Non-smooth optimization for the estimation of cellular immune components in a tumoral environment

Quentin Klopfenstein

UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

INSTITUT DE MATHÉMATIQUES
DE BOURGOGNE

THÈSE DE DOCTORAT DE L'ÉTABLISSEMENT
UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ
PRÉPARÉE A L'INSTITUT DE MATHÉMATIQUES DE BOURGOGNE

École doctorale n^o 553: Carnot-Pasteur

Non-smooth optimization for the estimation of cellular immune components in a tumoral environment

Optimisation non-lisse pour l'estimation de composants immunitaires cellulaires dans un environnement tumoral

THÈSE

Pour l'obtention du titre de

DOCTEUR EN SCIENCES
SPÉCIALITÉ MATHÉMATIQUES APPLIQUÉES

Présentée par

Quentin KLOPFENSTEIN

Thèse présentée devant le jury composé de

Chloé-Agathe AZENCOTT	Mines ParisTech	Examinatrice
Hervé CARDOT	Université de Bourgogne Franche-Comté	Co-directeur
Jalal FADILI (Président)	ENSICAEN	Examinateur
Enrico GLAAB	Université du Luxembourg	Examinateur
Sophie LAMBERT-LACROIX	Université Grenoble Alpes	Rapportrice
Jérôme MALICK	CNRS, Université Grenoble Alpes	Rapporteur
Nelly PUSTELNIK	CNRS, ENS de Lyon	Examinatrice
Samuel VAITER	CNRS, Université de Bourgogne Franche-Comté	Directeur

To my wife, Emily

Remerciements/Acknowledgements

Je débute ces remerciements par mes directeurs de thèse, Samuel et Hervé. Je vous remercie d'avoir accepté de construire ce sujet de thèse il y a 3 ans de cela. Hervé, je tiens à te remercier parce que depuis le master MIGS tu m'as souvent aidé et conseillé dans les choix et les décisions à prendre. Encore aujourd'hui tu continues à me donner de précieux conseils pour ce qui m'attend après la thèse. Samuel, je te remercie pour ta disponibilité, ton écoute ainsi que ta capacité à m'encourager et à me motiver. J'ai beaucoup apprécié travailler avec toi et appris énormément sous ta direction. Je suis très heureux d'avoir été ton premier doctorant et je te suis reconnaissant pour l'ensemble de ces trois années.

Je tiens à remercier Jérôme Malick et Sophie Lambert-Lacroix d'avoir accepté d'être les rapporteurs de cette thèse. Merci pour votre lecture attentive et vos conseils pour l'amélioration de ce manuscrit. I would like to thank also Nelly Pustelnik, Enrico Glaab, Jalal Fadili and Chloé-Agathe Azencott for being part of the jury.

Au cours de ces années de thèse, j'ai eu la chance de faire quelques rencontres : Nicolas je garderai un très bon souvenir de ma première conférence à Guidel dans notre petit cottage et les soirées passées à jouer à Overcooked, ainsi que de tous les autres moments que nous avons pu passer ensemble. Merci également à toi Cindy d'avoir partagé ce bureau pendant ces trois années, pour les pauses café et les discussions en tous genres.

Au cours de cette thèse j'ai aussi eu la chance de pouvoir collaborer avec plusieurs personnes et notamment Quentin. Quentin, je te remercie pour tout ce temps que l'on a passé ensemble à coder, réfléchir aux preuves, à la rédaction des articles et tant d'autres choses. J'ai grandement apprécié travailler avec toi aussi bien sur le côté scientifique que sur le plan humain. Je te remercie pour tout ce que j'ai pu apprendre à tes côtés et ce que nous avons pu découvrir ensemble. J'espère que nous pourrons continuer à collaborer dans les prochaines années. Merci également à Joseph Salmon, Alexandre Gramfort, Mathieu Blondel et Mathurin Massias pour les différents projets sur lesquels nous avons pu travailler ensemble. Merci pour vos conseils et votre supervision.

Je tiens également à te remercier Samuel Herrmann pour tes précieux conseils et ces temps de discussion que tu m'as accordé tout au long de la thèse. A mes débuts à l'université, tu as d'abord été un professeur que j'ai beaucoup apprécié et puis aujourd'hui un ami et un collègue avec qui j'aime discuter et échanger. En partageant ton expérience et ton vécu, tu m'as très souvent encouragé.

Merci également à Caroline et François du CGFL de m'avoir proposé ce sujet de mémoire

qui portait sur l'estimation de cellules immunitaires il y a déjà 4 ans. Je suis très reconnaissant pour tout ce que vous m'avez appris pendant ces 18 mois qui ont précédé cette thèse. Une pensée pour mes anciens collègues de la PTBC également qui m'ont donné goût à la recherche dans le monde biomédical: Valentin, Emeric, Corentin, Marion et Sylvain.

En faisant l'ensemble de mes études sur Dijon, j'ai eu l'occasion d'être entouré par beaucoup de personnes avec qui j'aime partager de bons moments. Merci à ma famille de l'EPEDE pour votre soutien, votre amitié et pour tout ce que nous avons vécu ensemble au cours de ces années. Merci tout particulièrement à Julian et Julie et à vos deux garçons pour votre amitié sans failles, votre soutien et tous les bons moments partagés ensemble. Vous m'avez très souvent encouragé et soutenu au cours de l'avancement de cette thèse.

Cette thèse n'aurait jamais été possible sans le soutien de ma famille que j'aime du plus profond de mon coeur. Tout d'abord merci à mes grands-parents Lionel, Violaine, Gilbert et Jacqueline pour tout ce que vous m'avez donné et apporté tout au long de ces années. Merci à mes deux soeurs, Jessica et Manon, ainsi qu'à mon petit frère, Benjamin d'avoir toujours été là pour moi quand j'en avais besoin et d'écouter (parfois) avec attention ce qui me passionne dans ce métier de chercheur. J'en viens aux personnes à qui je dois énormément et que je tiens particulièrement à remercier, mes parents André et Magali. Merci parce que vous m'avez donné l'opportunité de faire des études et que vous m'avez toujours soutenu dans mes projets. Merci pour votre éducation, votre amour et les sacrifices réalisés afin que je puisse soutenir cette thèse. Cette réussite aujourd'hui est en grande partie grâce à vous et pour vous.

I would like to thank my parents-in-law, Earl and Karen for their unconditional love and support. You are both examples to me in the way you are and what you do. This has motivated me to pursue this thesis until its end. Finally, last but not least, thank you Emily, my dear wife, for your love and your support through these three years. You have always been by my side, pushing me when I needed to, encouraging me as well. Thank you for sharing my life and for being a great mom to our little girl Lucy. Both of you are my family that I cherish, I appreciate every day that I spend with you two.

Je termine ces remerciements avec cette pensée:

Soli Deo Gloria

Abstract

This thesis is concerned with the estimation of different immune cells proportions present in a tumor from genomic data. This estimation process boils down to a linear inverse problem. The state-of-the-art method is based on the Support Vector Regression estimator but does not take into account the constraints related to the estimation of proportions. We propose an estimator based on the Support Vector Regression which is constrained to be a vector of proportions directly in the estimation process and more generally to meet polyhedral constraints. We study the resolution of the optimization problem using an optimization algorithm called coordinate descent. Then, we propose a method to automatically select the hyperparameters in order to avoid using the grid-search method which can be applied for any separable non-smooth optimization problem arising in machine learning. This method is a first-order method that relies on automatic differentiation techniques to compute the gradient with respect to the hyperparameters given a measure of performance. Finally, we show how the developed tools can be used to solve the inverse problem mentioned above and the improvements obtained in comparison to the state-of-the-art methods.

Key words: Inverse problem, Non-smooth optimization, Coordinate descent, Hyperparameters selection, Automatic differentiation, Support Vector Regression, Biomedical application

Résumé

Cette thèse s'intéresse à l'estimation de la proportion des différents types de cellules immunitaires présents dans une tumeur à partir de donnés génomiques. Cette estimation revient à résoudre un problème inverse linéaire bruité. Nous proposons un estimateur basé sur la Support Vector Régression qui intègre dans le processus d'estimation les contraintes liées à l'estimation de proportions et de manière plus générale à toutes contraintes polyédriques. Nous étudions la résolution du problème d'optimisation sous-jacent en utilisant une méthode d'optimisation appelée la descente par coordonnées. Par la suite, nous proposons une méthode de sélection automatique des hyperparamètres pour s'affranchir de la méthode de sélection sur une grille qui est coûteuse dans le cas de la Support Vector Régression par exemple. Cette méthode est une méthode du premier ordre qui utilise les techniques de différentiation automatique afin de calculer le gradient en fonction des hyperparamètres et permet d'obtenir les paramètres optimaux étant donné un critère de performance pour tout problème d'optimisation non-lisse et séparable. Enfin, nous montrons comment les outils développés au cours de cette thèse peuvent être appliqués pour résoudre le problème inverse mentionné ci-dessus et les gains obtenus par rapport aux méthodes état de l'art.

Mots clés: Problème inverse, Optimisation non-lisse, Descente par coordonnées, Sélection de paramètres, Différentiation automatique, Support Vector Régression, Application biomédicale

Contents

1 Introduction

Contents

In 1961, William G. Cochran, professor of statistics at Harvard University, stated that "Mathematical reasoning can contribute to biology in many ways. It may enable the biologist to obtain quantitative estimates in situations in which his information has previously been qualitative" (Cochran, 1961). Nowadays, his statement is true more than ever: mathematics is a large part of biomedical research. For example, Statistics have been applied to clinical trials for many years and give key reasoning to assess the efficiency of a treatment over another one or over a placebo effect. Cells detection on an image, differential expression analysis for genes, survival analysis are only a few examples that highlight the importance of mathematics in the medical field.

These past years, the exponential growth of the volume of data generated by the biomedical field (Luo et al., 2016) have forced biologists and physicians to face many challenges

related to the storage, the processing and the analysis of this flow of information. Tools that would help them find relevant information in the mass of available data are crucial to continue the improvement of current medicine.

In this thesis, our goal is to contribute to this interaction between mathematics and biomedical research. The mathematical questions and challenges faced throughout this work are deeply rooted to a specific application in cancer research that we are detailing now.

1.1 Medical context

Cancer is a major problem of our society, with new statistics giving more than 17 millions new cases worldwide in 2020 (Sung et al., 2021). This disease is the second leading cause of death in the world. Behind this general word *cancer* hides about 100 different types of cancers located in different organs; each of them having its specificities. Recent medical research in the field have lead to this general conclusion that two patients' cancers will never be the same even if they are located in the same organ (Krzyszczyk et al., 2018). This observation is key; for a long time cancers were treated similarly if located at the same place. This generic type of treatment was sometimes ineffective and not adapted to each patient.

The development of genomic sequencing technologies have paved the way for *personalized medicine* sometimes called *precision medicine*. The idea is simple, considering each tumor as unique and find the treatment that suits the best to the patient. This need of precision medicine became even clearer with the development of the immunotherapy treatment.

Immunotherapy. There exists several possible treatments for a patient suffering from cancer: surgery, chemotherapy, targeted therapy, hormone therapy, immunotherapy to only cite a few. This last one, immunotherapy, is recent and has been described in 2013 as "*Breakthrough of the year*" by Couzin-Frankel (2013), leading to a great hope in the fight against cancer. The idea behind immunotherapy is to use the patient's immune system to fight cancer. Indeed, immune cells can be found in and around a tumor showing a immune response towards those abnormal cancer cells forming the tumor. Unfortunately, these cancer cells are often able to *hide* from the immune cells by changing their genetic information to look like normal cells. They are also able to turn off the immune response in different ways. Despite these facts, several studies have showed that a tumor infiltrated with immune cells often has a better prognosis than a tumor without immune cells (Crisc-

itiello et al., 2016; Stanton et al., 2016; Glaire et al., 2019; Reichling et al., 2020). From these conclusions, medical researchers have proposed different treatments that would enhance the immune response against the tumor.

Early on, this new treatment seemed to be very promising and encouraging results confirmed the expectations (for lung cancers for example (Reck et al., 2016)). However a few years later, the results are mixed. Immunotherapy can be very effective in some types of cancer but several studies showed that only a few patients were responding to the treatment. In other words, the patients for which the treatment was working had a good benefit from it but a majority of patients did not respond to it (Ventola, 2017). Today the challenge is to understand the reasons of this observation and being able to characterize, before starting the treatment, which patient will be able to benefit from immunotherapy. To answer these questions, there is a need to better understand the tumor composition and its environment. A tumor is not only composed of cancer cells but it also includes non-malignant cells, secreted proteins and blood vessels (Ansell and Vonderheide, 2013). Among these non-malignant cells, there exists a large number of immune cells that can be found inside a tumor as illustrated in Figure 1.1. These different types of immune cells have different roles regarding the growth and the spread of a tumor. For example, the quantity of CD8 T-cells inside a tumor was shown to be associated with longer survival rate (Wahlin et al., 2007) and leads to a better prognosis. On the contrary other types of immune cells are associated to a poorer survival rate (Barnes and Amir, 2017). These findings lead to the hypothesis that the composition of the tumor influences the efficiency of the immunotherapy treatment. Today, several questions remain unanswered about the interactions between these cells, their influence on the tumor growth and what would be a *favorable* environment for immunotherapy. One of the keys to answer these questions is to be able to describe precisely the tumoral environment by quantifying its cells.

Quantifying cells. There exist different methods to quantify the types of cells that can be found inside a tumor. Three of them will be presented here to better understand the strengths and weaknesses of these methods.

Flow cytometry is commonly used in medical research to count cells and identify them. The idea behind the process is to bring the cells one by one in front of a laser. Then captors will measure the scattered light coming out of the cells. Cells can be marked with fluorescent markers to be easily identifiable. The major drawbacks of this technique are the limited number of populations that can be studied at the same time and the complexity of the analysis needed to interpret the data.

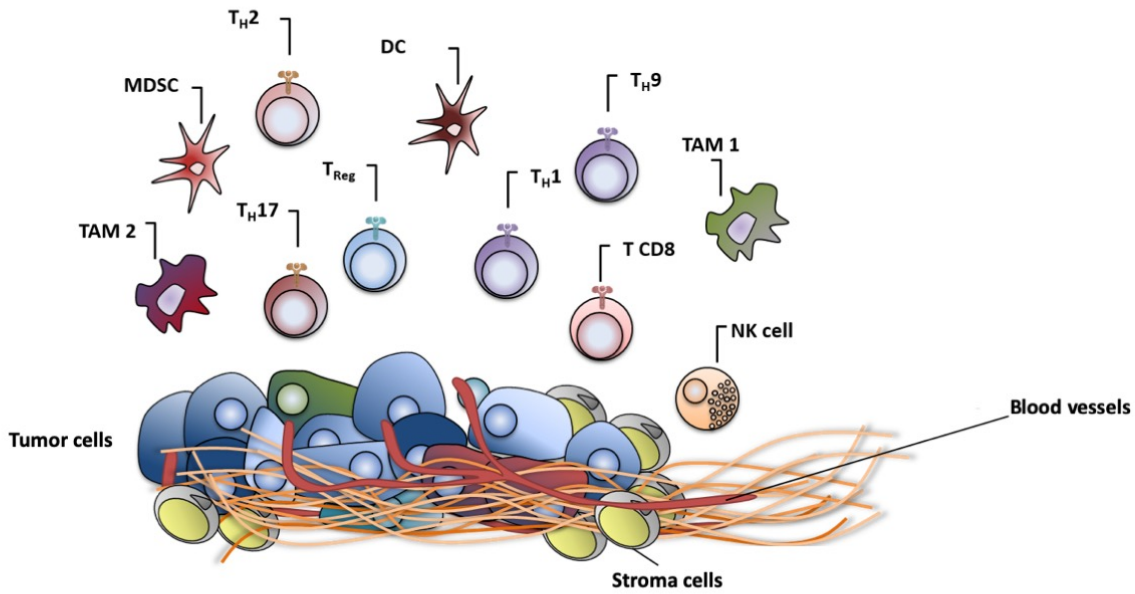

Figure 1.1 – **Tumor composition.** Schematic composition of a tumor showing the different types of cells that can be found inside it. A tumor is not only composed of tumor cells but also of blood vessels, stroma cells and immune cells that are of interest for the immunotherapy treatment. The cells on the top of the figure are the immune cells, illustrating the existence of different types of immune cells within a tumor, each of them has a specific role in the immune system.

Another possible method is the immunohistochemestry. The first step is to prepare a slice of tissue that needs to be analyzed coming from a biopsy. Then the idea is to stain the cells of interest by targeting a specific protein with an antibody. It will create a reaction and stain the targeted type of cells (see the cells in brown in Figure 1.2). Afterwards, the slide containing the tissue and the stained cells is digitalized and then the image is analyzed. Afterwards, the task is to detect the stained cells and count them on the image. This technique is widely used in the cancer medical field for different purposes: disease diagnosis, research and drugs development. This method suffers from the same drawback as the flow cytometry, it is very limited in the number of cell populations that can be studied at the same time. As an answer to these limitations, scientists have developed computational methods based on genomic data.

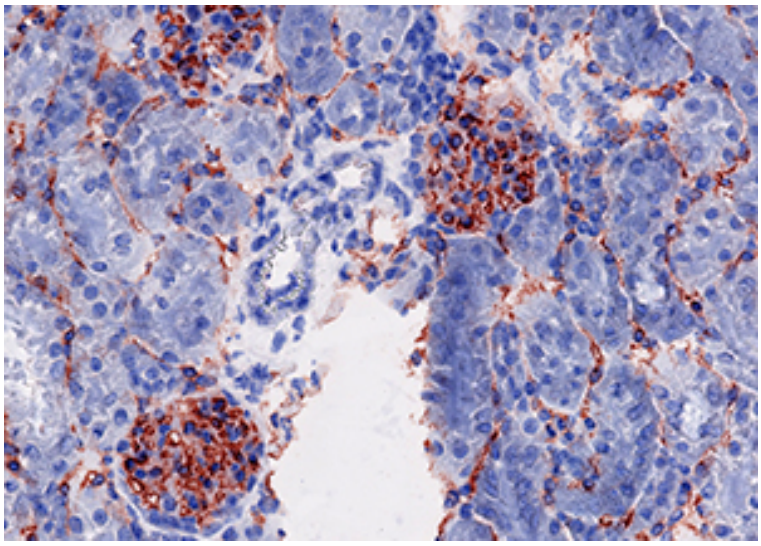

Figure 1.2 – **Immunohistochemestry.** Cells of interest are stained in brown whereas other cells appear blue. (*source:* https://resources.rndsystems.com/)

Genomic data refers to the data studying the genome, *i.e.*, the complete set of genes (DNA) of an organism. To be more precise, the genomic data mostly used in the rest of this work is called transcriptome which is the complete set of all RNA transcripts in an individual or a cell. The first transcriptome studies were based on the microarray technology (or biochip). A microarray is a glass slide on which DNA/RNA molecules are fixed in a predefined location. To keep it simple, one can have in mind that each location uniquely corresponds to a gene. Then, the studied DNA/RNA will bound to a specific location because of the complementarity of the nucleic acid sequences. Finally, each location is excited by a laser and scanned at different wavelengths. An illustration of the process can be found in Figure 1.3.

To be usable, the data is preprocessed using different normalization techniques (see Ni et al. (2008); Rao et al. (2008) for more details on normalization techniques and comparison) that will not be detailed here, it goes beyond the scope of this thesis. However, at the end of the process, genomic data can be represented by a matrix containing the gene expression of different cells or individuals. For example, we can consider that the gene expression matrix has n genes and p different cells for which the transcriptome was studied. The value X_{ij} then represents the expression value of the gene i in the cell j.

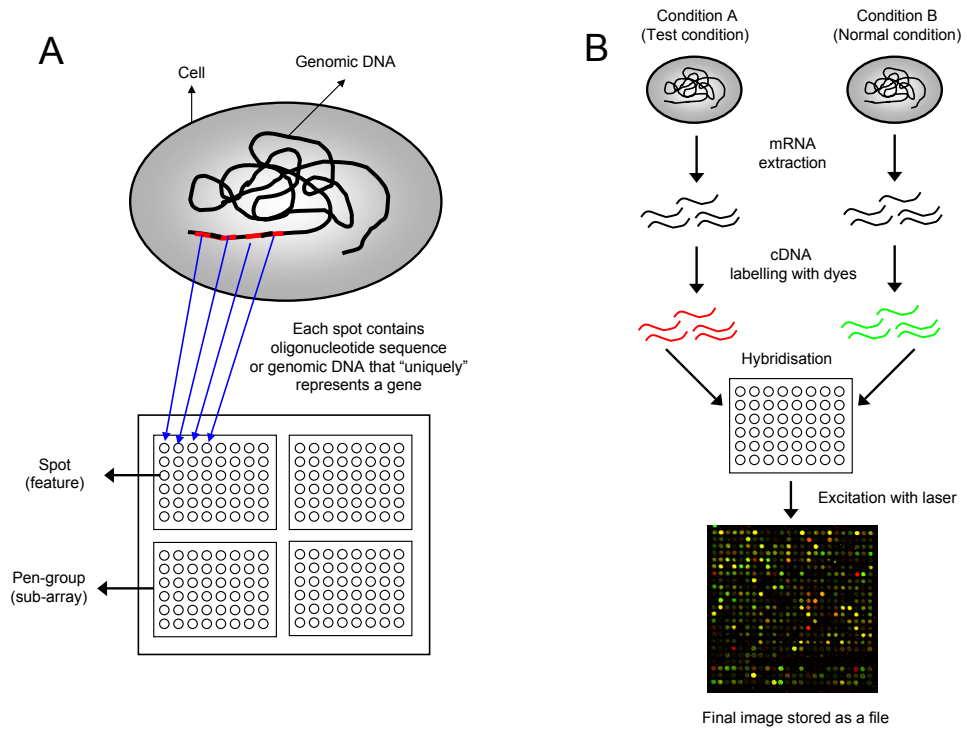

Figure 1

Figure 1.3 – **Microarray.** Illustration of the process to obtain the gene expression level using the microarray technique. This technique is used to obtain the gene expression level from the RNA of a cell. (*source:* https://www.mrc-lmb.cam.ac.uk/genomes/madanm/microarray/)

$$X = \begin{array}{c} \begin{array}{cccc} \textbf{Cell 1} & \textbf{Cell 2} & \dots & \textbf{Cell p} \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} \end{array} \begin{array}{l} \textbf{Gene 1} \\ \textbf{Gene 2} \\ \vdots \\ \textbf{Gene n} \end{array}$$

1.2 Inverse problem to estimate cell quantities

Mathematical modeling. To estimate the quantities of cells inside a tumor, biostatisticians have proposed a mathematical modeling based on genomic data. They consider the RNA coming from a tumor as a *mixed* signal composed of *pure* signals coming from all the different cells that compose a tumor. Their assumptions is that there is a direct relation

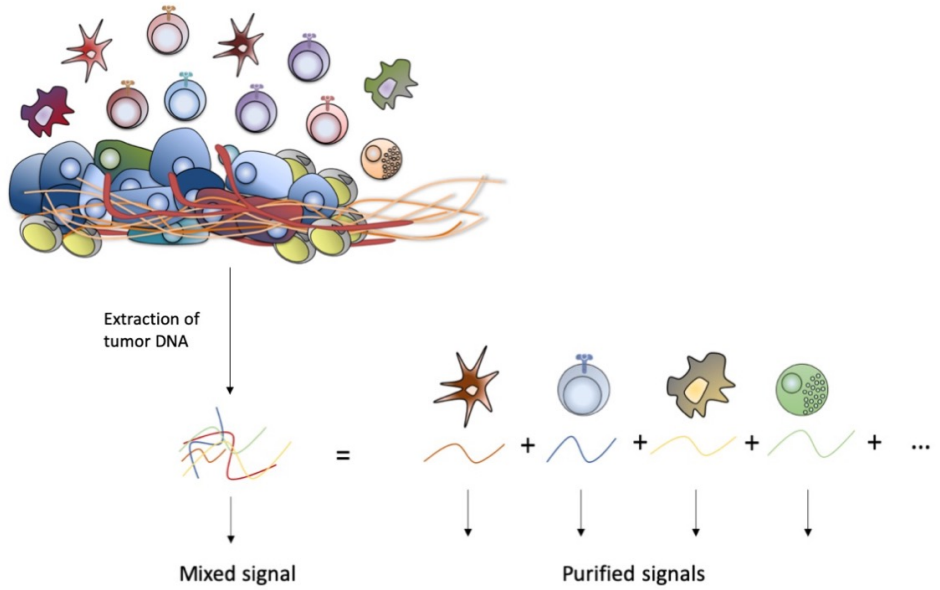

Figure 1.4 – **Linear inverse problem.** Schematic representation of the linear dependance assumed between the RNA coming from a tumor and the *pure* RNA coming from the different cells that compose it. The signal coming from the tumor is modelized as the weighted sum of signals coming from the different cells that compose it. The question is to retrieve the weights (proportions) of cells that lead to the observed mixed signal in the tumor.

between the quantity of cells inside a tumor and its gene expression level. A linear model was then proposed to modelize the gene expression level of a tumor as a function of the expression level of the different cells. From the tumor, it can be observed a vector $y \in \mathbb{R}^n$ and we are trying to retrieve the quantity of cells that lead to observing this vector supposing that there is a linear relationship between their quantity and the genes expression in y. As a result, if one has access to the *pure* transcriptome of the cells of interest, finding the quantities of these cells inside the mixed signal would resort to solving a linear inverse problem as illustrated in Figure 1.4. Formally, we denote $y \in \mathbb{R}^n$ the *mixed* signal coming from a tumor and $X \in \mathbb{R}^{n \times p}$ the matrix containing the *pure* transcriptome of the cells of interest then the linear model writes:

$$y = X\beta + \varepsilon \ , \tag{1.1}$$

where $\varepsilon \in \mathbb{R}^n$ is the noise. The inverse problem is the reconstruction of β when observing y and X.

