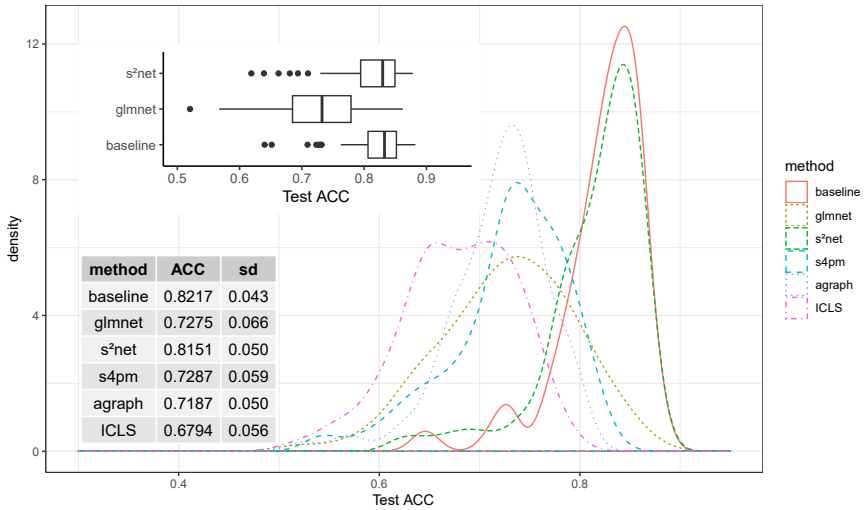


Figure 5.9: Density estimation of the (test) MSE of each method for 100 repetitions (*auto-mpg-P2* data).

Graff, 2017). It classifies 4601 e-mails as *spam* or *non-spam*. There are 57 explanatory variables indicating the frequency of certain words and characters in the e-mail. This data set was also studied by Kawakita and Kanamori, 2013 in a semi-supervised context. To adapt it to our semi-supervised set-up, we have split the data according to variable `Internet` (e-mails from the source domain containing the word internet in the body of the message). This partition yields to different balances of the response variable in the source and target domains, which suggests an additional complexity for the prediction.

Figure 5.10 displays the empirical distribution of the accuracy in the test set for the *spambase* data. We notice that s²net outperforms glmnet by a margin close to 10%. However – and this is why it is important to have a baseline method to compare – the supervised version of s²net performs very similarly (slightly better). In this case, there is no advantage in using the unlabeled data, but the optimization method itself that computes the coefficient estimations for s²net and baseline is showing good performance.



method	ACC	sd
baseline	0.8217	0.043
glmnet	0.7275	0.066
s²net	0.8151	0.050
s4pm	0.7287	0.059
agraph	0.7187	0.050
ICLS	0.6794	0.056

Figure 5.10: Density estimation of the (test) accuracy of each method for 100 repetitions (*spambase* data).

5.7 Conclusions

In this paper we have introduced s²net, a semi-supervised elastic-net for generalized linear models. Furthermore, we showed that s²net generalizes both JT and ExtJT, in addition to the supervised elastic-net for generalized linear models, and thus with the appropriate choice of hyper-parameters s²net defaults to the supervised solution if the unlabeled information is not relevant. Our method was tested using both real and synthetic data sets, and the experiments confirmed our approach as a good alternative to the elastic-net in the semi-supervised context.

We introduced a general optimization framework, that implements the FISTA algorithm to solve the elastic-net for a generic loss function. We believe our implementation can be easily adapted to solve other extensions of lasso, such as the group-lasso and the sparse-group lasso. In addition, we observed a relative improvement of using gradient-descent to optimize (5.6) with respect to coordinate-descent, demonstrated by the fact that our elastic-net baseline sometimes outperforms glmnet (Tables 5.1, 5.2, 5.3, and Figure 5.10).

The simulation design studied in Section 5.5.1 highlighted a scenario where s²net clearly outperforms all the other methods. We believe the increased performance is due to the fact that the underlying model's coefficient are different for the source and target domains. Since s²net uses the information in the unlabeled data (in contrast to the elastic-net), it can learn that change and adapt. Compared to other semi-supervised methods, s²net has the advantage of separating the shift from the covariance information, which adds flexibility to the model. Additionally, s²net brings nice properties of elastic-net to the semi-supervised framework, such as the sparsity in the solution.