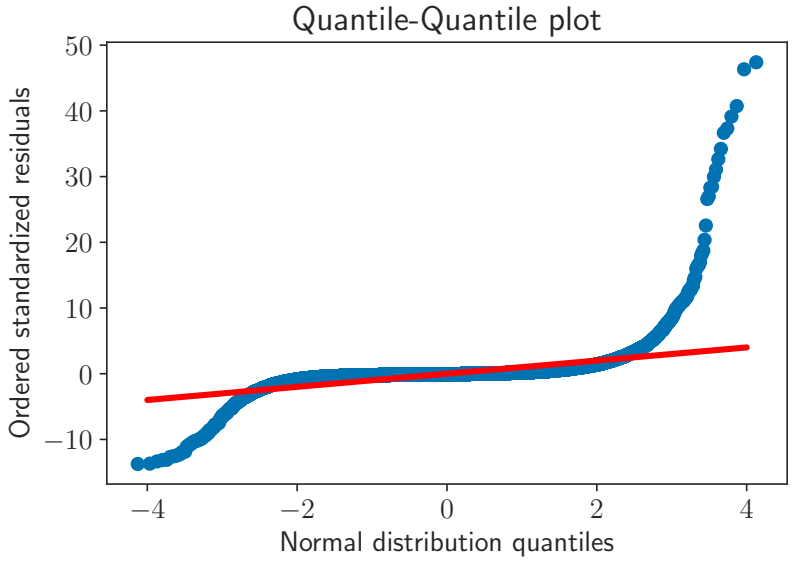

Figure 1.5 – **Quantile-Quantile plot.** Quantile-Quantile plot of the standardized residuals of the Ordinary Least Squares estimator to check for normality distribution. It illustrates the fact that the noise in the genomic data does not follow a Gaussian distribution but it is heavy-tailed and skewed.

This linear inverse problem is overdetermined since $n \gg p$. Generally, the number of cells populations p stays small (maximum 30 populations) whereas at first n is close to $20{,}000$. A natural way to obtain an estimation of the vector $\beta \in \mathbb{R}^p$ is to use the Ordinary Least Squares (OLS) estimator which is the solution of the following optimization problem:

$$\hat{\beta}_{\text{OLS}} = \underset{\beta \in \mathbb{R}^p}{\arg\min} \frac{1}{2} \|y - X\beta\|^2 \ . \tag{1.2}$$

If X has full rank, Equation (1.2) has a unique solution:

$$\hat{\beta}_{\text{OLS}} = (X^\top X)^{-1} X^\top y \ . \tag{1.3}$$

If the noise $\varepsilon \in \mathbb{R}^n$ is *i.i.d.*, homoscedastic and uncorrelated, estimating β by Equation (1.2) would lead to the Best Linear Unbiased Estimator (BLUE) (Gauss-Markov theorem). However, the quantile-quantile plot of the OLS residuals in Figure 1.5 shows that the residuals of the OLS are heavy-tailed and skewed which suggests that using a biased estimator might decrease the variance and improve the quality of the estimator. The noise comes from different elements: the preparation of the tissue by the biologists, the data acquisition device and the biological environment can all induce noise. In this case the noise is

probably heavy-tailed and asymmetric (skewed) as pointed out by Purdom and Holmes (2005).

Apart from the heavy tailed noise, there are also constraints on this estimator β coming from the nature of what is to be estimated. First, quantities of cells are estimated which means that the coefficients of β have to be positive. Second, β is often interpreted as proportions with the constraint that its coefficients sum to one. A problem of interest is then to take into account this prior information in the estimation process as we will see in details later on.

Note that in the biostatistic field, this linear inverse problem is called deconvolution as mentioned in Lu et al. (2003) and Abbas et al. (2009) who were the first to propose the linear inversion for immune cells in blood. We want to address here that there is no direct link with the deconvolution process coming from the imaging field. For mathematicians, this problem is a linear inverse problem but the literature in biostatistic has called the estimation process *deconvolution*.

It is worth mentioning that a closely related inverse problem arises in hyperspectral imaging named *hyperspectral unmixing*. The goal is to estimate the proportions of different objects present in one pixel given a dictionary of objects for which we know the spectral band signal. The positivity and the sum-to-one constraints also appear in the estimation process and are often taken into account as prior information. One of the main differences between hyperspectral unmixing and the estimation of cells proportions is that the estimator proposed in hyperspectral unmixing often has a sparse prior (see Bioucas-Dias et al. (2012) for a review). In our case, the sparsity prior does not seem relevant. Even if a given cell type is present in a very small quantity, the information about this quantity could be important and there is no reason to consider that only a few numbers of cell types can be present inside the tumor. Most of the cell types can potentially be present inside the tumor. The state-of-the-art methods in hyperspectral unmixing cannot be applied directly to our inverse problem. The type of data and the priors on the estimator result in different challenges and questions.

Challenges. The raw design matrix of the linear model X contains the gene expression level of approximately $20{,}000$ genes. When considering cells inside a tumor, only a small fraction of these genes will be expressed differently between the different types of cells. It means that a large number of the genes (the rows) present in the design matrix are not informative; they will not reflect the quantity of cells present inside the tumor since they are equally expressed by all the different types of cells. From a mathematical point of

view, it means that the columns of the matrix are highly correlated when considered as a whole as depicted in Figure 1.6. In other terms, X is ill-conditioned and multicollinearity is present in the regression problem.

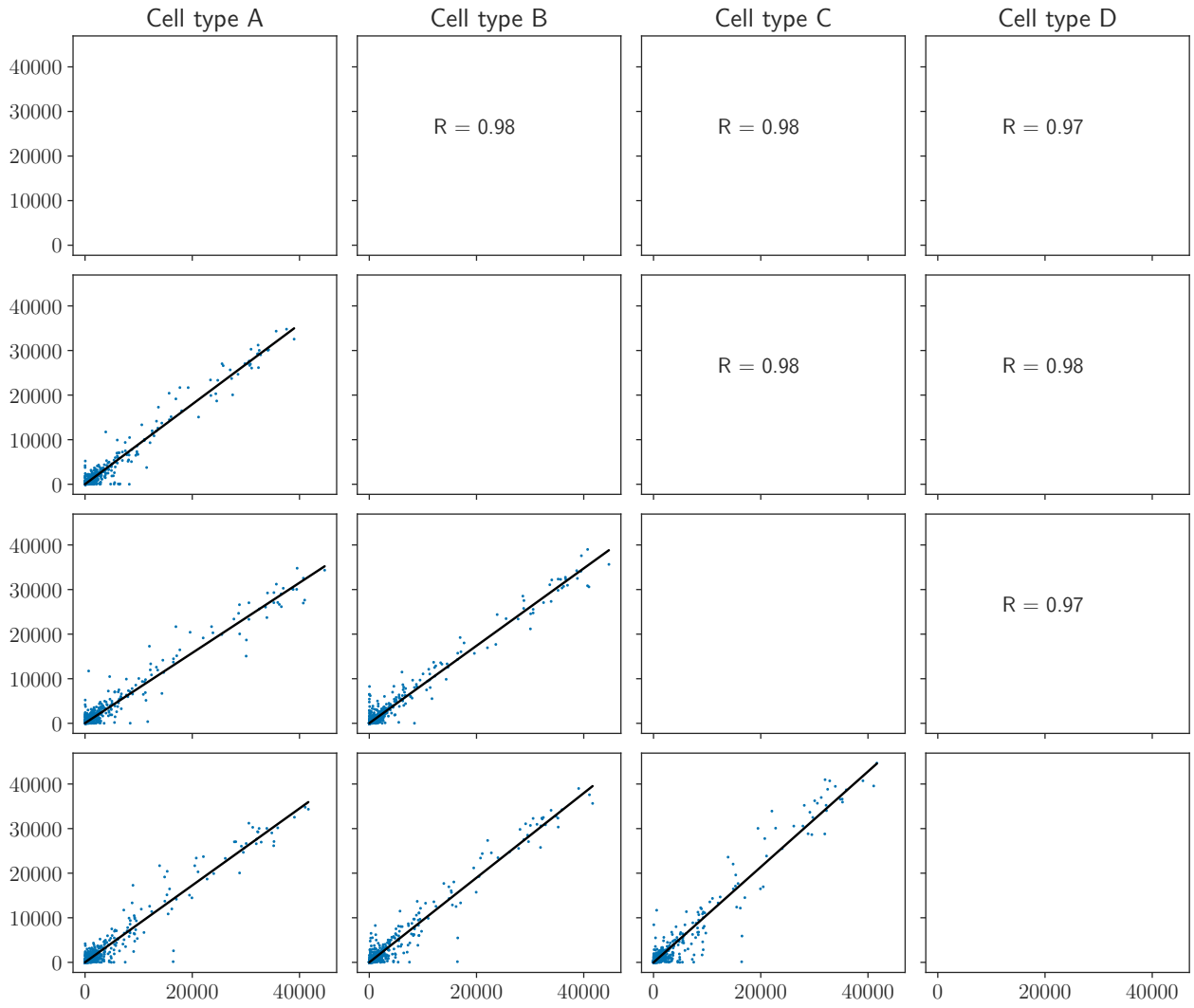

Figure 1.6 – **Multicollinearity in the design matrix X.** Scatter plots of the genes expression of the different types of cells present in the design matrix X. The cell types are compared 2 by 2 to illustrate their high correlation when considering the 5000 first genes.

When considering the Ordinary Least Squares estimator, the conditioning of the matrix X impacts the stability of the estimator. Let us consider the linear system $Ax = b$, with $A \in \mathbb{R}^{n \times p}$ and $b \in \mathbb{R}^p$. Let $\delta b \in \mathbb{R}^n$ be a small perturbation of the observation vector which can occur for example if the vector is perturbed by noise. Let δx the solution of the perturbed system *i.e.*, $A\delta x = b + \delta b$. The stability of the OLS estimator is then controlled

by the following inequality that bounds the relative error on the solution x:

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A^\top A)\frac{\|\delta b\|}{\|b\|} \quad , \tag{1.4}$$

where $\kappa(A)$ is the condition number of A *i.e.*, $\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$ where $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ are the maximal and minimal singular values of A. It means that the relative error on the solution x is bounded by the relative perturbation multiplied by the condition number. If the matrix A is ill-conditioned, a small change in the vector b will imply a large error on the solution x; hence the lack of stability.

One of the many possible ways to deal with ill-conditioned design matrix X is to regularize the estimator. A famous example of such regularization is the Tikhonov regularization (Tikhonov, 1943). The ridge regression (Hoerl and Kennard, 1970) is the quadratic data fitting term with an added Tikhonov regularization, it is obtained as the solution of

$$\hat{\beta} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|^2 + \frac{\lambda}{2}\|\beta\|^2 \quad , \tag{1.5}$$

where $\lambda > 0$ controls the tradeoff between the data fidelity term and the regularization. The solution of this optimization problem is given by

$$\hat{\beta} = (X^\top X + \lambda \,\mathrm{Id})^{-1} X^\top y \quad . \tag{1.6}$$

Interestingly, the condition number of the matrix $X^\top X + \lambda \,\mathrm{Id}$ writes

$$\kappa(X^\top X + \lambda \,\mathrm{Id}) = \frac{\sigma_{\max}(X)^2 + \lambda}{\sigma_{\min}(X)^2 + \lambda} \quad . \tag{1.7}$$

The value of $\lambda > 0$ can be chosen to greatly improve the condition number and hence increase the stability of the estimator with respect to some perturbation.

However, the classical process in the field of estimating cell proportions is to preselect the genes (the rows) of the design matrix X. From a biological point of view, the idea is to get rid of the genes that are not related to the cells considered. In biostatistics, it comes down to performing differential expression analysis, which is a very common methodology. The goal is to find the set of genes that characterizes a cell in comparison to the other cells considered, the genes that are *overexpressed* or *underexpressed* for certain cell types. It means that we seek genes that have a high or low value in comparison to all the other cell types. These methods rely on performing statistical tests (such as *t-test*) to find the genes that are

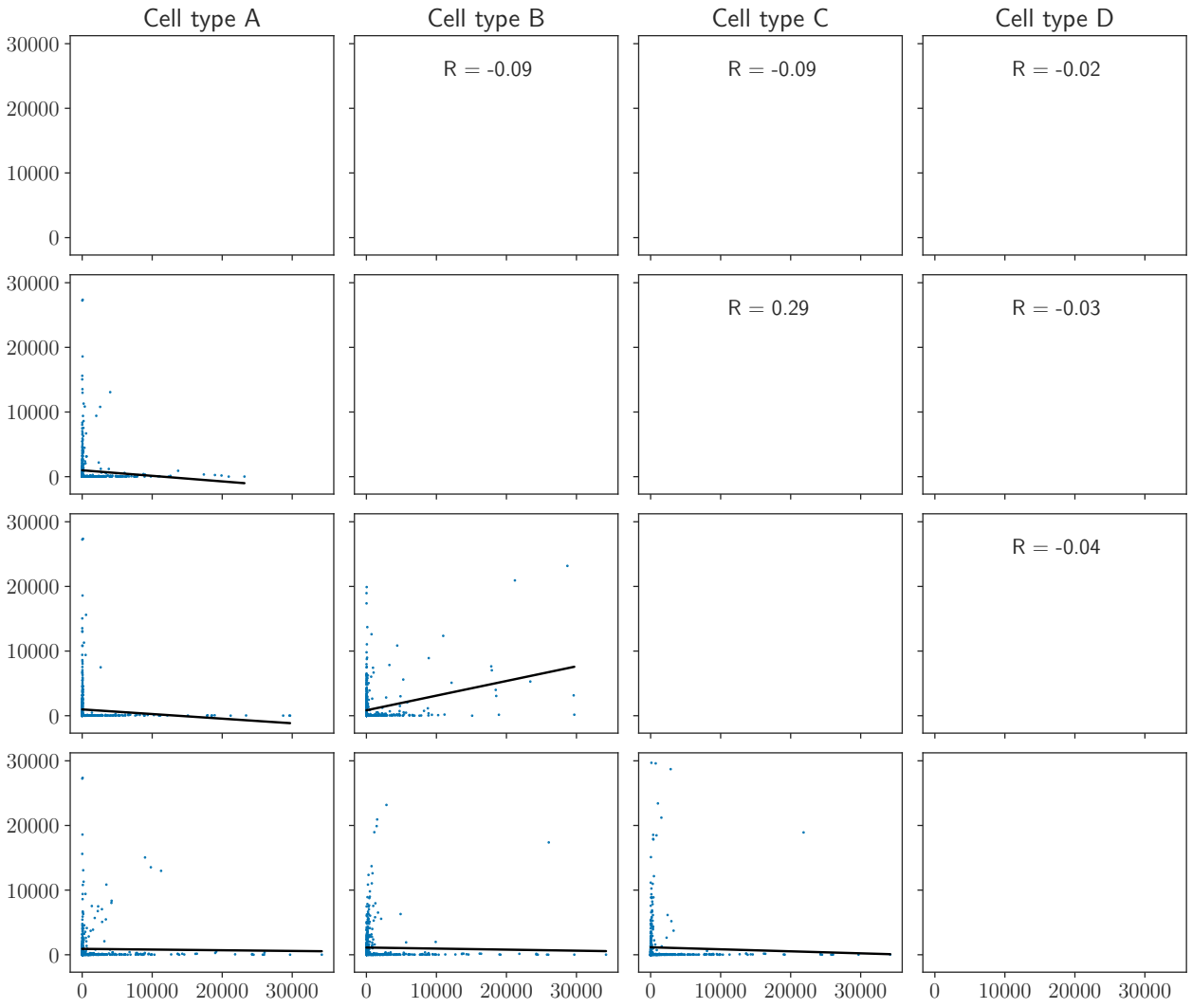

Figure 1.7 – **Signature matrix X after gene selection.** Scatter plots of the variables of the signature matrix 2 by 2 to illustrate that the preprocessing selection of genes leads to decorrelated columns in the matrix X.

differentially expressed between the different cells. The significant genes will be kept and a new design matrix is built from these genes. There is no consensus in the methodology to build such a matrix. We refer to Newman et al. (2015) that suggest for example to incrementally add the significant genes and keep the submatrix that has the lowest condition number. This matrix is often called the signature matrix as it is composed of the genes that characterize the cell types.

This preprocessing step drastically reduces the number of rows of the design matrix. Coming back to our example, the signature matrix is now composed of 584 genes. As a result of the preprocessing step, the columns of X are decorrelated as shown in Figure 1.7. The

estimation is now less sensitive to noise since the preprocessing step has also reduced the condition number (divided by 3 in our example). Often biostatisticians represent this signature matrix by a heatmap showing the row *z-score i.e.,*

$$Z_{ij} = \frac{(X_{ij} - \text{mean}(X_{i:}))}{\text{std}(X_{i:})} \quad ,$$

where mean(x) is the average of the coefficients of x and std(x) the standard deviation.

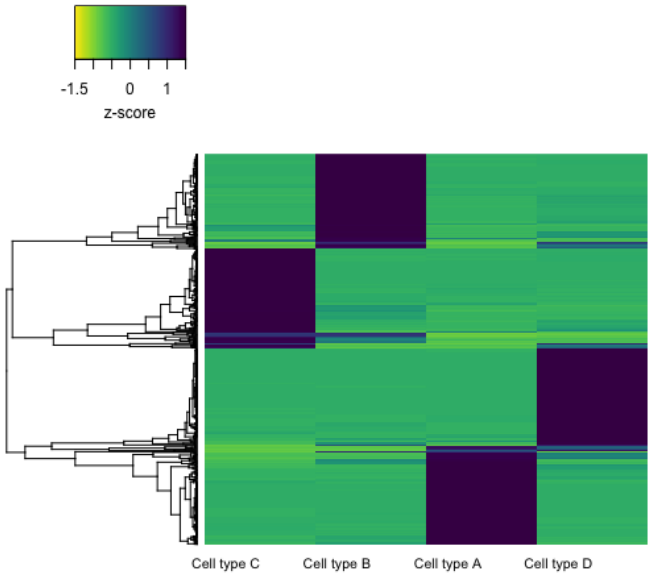

Figure 1.8 – **Heatmap of the signature matrix X after gene selection.** Heatmap representing the z-score of the signature matrix used for the estimation process. The genes overexpressed in one cell type in comparison to all the others will have a high z-score (blue on the figure). A hierarchical clustering on the genes shows that we have selected four blocks of genes, each of them characterizes a cell type.

The different immune cells that are interesting are organized in a structural way. Our goal here is not to fully explain the different immune cells of the human body nor to explain their role. We just want to highlight that this preprocessing step gets harder when the considered cells are closely related. Loosely speaking, cells can be divided in different families and some look like each other from a genomic point of view. Estimating cell quantities that are highly similar becomes a challenge.

In this thesis, we consider that the signature matrix is known and fixed. Trying to improve the signature construction is a whole literature on its own and requires a deep understanding of the relationship between cells and genes. We focus our research on the estimation

process once the signature matrix is known and fixed.

1.3 State of the art

We now focus on the estimation process, *i.e.,* the design matrix X is known and fixed and given the vector of observation $y \in \mathbb{R}^n$, one would like to retrieve the proportions $\beta \in \mathbb{R}^p$. As mentioned earlier, β is interpreted as proportions, the proposed estimator should take into account the related constraints. It means that estimating β will not be a simple linear problem but a constrained linear problem. The constraints on β are that it has to belong to the positive orthant of $\mathbb{R}^p$ and that its coefficients sum to one *i.e.,*

$$\beta \in \{x \in \mathbb{R}^p : x_i \geq 0, \sum_{i=1}^{p} x_i = 1\} \ , \tag{1.8}$$

this set is called the simplex.

We will now describe the methods proposed in the literature to estimate these proportions. The linearity assumptions and the first estimator of Equation (1.1) was proposed by Abbas et al. (2009). They estimated $\beta \in \mathbb{R}^p$ using the Ordinary Least Squares (OLS) estimator (Galton, 1886) *i.e.,* doing a simple linear regression, solving Equation (1.2). To be able to interpret β as a vector of proportions, they projected the given estimator $\hat{\beta}_{\text{OLS}}$ onto the positive orthant and then divided by the sum of the remaining coefficients. Note that minimizing Equation (1.2) and then projecting onto the set of constraints generally does not lead to the solution of minimizing Equation (1.2) under the constraint that β is positive and sum-to-one.

A second line of work (Qiao et al., 2012; Gong et al., 2011), considered the estimator given by constrained version of Equation (1.2) namely the Non-Negative Least Squares (NNLS) estimator (Lawson and Hanson, 1995) obtained as solution of:

$$\begin{aligned} \hat{\beta}_{\text{NNLS}} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \frac{1}{2}||y - X\beta||^2 \\ s.t.\ \beta_j \geq 0 \ , \end{aligned} \tag{1.9}$$

and the Simplex Ordinary Least Squares (SOLS) which is the result of this optimization

problem:

$$\begin{aligned}\hat{\beta}_{\text{SOLS}} \in \underset{\beta\in\mathbb{R}^p}{\arg\min}\ & \frac{1}{2}||y - X\beta||^2 \\ s.t.\ & \beta_j \geq 0, \\ & \sum_{j=1}^{p} \beta_j = 1\ .\end{aligned} \tag{1.10}$$

All these estimators based on the quadratic loss are affected by noise that does not follow a Gaussian distribution. More importantly the estimation performances are largely affected by heavy tailed-noise which appears to be the case in our application. It resulted in the biostatisticians community of a discussion about robustness of these estimators, meaning their ability to perform well under noisy data.

On microarray data, the considered state-of-the-art method for estimating the proportions of cells inside a tumor was proposed by Newman et al. (2015). They proposed an estimator based on the ε-insensitive loss namely the Support Vector Regression (SVR) estimator. The ε-loss is given by the following formula:

$$L_\varepsilon(y, t) = \max\{0, |y - t| - \varepsilon\}\ . \tag{1.11}$$

The first variant of SVR estimator was proposed by Drucker et al. (1996) and depends on two hyperparameters: C and ε. It is obtained as the solution of

$$\hat{\beta} \in \underset{\beta\in\mathbb{R}^p}{\arg\min}\ \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} L_\varepsilon(y_i, X_{i:}\beta)\ . \tag{1.12}$$

This optimization is often written equivalently

$$\begin{aligned}\hat{\beta} \in \underset{\beta\in\mathbb{R}^p, \xi, \xi^*\in\mathbb{R}^n}{\arg\min}\ & \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ s.t.\ & y_i - X_{i:}\beta - \beta_0 \leq \varepsilon + \xi_i \\ & X_{i:}\beta + \beta_0 - y_i \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0\ .\end{aligned} \tag{1.13}$$

A second variant exists and was proposed by Schölkopf et al. (1999), the parameter ε is part of the variables to optimize in the optimization problem and a new hyperparameter $\nu \in [0, 1]$ appears. This estimator is the one retained by Newman et al. (2015) for the

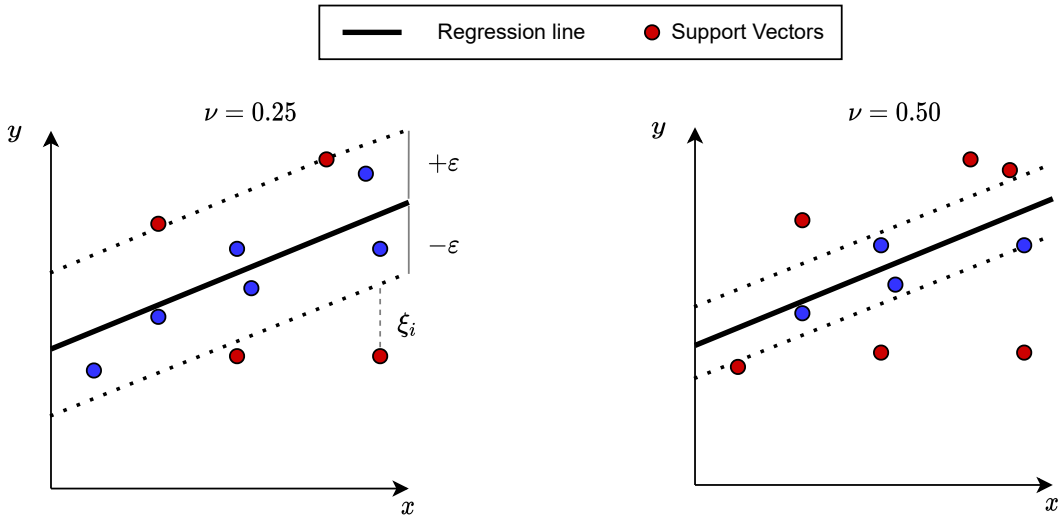

Figure 1.9 – Support Vector Regression with one explanatory variable. The points outside the ε tube (in red) are the support vectors. Changing the hyperparameter ν changes the size of the tube and the number of support vectors.

estimation of cells proportions.