Computational details

All the experiments in Sections 5.6 and 5.5 were conducted in the same HPC cluster[2], specifically 8 nodes with Intel(R) Xeon(R) CPUs E5-2680 v2, 128G RAM, running Linux 3.10.0 and R (3.6.1 – platform x86_64-conda_cos6-linux-gnu (64-bit) – Anaconda Inc.).

To select the hyper-parameters of all the methods we used random search with 1000 iterations. For s²net and baseline, we took $\lambda_1, \lambda_2 \sim 2^{U[-8,1]}$, and $\gamma_1, \gamma_3 \sim 2^{U[-8,1]}, \gamma_2 \sim 2^{U[-1,10]}$ (s²net). For glmnet and JT, $\alpha \sim U[0,1], \lambda \sim 2^{U[-8,1]}$, and $\gamma_1(\tau) \sim 2^{U[-8,1]}, \gamma_2(\gamma) \sim 2^{U[-1,10]}$ (JT). For s4pm and agraph, $lams, gams, hs \sim 2^{U[-8,1]}$, and for ICLS, $\lambda_1, \lambda_2 \in 2^{U[-8,1]}$. The code for the simulations and data analyses is available online[3].

Acknowledgments

We gratefully acknowledge the help provided by Prof. Mark Culp, who gave us access to the source code of the methods JT, s4pm and agraph, compared in our simulations and data analyses.

[2] www.hpc.dtu.dk

[3] https://github.com/jlaria/s2net-paper

Bibliography for Chapter 5

Amini, Massih-Reza and Patrick Gallinari (2002). "Semi-supervised logistic regression". In: *ECAI*, pp. 390–394.

Andries, Erik, John H Kalivas, and Anit Gurung (2019). "Sample and feature augmentation strategies for calibration updating". In: *Journal of Chemometrics* 33.1, e3080.

Beck, Amir and Marc Teboulle (2009). "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1, pp. 183–202.

Bergstra, James and Yoshua Bengio (2012). "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* 13.Feb, pp. 281–305.

Bühlmann, Peter and Sara Van De Geer (2011). *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media.

Chapelle, O., B. Schölkopf, and A. Zien (2010). *Semi-supervised Learning*. Adaptive computation and machine learning. MIT Press. ISBN: 9780262514125. URL: https://books.google.dk/books?id=zHAOQgAACAAJ.

Culp, Mark (2013). "On the Semisupervised Joint Trained Elastic Net". In: *Journal of Computational and Graphical Statistics* 22.2, pp. 300–318.

Culp, Mark Vere and Kenneth Joseph Ryan (2018). "SemiSupervised: Scalable Semi-Supervised Routines for Real Data Problems". In:

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml.

Eddelbuettel, Dirk and James Joseph Balamuta (Aug. 2017). "Extending extitR with extitC++: A Brief Introduction to extitRcpp". In: *PeerJ Preprints* 5, e3188v1. ISSN: 2167-9843. DOI: 10.7287/peerj.preprints.3188v1. URL: https://doi.org/10.7287/peerj.preprints.3188v1.

Eddelbuettel, Dirk and Romain François (2011). "Rcpp: Seamless R and C++ Integration". In: *Journal of Statistical Software* 40.8, pp. 1–18. DOI: 10.18637/jss.v040.i08. URL: http://www.jstatsoft.org/v40/i08/.

Eddelbuettel, Dirk and Conrad Sanderson (Mar. 2014). "RcppArmadillo: Accelerating R with high-performance C++ linear algebra". In: *Com-*

putational Statistics and Data Analysis 71, pp. 1054–1063. URL: http://dx.doi.org/10.1016/j.csda.2013.02.005.

Friedman, Jerome, Trevor Hastie, and Rob Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent". In: *Journal of statistical software* 33.1, p. 1.

Genkin, Alexander, Anirvan M Sengupta, and Dmitri Chklovskii (2019). "A Neural Network for Semi-supervised Learning on Manifolds". In: *International Conference on Artificial Neural Networks*. Springer, pp. 375–386.

Ji, Xu, João F Henriques, and Andrea Vedaldi (2019). "Invariant information clustering for unsupervised image classification and segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9865–9874.

Kawakita, Masanori and Takafumi Kanamori (2013). "Semi-supervised learning with density-ratio estimation". In: *Machine learning* 91.2, pp. 189–209.

Krijthe, Jesse H and Marco Loog (2015). "Implicitly constrained semi-supervised least squares classification". In: *International symposium on intelligent data analysis*. Springer, pp. 158–169.