We will then refer in the following to ε-SVR for the one proposed by Drucker et al. (1996) and ν-SVR for the one proposed by Schölkopf et al. (1999). The ν-SVR estimator is obtained by solving the following optimization problem:

$$\begin{aligned} \hat{\beta}_{\text{SVR}} \in \underset{\beta,\beta_0,\xi_i,\xi_i^*,\varepsilon}{\arg\min} \quad & \frac{1}{2}||\beta||^2 + C(\nu\varepsilon + \frac{1}{n}\sum_{i=1}^{n}(\xi_i + \xi_i^*)) \\ s.t. \quad & y_i - X_{i:}\beta - \beta_0 \leq \varepsilon + \xi_i \\ & X_{i:}\beta + \beta_0 - y_i \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \varepsilon \geq 0 \ . \end{aligned} \tag{1.14}$$

The advantage of this modification is that the parameter ε that controls the side of the ε-insensitive tube (see Figure 1.9) is automatically computed. We will now describe the idea behind this estimator. At each point $X_{i:} \in \mathbb{R}^p$, an error ε is allowed. If the error is larger than ε, it is captured in the slack variables appearing in the optimization problem ξ_i, ξ_i^*. The hyperparameter $C > 0$ controls the tradeoff between the ℓ_2 regularization and the data fidelity term namely the ε-insentive loss (or equivalently the slack variables). The value of ε is obtained by a trade off against model complexity and slack variables

via a constant $\nu > 0$. From the KKT conditions, it can be shown that all problems given by Equation (1.14) with $\nu \geq 1$, lead to the same solution (Schölkopf and Smola, 2002, Chapter 9, p.262). The points $X_{i:}$ that are lying on or outside the ε-tube are called support vectors as shown in Figure 1.9. This new formulation of the problem has an interesting interpretation: ν is an upper bound on the proportions of points $X_{i:}$ lying outside the tube (also called fraction of error) and it is a lower bound on the proportions of support vectors.

A very interesting property of the SVR is that only the points outside the ε-tube are penalized whereas the points inside the tube have zero loss. This does not mean that only the points outside the tube determine the regression, it is the contrary: a point outside the tube can be moved anywhere in its living space as long as it stays outside the tube without changing the linear estimation. This is certainly the reason why this estimator has shown to perform well in estimating cells from microarray data. As stated earlier, the noise in the data is heavy tailed, the SVR can be robust to the points (genes in our application) that are hit by the tail of the noise distribution.

The selection of support vectors for the estimation acts like a gene selection (*i.e.*, selection of the rows of the matrix X), the very noisy genes are not kept for the estimation. Even if, the method seems to perform well in the settings presented in Newman et al. (2015) looking at how the complete method named Cibersort was implemented raises a few questions and concerns. First of all, the ν-SVR estimator does not take into account the underlying constraints from estimating a vector of proportions. The estimated proportions obtained by Cibersort are then obtained by setting to zero the negative coefficients of $\hat{\beta}_{\text{SVR}}$ and then dividing each coefficients by the sum of the remaining coefficients leading to a vector of proportions. As already mentioned, solving Equation (1.14) and then projecting the results onto the set of constraints does not generally lead to the minimum with the additional constraints. Second, the choice of hyperparameters is done as follows: the parameter C is always set to be equal to 1. The parameter ν is chosen in the set $\{0.25, 0.5, 0.75\}$ and the one leading to the minimal root mean squared error is kept for the estimation process.

A first natural improvement of their method would be to study the SVR estimator with the positivity and sum-to-one constraints. However, the new underlying optimization problem has not been studied in the literature. It raises the question on how to efficiently solve this new optimization problem. Secondly, the performance of an estimator can be drastically affected by the choice of the hyperparameters, we believe that choosing C and ν in a different way could improve the estimation process. Our goal is to avoid grid-search

which is costly to set the two hyperparameters of the ν-SVR.

The following questions summarize the main directions of this thesis:

- Can we solve the support vector regression with the proportions constraints optimization problem efficiently and have convergence guarantees towards a solution?
- Can we ease the hyperparameter selection process? We would like to avoid the grid-search selection which is costly for two hyperparameters in the SVR case.
- Finally, does taking the constraints into account and choosing the hyperparameters more *carefully* improve the quality of the estimation?

We now briefly present the main contributions of this thesis to answer these questions.

1.4 Coordinate descent for non-smooth optimization

Solving the optimization problem of the SVR estimator is usually done using an optimization algorithm called coordinate descent. Generally, the coordinate descent is applied to the dual optimization problem as proposed by Ho and Lin (2012). More generally, the optimization problem that we are interested in solving is as follows

$$x^\star \in \arg\min_{x \in \mathbb{R}^p} f(x) + \underbrace{\sum_{j=1}^{p} g_j(x_j)}_{=g(x)} \ , \tag{1.15}$$

where f is a smooth (∇f is Lipschitz) convex function and the functions g_j are convex (possibly non-differentiable). This composite minimization can be solved using proximal algorithms such as proximal gradient descent (Lions and Mercier, 1979; Combettes and Wajs, 2005) or proximal coordinate descent (Tseng and Yun, 2009). We now give the definition of the proximal operator of a general convex function g.

Definition 1.1 (Proximal operators.)**.** The proximal operator of a proper (Definition 2.4), closed (Definition 2.5), convex function $g : \mathbb{R}^p \to \bar{\mathbb{R}}$ is the function denoted $\mathrm{prox}_g : \mathbb{R}^p \to \mathbb{R}^p$ defined by

$$\mathrm{prox}_g(x) \triangleq \arg\min_{y \in \mathbb{R}^p} \frac{1}{2}\|y - x\|^2 + g(y) \ .$$

Several estimators used for regression tasks or classification tasks are the results of an optimization that can be written like Equation (1.15): the Lasso (Tibshirani, 1996), the elastic net (Zou and Hastie, 2005), the sparse logistic regression (Koh et al., 2007) and the SVM/SVR (Boser et al., 1992; Schölkopf et al., 1999). The proximal coordinate descent algorithm is a very efficient tool to solve these types of optimization problem and we now state some convergence properties of this algorithm.

1.4.1 Local linear convergence of the coordinate descent algorithm

These non-smooth optimization problems coming from machine learning generate solution that are *structured*. For example, the ℓ_1 sparse regularization where all $g_j = |\cdot|$ leads to solutions that only have a few non-zero coefficients. The structure induced by the ℓ_1 norm is carried out by the notion of support which is the set of non-zero coefficients of the solution. This notion of support can be generalized for general separable convex functions g_j.

Definition 1.2 (Generalized support)**.** The *generalized support* $\mathcal{S}_x \subseteq \{1, \dots, p\}$ is the set of indices $j \in \{1, \dots, p\}$ where g_j is differentiable at x_j:

$$\mathcal{S}_x \triangleq \{j \in [p] : \partial g_j(x_j) \text{ is a singleton}\} \ ,$$

where ∂g denotes the subdifferential of the function g (see Definition 2.11)

An important question about the fact of using an iterative algorithm to solve Equation (1.15) is wether we identify the good support of the solution after a finite number of iterations? In other words, does it exist an iteration $K > 0$ such that for all $k \geq K$, we have $x^{(k)}_{\mathcal{S}^c_{x^\star}} = x^\star_{\mathcal{S}^c_{x^\star}}$ with $x^\star$ being a solution of Equation (1.15). The finite support identification property was proved for the coordinate descent in Nutini et al. (2017, Lemma 3) and a similar result with a different proof technique is given in Chapter 3. Moreover, proximal gradient descent enjoys linear convergence rate once the support is identified (Liang et al., 2014) and we show that it is the case for coordinate descent as well.

Main contribution. In order to study local linear convergence, we consider the fixed point iteration of a complete epoch (an epoch is a complete pass over all the coordinates). A full epoch of cyclic coordinate descent can be written:

$$x^{(k+1)} = \psi(x^{(k)}) \triangleq \mathcal{P}_p \circ \dots \circ \mathcal{P}_1(x^{(k)}) \ , \tag{1.16}$$

where $\mathcal{P}_j$ are coordinatewise sequential applications of the proximal operator $\mathcal{P}_j : \mathbb{R}^p \to \mathbb{R}^p$ and $\gamma_j > 0$:

$$x \mapsto \begin{pmatrix} x_1 \\ \vdots \\ x_{j-1} \\ \operatorname{prox}_{\gamma_j g_j}\left(x_j - \gamma_j \nabla_j f(x)\right) \\ x_{j+1} \\ \vdots \\ x_p \end{pmatrix} .$$

The following theorem states that after support identification the coordinate descent algorithm converges linearly towards to the solution.

Theorem 1.1 (Local linear convergence)**.** *Consider a solution $x^\star$ of Equation (1.15) and $\mathcal{S} = \mathcal{S}_{x^\star}$ its generalized support. Suppose*

1. *The solution is non-degenerated* i.e., $-\nabla f(x^\star) \in \operatorname{ri}(\partial g(x^\star))$ *where* $\operatorname{ri}(C)$ *is the relative interior*[1] *of a convex set C and ∂g is the subdifferential of g.*
2. *For all $j \in \mathcal{S}$, g_j is locally $\mathcal{C}^2$ and f is locally $\mathcal{C}^2$ around $x^\star$.*
3. *The restricted injectivity condition hold* i.e., $\nabla^2_{\mathcal{S},\mathcal{S}} f(x^\star) \succ 0$ *(Hessian restricted to the rows and columns of indices in $\mathcal{S}$).*
4. *The sequence $(x^{(k)})_{k\geq 0}$ generated by the coordinate descent algorithm converges to $x^\star$.*
5. *The model has been identified* i.e., *there exists $K \geq 0$ such as for all $k \geq K$*

$$x^{(k)}_{\mathcal{S}^c} = x^\star_{\mathcal{S}^c} .$$

Then $(x^{(k)})_{k\geq K}$ converges linearly towards $x^\star$. More precisely, for any $\nu \in [\rho(\mathcal{J}\psi_{\mathcal{S},\mathcal{S}}(x^\star)), 1[$, there exists a constant C such that for all $k \geq K$,

$$\|x^{(k)}_{\mathcal{S}} - x^\star_{\mathcal{S}}\| \leq C\nu^{(k-K)}\|x^{(K)}_{\mathcal{S}} - x^\star_{\mathcal{S}}\| .$$

We denote by $\mathcal{J}f(x)$ the Jacobian of a function f at x and $\rho(M)$ corresponds to the spectral radius of a matrix M.

[1] see Definition 2.10 in Chapter 2

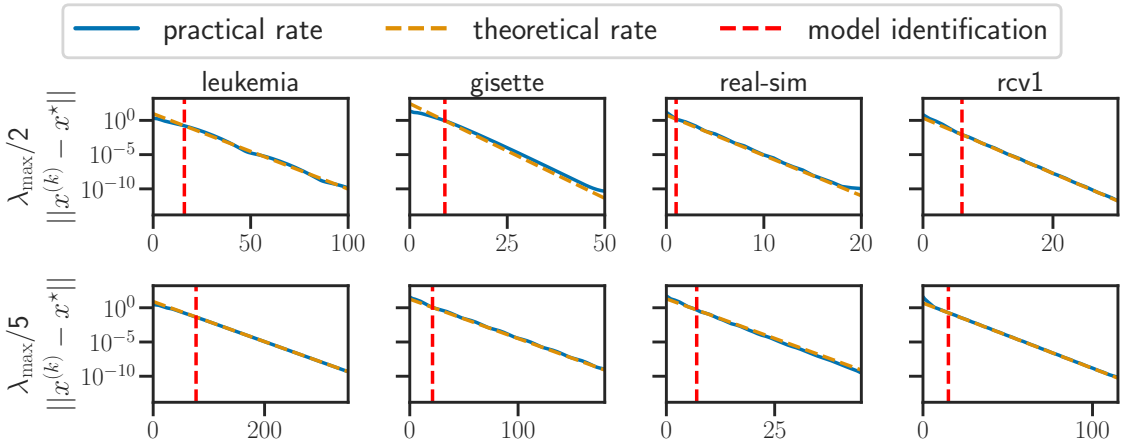

Figure 1.10 – **Lasso, linear convergence.** Distance to optimum, $\|x^{(k)} - x^\star\|$, as a function of the number of iterations k, on 4 different datasets: *leukemia*, *gisette*, *rcv1*, and *real-sim*. The regularization parameter was chosen proportional to $\lambda_{\max} = \frac{\|A^\top b\|}{2n}$.

We showed on several real datasets that our theoretical rates derived from Theorem 1.1 match empirical rates as shown in Figure 1.10 for the case of the Lasso where $f(x) = \frac{1}{2n}\|Ax - b\|^2$ with $A \in \mathbb{R}^{n \times p}$ being the design matrix and $b \in \mathbb{R}^n$ the observation vector and $g_j(x_j) = \lambda |x_j|$ for a $\lambda > 0$. This theorem is proved in Chapter 3.

Relation to previous works. Local linear convergence was proved for the ISTA and FISTA algorithm on the Lasso problem (Tao et al., 2016) studying the spectral properties of the recurrence matrix used for the updates of both algorithms. This result has been extended to the generic proximal gradient descent algorithm in Liang et al. (2014) supposing that g is a partly smooth function. The partly smooth class of functions was defined in Lewis (2002) and unifies most of the non-smooth functions known in machine learning. It defines properly the structure that we mentioned in the non-smooth optimization problems cited earlier. In our work, we consider the coordinate descent algorithm with the assumption that g is separable *i.e.*, $g(x) = \sum_{j=1}^p g_j(x_j)$. For a solution $x^\star$ of Equation (1.15), we proved that the class of separable functions that are $\mathcal{C}^2$ on $\mathcal{S}_{x^\star}$ at $x^\star$ is equivalent to the fact that these functions are partly smooth. Once stated, Theorem 1.1 shows that the coordinate descent algorithm enjoys the same local linear convergence property than the proximal gradient descent. The local linear convergence property was also proved for the SAGA and prox-SVRG algorithms in Poon et al. (2018) under the same framework of assumptions.

This local linear convergence behavior of the previously mentioned algorithms raises a natural question about how fast we identify the structure induced by the non-smooth

function g. This notion is also known as active set complexity defined in Nutini et al. (2019) which is the number of iterations needed to identify the structure. Under the assumptions that g is separable, Nutini et al. (2019) gave a bound on the number of iterations needed for the proximal gradient descent algorithm to identify the structure. This results was extended for f being strongly convex with separable functions g for the cyclic or greedy coordinate descent algorithm in Nutini et al. (2017). In our work, we only prove that the local linear convergence regime happens after structure identification but we do not prove that it starts immediately after identification even if it seems to be the case in practice.

1.4.2 Convergence of the generalized Sequential Minimal Optimization algorithm

The Support Vector Regression (SVR) is a widely used estimator to estimate linear or non-linear functions. Motivated by the addition of probability constraint on the estimator in the linear case, we consider general linear constraints added in the optimization problem of the ν-SVR estimator (Schölkopf et al., 1999). The optimization problem that we are seeking to solve reads for a design matrix $X \in \mathbb{R}^{n \times p}$ and an observation vector $y \in \mathbb{R}^n$:

$$\begin{aligned} \min_{\beta, \beta_0, \xi_i, \xi_i^*, \varepsilon} \quad & \frac{1}{2}||\beta||^2 + C\left(\nu\varepsilon + \frac{1}{n}\sum_{i=1}^{n} L_\varepsilon(y_i, X_{i:}\beta)\right) \\ \text{subject to} \quad & \xi_i, \xi_i^* \geq 0, \varepsilon \geq 0 \\ & A\beta \leq b \\ & \Gamma\beta = d \ , \end{aligned} \tag{LSVR-P}$$

where $A \in \mathbb{R}^{k_1 \times p}$, $\Gamma \in \mathbb{R}^{k_2 \times p}$, $\beta \in \mathbb{R}^p$, ξ, $\xi^* \in \mathbb{R}^n$ and $\beta_0 \in \mathbb{R}$, $\varepsilon > 0$.

For example if $A = -\operatorname{Id}_p$, $b = 0$, $\Gamma = 0$ and $d = 0$, the resulting optimization problem adds non-negative prior on the SVR estimator which can be seen as the counter part of the Non-Negative Least Squares regression (Lawson and Hanson, 1995). Considering $A = -\operatorname{Id}_p$, $b = 0$, $\Gamma = (1, \ldots, 1)$ and $d = 1$ leads to the Simplex SVR with positive and sum-to-one constraints which is well suited for the estimation of cells proportions.

The SVR optimization problem without linear constraints is usually solved in its dual using dual coordinate descent algorithm (Ho and Lin, 2012) or a variant named Sequential Minimal Optimization (SMO) algorithm (Platt, 1999). The dual problem of Problem (LSVR-P) will be explicitly derived in Chapter 4 but can be written:

$$
\begin{aligned}
\min_{\theta \in \mathbb{R}^{2n+k_1+k_2}} \quad & f(\theta) = \frac{1}{2}\theta^\top Q\theta + l^\top \theta \\
\text{subject to} \quad & 0 \leq \theta_i \leq \frac{C}{n}, \forall i \in \{1, \dots, 2n\} \\
& \sum_{i=1}^{2n} \theta_i = C\nu \\
& \sum_{i=1}^{n} \theta_i - \theta_{i+n} = 0 \\
& \theta_i \geq 0, \forall i \in \{2n+1, \dots, 2n+k_1\} \ ,
\end{aligned}
\tag{LSVR-D}
$$

with Q a semidefinite positive matrix.

The two equality constraints in Problem (LSVR-D) link some variables together making classical coordinate descent impossible to use. The SMO algorithm updates two variables at a time ensuring that the two equality constraints are satisfied after each update.

Main contribution. We show that the linear constrained ν-SVR dual optimization problem has a convex quadratic objective function under convex constraints mixing blocks of variables that are separable and non-separable. We propose, in Chapter 4, a generalization of the SMO algorithm proposed by Platt (1999) to solve Problem (LSVR-D) and prove the following convergence theorem:

Theorem 1.2. *Let us suppose that $\{x \in \mathbb{R}^p : Ax \leq b, \Gamma x = d\}$ defines a non-empty polyhedron. Then the sequence of iterates $(\theta^k)_{k\geq 0}$, defined by the generalized SMO algorithm, converges to an optimal solution of the optimization Problem (LSVR-D).*

The detailed algorithm and proof of convergence are given in Chapter 4.

Relation to previous works. The convergence of the SMO algorithm was proved by Keerthi and Gilbert (2002). Later, Lopez and Dorronsoro (2012) gave a simpler proof of convergence. We show in Chapter 4 that the proof technique used in Lopez and Dorronsoro (2012) can be extended to the linear constraints case with some new arguments. Concerning the convergence rates, She and Schmidt (2017) proved linear convergence rate when considering random uniform selection of the blocks to update. Our proposed algorithm is a type of greedy algorithm that selects the pair of variables (or single variable) that violate the Karush–Kuhn–Tucker (KKT) optimality conditions the most.

Convergence guarantees for the coordinate descent algorithm are generally stated assum-

ing that the non-smooth function is separable. The classical SMO (Platt, 1999) is an example of coordinate descent variant that can be used without the separability assumptions, here with equality constraints linking the variables together. For the random coordinate descent, Necoara and Patrascu (2014) proposed a variant of the coordinate descent that can be used for composite minimization problem (Equation (1.15)) with the addition of one single equality constraints breaking the separability assumptions. Their proposed algorithm is very similar to the SMO algorithm but encompasses a wider class of optimization problem and uses a different index selection strategy (random vs greedy). This result was then extended by Reddi et al. (2014) with a linear system linking the variables together *i.e.,* a system of equality constraints using a random index selection. Our algorithm is a special instance of a coordinate descent variant that solves a problem with variables that are separable and variables that are linked by equality constraints. To our knowledge, general proof of convergence with greedy index selection for this type of problems does not exist and it is a first step towards considering more general optimization problems that could be solved using coordinate descent variants weakening the separability assumption in greedy type coordinate descent.

1.5 Automatic hyperparameters selection for non-smooth convex models

After focusing on optimization tools that would help us in the process of solving the underlying optimization problem for our cells quantification application, we turn towards the field of hyperparameters selection. Selecting hyperparameters in machine learning models can be a difficult task mainly when the number of hyperparameters is large. Taking the SVR as an example, having two hyperparameters to set is usually done using a predefined grid of hyperparameters on which we test the performance of the model for each point of the grid. This can be very costly and using a grid of ten values for each parameters in the SVR already requires to solve a hundred times the optimization problem.

The hyperparameter selection step requires a measure of performance for a given estimator called a criterion. Formally, a criterion is a function $\mathcal{C} : \mathbb{R}^p \to \mathbb{R}$, it is typically chosen to promote good generalization error, *e.g.,* the held-out loss (Devroye and Wagner, 1979), the cross-validation loss (CV, Stone and Ramer 1965, see Arlot and Celisse 2010 for a survey), or to reduce model complexity, *e.g.,* AIC (Akaike, 1974), BIC (Schwarz, 1978) or SURE (Stein, 1981) criteria (see Table 1.1 for some common examples). Selecting hyperparam-

Criterion	Problem type	Criterion $\mathcal{C}(\beta)$
Held-out mean squared error	Regression	$\frac{1}{n}\|y^{\text{val}} - X^{\text{val}}\beta\|^2$
Stein unbiased risk estimate (SURE)[2]	Regression	$\|y - X\beta\|^2 - n\sigma^2 + 2\sigma^2 \text{dof}(\beta)$
Held-out logistic loss	Classification	$\frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i^{\text{val}} X_{i:}^{\text{val}}\beta})$
Held-out smoothed Hinge loss[3]	Classification	$\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y_i^{\text{val}}, X_{i:}^{\text{val}}\beta)$

Table 1.1 – Examples of outer criteria used for hyperparameter selection where y^{val} and X^{val} are the validation set for the held-out loss.

eters given a criterion can be cast as a bilevel optimization problem (Colson et al., 2007)

$$\begin{aligned} &\underset{\lambda \in \mathbb{R}^r}{\arg\min} \left\{ \mathcal{L}(\lambda) \triangleq \mathcal{C}\left(\hat{\beta}^{(\lambda)}\right) \right\} \\ &s.t.\ \hat{\beta}^{(\lambda)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min}\, \Phi(\beta, \lambda) \ . \end{aligned} \tag{1.17}$$

Using the grid-search to solve this optimization problem can be seen as using zero-order optimization method *i.e.*, solving the optimization problem only with the evaluation of the function as information.

On the other hand, when the hyperparameter space is continuous and the (regularization path) function $\lambda \mapsto \hat{\beta}^{(\lambda)}$ is well-defined and (almost everywhere) differentiable, first-order optimization methods are well suited to solve the bilevel optimization Problem (1.17). Using the chain rule, the gradient of $\mathcal{L}$ *w.r.t.* λ, also called *hypergradient*, evaluates to

$$\nabla_\lambda \mathcal{L}(\lambda) = \hat{\mathcal{J}}_{(\lambda)}^\top \nabla \mathcal{C}(\hat{\beta}^{(\lambda)}) \ , \tag{1.18}$$

with $\hat{\mathcal{J}}_{(\lambda)} \in \mathbb{R}^{p \times r}$ the *Jacobian* of the function $\lambda \mapsto \hat{\beta}^{(\lambda)}$. The main challenge of applying first-order methods to solve Problem (1.17) is evaluating the hypergradient in Equation (1.18). There are three main algorithms to compute the hypergradient $\nabla_\lambda \mathcal{L}(\lambda)$: *implicit differentiation* (Larsen et al., 1996), *automatic differentiation* using the *backward (or reverse) mode* (Linnainmaa, 1970) or *forward mode* (Wengert, 1964). Once the hypergradient in Equation (1.18) has been computed, one can solve Problem (1.17) by using a first-order optimization scheme, for instance, gradient descent, with a step size $\rho > 0$: $\lambda^{(t+1)} = \lambda^{(t)} - \rho \nabla_\lambda \mathcal{L}(\lambda^{(t)})$.

[2]For a linear model $y = X\beta + \varepsilon$, with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, the degree of freedom (dof, Efron 1986) is defined as $\text{dof}(\beta) = \sum_{i=1}^{n} \text{cov}(y_i, (X\beta)_i)/\sigma^2$.

[3]The smoothed Hinge loss is given by $\mathcal{L}(x) = \frac{1}{2} - x$ if $x \leq 0$, $\frac{1}{2}(1-x)^2$ if $0 \leq x \leq 1, 0$ else .

1.5.1 Hypergradient computation in non-smooth convex learning

The goal is to propose an efficient method to compute the hypergradient $\nabla\mathcal{L}$ to use first order method for the resolution of Problem (1.17) when the lower problem is non-smooth.

We consider estimators that are the solution of the following generic optimization problem:

$$\hat{\beta}^{(\lambda)} \in \arg\min_{\beta\in\mathbb{R}^p} \Phi(\beta,\lambda) = f(\beta) + \underbrace{\sum_{i=1}^{n} g_j(\beta_j,\lambda)}_{=g(\beta,\lambda)} \ , \tag{1.19}$$

where ∇f is L-Lipschitz and g_j are proper, closed and convex. These types of optimization problems can be solved using iterative proximal algorithms. The solution $\hat{\beta}^{(\lambda)}$ satisfies the following fixed point equation (Combettes and Wajs, 2005), for any $\gamma > 0$:

$$\hat{\beta}^{(\lambda)} = \operatorname{prox}_{\gamma g}\left(\hat{\beta}^{(\lambda)} - \gamma\nabla f(\hat{\beta}^{(\lambda)})\right) \ . \tag{1.20}$$

Main contribution. We give an implicit differentiation formula for composite minimization of the form Equation (1.19) based on the notion of generalized support (Definition 1.2). The set of assumptions used for this theorem is not detailed here but will be given in Chapter 6.

Theorem 1.3 (Non-smooth implicit formula)**.** *Let $\lambda \in \mathbb{R}^r$. Let $\hat{\beta} \triangleq \hat{\beta}^{(\lambda)}$ be the solution of Equation (1.19), $\hat{\mathcal{S}}$ be its generalized support. Under a set of assumptions ensuring that the proximal operator is differentiable at the optimum, the Jacobian $\hat{\mathcal{J}}$ of the lower problem in Equation (1.19) is given by the following formula, with $\hat{z} = \hat{\beta} - \gamma X^\top \nabla f(X\hat{\beta})$:*

$$\begin{aligned}
\hat{\mathcal{J}}_{\hat{\mathcal{S}}^c} &= \partial_2 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}^c} \ , \\
\hat{\mathcal{J}}_{\hat{\mathcal{S}}} &= A^{-1}\left(\partial_2 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}} - \gamma\partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}} X_{:\hat{\mathcal{S}}}^\top \nabla^2 f(X\hat{\beta}) X_{:\hat{\mathcal{S}}^c} \hat{\mathcal{J}}_{\hat{\mathcal{S}}^c}\right) \ ,
\end{aligned}$$

where $A \triangleq \mathrm{Id}_{\hat{\mathcal{S}}} - \partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}}\left(\mathrm{Id}_{\hat{\mathcal{S}}} - \gamma X_{:\hat{\mathcal{S}}}^\top \nabla^2 f(X\hat{\beta}) X_{:\hat{\mathcal{S}}}\right)$.

Here the proximal operator of $g(\cdot,\lambda)$, is seen as a function ψ of β and λ:

$$\begin{aligned}
\mathbb{R}^p \times \mathbb{R}^r &\to \mathbb{R}^p \\
(\beta,\lambda) &\mapsto \operatorname{prox}_{g(\cdot,\lambda)}(\beta) = \psi(\beta,\lambda) \ .
\end{aligned}$$

We denote $\partial_1 \operatorname{prox}_g \triangleq \partial_1\psi$ and $\partial_2 \operatorname{prox}_g \triangleq \partial_2\psi$ where $\partial_1\psi$ is the Jacobian *w.r.t.* the first

variable and $\partial_2\psi$ the Jacobian *w.r.t.* the second variable. This theorem is proved in Chapter 6.

As stated in the theorem, the generalized sparsity induced by the non-smooth functions g_j can be taken into account to speed-up the computation of the hypergradient. Moreover, using an implicit differentiation technique allows us to use the solver of our choice, as long as it identifies the generalized support after a finite number of iterations.

We compared three different methods for the computation of the hypergradient of Problem (1.17), with the lower problem being the Lasso (Tibshirani, 1996) and the outer criterion being the held-out mean squared error (see Table 1.1):

- **Forward-mode PCD**: the classical forward differentiation of the proximal coordinate descent (PCD) algorithm used to solve the Lasso. This algorithm does not take into account the sparsity induced by the non-smooth functions g_j.
- **Implicit diff**: the Lasso is solved using the proximal coordinate descent (PCD) algorithm and then in a second time, we solve the linear system given in Theorem 1.3.
- **Implicit diff. + Celer**: the solver of the Lasso is changed, to apply an accelerated version of the proximal coordinate descent named `Celer`. After solving the Lasso, the linear system given in Theorem 1.3 is solved.

As it can be seen on Figure 1.11, taking advantage of the sparsity of the Jacobian leads to a significant increase in the speed of the hypergradient computation (Forward-mode versus Implicit diff.). Moreover, thanks to the implicit differentiation method, we can use state-of-the-art solvers as long as they identify the support of the solution. We see that combining the acceleration technique `Celer` with the implicit differentiation formula largely increase the speed of the hypergradient computation.

Relation to previous works. Implicit differentiation for the selection of hyperparameters can be traced back to Bengio (2000). Twice differentiable functions Φ were considered and the implicit differentiation involves the resolution of a linear system of size $p \times p$. Our result shows that for Lasso-type models the Jacobian has the same sparsity pattern than the iterates. We can take advantage of this fact to speed-up the computation of the Jacobian by solving a linear system of size $|\hat{\mathcal{S}}| \times |\hat{\mathcal{S}}|$.

One of the key assumptions for our algorithm to work is that the iterative solver of the Lasso identifies the good support. It is true for both the proximal gradient descent (Hale et al., 2008) and the proximal coordinate descent algorithm (Massias et al., 2020).

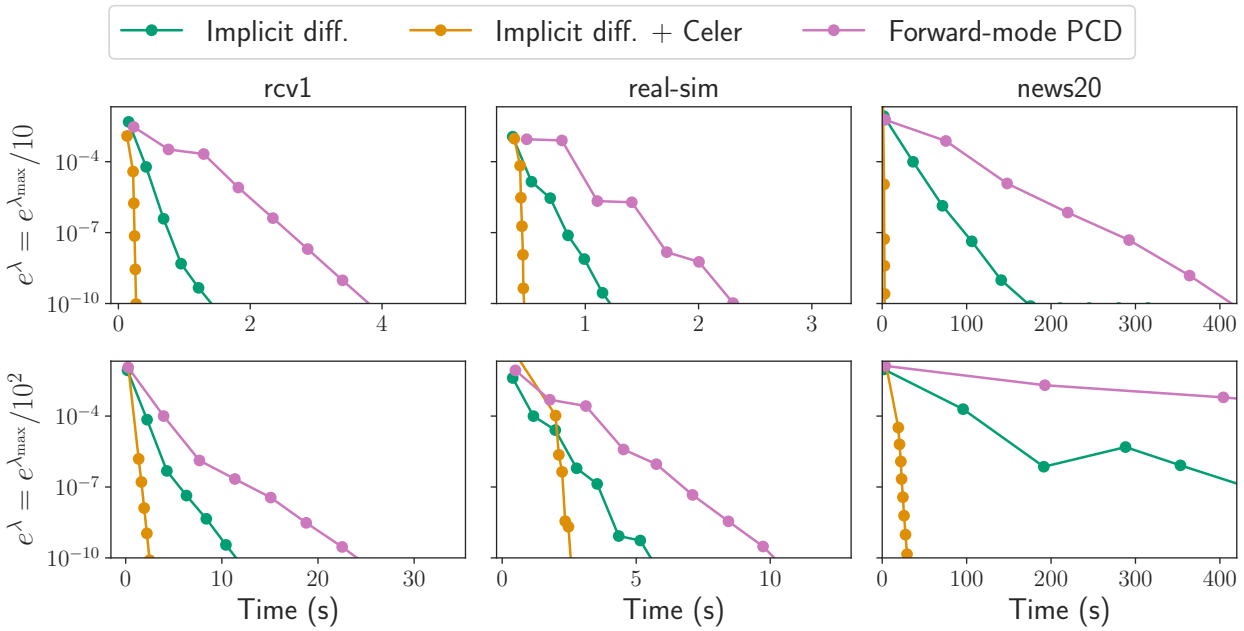

Figure 1.11 – **Lasso held-out, time to compute one hypergradient.** Absolute difference between the exact hypergradient (using $\hat{\beta}$) and the iterate hypergradient (using $\beta^{(k)}$) of the Lasso as a function of time. Results are for three datasets and two different regularization parameters. For the implicit differentiation, the lower problem is solved using proximal coordinate descent (Implicit diff.) or `Celer` (Massias et al. 2020, Implicit diff. + `Celer`).

Differentiating proximal algorithms was also considered in Deledalle et al. (2014) under the framework of weak differentiability. However, in their work they only considered the forward iterative differentiation. The major difference with our work is that thanks to the finite support identification property, we restrict the computation of the Jacobian on the support computing the gradient once a solution of the lasso-type models has been calculated.

Ochs et al. (2015) considered non-smooth lower level problems which can be solved iteratively with smooth updates. They proposed to compute the gradient of the bilevel optimization problem by computing the derivative of Bregman proximity operators which are supposed differentiable. By using the geometry of the non-smooth optimization problem via the generalized support, we are able to prove that classical proximal operators can be differentiated at optimum under mild assumptions. These iterative proximal algorithms such as proximal coordinate descent are state-of-the-art solvers for problems with ℓ_1 regularization for example leading to a faster convergence of the algorithms and faster computation of the gradient needed for the first order method.

1.5.2 Hyperparameter optimization in non-smooth convex learning

Capitalizing on the hypergradient computation method described in the previous section, we now turn towards the resolution of the bilevel optimization problem to perform hyperparameter optimization.

Main contribution. We propose a *heuristic* algorithm for the resolution of Problem (1.17) using a gradient descent algorithm, see Algorithm 1.

Algorithm 1 HEURISTIC GRADIENT DESCENT WITH APPROXIMATE GRADIENT

input : $X \in \mathbb{R}^{n \times p}, y \in \mathbb{R}^n, \lambda \in \mathbb{R}^r, (\epsilon_i)$
init : use_adaptive_step_size $=$ True
for $i = 1, \dots, iter$ **do**
  $\lambda^{\text{old}} \leftarrow \lambda$
  `// compute the value and the gradient`
  $\mathcal{L}(\lambda), \nabla\mathcal{L}(\lambda) \leftarrow$ Solution of the linear system given in Theorem 1.3
  if use_adaptive_step_size **then**
    $\alpha = 1/\|\nabla\mathcal{L}(\lambda)\|$
  $\lambda \mathrel{-}= \alpha\nabla\mathcal{L}(\lambda)$ `// gradient step`
  if $\mathcal{L}(\lambda) > \mathcal{L}(\lambda^{\text{old}})$ **then**
    use_adaptive_step_size $=$ False
    $\alpha \mathrel{/}= 10$
return λ

We compared our method based on Theorem 1.3 against other methods to select hyperparameters namely the *grid-search*, the *random search* and a Bayesian method named *SMBO* on the elastic net (Zou, 2006) estimator which requires 2 hyperparameters.

Figure 1.12 shows the trajectory of our first order method on the level set of the cross validation loss. It also illustrates that our method is faster than its competitors to find the two hyperparameters for the elastic net (Zou, 2006) that minimizes the cross validation loss (bottom of the figure).

Relation to previous works. Pedregosa (2016) studied hyperparameter optimization via implicit differentiation in the case of smooth bilevel optimization problem. In this case, he proved that his proposed algorithm *HOAG* converges to a stationnary point of Problem (1.17). Our algorithm is *heuristic* because we do not have guarantees on the convergence towards a solution of Problem (1.17) when the lower problem is non-smooth.

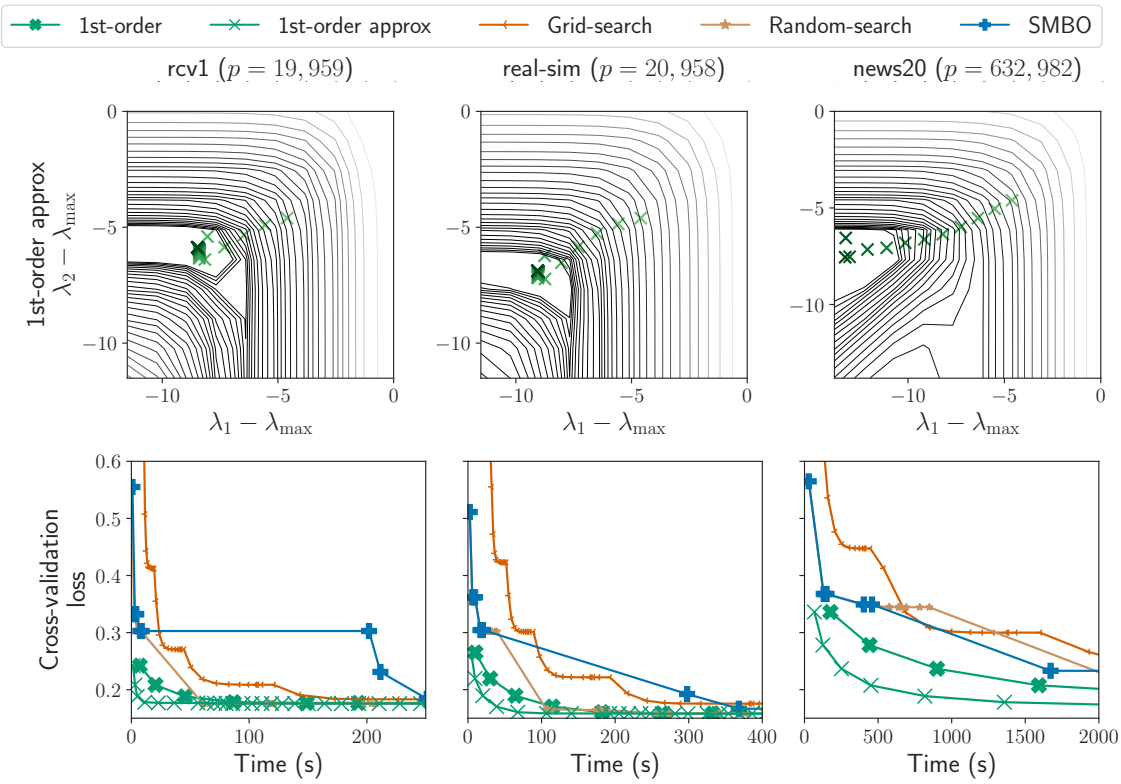

Figure 1.12 – **Elastic net cross-validation, time comparison (2 hyperparameters).** Level sets of the cross-validation loss (black lines, top) and cross-validation loss as a function of time (bottom) on *rcv1*, *real-sim* and *news20* datasets.

Extending this result to the non-smooth case is a very challenging question that is left for future work.

Open source package. The work presented in Chapter 6 and Chapter 7 has lead to the development of an open source package available in Python named **sparse-ho**. This package follows the same application programming interface than scikit-learn (Pedregosa et al., 2011).

The package is built following the structure of the bilevel optimization Problem (1.17). First, a model has to be chosen; several models widely used are already available for automatic hyperparameter selection such as the Lasso, the weighted Lasso, the elastic net, the sparse logistic regression, the SVM and the SVR. It corresponds to the lower problem in Problem (1.17). Then, a criterion is needed (outer problem) for the optimization of the hyperparameters, several options are available for regression such as the held-out Mean squared Error, the cross-validation Mean Squared Error or the SURE (Stein, 1981). The drawback of first order methods is that the criterion has to be *regular* and at least contin-

uous so the $0-1$ loss for classification cannot be used. However, we implemented the logistic loss, the smoothed hinge loss or the multiclass logistic loss. Finally, one has to choose a first order method to solve the bilevel optimization problem which can be gradient descent with constant step-size, line-search or ADAM (Kingma and Ba, 2014).

Examples, documentation and description of the package are available at https://qb3.github.io/sparse-ho/.

1.6 Estimating cells proportions with developed tools

The results of the two previous sections have lead to a new method to estimate cells proportions inside a tumor. The new estimator proposed takes advantage of the constrained Support Vector Regression which we can solve using the generalized SMO described above and the automatic hyperparameters selection. We could successfully apply this method on real datasets for the estimation of cells proportions and compared it with previous works.

Figure 1.13 compares our method with the state-of-the-art method named Cibersort (Newman et al., 2015) and the Simplex Ordinary Least Squares (SOLS) (Gong et al., 2011). We tested the ability of the different methods to be robust with respect to noise and unknown tumor content. We added log-Gaussian noise in the data $\mathcal{N}(0, \sigma^2)$ choosing σ as a percentage of $\sigma_{\max} = 11.6$ (taken from Newman et al. (2015)), the percentage was chosen between 0 and 1 with 30 different values. We artificially added tumor content in the data to replicate the condition of the cells inside a tumor and increased this tumor content from 0% to a 100% taking 30 different percentages. The estimation performance of the three estimators were compared using the Root Mean Squared Error (RMSE) and the correlation coefficient (R) between the true proportions and the estimated ones.

It can be seen that our proposed estimator the Simplex SVR has better estimation performance as the noise increased and is more robust to noise in comparison to Cibersort and the SOLS estimator.

As part of the Ph.D. thesis, I spent a year working one day per week inside a cancer institute being able to apply this estimator for clinical purposes as a replacement to teaching duties. This collaboration has lead to several articles being already published (Klopfenstein et al., 2019; Reichling et al., 2020).

The first article describes the use of immune cells proportions inside glioblastoma cancers to help refine patients' prognosis. This second article uses a random forest approach to

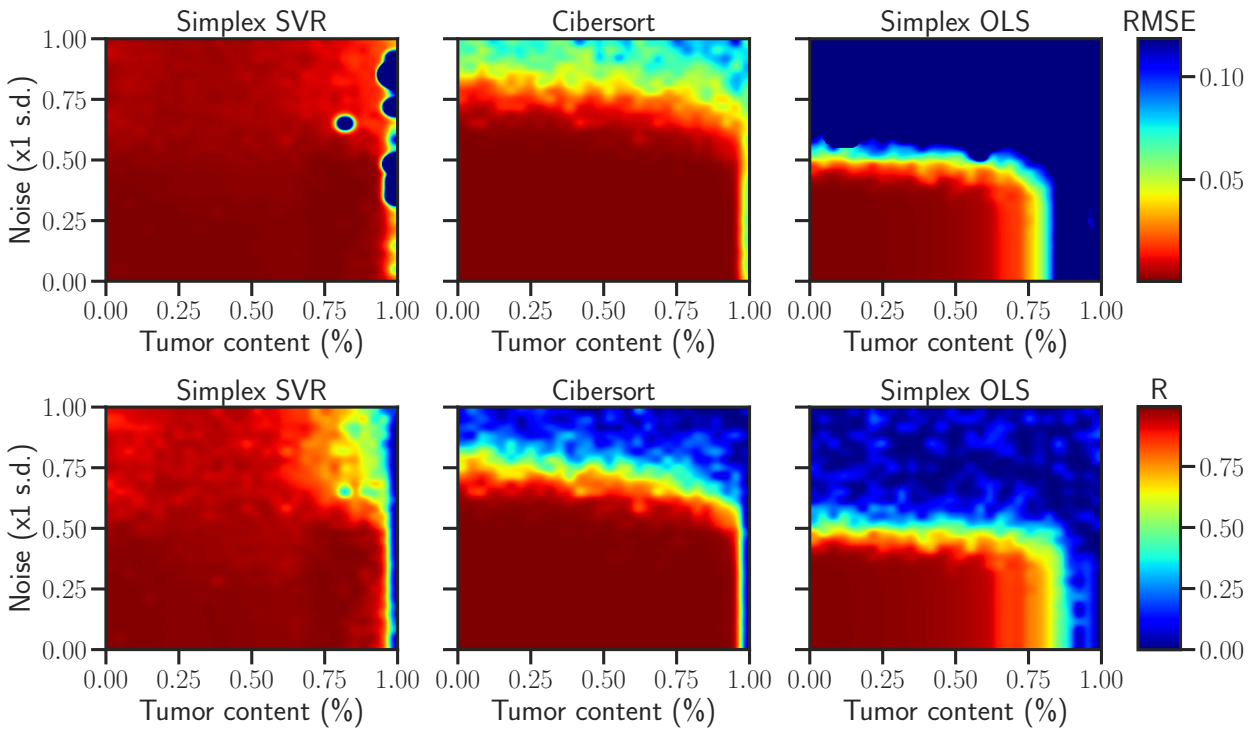

Figure 1.13 – **Robustness to noise and tumor content.** Heatmap representing the Root Mean Squared Error (RMSE) or the correlation coefficient (R) between the true proportions of cells and the estimated ones as a function of the tumor content percentage (x-axis) and the level of noise (y-axis). We compared three different estimator the Simplex Support Vector Regression, Cibersort and the Simplex Ordinary Least Squares.

automatically detect and compute cells of interest on an immunohistochemestry slide and it is not directly related to the topic of this thesis.

Another article is currently under review. In this paper, we show that the information about the proportions of cells is important and can be used for clinical purposes to estimate the risk of relapse for patients suffering from breast cancer and some results will be presented in Chapter 8 of this thesis.

1.7 Outline

This thesis is organized in 3 parts and 9 chapters that we briefly summarize here.

Chapter 2. This chapter presents the main mathematical tools used in this thesis. We recall classical definitions and results of convex analysis and convex optimization. We also present the coordinate descent algorithm which will be encountered throughout the rest of this manuscript.

Non-smooth optimization around coordinate descent. Part I focuses on the coordinate descent algorithm to solve composite minimization problems.

In Chapter 3, we prove an explicit equivalence between the partially smooth class of functions and the functions that are separable and locally $\mathcal{C}^2$ on the generalized support. Then, we prove model identification for the coordinate descent algorithm for this class of convex functions in Theorem 3.1 and prove local linear convergence once identification takes place (Theorem 3.2). We show on real datasets that our theoretical rates match the empirical convergence rates observed.

In Chapter 4, we study the Support Vector Regression optimization problem with linear constraints. We derive the dual formulation of the optimization problem and propose an efficient iterative algorithm to solve it. We prove in Theorem 4.1 the convergence of this algorithm towards a solution. We also show the utility of our proposed estimator on real datasets and various settings: non-negative regression, isotonic regression and regression with simplex constraints.

Publications/Preprints

- **Q. Klopfenstein** and S. Vaiter. *Linear Support Vector Regression with Linear Constraints*. Preprint. arXiv:1911.02306. 39 pages. 2019
- **Q. Klopfenstein***, Q. Bertrand*, A. Gramfort, J. Salmon, S. Vaiter. *Model identification and local linear convergence of coordinate descent*. Preprint. arXiv:2010.11825. 26 pages. 2020

Automatic hyperparameters selection for non-smooth convex models Part II focuses on hyperparameters selection for non-smooth convex models in machine learning.

In Chapter 5, we give an introduction on hyperparameters optimization. Selecting hyperparameters given a function that measures the quality of an estimator can be written as an bilevel optimization problem which can be solved efficiently using first order methods.

In Chapter 6, we also prove that the hypergradient of separable non-smooth lower problems can be computed using implicit differentiation (Theorem 6.1). Moreover, Theorem 6.4 prove that the convergence of the iterative Jacobian algorithm is linear once the structure induced by the sparsity has been identified. We show that our proposed method for the computation of the hypergradient outperforms classical tools based on generic differentiation packages or iterative differentiation.

In Chapter 7, we capitalise on the proposed hypergradient computation method to use a first order method to solve the bilevel optimization for the selection of hyperparameters. We then show that our proposed first order method outperforms state-of-the-art method for the setting of hyperparameters on various models: Lasso, elastic net, sparse logistic regression, SVM and the weighted Lasso.

Publications/Preprints

- Q. Bertrand*, **Q. Klopfenstein***, M. Blondel, S. Vaiter, A. Gramfort, J. Salmon. *Implicit differentiation of Lasso-type models for hyperparameter optimization*. ICML. 2020 11825. 26 pages. 2020
- Q. Bertrand*, **Q. Klopfenstein***, M. Blondel, S. Vaiter, A. Gramfort, J. Salmon. *Implicit differentiation for fast hyperparameter selection in non-smooth convex learning*. Preprint. 45 pages. 2021

Estimating cells proportions with developed tools Part III and final Chapter 8 aim at applying the mathematical tools introduced above for the estimation of cells proportions inside a tumor. We benchmark our proposed methods with different methods proposed in the literature and show that it can outperform the state-of-the-art methods in different settings. We also describe an example of clinical use of the information given by the estimation of the immune cells proportions for patients suffering of breast cancers. The information about the cells proportions can help the estimation of the risk of relapse for patients suffering from breast cancer.

- **Q. Klopfenstein**, V. Derangère, L. Arnould, M. Thibaudin, E. Limagne, F.Ghiringhelli, C. Truntzer, S. Ladoire. *Evaluation of Tumor Immune Contexture among Intrinsic Molecular Subtypes Helps to Predict Outcome in Early Breast Cancer*. Under review. 2021

The last chapter, Chapter 9, summarizes our contributions and discusses several possible future directions and open problems.

2 MATHEMATICAL BACKGROUND

Contents

In this chapter, we present the main mathematical tools that will be used throughout this manuscript. In particular, we present a first order optimization method called *coordinate descent* which will be the guideline of this work. We start in Section 2.1 by giving general mathematical notation that will appear in the following chapters. Section 2.2 recalls basic definitions and results of convex analysis, Section 2.3 introduces two algorithms used in smooth optimization and Section 2.4 presents their proximal variant for non-smooth optimization.