Larsen, Jacob Søgaard et al. (2020). "Semi-supervised covariate shift modelling of spectroscopic data". In: *Journal of Chemometrics*.

Mächler, M (2012). "Accurately Computing log (1- exp (-| a|))". In: *URL http://cran. r-project. org/web/packages/Rmpfr/vignettes/log1mexp-note. pdf*.

Oliver, Avital et al. (2018). "Realistic evaluation of deep semi-supervised learning algorithms". In: *Advances in Neural Information Processing Systems*, pp. 3235–3246.

Pedregosa, Fabian and Bart van Merrienboer (2019). *How to Evaluate the Logistic Loss and not NaN trying*. http://fa.bianp.net/blog/2019/evaluate_logistic/. (Visited on 09/27/2019).

Pohlert, Thorsten (2019). *PMCMRplus: Calculate Pairwise Multiple Comparisons of Mean Rank Sums Extended*. R package version 1.4.2. URL: https://CRAN.R-project.org/package=PMCMRplus.

Quinlan, J Ross (1993). "Combining instance-based and model-based learning". In: *Proceedings of the tenth international conference on machine learning*, pp. 236–243.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Ryan, Kenneth Joseph and Mark Vere Culp (2015). "On semi-supervised linear regression in covariate shift problems". In: *The Journal of Machine Learning Research* 16.1, pp. 3183–3217.

Sanderson, Conrad and Ryan Curtin (2016). "Armadillo: a template-based C++ library for linear algebra". In: *Journal of Open Source Software* 1.2, p. 26.

– (2019). "Practical Sparse Matrices in C++ with Hybrid Storage and Template-Based Expression Optimisation". In: *Mathematical and Computational Applications* 24.3, p. 70.

Tan, Ben, Junping Zhang, and Liang Wang (2011). "Semi-supervised elastic net for pedestrian counting". In: *Pattern Recognition* 44.10-11, pp. 2297–2304.

Tibshirani, Robert et al. (2012). "Strong rules for discarding predictors in lasso-type problems". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 74.2, pp. 245–266.

Zou, Hui and Trevor Hastie (2003). "Regression shrinkage and selection via the elastic net, with applications to microarrays". In: *Journal of the Royal Statistical Society: Series B. v67*, pp. 301–320.

Conclusions

This thesis has filled relevant gaps in the literature, exploring several penalized regression methods for generalized linear models. Two significant issues are identified in the Sparse Group Lasso regularization context, namely the correct hyper-parameter selection and the initial clustering specification.

To address the first issue, Chapter 2 presented the iterative Sparse Group Lasso, a coordinate descent algorithm for hyperparameter selection in the Sparse Group Lasso regularization context. There are in fact two versions of the iSGL, one that finds hyperparameters for each group, and a simpler version that only addresses the global penalties associated with the Lasso and Group Lasso terms. The advantages of iSGL were illustrated in a gene-expression dataset in Chapter 2, but also with a biomedical case study in Chapter 3, carried out as part of a collaboration with Hospital General Universitario Gregorio Marañón.

Although Chapter 3 partially approached the problem of variable clustering in the Sparse Group Lasso, it motivated a deep research into this second issue, provided in Chapter 4. The Group Linear Algorithm with Sparse Principal decomposition is, formally, an extension of the Sparse Group Lasso (zeroing one hyperparameter gives exactly the Sparse Group Lasso solution without groups). GLASP, unlike Sparse Group Lasso, does not require a prior specification of clusters

between the variables. Besides, GLASP can be considered as a supervised variable clustering algorithm. The advantages of GLASP were illustrated using both real and simulated data, and its source code is available for the community to use.

A contribution of regularization methods to transfer learning models was highlighted in Chapter 5, where the Semi-Supervised Elastic Net was compared with other state-of-the-art algorithms for explainable variable selection in the semi-supervised learning context. Although this context is different from the supervised one, Chapter 5 showed that the Semi-Supervised Elastic Net can be solved with the same tools used for supervised regression problems with regularization terms.

The source code of the contributions of this thesis is available as three different R packages: **sglfast**[4], **s2net**[5] and **glasp**[6].

Besides its main contributions, this thesis has identified areas for further research, which can be summarized as follows.