2.1 General notation

In the following, the symbol $\triangleq$ means equal by definition. We consider the vector space $\mathbb{R}^n$ with its associated Euclidean structure and its related scalar product denoted $\langle \cdot, \cdot \rangle$ given by:

$$\forall x, y \in \mathbb{R}^n, \langle x, y \rangle = \sum_{i=1}^{n} x_i y_i \ .$$

We also denote its associated ℓ_2 norm

$$\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2} \ .$$

Another important norm in this manuscript is the ℓ_1 norm given for all $x \in \mathbb{R}^n$ by

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \ .$$

We denote by $(e_1, \ldots, e_n)$ the Euclidean base of $\mathbb{R}^n$. The vector $\mathbb{1}$ denotes the vector with entries all equal to one. Let $x \in \mathbb{R}^n$ be a column vector, we denote by $x^\top$ its related row vector. We use the notation $\odot$ as the coordinatewise multiplication between two vectors $x, y \in \mathbb{R}^n$ *i.e.,*

$$x \odot y = (x_1 y_1, \ldots, x_n y_n)^\top \ .$$

For $x \in \mathbb{R}^n$, $\gamma \in \mathbb{R}^n_+$ the weighted norm is denoted

$$\|x\|_\gamma \triangleq \sqrt{\sum_{i=1}^{n} \gamma_i x_i^2} \ .$$

Matrices. The set of matrices of size n by p is denoted $\mathbb{R}^{n\times p}$. For a matrix $M \in \mathbb{R}^{n\times p}$, we will write that $M \succ 0$ ($M \succeq 0$) to say that it is definite positive (resp. semidefinite positive). We denote by $M^\top$ its transpose matrix and by $\mathrm{Tr}(M)$ its trace. We write the row wise multiplication of a vector $x \in \mathbb{R}^n$ and a matrix $M \in \mathbb{R}^{n\times p}$ by $x \odot M \in \mathbb{R}^{n\times p}$. The i^{th} row of a matrix M is denoted $M_{i:}$ and its j^{th} column by $M_{:j}$. The spectral radius of a matrix $M \in \mathbb{R}^{n\times n}$ is denoted $\rho(M) = \max_i |\lambda_i|$, where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of M (possibly complex). We denote by $\|M\|_2$ the matrix norm equal to the maximum singular value of M. The singular values of a matrix M will be written $\sigma_i(M)$.

The Mahalanobis norm of a vector $x \in \mathbb{R}^p$ for a given matrix $A \succ 0$ is noted

$$\|x\|_A \triangleq \sqrt{x^\top A^{-1} x} \ .$$

Sets. The set of integers $\{1, \ldots p\}$ is denoted by $[p]$. Let E be a set, for a subset $I \subseteq E$, we denote by I^c its complement and $|I|$ its cardinality. The Cartesian product between sets A and B is written $A \times B$. The closed ball of radius $\varepsilon > 0$ and center $x \in \mathbb{R}^n$ is denoted

by

$$\mathcal{B}(x, \varepsilon) = \{y \in \mathbb{R}^n : \|y - x\| \leq \varepsilon\} \ .$$

Derivatives. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, we write $\nabla f(x)$ its gradient at $x \in \mathbb{R}^n$ and $\nabla_i f(x)$ the i^{th} coordinate of the gradient at x. For a function $f : \mathbb{R}^n \to \mathbb{R}^p$, such that $x \mapsto \begin{pmatrix} f_1(x) \\ \vdots \\ f_p(x) \end{pmatrix}$, we denote $\mathcal{J}f(x) \in \mathbb{R}^{p \times n}$ its Jacobian at $x \in \mathbb{R}^n$ given by

$$\mathcal{J}f(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x) & \dots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_p}{\partial x_1}(x) & \dots & \frac{\partial f_p}{\partial x_n}(x) \end{pmatrix} \ .$$

We recall the definition of weak differentiability (see Evans and Gariepy (1992) for more details).

Definition 2.1. Let $f : x \in \Omega \to \mathbb{R}$, where Ω is an open subset of $\mathbb{R}^n$, be a function locally integrable. Its weak derivative with respect to x_i in Ω is the locally integrable function g_i such that

$$\int_\Omega g_i(x)\phi(x)dx = -\int_\Omega f(x)\frac{\partial \phi(x)}{\partial x_i}dx \ ,$$

holds for all functions ϕ that are continuously differentiable and with a compact support. A function is said to be weakly differentiable if all its weak partial derivatives exist.

For a vector-valued function $\psi : \mathbb{R}^p \times \mathbb{R}^r \mapsto \mathbb{R}^p$, we say that it is weakly differentiable if ψ_k is weakly differentiable for $k \in [m]$. Then we denote $\partial_1\psi$ the weak Jacobian *w.r.t.* the first variable and $\partial_2\psi$ the weak Jacobian *w.r.t.* the second variable. We denote $\hat{\mathcal{J}}_{(\lambda)} \triangleq (\nabla_\lambda \hat{\beta}_1^{(\lambda)}, \dots, \nabla_\lambda \hat{\beta}_p^{(\lambda)})^\top \in \mathbb{R}^{p \times r}$ the weak Jacobian of $\hat{\beta}^{(\lambda)}$ *w.r.t.* λ. Note that to ease the reading, we drop λ in the notation when it is clear from the context and use $\hat{\beta}$ and $\hat{\mathcal{J}}$.

If $\psi : \mathbb{R}^p \times \mathbb{R}^r \to \mathbb{R}$ and is twice differentiable, we denote by $\nabla_\beta \psi(\beta, \lambda) \in \mathbb{R}^p$ (resp. $\nabla_\lambda \psi(\beta, \lambda) \in \mathbb{R}^r$) its gradient *w.r.t.* the first variable (resp. the second variable). $\nabla_{\beta_j} f$ is the partial derivative $\frac{\partial \psi}{\partial \beta_j}$.

Definition 2.2 (Classes of regularity)**.** Let $U \subseteq \mathbb{R}^p$ be an open subset of $\mathbb{R}^p$. A function $f : U \to \mathbb{R}$ is said to be $\mathcal{C}^1$ on U if all its partial derivatives exist and are continuous.

The function f is said to be $\mathcal{C}^k$, $k \in \mathbb{N}$, on U if all its k^{th} order partial derivatives exist and are continuous.

Definition 2.3 (Local regularity)**.** Let $x \in \mathbb{R}^p$. The function f is said to be locally $\mathcal{C}^1$ around x if there exists $V \subseteq \mathbb{R}^p$ a neighborhood of x such that $f : V \to \mathbb{R}$ is $\mathcal{C}^1$. The function f is said to be locally $\mathcal{C}^k$, $k \in \mathbb{N}$, around x, if there exists $V \subseteq \mathbb{R}^p$ a neighborhood of x such that $f : V \to \mathbb{R}$ is $\mathcal{C}^k$.

We denote the right completion of the real line by $\bar{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$. The sign a of a vector $x \in \mathbb{R}^n$ is denoted $\operatorname{sign}(x)$ and each coordinate is obtained as $\operatorname{sign}(x)_i = \frac{x_i}{|x_i|}$ with the convention that $\operatorname{sign}(0) = 0$.

Model specific. The design matrix is $X \in \mathbb{R}^{n \times p}$ (corresponding to n samples and p features) and the observation vector is $y \in \mathbb{R}^n$. The underlying optimization problems of the models considered in this thesis can all be written under the general form:

$$\hat{\beta}^{(\lambda)} \in \operatorname*{arg\,min}_{\beta \in \mathbb{R}^p} \Phi(\beta, \lambda) \triangleq f(\beta) + \sum_{j=1}^{p} g_j(\beta_j, \lambda) \ , \tag{2.1}$$

where the regularization parameter, possibly multivariate, is denoted by $\lambda = (\lambda_1, \ldots, \lambda_r)^\top \in \mathbb{R}^r$. We denote $\hat{\beta}^{(\lambda)} \in \mathbb{R}^p$ the regression coefficients associated to λ *i.e.*, a solution of Problem (2.1) for a fixed λ.

2.2 Convex analysis

In this section, we recall essential elements of convex analysis that are the foundations of this work. We refer to Rockafellar (1997), Boyd and Vandenberghe (2004) and Hiriart-Urruty and Lemaréchal (1993a,b) for in depth references about this subject. We recall the definitions of the domain and epigraph of a function.

Definition 2.4 (Epigragh and domain)**.** The epigraph of a function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is a subset of $\mathbb{R}^{n+1}$ given by

$$\operatorname{epi}(f) \triangleq \{(x, t) \in \mathbb{R}^n \times \mathbb{R} : f(x) \leq t\} \ .$$

The (effective) domain of f is the set

$$\operatorname{dom}(f) \triangleq \{x \in \mathbb{R}^n : f(x) < +\infty\} \ .$$

A function f is said to be proper if its domain $\mathrm{dom}(f)$ is non-empty.

In the rest of this manuscript, we will consider functions that are proper even if not explicitly stated.

Definition 2.5 (Lower semicontinuity)**.** A function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is lower semi-continuous at $x \in \mathbb{R}^n$ if

$$\liminf_{z \to x} f(z) \geq f(x) \ .$$

The lower semicontinuity assumption on a function f is equivalent to the fact that $\mathrm{epi}(f)$ is closed (Rockafellar and Wets, 1998, Thm. 1.6), we also say that f is closed then.

Definition 2.6 (Convexity)**.** A set $C \subseteq \mathbb{R}^n$ is said to be convex if

$$\forall x, y \in C, \forall \lambda \in [0,1], \quad \lambda x + (1-\lambda) y \in C \ .$$

A function $f : \mathbb{R}^n \mapsto \bar{\mathbb{R}}$ is convex if

$$\forall x, y \in \mathrm{dom}\,(f), \forall \lambda \in [0,1], \quad f(\lambda x + (1-\lambda) y) \leq \lambda f(x) + (1-\lambda) f(y) \ .$$

It is said to be strictly convex if

$$\forall x, y \in \mathrm{dom}\,(f), \forall \lambda \in]0,1[, \quad x \neq y \Rightarrow f(\lambda x + (1-\lambda) y) < \lambda f(x) + (1-\lambda) f(y) \ .$$

It is strongly convex of modulus μ if for all $x, y \in \mathrm{dom}\,(f)$ and all $\lambda \in [0,1]$,

$$f(\lambda x + (1-\lambda) y) \leq \lambda f(x) + (1-\lambda) f(y) - \frac{\mu}{2} \lambda (1-\lambda) \|x - y\|^2 \ .$$

We recall the definition of the indicator of a set C.

Definition 2.7 (Indicator function)**.** Let $C \subseteq \mathbb{R}^n$ be a non-empty closed convex set. The indicator function of C is given by

$$\iota_C(x) \triangleq \begin{cases} 0, \text{ if } x \in C \\ +\infty, \text{ otherwise} \end{cases} \ .$$

An important example of indicator function of a set C will be the indicator of a closed interval, for example $[0, \lambda]$ for any $\lambda > 0$. This will appear in this manuscript as constraints

in the Support Vector Regression problem in Chapter 4.

Note also that a constrained optimization problem of the form:

$$x^\star \in \arg\min_{x \in C} f(x) \ ,$$

can be rewritten as an unconstrained one

$$x^\star \in \arg\min_{x \in \mathbb{R}^n} f(x) + \iota_C(x) \ .$$

Definition 2.8 (Affine hull)**.** The affine hull of a convex set C is the smallest affine set containing C *i.e.,*

$$\text{aff}(C) \triangleq \left\{ \sum_{i=1}^{k} \alpha_i x_i : k > 0, x_i \in C, \alpha_i \in \mathbb{R}, \sum_{i=1}^{k} \alpha_i = 1 \right\} \ .$$

Definition 2.9 (Linear hull)**.** The linear hull of a convex set C is the smallest linear subspace containing C denoted $\text{Lin}(C)$.

Definition 2.10 (Interior and relative interior)**.** The interior of a convex set C is denoted $\text{int}(C)$. Its relative interior $\text{ri}(C)$ is the interior relative to its affine hull.

Definition 2.11 (Subdifferential)**.** The subdifferential ∂f of a convex function f at x is the set

$$\partial f(x) \triangleq \{u \in \mathbb{R}^n : f(y) \geq f(x) + \langle u, y - x \rangle, \forall y \in \text{dom}(f)\} \ .$$

An element of $\partial f(x)$ is called a subgradient. Moreover if f is differentiable at x then $\partial f(x)$ is a singleton and $\partial f(x) = \{\nabla f(x)\}$. We now give a first order optimality condition based on the subdifferential of a function that extends the first order condition for differentiable function: $\nabla f(x^\star) = 0$.

Proposition 2.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function f, then $x^\star$ is a minimizer of f if, and only if, 0 is an element of the subdifferential of f at $x^\star$,* i.e., $0 \in \partial f(x^\star)$.

Definition 2.12 (Smoothness)**.** A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be L-smooth

if:

$$\forall x, y \in \mathbb{R}^n, f(x) \leq f(y) + \nabla f(y)^\top (x - y) + \frac{L}{2}\|x - y\|^2 \ .$$

It is equivalent to the fact that ∇f is a L-Lipschitz function *i.e.,*

$$\forall x, y \in \mathbb{R}^n, \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \ .$$

Moreover, if we suppose that f is twice differentiable then it also means for all $x \in \mathbb{R}^n$ that:

$$0 \preceq \nabla^2 f(x) \preceq L \operatorname{Id}_n \ .$$

Proximal operators.

Definition 2.13 (Proximal operators.)**.** The proximal operator of a convex function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is the function denoted $\operatorname{prox}_f : \mathbb{R}^n \to \mathbb{R}^n$ defined by

$$\operatorname{prox}_f(x) \triangleq \operatorname*{arg\,min}_{y \in \mathbb{R}^n} \frac{1}{2}\|y - x\|^2 + f(y) \ .$$

The function minimized on the righthand side is strongly convex if f is convex, hence has a unique minimizer for every $x \in \mathbb{R}^n$.

Proximal operators can be seen as a generalization of projection operators. If we consider a non-empty convex set C and its indicator function $f = \iota_C$, the proximal operator of f at x is the result of the following optimization problem

$$\operatorname{prox}_f(x) = \operatorname*{arg\,min}_{y \in C} \|y - x\|^2 \ ,$$

which is the Euclidean projection onto the set C. We now give an optimality condition for the proximal operator which establishes a link with the fixed point equation. This condition allows us to see the connection between the fixed point iteration induced by the proximal operator and the solution of a minimization problem.

Proposition 2.2. *Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ be a convex function and prox_f its proximal operator. Then we have the following property: $x^\star \in \mathbb{R}^n$ is a minimum of f if, and only if, $\operatorname{prox}_f(x^\star) = x^\star$ (see Bauschke and Combettes (2011, Chap. 10)).*

An important class of convex functions in this manuscript are the separable functions.

Definition 2.14. We say that $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is a separable function if f can be written for all $x \in \mathbb{R}^n$ as

$$f(x) = \sum_{i=1}^{n} f_i(x_i) \ .$$

The proximal operator of separable functions is equal to the proximal operator computed coordinatewise.

Proposition 2.3. *Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ be a convex and separable function* i.e., $f(x) = \sum_{i=1}^{n} f_i(x_i)$ *then the proximal operator of f is*

$$\big(\mathrm{prox}_f(x)\big)_i = \mathrm{prox}_{f_i}(x_i) \ .$$

This proposition is essential for the first order optimization algorithm called coordinate descent that we will present in Section 2.4.

We now give two examples of proximal operators that will be extensively used in the following chapters. The proximal operator of $f = \lambda\|\cdot\|_1 : \mathbb{R}^n \to \mathbb{R}$, is the soft thresholding operator, defined coordinatewise for $x \in \mathbb{R}^n$ and $\lambda > 0$:

$$\mathrm{prox}_{\lambda\|\cdot\|}(x)_i = \mathrm{ST}(x, \lambda)_i = \mathrm{sign}(x_i)(|x_i| - \lambda)_+, \quad \forall i \in [n] \ .$$

The second example is the proximal operator of the indicator function of the set $[0, \lambda]^n$, for any $\lambda > 0$. It is simply the projection onto the interval $[0, \lambda]$ coordinatewise, *i.e.,*

$$\mathrm{prox}_{\iota_{[0,\lambda]^n}}(x)_i = \mathcal{P}_{[0,\lambda]}(x_i) = \max(\min(0, x_i), \lambda), \quad \forall i \in [n] \ .$$

2.3 Smooth optimization

In this section, we present two algorithms that can be used to iteratively approach the minimum of a convex and differentiable function, namely the gradient descent algorithm and the coordinate descent algorithm.

Gradient descent. Let us consider the following optimization problem:

$$x^\star \in \arg\min_{x \in \mathbb{R}^p} \Phi(x) \ , \tag{2.2}$$

where $\Phi : \mathbb{R}^p \to \bar{\mathbb{R}}$ is a convex L-smooth function. Solving Equation (2.2) can be done using an iterative gradient descent algorithm. The solution $x^\star \in \mathbb{R}^p$ will be approximated by the following scheme, for any $0 < \gamma < \frac{2}{L}$:

$$x^{(k+1)} = x^{(k)} - \gamma \nabla \Phi(x^{(k)}) \ .$$

Note that in practice, we will choose $\gamma = \frac{1}{L}$ when L is known.

Under the assumption that Φ is L-smooth, it is possible to obtain the following rate of convergence (Nesterov, 2004, Corollary 2.1.2):

$$\Phi(x^{(k)}) - \Phi(x^\star) \leq \frac{2L\|x^{(0)} - x^\star\|^2}{k+4} \ .$$

If we assume that Φ is μ-strongly convex, we obtain a linear rate of convergence:

$$\Phi(x^{(k)}) - \Phi(x^\star) \leq \left(1 - \frac{\mu}{L}\right)^k (\Phi(x^{(0)}) - \Phi(x^\star)) \ .$$

Coordinate descent. Another first order method can be used to solved smooth problem like Equation (2.2). This optimization algorithm is simple and is widely used to solve large scale problems in machine learning (see for example Wright (2015)). The main idea of the algorithm is the following: choose the index of a coordinate that will be updated and perform an exact (or approximated) minimization with respect to this chosen variable while all the others remain constant. In terms of trajectory of the iterative algorithm, each update only moves the iterate along one axis as shown in Figure 2.1. There exists several variants of coordinate descent depending on the update rule and the index selection. Concerning the update rule, a natural way to use coordinate descent is to perform exact coordinate minimization iteratively, for $j \in [p]$:

$$x_j^{(k+1)} \in \arg\min_{x_j \in \mathbb{R}} \Phi\left(x_1^{(k+1)}, \ldots, x_{j-1}^{(k+1)}, x_j, x_{j+1}^{(k)}, \ldots, x_p^{(k)}\right) \ . \tag{2.3}$$

Breaking up the optimization Equation (2.2) into subproblems of the form Equation (2.3) makes practical sense if the minimization problem of size one is *easy* to solve. Supposing that the minimum of each subproblem is unique, it is possible to show that exact coor-

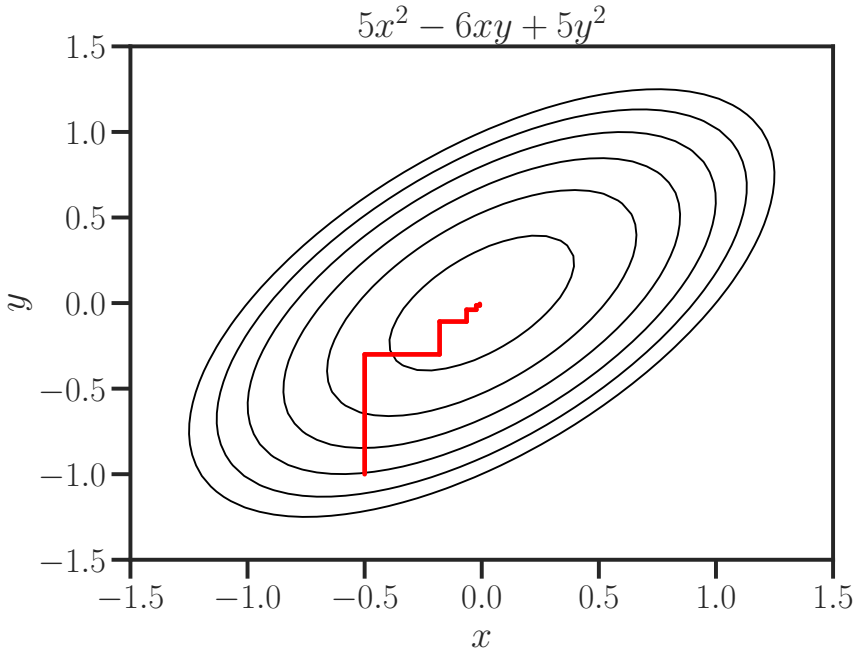

Figure 2.1 – **Coordinate descent algorithm.** Coordinate descent applied to the function $f(x, y) = 5x^2 - 6xy + 5y^2$ with initial point $(-0.5, -1.0)$. The algorithm updates one coordinate at a time while the other one remains constant.

dinate minimization converges towards a solution of Equation (2.2) when Φ is smooth (Bertsekas, 2015, Prop. 6.5.1).

Computing exact minimization at each step might be difficult in some cases. Another version of coordinate descent is used for differentiable functions relying on a quadratic approximation of Φ, the update can then be seen as a coordinate gradient descent step. Before giving the update formula, we need to define the coordinatewise Lipschitz constant of a function.

Definition 2.15. Let $f : \mathbb{R}^p \to \mathbb{R}$ be a function. We say that $L_j > 0$ is a local Lipschitz constant of f, if for all $x \in \mathbb{R}^p$, $h \in \mathbb{R}$:

$$||f_j(x + e_j h) - f_j(x)|| \leq L_j |h| \ .$$

Algorithm 2 CYCLIC COORDINATE DESCENT

input : $L_1, \dots, L_P \in \mathbb{R}_+, n_{\text{iter}} \in \mathbb{N}, x^{(0)} \in \mathbb{R}^p$

for $k = 0, \dots, n_{iter}$ **do**

  $x^{(0,k)} \leftarrow x^{(k)}$

  for $j = 1, \dots, p$ **do** `// index selection`

    $x^{(j,k)} \leftarrow x^{(j-1,k)}$

    $x_j^{(j,k)} \leftarrow x_j^{(j-1,k)} - \frac{1}{L_j} \nabla_j \Phi(x^{(j-1,k)})$ `// Coordinate gradient step`

  $x^{(k+1)} \leftarrow x^{(p,k)}$

return $x^{n_{iter}+1}$

The update for the coordinate gradient descent update then reads:

$$x_j^{(j,k)} = x_j^{(j-1,k)} - \frac{1}{L_j} \nabla_j \Phi(x^{(j-1,k)}) \ ,$$

where L_j is the local Lipschitz constant of $\nabla \Phi$.

As shown in Algorithm 2, the notation $x_j^{(j,k)}$ allows us to keep track of the epoch, meaning the number of full passes on the coordinates denoted by k and at the same time the current updated coordinate j. Once all the coordinates have been updated once, we increase the number of epoch by one and repeat until a stopping criterion is met. This notation can be found in Beck and Tetruashvili (2013).

Note that if Φ is a quadratic function both approaches, the exact minimization and the coordinate gradient step with step size $\frac{1}{L_j}$, lead to the same iterative scheme.

We now present different ways to select the coordinate $j \in [p]$ that will be updated, also called the index-rule. There exists three different families of index rules:

- The most natural approach to pick an index is going through all the coordinates $j \in [p]$ cyclically. This choice of index is deterministic and is easy to implement. There also exists some variants which allow the permutation of the set $[p]$ after each epoch (*i.e.*, a complete pass on all coordinates) or once and for all. The algorithm obtained with this choice of index selection is simple to implement and is given in Algorithm 2. As references for cyclic coordinate descent, we can refer to Luo and Tseng (1992).

- Another possibility is to select the coordinate at random for a given random distribution. The easiest distribution to choose is the uniform distribution over all the coordinates as in Nesterov (2012) but other choices can be made to improve convergence based on feature importance score (Zhang, 2004).

- Finally the last index rule is the greedy coordinate descent. For example, the index can be chosen such as the decrease in the objective function is maximal or choosing the index corresponding to the largest gradient coordinate in absolute value *i.e.*,

$$j = \arg\max_{i \in [p]} \|\nabla_j f(x^{(j-1,k)})\| \ .$$

  This index rule is called the Gauss-Southwell rule and other variants exist as well such as the Gauss-Southwell Lipschitz selection (Nutini et al., 2015).

Linear regression. We now give an example of a smooth optimization problem that can be solved using gradient descent or coordinate descent. Let us consider the following Ordinary Least Squares (OLS) problem for $b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times p}$:

$$x^\star \in \arg\min_{x \in \mathbb{R}^p} \frac{1}{2}\|Ax - b\|^2 \ .$$

Applying the gradient descent scheme to this problem leads to the following update with $L = \|X\|_2^2$:

$$x^{(k+1)} = x^{(k)} - \frac{1}{L} A^\top (Ax^{(k)} - b) \ .$$

Considering the cyclic coordinate gradient descent for this problem leads to the coordinates updates with $L_j = \|X_{:j}\|^2$:

$$x_j^{(j,k)} = x_j^{(j-1,k)} - \frac{1}{L_j} A_{:j}^\top (Ax^{(j-1,k)}) - b) \tag{2.4}$$

Figure 2.2 shows the different convergence trajectories of the linear regression optimization problem. The curves represent the objective function at iteration k minus the objective function at the optimum for the gradient descent and at epoch k for the coordinate descent. The question here is to know if it is fair to compare a whole epoch of coordinate descent versus one iteration of gradient descent. The response is yes in terms of computation cost, if we use cleverly the fact that we update one coordinate at a time. An iteration of gradient descent cost $\mathcal{O}(np)$ operations. When considering coordinate descent, one can notice that the vector $r = Ax^{(0)} - b$ can be computed once at the beginning of the process. Then Equation (2.4) becomes

$$x_j^{(j,k)} = x_j^{(j-1,k)} - \frac{1}{L_j} A_{:j}^\top r \ , \tag{2.5}$$

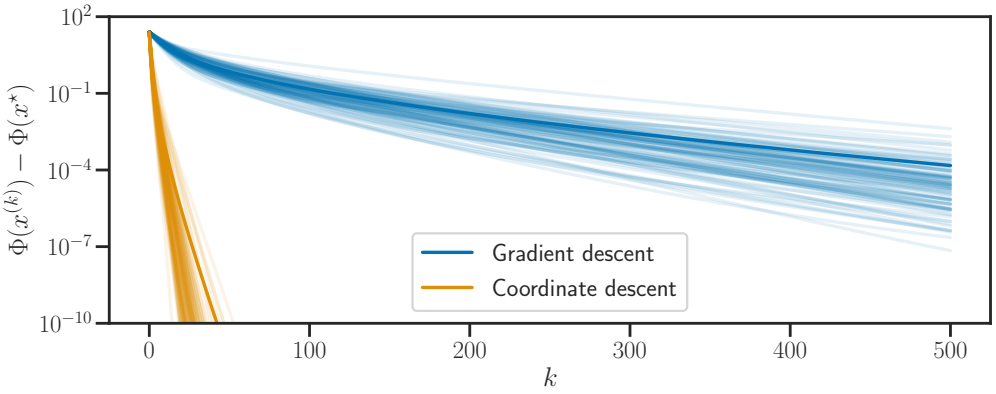

Figure 2.2 – **Coordinate descent versus Gradient descent.** Convergence speed of coordinate descent versus gradient descent to solve 50 linear regression problems with $A \in \mathbb{R}^{50\times 20}$ and $b \in \mathbb{R}^{50}$.

and r can be updated by $r = (x_j^{\text{old}} - x_j^{(k,j)})A_{:j}$. Therefore, an update cost $\mathcal{O}(n)$ operations: $\mathcal{O}(n)$ for the multiplication $A_{:j}^\top r$ and $\mathcal{O}(n)$ to keep r updated. A whole epoch then costs $\mathcal{O}(np)$, the same as gradient descent.

Empirically, coordinate descent is often much faster than gradient descent as illustrated in Figure 2.2. There is a lack of theory backing up these empirical observations but a possible explanation is that we use the information about updated gradient for each coordinate in the coordinate descent algorithm whereas for the gradient descent algorithm, the gradient is only updated at each iteration. Coordinate descent is an efficient algorithm to solve some smooth optimization problems but its main advantage is its efficiency to solve composite minimization problems that arise in machine learning as we will now describe.

2.4 Non-smooth optimization

This section contains the class of optimization problems that will be considered in this entire work. We present how the gradient descent and coordinate descent algorithms can be extended to solve composite minimization problems.

Proximal gradient descent. Let us now consider a composite minimization problem such that

$$x^\star \in \arg\min_{x\in\mathbb{R}^p} \Phi(x) = f(x) + g(x) \ , \tag{2.6}$$

where $f : \mathbb{R}^p \to \mathbb{R}$ is a L-smooth convex function and $g : \mathbb{R}^p \to \bar{\mathbb{R}}$ is convex, proper and closed, possibly non-differentiable. Optimality conditions for Equation (2.6) can be written as follows, for any $\gamma > 0$:

$$\begin{aligned} -\nabla f(x^\star) \in \partial g(x^\star) &\Leftrightarrow \gamma \nabla f(x^\star) \in -\gamma \partial g(x^\star) \\ &\Leftrightarrow x^\star + \gamma \nabla f(x^\star) \in x^\star - \gamma \partial g(x^\star) \\ &\Leftrightarrow x^\star \in x^\star - \gamma \nabla f(x^\star) - \gamma \partial g(x^\star) \ . \end{aligned}$$

Using the following characterization of the proximal operator of a function $g : \mathbb{R}^p \to \mathbb{R}$, $\forall v \in \mathbb{R}^p$ such that $u = \operatorname{prox}_{\gamma g}(v)$ we have that:

$$\frac{1}{\gamma}(v - u) \in \partial g(u) \text{ hence } u \in v - \gamma \partial g(u) \ . \tag{2.7}$$

Using Equation (2.7), we obtain the following fixed point equation (Combettes and Wajs, 2005):

$$x^\star = \operatorname{prox}_{\gamma g}\left(x^\star - \gamma \nabla f(x^\star)\right) \ .$$

This fixed point equation suggest an iterative algorithm to obtain a solution of Equation (2.6) that is called proximal gradient descent algorithm or Forward-Backward algorithm (Lions and Mercier, 1979; Combettes and Wajs, 2005) and its iteration update reads:

$$x^{(k+1)} = \operatorname{prox}_{\gamma g}\left(x^{(k)} - \gamma \nabla f(x^{(k)})\right) \ .$$

The next iterate is obtained as the composition of a gradient step on the smooth function f composed with the proximal operator of the non-differentiable function g. Under the assumptions that f is L-smooth and g is a proper, closed, convex function; this method converges with rate $\mathcal{O}(1/k)$ (same as gradient descent) with a fixed step size $\gamma = \frac{1}{L}$ (which we will choose in practice when L is known). Actually, the convergence is guaranteed for step size $0 < \gamma < \frac{2}{L}$ but the method is no longer a "majorization-minimization" method for step sizes larger than $\frac{1}{L}$ (Combettes and Pesquet, 2011).

Proximal coordinate descent. The proximal gradient descent algorithm can be seen as the counter part of the gradient descent method to solve composite minimization problem (Equation (2.6)). Generally, the coordinate descent algorithm does not have convergence guarantees without more assumptions. However, if we assume that g is a separable func-

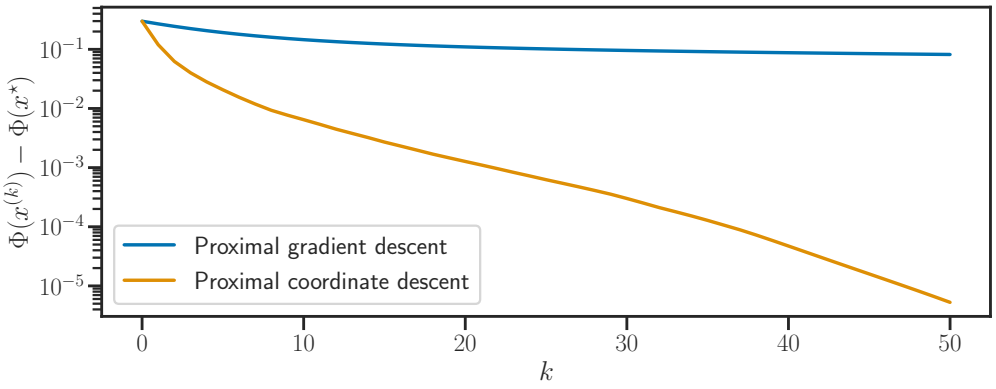

Figure 2.3 – **Proximal coordinate descent versus Proximal gradient descent.** Convergence speed of proximal coordinate descent and proximal gradient descent to solve the Lasso optimization problem on the *colon* dataset from *libsvm*. The regularization parameter was chosen as $\lambda = \frac{\lambda_{\max}}{10}$, where $\lambda_{\max} = \left\|X^\top y\right\|_\infty / n$.

tion (Definition 2.14). The composite minimization problem can then be written:

$$x^\star \in \arg\min_{x \in \mathbb{R}^p} \Phi(x) = f(x) + \underbrace{\sum_{j=1}^{p} g_j(x_j)}_{\triangleq g(x)} \quad . \tag{2.8}$$

Equation (2.8) will be encountered throughout this thesis (Chapters 3, 6 and 7) and we suppose that these general assumptions remain true.

Assumption 2.1 (Smoothness)**.** $f : \mathbb{R}^p \to \mathbb{R}$ is a convex and differentiable function, with a Lipschitz gradient. We denote by L its global Lipschitz constant and by L_j its local Lipschitz constants.

Assumption 2.2 (Proper, closed, convex)**.** For any $j \in [p]$, g_j is proper, closed and convex.

Assumption 2.3 (Existence)**.** The problem admits at least one solution:

$$\arg\min_{x \in \mathbb{R}^p} \Phi(x) \neq \emptyset \quad .$$

Assumption 2.4 (Non degeneracy)**.** The problem is non-degenerated:

$$\forall x^\star \in \arg\min_{x \in \mathbb{R}^p} \Phi(x), \quad -\nabla f(x^\star) \in \mathrm{ri}\left(\partial g(x^\star)\right) \quad .$$

Assumptions 2.1 to 2.3 ensure that one can use proximal coordinate descent to solve Equation (2.8) and convergence towards a solution is guaranteed. Assumption 2.4 can be seen

as a generalization of qualification constraints (Hare and Lewis, 2007, Sec. 1).

Definition 2.16 (Regularity of the composite minimization)**.** We will say that Equation (2.8) is regular if Assumptions 2.1 to 2.3 hold for the composite minimization problem.

The important property here is that the proximal operator of a separable function g is the proximal operator of each g_j coordinatewise. This result directly leads to a fixed point equation coordinatewise with local step size $\gamma_j > 0$:

$$x_j^\star = \operatorname{prox}_{\gamma_j g_j}\left(x_j^\star - \gamma_j \nabla_j f(x^\star)\right) \quad .$$

In this setting, the coordinate descent algorithm has a proximal version counter part which will be the common thread of the manuscript. A coordinate update writes:

$$x_j^{(j,k)} = \operatorname{prox}_{\gamma_j g_j}\left(x_j^{(j-1,k)} - \gamma_j \nabla_j f(x^{(j-1,k)})\right) \quad .$$

The separability assumption on g might seem restrictive but in modern machine learning, a wide class of estimators are obtained by solving Equation (2.6) with a separable non-differentiable function g. The most famous examples are probably the regularized optimization problems such as the Lasso (Tibshirani, 1996), the sparse logistic regression (Hoerl and Kennard, 1970) and the Elastic net (Zou, 2006). Another family of estimators can be obtained via solving an optimization with coordinate descent: the Support Vector Machine (Boser et al., 1992) estimator for classification and regression.

The (proximal) coordinate descent algorithm has proved to be very efficient in large scale problems and is nowadays implemented in the most used machine learning packages like scikit-learn (Pedregosa et al., 2011) and glmnet (Friedman et al., 2010). In practice, the step sizes γ_j will be chosen to be equal to the inverse of the local Lipschitz constants *i.e.*, $\gamma_j = \frac{1}{L_j}$. This algorithm is guaranteed to converge towards a solution of Equation (2.6) (Tseng and Yun, 2009) and has a convergence rate of $\mathcal{O}(1/k)$ (Sun and Hong, 2015). Although, its theoretical convergence rate is the same as the proximal gradient descent, empirically the proximal coordinate descent is faster than proximal gradient descent as illustrated in Figure 2.3 where there proximal coordinate descent outperforms the proximal gradient descent by several orders of magnitude.

Notion of support. The non-smooth function g in Equation (2.8) generates solution that are structured as we will see in more details in Chapter 3. The information about the structure is carried out by the notion of support that is usually well known for the ℓ_1-norm but can be extended to more general functions by the following definition.

Definition 2.17 (Generalized support, Nutini et al. 2019, Def. 1)**.** For a vector $x \in \mathbb{R}^p$, its *generalized support* $\mathcal{S}_x \subseteq [p]$ is the set of indices $j \in [p]$ such that g_j is differentiable at x_j:

$$\mathcal{S}_x \triangleq \{j \in [p] : \partial g_j(x_j) \text{ is a singleton}\} \ .$$

An iterative algorithm is said to achieve **finite support identification** if its iterates $x^{(k)}$ converge to $x^\star \in \arg\min_{x \in \mathbb{R}^p} \Phi(x)$, and there exists $K \geq 0$ such that for all $j \notin \mathcal{S}_{x^\star}$, for all $k \geq K, x_j^{(k)} = x_j^\star$.

Examples. For the ℓ_1 norm (promoting sparsity) with $\lambda > 0$, $g_j(x_j^\star) = \lambda |x_j^\star|$, the generalized support is $\mathcal{S}_{x^\star} \triangleq \{j \in [p] : x_j^\star \neq 0\}$. This set corresponds to the indices of the non-zero coefficients, which is the usual support definition. For the SVM estimator, $g_j(x_j^\star) = \iota_{[0,C]}(x_j^\star)$ for a given hyperparameter $C > 0$. This function is non-differentiable at 0 and at C. The generalized support for the SVM estimator then corresponds to the set of indices such that $x_j^\star \in]0, C[$.

This notion of support will be key to show local convergence properties and to take advantage of the general sparsity pattern to speed-up computation of algorithms.

Part I

Non-smooth optimization around coordinate descent

3 Local linear convergence of coordinate descent

Contents

In this chapter, we prove convergence properties of the proximal coordinate descent algorithm. Non-smooth optimization problems arising in machine learning are often highly structured and the question of structure recovery for iterative algorithm is key. We first prove, in Section 3.2, that there is an explicit equivalence between separable functions that are locally $\mathcal{C}^2$ around the optimum on the generalized support and the class of partly smooth functions. In Section 3.3, we show that cyclic proximal coordinate descent achieves model identification in finite time. This result already existed in the literature but the proof and the tools that we propose here are different. In addition, we prove in Section 3.4 explicit local linear convergence rates for coordinate descent. Finally in Section 3.5, we illustrate these results on various estimators and on real datasets. Our experiments demonstrate that these rates match empirical results well.

3.1 Introduction

3.1.1 Coordinate descent

Over the last two decades, coordinate descent (CD) algorithms have become a powerful tool to solve large scale optimization problems (Friedman et al., 2007, 2010). Many applications coming from machine learning or compressed sensing have lead to optimization problems that can be solved efficiently via CD algorithms: the Lasso (Tibshirani, 1996; Chen et al., 1998), the elastic net (Zou and Hastie, 2005) or support-vector machine (Boser et al., 1992). All the previously cited estimators are based on an optimization problem which can be written:

$$x^\star \in \arg\min_{x \in \mathbb{R}^p} \{\Phi(x) \triangleq f(x) + \underbrace{\sum_{j=1}^{p} g_j(x_j)}_{\triangleq g(x)}\} \ , \tag{3.1}$$

with f a convex smooth (*i.e.*, with a Lipschitz gradient) function and g_j proper, closed and convex functions. In the past twenty years, the popularity of CD algorithms has greatly increased due to the well suited structure of the new optimization problems mentioned above (*i.e.*, separability of the non-smooth term), as well as the possible parallelization of the algorithms (Fercoq and Richtárik, 2015).

The key idea behind CD (Algorithm 3) is to solve small and simple subproblems iteratively until convergence. More formally, for a function $\Phi : \mathbb{R}^p \mapsto \mathbb{R}$, the idea is to minimize successively one dimensional functions $\Phi_{|x_j} : \mathbb{R} \mapsto \mathbb{R}$, updating only one coordinate at a time, while the others remain unchanged. There exists many variants of CD algorithms, the main branching being:

- **The index selection.** There are different ways to choose the index of the updated coordinate at each iteration. The main variants can be divided in three categories, **cyclic** CD (Tseng and Yun, 2009) when the indices are chosen in the set $[p]$ cyclically. **Random** CD (Nesterov, 2012), where the indices are chosen following a given random distribution. Finally, **greedy** CD *picks* an index, optimizing a given criterion: largest decrease of the objective function, or largest gradient norm (Gauss-Southwell rule), for instance.
- **The update rule.** There also exists several possible schemes for the coordinate update: exact minimization, coordinate gradient descent or prox-linear update (see Shi et al. 2016, Sec. 2.2 for details).

In this chapter, we will focus on **cyclic** CD with **prox-linear** update rule (Algorithm 3): a popular instance, *e.g.*, the one coded in popular packages such as `glmnet` (Friedman et al., 2007) or `scikit-learn` (Pedregosa et al., 2011).

Algorithm 3 PROXIMAL COORDINATE DESCENT

input : $\gamma_1, \dots, \gamma_p \in \mathbb{R}_+, n_{\text{iter}} \in \mathbb{N}, x^{(0)} \in \mathbb{R}^p$

for $k = 0, \dots, n_{iter}$ **do**

  $x^{(0,k)} \leftarrow x^{(k)}$

  for $j = 1, \dots, p$ **do** `// index selection`

    $x^{(j,k)} \leftarrow x^{(j-1,k)}$

    $x_j^{(j,k)} \leftarrow \text{prox}_{\gamma_j g_j}\left(x_j^{(j-1,k)} - \gamma_j \nabla_j f(x^{(j-1,k)})\right)$ `// Update rule`

  $x^{(k+1)} \leftarrow x^{(p,k)}$

return $x^{n_{iter}+1}$

Among the methods of coordinate selection, **random** CD has been extensively studied, especially by Nesterov (2012) for the minimization of a smooth function f. It was the first paper proving global non-asymptotic $\mathcal{O}(1/k)$ convergence rate in the case of a smooth and convex f. This work was later extended to composite optimization $f + \sum_j g_j$ for non-smooth separable functions (Richtárik and Takáč, 2014; Fercoq and Richtárik, 2015). Refined convergence rates were also shown by Shalev-Shwartz and Tewari (2011); Shalev-Shwartz and Zhang (2013). These convergence results have then been extended to coordinate descent with equality constraints (Necoara and Patrascu, 2014) that induce non-separability as found in the SVM dual problem in the presence of a¡ bias term. Different distributions have been considered for the index selection such as uniform distribution (Fercoq and Richtárik, 2015; Nesterov, 2012; Shalev-Shwartz and Tewari, 2011; Shalev-Shwartz and Zhang, 2013), importance sampling (Leventhal and Lewis, 2010; Zhang, 2004) and arbitrary sampling (Necoara and Patrascu, 2014; Qu and Richtárik, 2016a,b).

On the cyclic coordinate descent side, Luo and Tseng (1992); Tseng (2001); Tseng and Yun (2009); Razaviyayn et al. (2013) have shown convergence results for (block) CD algorithms for non-smooth optimization problems (without rates[1]). Then, Beck and Tetruashvili (2013) showed $\mathcal{O}(1/k)$ convergence rates for Lipschitz convex functions and linear convergence rates in the strongly convex case. Saha and Tewari (2013) proved $\mathcal{O}(1/k)$ convergence rates for composite optimization $f + \|\cdot\|_1$ under "isotonicity" condition. Sun and Hong (2015); Hong et al. (2017) have extended the latter results and showed $\mathcal{O}(1/k)$ convergence rates with improved constants for composite optimization $f + \sum_j g_j$. Li et al. (2017) have extended the work of Beck and Tetruashvili (2013) to the non-smooth case and

[1]Note that some local rates are shown in Tseng and Yun (2009) but under some strong hypothesis.

refined their convergence rates in the smooth case. Finally, as far as we know, the work by Xu and Yin (2017) is the first one tackling the problem of local linear convergence. They have proved local linear convergence under the very general Kurdyka-Lojasiewicz hypothesis, relaxing convexity assumptions. Following the line of work by Liang et al. (2014), we use a more restrictive framework that allows to achieve finer results: model identification as well as improved local convergence results.

3.1.2 Model identification

Non-smooth optimization problems coming from machine learning such as the Lasso or the support-vector machine (SVM) generally generate solutions lying onto a low-complexity model. For the Lasso, for example, a solution $x^\star$ has typically only a few non-zeros coefficients: it lies on the model set $T_{x^\star} = \{u \in \mathbb{R}^p : \text{supp}(u) \subseteq \text{supp}(x^\star)\}$, where $\text{supp}(x)$ is the support of x, *i.e.*, the set of indices corresponding to the non-zero coefficients. A question of interest in the literature is: does the algorithm achieve model identification after a finite number of iterations? Formally, does it exist $K > 0$ such that for all $k > K$, $x^{(k)} \in T_{x^\star}$? For the Lasso the question boils down to "does it exist $K > 0$ such that for all $k > K$, $\text{supp}(x^{(k)}) \subseteq \text{supp}(x^\star)$"? This finite time identification property is paramount for features selection (Tibshirani, 1996), but also for potential acceleration methods (Massias et al., 2018) of the CD algorithm, as well as model selection (Bertrand et al., 2020).

Finite model identification was first proved in Bertsekas (1976) for the projected gradient method with non-negative constraints. In this case, after a finite number of steps the sparsity pattern of the iterates is the same as the sparsity pattern of the solution. It means that for k large enough, $x_i^{(k)} = 0$ for all i such that $x_i^\star = 0$. Then, many other results of finite model identification have been shown in different settings and for various algorithms. For the projected gradient descent algorithm, identification was proved for polyhedral constraints (Burke and Moré, 1988), for general convex constraints (Wright, 1993), and even non-convex constraints (Hare and Lewis, 2004). More recently, identification was proved for proximal gradient algorithm (Lions and Mercier, 1979; Combettes and Wajs, 2005), for the ℓ_1 regularized problem (Hare, 2011). Liang et al. (2014, 2017); Vaiter et al. (2018) have shown model identification and local linear convergence for proximal gradient descent. These results have then been extended to other popular machine learning algorithms such as SAGA, SVRG (Poon et al., 2018) and ADMM (Poon and Liang, 2019). Some identification results have been shown for CD on specific models (She and Schmidt, 2017; Massias et al., 2019) or variants of CD (Wright, 2012), in general, under restrictive hypothesis. Model identification property for CD algorithm is not new and was first stated

in Nutini et al. (2017). We do not claim novelty here but the proof technique and the point of view is different in our approach.

3.2 Structure for separable non-smooth convex functions

As stated before, the solutions of the Lasso are structured. Using an iterative algorithm like the coordinate descent to find an approximate solution (since we stop after a finite number of iterations) brings the question of structure recovery. For the Lasso, the underlying structure, also called model (Candès and Recht, 2012), is identified by the Forward-Backward algorithm. It means that after a finite number of iterations, the iterative algorithm leads to an approximated solution that shares a similar structure than the true solution of the optimization problem (Liang et al., 2014; Vaiter et al., 2018; Fadili et al., 2018). For the Lasso, the underlying model is related to the notion of support: *i.e.,* the non-zero coefficients and this notion of sparsity can be generalized for the case of completely separable functions using Definition 2.17. This notion can be unified with the definition of model subspace from Vaiter et al. (2015, Sec. 3.1):

Definition 3.1 (Model subspace, Vaiter et al. 2015)**.** We denote the model subspace at x:

$$T_x = \{u \in \mathbb{R}^p : \forall j \in \mathcal{S}_x^c, u_j = 0\} \ , \tag{3.2}$$

where $\mathcal{S}_x$ is the generalized support of x (see Definition 2.17).

Examples in machine learning.
The ℓ_1 norm. The function $g(x) = \sum_{i=1}^p |x_i|$ is certainly the most popular non-smooth convex regularizer promoting sparsity. Indeed, the ℓ_1 norm generates structured solution with model subspace (Vaiter et al., 2018). We have that $\mathcal{S}_x = \{j \in [p] : x_j \neq 0\}$ since $|\cdot|$ is differentiable everywhere but not at 0, and the model subspace reads:

$$T_x = \{u \in \mathbb{R}^p : \text{supp}(u) \subseteq \text{supp}(x)\} \ . \tag{3.3}$$

The box constraints indicator function $\iota_{[0,C]}$. This indicator function appears for instance in box constrained optimization problems such as the dual problem of the SVM. Let $\mathcal{I}_x^0 = \{j \in [p] : x_j = 0\}$ and $\mathcal{I}_x^C = \{j \in [p] : x_j = C\}$, then

$$T_x = \{u \in \mathbb{R}^p : \mathcal{I}_x^0 \subseteq \mathcal{I}_u^0 \text{ and } \mathcal{I}_x^C \subseteq \mathcal{I}_u^0\}.$$

For the SVM, model identification boils down to finding the active set of the box con-

strained quadratic optimization problem after a finite number of iterations.

To prove model identification for the CD algorithm, we need to rely on the assumption that the non-smooth function g is regular enough, or more precisely *partly smooth*. Loosely speaking, a partly smooth function behaves smoothly as it lies on the related model and sharply if we move normal to that model. Formally, we recall the definition of partly smooth functions restricted to the case of proper, lower semicontinuous and convex functions.

Definition 3.2 (Partial smoothness)**.** Let $g : \mathbb{R}^p \to \mathbb{R}$ be a proper closed convex function. g is said to be partly smooth at x relative to a set $\mathcal{M} \subseteq \mathbb{R}^n$ if there exists a neighborhood $\mathcal{U}$ of x such that

- **(Smoothness)** $\mathcal{M} \cap \mathcal{U}$ is a $\mathcal{C}^2$-manifold and g restricted to $\mathcal{M} \cap \mathcal{U}$ is $\mathcal{C}^2$,
- **(Sharpness)** The tangent space of $\mathcal{M}$ at x is the model tangent space T_x where $T_x = \text{Lin}(\partial g(x))^\perp$,
- **(Continuity)** The set valued mapping ∂g is continuous at x relative to $\mathcal{M}$.

The class of partly smooth functions was first defined in Lewis (2002). It encompasses a large number of known non-smooth machine learning optimization penalties, such as the ℓ_1-norm or box constraints to only name a few, see Vaiter et al. (2018, Section 2.1) for details. Interestingly, this framework enables powerful theoretical tools on model identification such as Hare and Lewis (2004, Thm. 5.3). For separable functions, the next lemma gives an explicit link between the generalized support (Definition 2.17) (Sun et al., 2019) and the framework of partly smooth functions (Hare and Lewis, 2004).