1. To explore extensions of the iterative Sparse Group Lasso to deal with the regularization parameter selection of GLASP. Chapter 4 addressed the penalty parameter selection using available solutions based on random search and bayesian optimization, but an iterative approach could lead to better results.

2. To study the methodologies discussed in this thesis when the response variable is multivariate. In this case, $\boldsymbol{\beta}$ is a matrix, and thus the groups are both row-wise and column-wise. A Sparse Group Lasso with multivariate response was discussed by Vincent and Hansen, 2014, and Multivariate Cluster Elastic Nets have been proposed by Price and Sherwood, 2017 and, recently, Ren, Kang, and Lu, 2020. An extension of GLASP in this direction seems reasonable.

3. To analyze practical scenarios of the algorithms discussed here. One emerging field of application is the financial data, where variable selection and variable clustering techniques would provide deep insight into underlying structures. Recent applica-

[4] https://github.com/jlaria/sglfast
[5] https://cran.r-project.org/package=s2net
[6] https://github.com/jlaria/glasp

tions of variable selection techniques in finance can be found in Uniejewski, Marcjasz, and Weron, 2019 and Hosaka, 2019.

4. To examine the possible integration of GLASP and the algorithm described in Chapter 3 (instead of iSGL). This combined approach would be a novel ensemble method in the high dimensional context.

5. To model survival after treatment of patients in a biomedical study in collaboration with Hospital General Universitario Gregorio Marañón. The data investigated in Chapter 3 is part of a broader study in which, besides measuring the binary response to treatment, the time until relapse is also quantified. This corresponds to high-dimensional right-censored survival data.

6. To improve with a rigorous theoretical justification the definition of the variable importance index introduced in Chapter 3, and to compare this technique with other state-of-the-art methods that weight variables based on their relative importance in the models.

7. To extend the iterative Sparse Group Lasso, the Semi-Supervised Elastic Net and GLASP to other risk functions, covering other generalized linear models apart from linear and logistic regression. In this sense, GLASP is superior to the rest because it covers linear, logistic, and survival (including right-censored) models. Besides, both **glasp** and **s2net** packages have been implemented following a modular approach – the optimization and the model specification are implemented in separate classes – and thus adding a new risk function should be straightforward.

8. To investigate the implementation of the methods discussed in this thesis in a high performance computing context. The major limitation of GLASP is that its performance is severely affected by the number of training observations. Empirically, we noticed that sample sizes in the order of thousands when the number of variables was also large, penalized the computational performance significantly. However, the block-gradient descent method to solve the optimization (FISTA) could be paralleled by groups. Moreover, to this date, not even the Sparse Group

Lasso has been implemented in a scalable framework, and it is possible because Elastic Net is available in a parallel framework (through Spark's **MLib**). The underlying optimization algorithm that the Elastic Net model pipeline uses in this case is described in Andrew and Gao, 2007.

This thesis set out to find new methods for variable selection in generalized linear models, and delivered its primary objective. It has offered a framework for the exploration of modern explainable machine learning and statistical algorithms, paving the way for future research, which is already underway.

Bibliography for Conclusions

Andrew, Galen and Jianfeng Gao (2007). "Scalable training of L 1-regularized log-linear models". In: *Proceedings of the 24th international conference on Machine learning*, pp. 33–40.

Hosaka, Tadaaki (2019). "Bankruptcy prediction using imaged financial ratios and convolutional neural networks". In: *Expert systems with Applications* 117, pp. 287–299.

Price, Bradley S and Ben Sherwood (2017). "A cluster elastic net for multivariate regression". In: *The Journal of Machine Learning Research* 18.1, pp. 8685–8723.

Ren, Sheng, Emily L Kang, and Jason L Lu (2020). "MCEN: a method of simultaneous variable selection and clustering for high-dimensional multinomial regression". In: *Statistics and Computing* 30.2, pp. 291–304.

Uniejewski, Bartosz, Grzegorz Marcjasz, and Rafał Weron (2019). "Understanding intraday electricity markets: Variable selection and very short-term price forecasting using LASSO". In: *International Journal of Forecasting* 35.4, pp. 1533–1547.

Vincent, Martin and Niels Richard Hansen (2014). "Sparse group lasso and high dimensional multinomial classification". In: *Computational Statistics & Data Analysis* 71, pp. 771–786.

Cover designed by starline / Freepik

Universidad
Carlos III de
Madrid

Av. Universidad 30
28912 Leganés, Madrid

www.uc3m.es