Lemma 3.1. *Let $x^\star \in \text{dom}\,(g)$. If for every $j \in \mathcal{S}_{x^\star}$, g_j is locally C^2 around $x_j^\star$ if, and only if, g is partly smooth at $x^\star$ relative to $x^\star + T_{x^\star}$.*

Proof. We need to prove the three properties of the partial smoothness (Definition 3.2).

Smoothness. Let us write $\mathcal{M}_{x^\star} = x^\star + T_{x^\star}$ the affine space directed by the model subspace and pointed by $x^\star$. In particular, it is a C^2-manifold.

For every $j \in \mathcal{S}_{x^\star}$, g_j is locally C^2 around $x_j^\star$, hence there exists a neighborhood U_j of $x_j^\star$ such that the restriction of g_j to U is twice continuously differentiable. For $j \in \mathcal{S}_{x^\star}^c$, let's write $U_j = \mathbb{R}$. Take $U = \bigotimes_{j\in[p]} U_j$. This a neighborhood of $x^\star$ (it is open, and contains $x^\star$). Consider the restriction $g_{|\mathcal{M}_{x^\star}}$ of g to $\mathcal{M}_{x^\star}$. It is C^2 at each point of U since each coordinates (for $j \in \mathcal{S}_{x^\star}$) are C^2 around U_j.

Sharpness. Since g is completely separable, we have that $\partial g(x^\star) = \partial g_1(x_1^\star) \times \ldots \times \partial g_p(x_p^\star)$. Note that $\partial g_j(x_j^\star)$ is a set valued mapping which is equal to the singleton $\{\nabla_j g(x_j^\star)\}$ if g_j is differentiable at $x_j^\star$ or it is equal to an interval. The model tangent space $T_{x^\star}$ of g at $x^\star$ is given by

$$T_{x^\star} = \text{span}(\partial g(x^\star))^\perp \quad \text{where} \quad \text{span}(\partial g(x^\star)) = \text{aff}(\partial g(x^\star)) - e_{x^\star} \ , \tag{3.4}$$

with

$$e_{x^\star} = \underset{e \in \text{aff}(\partial g(x^\star))}{\arg\min} \ ||e|| \ , \tag{3.5}$$

called the model vector.

In the particular case of separable functions, we have that

$$\text{aff}\left(\partial g(x^\star)\right) = \text{aff}\left(\partial g_1(x_1^\star) \times \ldots \times \partial g_p(x_p^\star)\right) = \text{aff}\left(\partial g_1(x_1^\star)\right) \times \ldots \times \text{aff}\left(\partial g_p(x_p^\star)\right) \ .$$

In this case,

$$\text{aff}\left(\partial g_j(x_j^\star)\right) = \begin{cases} \{\nabla_j g(x_j^\star)\} & \text{if } j \in \mathcal{S}_{x^\star} \\ \mathbb{R} & \text{otherwise} \end{cases} \quad \text{and} \quad e_{x_j^\star} = \begin{cases} \nabla_j g(x_j^\star) & \text{if } j \in \mathcal{S}_{x^\star} \\ 0 & \text{otherwise} \ . \end{cases} \tag{3.6}$$

Thus we have that

$$\text{span}\left(\partial g(x^\star)\right) = \text{aff}\left(\partial g(x^\star)\right) - e_{x^\star} = \{x \in \mathbb{R}^p : \forall j' \in \mathcal{S}_{x^\star}, x_{j'} = 0\} \ .$$

Then

$$T_{x^\star} = \text{span}\left(\partial g(x^\star)\right)^\perp = \{x \in \mathbb{R}^p : \forall j' \in \mathcal{S}_{x^\star}^c, x_{j'} = 0\} \ . \tag{3.7}$$

Continuity. We are going to prove that ∂g is lower semicontinuous at $x^\star$ relative to $\mathcal{M}_{x^\star}$, *i.e.,* that for any sequence $(x^{(k)})$ of elements of $\mathcal{M}_{x^\star}$ converging to $x^\star$ and any $\bar{\eta} \in \partial g(x^\star)$, there exists a sequence of subgradients $\eta^{(k)} \in \partial g(x^{(k)})$ converging to $\bar{\eta}$.

Let $x^{(k)}$ be a sequence of elements of $\mathcal{M}_{x^\star}$ converging to $x^\star$, or equivalently, let $t^{(k)}$ be a sequence of elements of $T_{x^\star}$ converging to 0, and let $\bar{\eta} \in \partial g(x^\star)$.

For $j \in \mathcal{S}_{x^\star}$, we choose $\eta_j^{(k)} \triangleq g_j'(x_j^\star + t_j^{(k)})$, using the smoothness property we have $\eta_j^{(k)} \triangleq \bar{\eta}_j$. For all $j \in \mathcal{S}_{x^\star}^c$ $x_j^{(k)} = x_j^\star$ we choose $\eta_j^{(k)} \triangleq \bar{\eta}_j$, since $x^{(k)} \in \mathcal{M}_{x^\star}$, we have $\eta_j^{(k)} \in \partial g(x^{(k)})$.

We have that $\eta^{(k)} \in \partial g(x^k)$ and $\eta^{(k)}$ converges towards $\bar{\eta}$ since g_j' is C^1 around $x_j^\star$ for $j \in \mathcal{S}_{x^\star}$, hence, $g_j'(x_j^\star + t_j^{(k)})$ converges to $g_j'(x_j^\star) = \bar{\eta}_j$. Thus, it proves that g is partly smooth at $x^\star$ relative to $x^\star + T_{x^\star}$.

Let us now suppose that g is partly smooth at $x^\star$ relative to $\mathcal{M}_{x^\star}$. By Definition 3.2, there exists $U \subset \mathbb{R}^p$ a neighboorhood of $x^\star$ such that $g_{|\mathcal{M}_{x^\star} \cap U}$ is $\mathcal{C}^2$. Since g is separable, $\mathcal{M}_{x^\star} = x^\star + T_{x^\star}$ with $T_{x^\star} = \{x \in \mathbb{R}^p : \forall j' \in \mathcal{S}_{x^\star}^c, x_j' = 0\}$. Let $y \in \mathcal{M}_{x^\star} \cap U$, we have that $\forall j' \in \mathcal{S}_{x^\star}^c, y_{j'} = x_{j'}^\star$. Let us write $U = \bigotimes_{j \in [p]} U_j$, then for all $j \in \mathcal{S}_{x^\star}$, $y \in \mathcal{M}_{x^\star} \cap U$ implies that $y_j \in U_j$. Thus, for all $j \in \mathcal{S}_{x^\star}$, we have that g_j is $\mathcal{C}^2$ on U_j that contains $x_j^\star$. □

3.3 Model identification for CD

We now turn to our identification result. To ensure model identification, we need the following (mild) assumption:

Assumption 3.1 (Locally $\mathcal{C}^2$)**.** Let $x^\star$ be a solution of Equation (3.1). For all $j \in \mathcal{S}_{x^\star}$, g_j is locally $\mathcal{C}^2$ around $x_j^\star$, and f is locally $\mathcal{C}^2$ around $x^\star$.

It is satisfied for the Lasso and the dual SVM problem mentioned above, but also for sparse logistic regression and elastic net. The following theorem shows that the CD (Algorithm 3) has the model identification property with local constant step size $0 < \gamma_j \leq 1/L_j$:

Theorem 3.1 (Model identification of CD)**.** *Assume that Equation (3.1) is regular (Definition 2.16). We denote $x^\star \in \arg\min_{x \in \mathbb{R}^p} \Phi(x)$ and $\mathcal{S} = \mathcal{S}_{x^\star}$. Suppose*

1. *Assumption 3.1 hold,*
2. *$x^\star$ is non-degenerated (Assumption 2.4),*
3. *The sequence $(x^{(k)})_{k \geq 0}$ generated by Algorithm 3 converges to $x^\star$.*

Then, Algorithm 3 identifies the model after a finite number of iterations, which means that there exists $K > 0$ such that for all $k \geq K$, $x_{\mathcal{S}^c}^{(k)} = x_{\mathcal{S}^c}^\star$.

This result implies that for k large enough, $x^{(k)}$ shares the support of $x^\star$ (potentially smaller).

Proof. Using Lemma 3.1, we have that the regularity of Equation (3.1) (Definition 2.16) and Assumptions 2.4 and 3.1 imply that g is *partly smooth* (Lewis, 2002) at $x^\star$ relative to the affine space $x^\star + T_{x^\star}$.

We now prove that for the CD Algorithm 3: $\operatorname{dist}\left(\partial\Phi(x^{(k)}), 0\right) \to 0$, when $k \to \infty$.

As written in Algorithm 3, one update of coordinate descent reads:

$$\frac{1}{\gamma_j} x_j^{(j-1,k)} - \nabla_j f\left(x^{(j-1,k)}\right) - \frac{1}{\gamma_j} x_j^{(j,k)} \in \partial g_j\left(x_j^{(j,k)}\right)$$
$$\frac{1}{\gamma_j} x_j^{(k)} - \nabla_j f\left(x^{(j-1,k)}\right) - \frac{1}{\gamma_j} x_j^{(k+1)} \in \partial g_j\left(x_j^{(k+1)}\right) \quad .$$

Since g is separable with non-empty subdifferential, the coordinatewise subdifferential of g is equal to the subdifferential of g, we then have

$$\frac{1}{\gamma} \odot x^{(k)} - \left(\nabla_j f\left(x^{(j-1,k)}\right)\right)_{j\in[p]} - \frac{1}{\gamma} \odot x^{(k+1)} \in \partial g(x^{(k+1)}) \quad , \tag{3.8}$$

which leads to

$$\frac{1}{\gamma} \odot x^{(k)} - \left(\nabla_j f\left(x^{(j-1,k)}\right)\right)_{j\in[p]} - \frac{1}{\gamma} \odot x^{(k+1)} + \nabla f(x^{(k+1)}) \in \partial \Phi(x^{(k+1)}) \quad . \tag{3.9}$$

To prove support identification using Hare and Lewis (2004, Thm. 5.3), we need to bound the distance between $\partial\Phi(x^{(k+1)})$ and 0, using Equation (3.9):

$$\begin{aligned}
\operatorname{dist}\left(\partial\Phi(x^{(k+1)}), 0\right)^2 &\leq \sum_{j=1}^{p} \left| \frac{x_j^{(k)}}{\gamma_j} - \nabla_j f(x^{(j-1,k)}) - \frac{x_j^{(k+1)}}{\gamma_j} + \nabla_j f(x^{(k+1)}) \right|^2 \\
&\leq ||x^{(k)} - x^{(k+1)}||_{\gamma^{-1}}^2 + \sum_{j=1}^{p} \left| \nabla_j f\left(x^{(j-1,k)}\right) - \nabla_j f\left(x^{(k+1)}\right) \right|^2 \\
&\leq ||x^{(k)} - x^{(k+1)}||_{\gamma^{-1}}^2 + L^2 \sum_{j=1}^{p} \|x^{(j-1,k)} - x^{(k+1)}\|^2 \\
&\leq \underbrace{||x^{(k)} - x^{(k+1)}||_{\gamma^{-1}}^2 + L^2 \sum_{j=1}^{p} \sum_{j'\geq j}^{p} \left| x_{j'}^{(k)} - x_{j'}^{(k+1)} \right|^2}_{\to 0 \text{ when } k\to\infty} \quad .
\end{aligned}$$

We thus have:

- $\operatorname{dist}\left(\partial\Phi(x^{(k+1)}), 0\right) \to 0$
- $\Phi(x^{(k)}) \to \Phi(x^\star)$ because Φ is prox-regular (since it is convex, see Poliquin and Rockafellar 1996b) and subdifferentially continuous.

Then the conditions to apply Hare and Lewis (2004, Th. 5.3) are met and hence we have model identification after a finite number of iterations. □

Comments on Theorem 3.1. It unifies several results found in the literature: Massias et al. (2019) showed model identification for the Lasso, solved with coordinate descent, but requiring uniqueness assumption. As mentioned in the introduction, this theorem was already stated in Nutini et al. (2017, Lemma 3) but the proof technique is different. The proof of Nutini et al. (2017, Lemma 3) is based on an explicit manipulation of the expression of the minimum distance to the boundary of the subdifferential, whereas our proof is based on Lemma 3.1, a geometrical statement showing that any separable function regular enough enjoys a natural control on this distance, thanks to partial smoothness theory.

3.4 Local convergence rates

In this section, we prove the local linear convergence of the CD Algorithm 3. After model identification, there exists a regime where the convergence towards a solution of Equation (3.1) is linear. Local linear convergence was already proved in various settings such as for ISTA and FISTA algorithms (*i.e.*, with an ℓ_1 penalty, Tao et al. 2016) and then for the general Forward-Backward algorithm (Liang et al., 2014).

Local linear convergence requires an additional assumption: *restricted injectivity*. It is classical for this type of analysis as it can be found in Liang et al. (2017) and Poon and Liang (2019).

Assumption 3.2. (Restricted injectivity) For a solution $x^\star \in \arg\min_{x \in \mathbb{R}^p} \Phi(x)$, the restricted Hessian to its generalized support $\mathcal{S} = \mathcal{S}_{x^\star}$ is definite positive, *i.e.*,

$$\nabla^2_{\mathcal{S},\mathcal{S}} f(x^\star) \succ 0 \ . \tag{3.10}$$

For the Lasso, Assumption 3.2 is a classical necessary condition to ensure uniqueness of the minimizer (Fuchs, 2004).

In order to study local linear convergence, we consider the fixed point iteration of a complete epoch (an epoch is a complete pass over all the coordinates). A full epoch of CD can be written:

$$x^{(k+1)} = \psi(x^{(k)}) \triangleq \mathcal{P}_p \circ \ldots \circ \mathcal{P}_1(x^{(k)}) \ , \tag{3.11}$$

where $\mathcal{P}_j$ are coordinatewise sequential applications of the proximity operator $\mathcal{P}_j : \mathbb{R}^p \to$

$\mathbb{R}^p$:

$$x \mapsto \begin{pmatrix} x_1 \\ \vdots \\ x_{j-1} \\ \operatorname{prox}_{\gamma_j g_j}\left(x_j - \gamma_j \nabla_j f(x)\right) \\ x_{j+1} \\ \vdots \\ x_p \end{pmatrix} \quad .$$

Thanks to model identification (Theorem 3.1), we are able to prove that once the generalized support is correctly identified, there exists a regime where CD algorithm converges linearly towards the solution:

Theorem 3.2 (Local linear convergence)**.** *Assume Equation (3.1) is regular (Definition 2.16). We denote $x^\star \in \arg\min_{x\in\mathbb{R}^p} \Phi(x)$ and $\mathcal{S} = \mathcal{S}_{x^\star}$. Suppose*

1. *$x^\star$ is non-degenerated (Assumption 2.4)*
2. *Assumptions 3.1 and 3.2 hold.*
3. *The sequence $(x^{(k)})_{k\geq 0}$ generated by Algorithm 3 converges to $x^\star$.*
4. *The model has been identified* i.e.*, there exists $K \geq 0$ such as for all $k \geq K$*

$$x^{(k)}_{\mathcal{S}^c} = x^\star_{\mathcal{S}^c} \quad .$$

Then $(x^{(k)})_{k\geq K}$ converges linearly towards $x^\star$. More precisely, for any $\nu \in [\rho(\mathcal{J}\psi_{\mathcal{S},\mathcal{S}}(x^\star)), 1[$, there exists $K > 0$ and a constant C such that for all $k \geq K$,

$$\|x^{(k)}_{\mathcal{S}} - x^\star_{\mathcal{S}}\| \leq C\nu^{(k-K)}\|x^{(K)}_{\mathcal{S}} - x^\star_{\mathcal{S}}\| \quad .$$

Proof. To simplify the notations, $\mathcal{S} \triangleq \mathcal{S}_{x^\star}$. Let us also write the element of $\mathcal{S}$ as follows: $\mathcal{S} = \{j_1, \ldots, j_{|\mathcal{S}|}\}$.

We also define $\pi^{x^\star_{\mathcal{S}^c}} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^p$ for all $x_{\mathcal{S}} \in \mathbb{R}^{|\mathcal{S}|}$ and all $j \in \mathcal{S}$ by

$$\left(\pi^{x^\star_{\mathcal{S}^c}}(x_{\mathcal{S}})\right)_j = \begin{cases} x_j & \text{if } j \in \mathcal{S} \\ x^\star_j & \text{if } j \in \mathcal{S}^c \end{cases} , \tag{3.12}$$

and for all $j_s \in \mathcal{S}$, $\tilde{\mathcal{P}}_{j_s}^{x^\star_{\mathcal{S}^c}} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ is the function defined for all $x_{\mathcal{S}} \in \mathbb{R}^{|\mathcal{S}|}$ and all $j \in \mathcal{S}$ by

$$\left(\tilde{\mathcal{P}}_{j_s}^{x^\star_{\mathcal{S}^c}}(x_{\mathcal{S}})\right)_j = \begin{cases} x_j & \text{if } j \neq j_s \\ \operatorname{prox}_{\gamma_j g_j}\left(x_{j_s} - \gamma_{j_s} \nabla_{j_s} f(\pi^{x^\star_{\mathcal{S}^c}}(x_{\mathcal{S}}))\right) & \text{if } j = j_s \ . \end{cases} \tag{3.13}$$

Once the model is identified (Theorem 3.1), we have that there exists $K \geq 0$ such that for all $k \geq K$, we have that

$$x_{\mathcal{S}^c}^{(k)} = x^\star_{\mathcal{S}^c} \quad \text{and} \quad x_{\mathcal{S}}^{(k+1)} = \tilde{\psi}(x_{\mathcal{S}}^{(k)}) \triangleq \mathcal{P}_{j_{|\mathcal{S}|}}^{x^\star_{\mathcal{S}^c}} \circ \ldots \circ \mathcal{P}_{j_1}^{x^\star_{\mathcal{S}^c}}(x_{\mathcal{S}}^{(k)}) \ . \tag{3.14}$$

When no confusion is possible, we denote by $\tilde{\mathcal{P}}_j$ the function $\tilde{\mathcal{P}}_{j_s}^{x^\star_{\mathcal{S}^c}}$, hence still dependant on $x^\star_{\mathcal{S}^c}$. The following lemma shows that $\tilde{\mathcal{P}}_j$ is differentiable at the optimum.

Sketch of proof:

- The first part of the proof, *i.e.*, Lemmas 3.2 and 3.3, aims at proving that $x \mapsto \operatorname{prox}_{\gamma_j g_j}(x_j - \gamma_j \nabla_j)$ is differentiable for all $j \in [p]$ and that its derivative is 0 for all $j \notin \mathcal{S}$. These results allow us to compute the Jacobian of $\tilde{\psi}$ at $x^\star_{\mathcal{S}}$.
- Then Lemmas 3.4 to 3.8 prove that $\rho(\mathcal{J}\tilde{\psi}(x^\star_{\mathcal{S}})) < 1$.
- Finally, all the conditions are met to apply Polyak (1987)[Theorem 1, Section 2.1.2]. The latter reference provides sufficient conditions for local linear convergence of sequences based on fixed point iterations.

Lemma 3.2. *For all $j \in \mathcal{S}$, $\tilde{\mathcal{P}}_j$ is differentiable at $x^\star_{\mathcal{S}}$.*

Proof. From Assumption 3.1, we know there exists a neighboorhood of $x_j^\star$ denoted $\mathcal{U}$ such that, for $j \in \mathcal{S}$, the restriction of g_j to $\mathcal{U}$ is $\mathcal{C}^2$ on $\mathcal{U}$. In particular, it means that $x_j^\star$ is a differentiable point of g_j and given a pair $(u, v) \in \mathcal{U} \times \mathbb{R}^p$ such that

$$u = \operatorname{prox}_{\gamma_j g_j}(v) \in \mathcal{U} \ , \tag{3.15}$$

we have $\frac{1}{\gamma_j}(v - u) \in \partial g_j(u)$ becomes

$$\frac{1}{\gamma_j}(v - u) = g_j'(u) \Leftrightarrow v = u + \gamma_j g_j'(u) \Leftrightarrow v = (\mathrm{Id} + g_j')(u) \ . \tag{3.16}$$

Let $H(u) = (\mathrm{Id} + g_j')(u)$, since g_j is twice differentiable at u, we have that

$$H'(u) = 1 + \gamma_j g_j''(u) \ . \tag{3.17}$$

Thus, $H' : \mathcal{U} \mapsto \mathbb{R}$ is continuous and then $H : \mathcal{U} \mapsto \mathbb{R}$ is continuously differentiable. Hence $F(v, u) \triangleq v - H(u)$ is $\mathcal{C}^1$ and $F(v, u) = 0$. By convexity of g, we have $g_j''(u) \geq 0$ and

$$\frac{\partial F}{\partial u}(v, u) = -H'(u) = -1 - \gamma_j g''(u) \neq 0 \ . \tag{3.18}$$

Using the implicit functions theorem, we have that there exists an open interval $\mathcal{V} \subseteq \mathbb{R}$ with $v \in \mathcal{V}$ and a function $h : \mathcal{V} \mapsto \mathbb{R}$ which is $\mathcal{C}^1$ such as $u = h(v)$.

Using (3.15) we thus have with the choice $u = x_j^\star$, $v = x_j^\star - \gamma_j \nabla_j f(x^\star)$ that the map h coincides with $\mathrm{prox}_{\gamma_j g_j}$ on $\mathcal{V}$ and is differentiable at $v = x_j^\star - \gamma_j \nabla_j f(x^\star) \in \mathcal{V}$. It follows that $\tilde{\mathcal{P}}_j$ is differentiable at $x_{\mathcal{S}}^\star$. □

For the sake of completness, we show that in fact $\mathrm{prox}_{\gamma_j g_j}$ is also differentiable on the complement of the generalized support at $x_j^\star - \nabla_j f(x^\star)$.

Lemma 3.3. *For all $j \in \mathcal{S}^c$, $\mathrm{prox}_{\gamma_j g_j}$ is constant around $x_j^\star - \gamma_j \nabla_j f(x^\star)$. Moreover, the map $x \mapsto \mathrm{prox}_{\gamma_j g_j}(x_j - \gamma_j \nabla_j f(x))$ is differentiable at $x^\star$ with gradient 0.*

Proof. Let $\partial g_j(x_j^\star) = [a; b]$ and let $z_j^\star = x_j^\star - \nabla_j f(x^\star)$, then combining the fixed point equation and Assumption 2.4 leads to:

$$\frac{1}{\gamma_j}(z_j^\star - x_j^\star) \in \mathrm{ri}\left(\partial g_j(x_j^\star)\right) =]a; b[\ . \tag{3.19}$$

Thus,

$$z_j^\star \in]\gamma_j a + x_j^\star; \gamma_j b + x_j^\star[\ . \tag{3.20}$$

For all $v \in]\gamma_j a + x_j^\star; \gamma_j b + x_j^\star[$, we have $\frac{1}{\gamma_j}(v - x_j^\star) \in]a; b[= \mathrm{ri}\left(\partial g_j(x_j^\star)\right)$, *i.e.*, $\mathrm{prox}_{\gamma_j g_j}(v) = x_j^\star$. As f is $\mathcal{C}^2$ in $x^\star$, we have that $x \mapsto \mathrm{prox}_{\gamma_j g_j}(x_j - \nabla_j f(x))$ is differentiable at $x^\star$ with gradient being 0. □

From Lemma 3.2, we have that $\tilde{\mathcal{P}}_j$ is differentiable at $x_{\mathcal{S}}^\star$ for all $j \in \mathcal{S}$. Since $x^\star$ is an optimal

point, the following fixed points equation holds:

$$x_j^\star = \operatorname{prox}_{\gamma_j g_j}\left(x_j^\star - \gamma_j \nabla_j f(x^\star)\right) \quad . \tag{3.21}$$

The map $\tilde{\psi}$ is then differentiable at $x_{\mathcal{S}}^\star$ since it is obtained as the composition of differentiable functions and that each function $\tilde{\mathcal{P}}_j$ is evaluated at a differentiable point (only one coordinate change at each step).

To compute, the Jacobian of $\tilde{\mathcal{P}}_j$ at $x_{\mathcal{S}}^\star$, let us first notice that

$$\mathcal{J}\mathcal{P}_j(x_{\mathcal{S}}^\star)^\top = \left(\; e_1 \;\middle|\; \ldots \;\middle|\; e_{j-1} \;\middle|\; v_j \;\middle|\; e_{j+1} \;\middle|\; \ldots \;\middle|\; e_s \; \right) \quad , \tag{3.22}$$

where $v_j = \partial_x \operatorname{prox}_{\gamma_j g_j}\left(z_j^\star\right)\left(e_j - \gamma_j \nabla^2_{j,:} f(x^\star)\right)$ and $z_j^\star = x_j^\star - \gamma_j \nabla_j f(x^\star)$. This matrix can be rewritten as

$$\begin{aligned}
\mathcal{J}\tilde{\mathcal{P}}_j(x_{\mathcal{S}}^\star) &= \mathrm{Id}_{|S|} - e_j e_j^\top + \partial_x \operatorname{prox}_{\gamma_j g_j}\left(z_j^\star\right)\left(e_j e_j^\top - \gamma_j e_j e_j^\top \nabla^2 f(x^\star)\right) \\
&= \mathrm{Id}_{|S|} - e_j e_j^\top \gamma_j \partial_x \operatorname{prox}_{\gamma_j g_j}\left(z_j^\star\right)\left(\operatorname{diag}(u) + \nabla^2 f(x^\star)\right) \\
&= \mathrm{Id}_{|S|} - e_j e_j^\top \gamma_j \partial_x \operatorname{prox}_{\gamma_j g_j}\left(z_j^\star\right) M \\
&= M^{-1/2}\left(\mathrm{Id}_{|S|} - M^{1/2} e_j e_j^\top \gamma_j \partial_x \operatorname{prox}_{\gamma_j g_j}\left(z_j^\star\right) M^{1/2}\right) M^{1/2} \\
&= M^{-1/2}\left(\mathrm{Id}_{|S|} - B_j\right) M^{1/2} \quad ,
\end{aligned} \tag{3.23}$$

where

$$M \triangleq \nabla^2_{\mathcal{S},\mathcal{S}} f(x^\star) + \operatorname{diag}(u) \quad , \tag{3.24}$$

and $u \in \mathbb{R}^{|\mathcal{S}|}$ is defined for all $j \in \mathcal{S}$ by

$$u_j = \begin{cases} \frac{1}{\gamma_j \partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star)} - \frac{1}{\gamma_j} & \text{if } \operatorname{prox}_{\gamma_j g_j}(z_j^\star) \neq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{3.25}$$

and

$$B_j = M^{1/2}_{:,j} \gamma_j \partial_x \operatorname{prox}_{\gamma_j g_j}\left(z_j^\star\right) M^{1/2\top}_{:,j}. \tag{3.26}$$

Since only one coordinate change at each step, the chain rule leads to

$$\begin{aligned}\mathcal{J}\tilde{\psi}(x^\star_{\mathcal{S}}) &= \mathcal{J}\mathcal{P}_{j_s}(x^\star_{\mathcal{S}})\mathcal{J}\mathcal{P}_{j_{s-1}}(x^\star_{\mathcal{S}})\dots\mathcal{J}\mathcal{P}_{j_1}(x^\star_{\mathcal{S}}) \\ &= M^{-1/2}\underbrace{(\mathrm{Id}-B_{j_s})\dots(\mathrm{Id}-B_{j_1})}_{A}M^{1/2}\end{aligned}$$

The next series of lemma will be useful to prove that the spectral radius $\rho\left(\mathcal{J}\tilde{\psi}(x^\star_{\mathcal{S}})\right) < 1$.

Lemma 3.4. *The matrix M defined in (3.24) is symmetric definite positive.*

Proof. Using the non-expansivity of the proximal operator, and the property $\partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star) > 0$ for $j \in \mathcal{S}$, $\operatorname{diag}(u)$ is a symmetric semidefinite matrix, so M is a sum of a symmetric definite positive matrix and a symmetric semidefinite matrix, hence M is symmetric definite positive. □

Lemma 3.5. *For all $j \in \mathcal{S}$, the matrix B_j defined in (3.26) has spectral norm bounded by 1,* i.e., $\|B_j\|_2 \leq 1$.

Proof. B_j is a rank one matrix which is the product of $\gamma_j\partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star)M^{1/2}_{:,j}$ and $M^{1/2\top}_{:,j}$, its non-zeros eigenvalue is thus given by

$$\begin{aligned}\|B_j\|_2 &= \left|M^{1/2\top}_{:,j}\gamma_j\partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star)M^{1/2}_{:,j}\right| \\ &= \left|\gamma_j\partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star)M_{j,j}\right| \\ &= \left|\gamma_j\partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star)\left(\underbrace{\nabla^2_{j,j}f(x^\star)}_{0\leq} + \underbrace{\left(\frac{1}{\gamma_j\partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star)} - \frac{1}{\gamma_j}\right)}_{0\leq}\right)\right| . \end{aligned} \tag{3.27}$$

By positivity of the two terms,

$$\begin{aligned}\|B_j\|_2 &= \gamma_j\partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star)\underbrace{\nabla^2_{j,j}f(x^\star)}_{\leq L_j \leq \frac{1}{\gamma_j}} + \left(1 - \partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star)\right) \\ &\leq \partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star) + \left(1 - \partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star)\right) \\ &\leq 1 \ . \end{aligned} \tag{3.28}$$

□

Lemma 3.6. *For all $j \in \mathcal{S}$, $B_j/\|B_j\|$ is an orthogonal projector onto* $\mathrm{Span}(M_{:,j}^{1/2})$.

Proof. It is clear that $B_j/||B_j||$ is symmetric.

We now prove that it is idempotent, *i.e.*, $(B_s/||B_s||)^2 = B_s/||B_s||$.

$$\begin{aligned} B_j^2/||B_j||^2 &= (\gamma_j \partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star))^2 M_{:,j}^{1/2} M_{:,j}^{1/2\top} M_{:,j}^{1/2} M_{:,j}^{1/2\top}/||B_j||^2 \\ &= (\gamma_j \partial_x \operatorname{prox}_{\gamma_j g_j}(z_j^\star))||B_j|| M_{:,j}^{1/2} M_{:,j}^{1/2\top}/||B_j||^2 \\ &= B_j/||B_j|| \quad . \end{aligned}$$

Hence, $B_j/||B_j||$ is an orthogonal projector. □

Lemma 3.7. *For all $j \in \mathcal{S}$ and for all $x \in \mathbb{R}^S$, if $\|(\mathrm{Id} - B_j)x\| = \|x\|$ then $x \in \mathrm{Span}(M_{:,j}^{1/2})^\perp$.*

Proof.

$$\begin{aligned} \mathrm{Id} - B_j &= \mathrm{Id} - \|B_j\| \frac{B_j}{\|B_j\|} \\ &= (1 - \|B_j\|)\,\mathrm{Id} + \|B_j\|_2\,\mathrm{Id} - \|B_j\|_2 \frac{B_j}{\|B_j\|_2} \\ &= (1 - \|B_j\|)\,\mathrm{Id} + \|B_j\| \underbrace{\left(\mathrm{Id} - \frac{B_j}{\|B_j\|_2}\right)}_{\text{projection onto } M_{:,j}^{1/2\perp}} \quad . \end{aligned} \tag{3.29}$$

Let $x \notin \mathrm{Span}(M_{:,j}^{1/2})^\perp$, then there exists $\kappa \neq 0$, $x_{M_{:,j}^{1/2\perp}} \in \mathrm{Span}(M_{:,j}^{1/2})^\perp$ such that

$$x = \kappa M_{:,j} + x_{M_{:,j}^{1/2\perp}} \quad . \tag{3.30}$$

Combining Equations (3.29) and (3.30) leads to:

$$\begin{aligned} (\mathrm{Id} - B_j)x &= (1 - \|B_j\|_2)x + \|B_j\|_2 x_{M_{:,j}^{1/2\perp}} \\ \|(\mathrm{Id} - B_j)x\| &\leq \underbrace{|1 - \|B_j\|_2|}_{=1-\|B_j\|_2} \|x\| + \|B_j\|_2 \underbrace{\|x_{M_{:,j}^{1/2\perp}}]\|}_{<\|x\|} \\ &< \|x\| \quad . \end{aligned}$$

□

Lemma 3.8. *The spectral norm of A is bounded by 1,* i.e.,

$$\|(\mathrm{Id} - B_{j_s}) \dots (\mathrm{Id} - B_{j_1})\|_2 = ||A||_2 < 1 \ .$$

Proof. Let $x \in \mathbb{R}^s$ such that $\|(\mathrm{Id} - B_{j_s}) \dots (\mathrm{Id} - B_{j_1})x\| = \|x\|$. Since

$$\|(\mathrm{Id} - B_{j_s} \dots (\mathrm{Id} - B_{j_1})\|_2 \leq \underbrace{\|(\mathrm{Id} - B_{j_s})\|_2}_{\leq 1} \times \cdots \times \underbrace{\|(\mathrm{Id} - B_{j_1})\|_2}_{\leq 1} \ ,$$

we thus have for all $j \in \mathcal{S}$, $\|(\mathrm{Id} - B_j)x\| = \|x\|$. One can thus successively apply Lemma 3.7 which leads to:

$$x \in \bigcap_{j \in \mathcal{S}} \mathrm{Span}\, M^{1/2^\perp}_{:,j} \Leftrightarrow x \in \mathrm{Span}\left(M^{1/2}_{:,j_1}, \dots, M^{1/2}_{:,j_s}\right)^\perp \ .$$

Moreover $M^{1/2}$ has full rank (see Lemma 3.4), thus $x = 0$ and

$$\|(\mathrm{Id} - B_{j_s}) \dots (\mathrm{Id} - B_{j_1})\|_2 < 1 \ .$$

□

From Lemma 3.8, $||A||_2 < 1$. Moreover A and $\mathcal{J}\tilde{\psi}(x^\star_{\mathcal{S}})$ are similar matrices, then $\rho(\mathcal{J}\tilde{\psi}(x^\star_{\mathcal{S}})) = \rho(A) \leq ||A||_2 < 1$.

To summarize, $x^\star_{\mathcal{S}}$ is the solution of a fixed point equation $\tilde{\psi}(x^\star_{\mathcal{S}}, x^\star_{\mathcal{S}^c}) = x^\star_{\mathcal{S}}$. From Lemma 3.2, $\tilde{\psi}(., x^\star_{\mathcal{S}^c})$ is differentiable at $x^\star_{\mathcal{S}}$ and the Jacobian at $x^\star_{\mathcal{S}}$ satisfies the condition $\rho(\mathcal{J}\tilde{\psi}(x^\star_{\mathcal{S}})) < 1$. Then all conditions are met to apply Polyak (1987)[Theorem 1, Section 2.1.2] which proves the local linear convergence.

□

3.5 Experiments

We now illustrate Theorems 3.1 and 3.2 on multiple datasets and estimators: the Lasso, the logistic regression and the SVM. In this section, we consider a design matrix $A \in \mathbb{R}^{n \times p}$ and a target $y \in \mathbb{R}^n$ for regression (Lasso) and $y \in \{-1, 1\}^n$ for classification (logistic regression and support-vector machine). We used classical datasets from `libsvm` (Chang and Lin, 2011) summarized in Table 3.1.

In Figures 3.1 to 3.3 the distance of the iterates to the optimum, $\|x^{(k)} - x^\star\|$ as a function

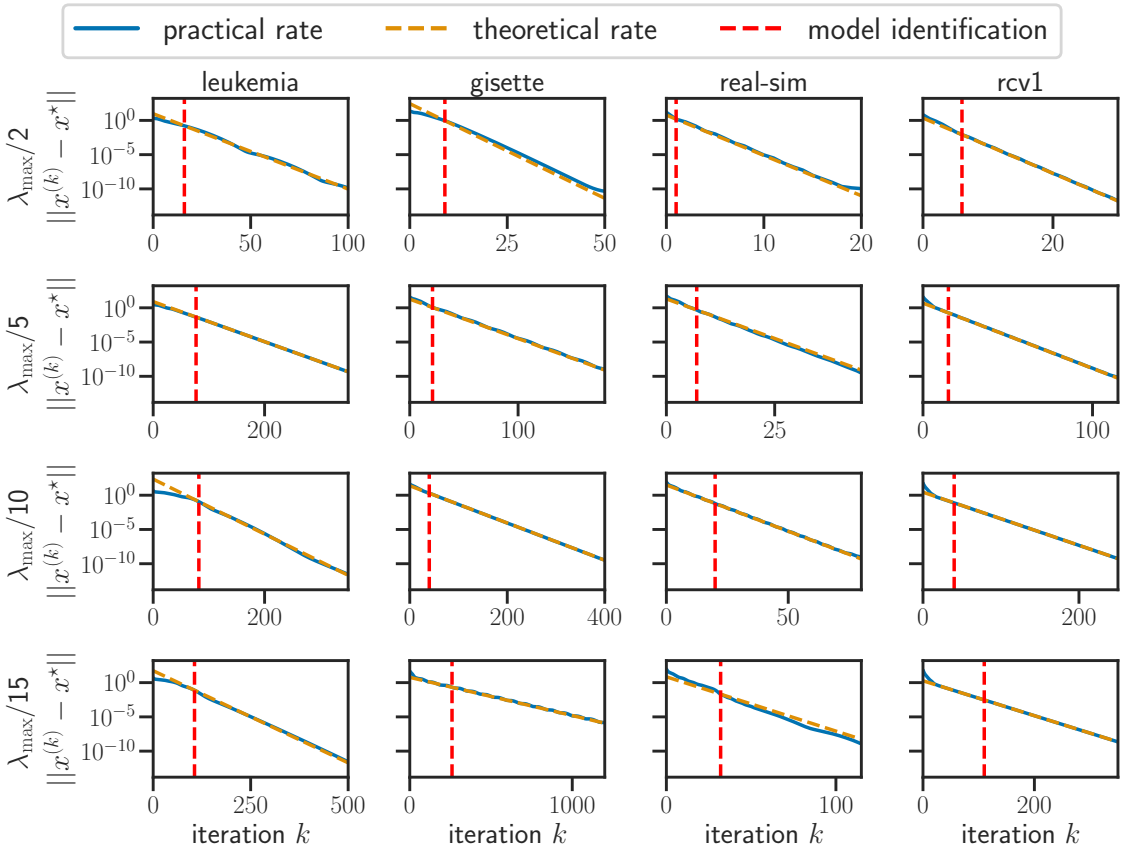

Figure 3.1 – **Lasso, linear convergence.** Distance to optimum, $\|x^{(k)} - x^\star\|$, as a function of the number of iterations k, on 4 different datasets: *leukemia*, *gisette*, *rcv1*, and *real-sim*.

of the number of iterations k is plotted as a solid blue line. The vertical red dashed line represents the iteration $k^\star$ where the model has been identified by CD (Algorithm 3) illustrating Theorem 3.1. The yellow dashed line represents the theoretical linear rate from Theorem 3.2. Theorem 3.2 gives the slope of the dashed yellow line, the (arbitrary) origin point of the theoretical rate line is chosen such that blue and yellow lines coincide at identification time, *i.e.*, all lines intersect at this point. More precisely, if $k^\star$ denotes the iteration where model identification happens, the equation of the dashed yellow line is:

$$h(k) = \|x^{(k^\star)} - x^\star\| \times \rho(\mathcal{J}\psi_{\mathcal{S},\mathcal{S}}(x^\star))^{(k-k^\star)} \quad . \tag{3.31}$$

Once a solution $x^\star$ has been computed, one can calculate $\mathcal{J}\psi_{\mathcal{S},\mathcal{S}}(x^\star)$ and its spectral radius for each estimator.

For the experiments we used three different estimators that we detail here.

Lasso. (Tibshirani, 1996) The most famous estimator based on a nonsmooth optimiza-

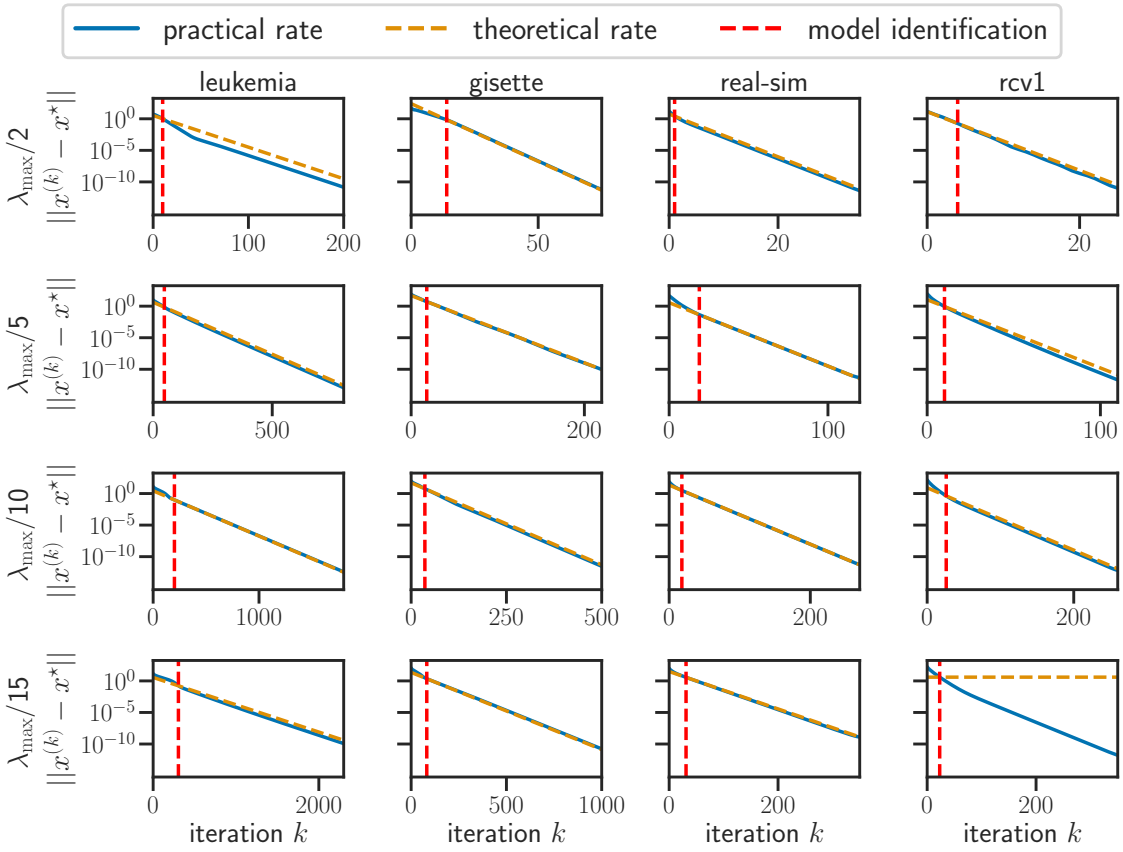

Figure 3.2 – **Sparse logistic regression, linear convergence.** Distance to optimum, $\|x^{(k)} - x^\star\|$, as a function of the number of iterations k, on 4 different datasets: *leukemia*, *gisette*, *rcv1*, and *real-sim*.

tion problem may be the Lasso. For a design matrix $A \in \mathbb{R}^{n \times p}$ and a target $y \in \mathbb{R}^n$ it writes:

$$\arg\min_{x \in \mathbb{R}^p} \frac{1}{2n} ||Ax - y||^2 + \lambda ||x||_1 \ . \tag{3.32}$$

Table 3.1 – Characteristics of the datasets.

Datasets	#samples n	#features p	density
leukemia	38	7129	1
gisette	6000	4955	1
rcv1	20,242	19,959	3.6×10^{-3}
real-sim	72,309	20,958	2.4×10^{-3}
20news	5184	155,148	1.9×10^{-3}

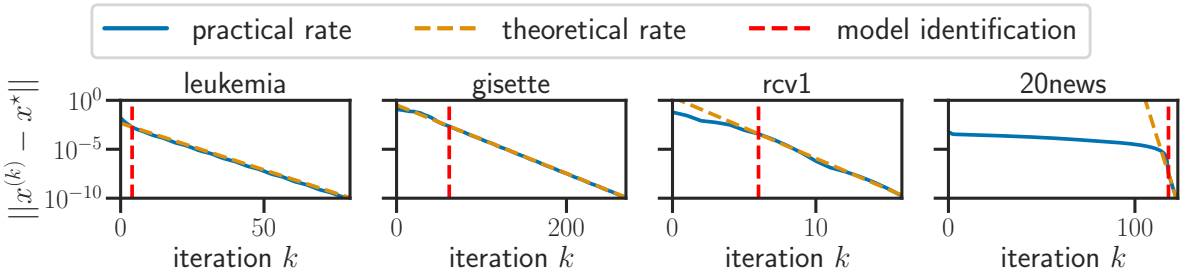

Figure 3.3 – **Support vector machine, linear convergence.** Distance to optimum, $\|x^{(k)} - x^\star\|$, as a function of the number of iterations k, on 4 different datasets: *leukemia*, *gisette*, *rcv1* and *20news*.

The CD update for the Lasso is given by

$$x_j \leftarrow \mathrm{ST}_{\gamma_j \lambda}\left(x_j - \gamma_j A_{:,j}^\top (y - Ax)\right) \quad , \tag{3.33}$$

where $\mathrm{ST}_\lambda(x) = \mathrm{sign}(x) \cdot \max(|x| - \lambda, 0)$. The solution of Equation (3.32) is obtained using Algorithm 3 with constant stepsizes $1/\gamma_j = \frac{\|A_{:,j}\|^2}{n}$.

Sparse logistic regression. The sparse logistic regression is an estimator for classification tasks. It is the solution of the following optimization problem, for a design matrix $A \in \mathbb{R}^{n \times p}$ and a target variable $y \in \{-1, 1\}^n$, with $\sigma(z) \triangleq \frac{1}{1+e^{-z}}$:

$$\arg\min_{x \in \mathbb{R}^p} -\frac{1}{n} \sum_{i=1}^n \log \sigma(y_i x^\top A_{i,:}) + \lambda \|x\|_1 \quad . \tag{3.34}$$

The CD update for the sparse logistic regression is

$$x_j \leftarrow \mathrm{ST}_{\gamma_j \lambda}\left(x_j - \gamma_j A_{:,j}^\top (y \odot (\sigma(y \odot Ax) - 1))\right) \quad . \tag{3.35}$$

The constant stepsizes for the CD algorithm to solve Equation (3.34) are given by $1/\gamma_j = \frac{\|A_{:,j}\|^2}{4n}$.

Support-vector machine. (Boser et al., 1992) The support-vector machine (SVM) primal optimization problem is, for a design matrix $A \in \mathbb{R}^{n \times p}$ and a target variable $y \in \{-1, 1\}^n$:

$$\arg\min_{x \in \mathbb{R}^p} \frac{1}{2}\|x\|^2 + C \sum_{i=1}^n \max\left(1 - y_i x^\top A_{i,:}, 0\right) \quad . \tag{3.36}$$

Table 3.2 – C values for SVM.

dataset	leukemia	gisette	rcv1	20news
C value	10	$1.5\ 10^{-2}$	$1.5\ 10^{-2}$	$5\ 10^{-1}$

The SVM can be solved using the following dual optimization problem:

$$\begin{aligned} &\arg\min_{w \in \mathbb{R}^n} \frac{1}{2} w^\top (y \odot A)(y \odot A)^\top w - \sum_{i=1}^n w_i \\ &\text{subject to } 0 \leq w_i \leq C \ . \end{aligned} \tag{3.37}$$

The CD update for the SVM reads:

$$w_i \leftarrow \mathcal{P}_{[0,C]} \left(w_i - \gamma_i ((y \odot A)_{i,:}^\top (y \odot Aw) - 1) \right) \ , \tag{3.38}$$

where $\mathcal{P}_{[0,C]}(x) = \min(\max(0, x), C)$. The stepsizes of the CD algorithm to solve Equation (3.37) are given by $1/\gamma_i = ||(y \odot A)_{i,:}||^2$. The values of the regularization parameter C for each dataset from Figure 3.3 are given in Table 3.2.

Comments on Figures 3.1 to 3.3. Finite time model identification and local linear convergence are illustrated on the Lasso, the sparse logistic regression and the SVM in Figures 3.1 to 3.3. As predicted by Theorem 3.1, the relative model is identified after a finite number of iterations. For the Lasso (Figure 3.1) and the sparse logistic regression (Figure 3.2), we observe that as the regularization parameter gets smaller, the number of iterations needed by the CD algorithm to identify the model increases. To our knowledge, this is a classical empirical observation, that is not backed up by theoretical results. After identification, the convergence towards a solution is linear as predicted by Theorem 3.2. The theoretical local speed of convergence provided by Theorem 3.2 seems like a sharp estimation of the true speed of convergence as illustrated by the three figures.

Note that on Figures 3.1 to 3.3 high values of λ (or small values of C) were required for the restricted injectivity Assumption 3.2 to hold. Indeed, despite its lack of theoretical foundation, it is empirically observed that, in general, the larger the value of λ, the smaller the cardinal of the generalized support: $|\mathcal{S}|$. It makes the restricted injectivity Assumption 3.2: $\nabla^2_{\mathcal{S},\mathcal{S}} f(x^\star) \succ 0$ easier to be satisfied. For instance, for $\lambda = \lambda_{\max}/20$, the restricted injectivity Assumption 3.2 was not verified for a lot of datasets for the Lasso and the sparse logistic regression (Figures 3.1 and 3.2). In the same vein, values of C for the SVM had to be chosen small enough, in order to make $|\mathcal{S}|$ not too large (Figure 3.3).

Note that finite time model identification is crucial to ensure local linear convergence, see for instance *20news* dataset on Figure 3.3. However there exists very few quantitative theoretical results for the convergence speed of the model identification. Nutini et al. (2019); Sun et al. (2019) tried to obtain some rates on the identification, quantifying "how much the problem is qualified", *i.e.*, how much Assumption 2.4 is satisfied. But these theoretical results do not seem to explain fully the experimental results of the CD: in particular the identification speed of the model compared to other algorithms.

Limits. We would like to point out the limit of our analysis illustrated for the case of $\lambda = \lambda_{\max}/15$ for the sparse logistic regression and the *rcv1* dataset in Figure 3.2. In this case, the solution may no longer be unique. The support gets larger and Assumption 3.2 is no longer met. In this case, the largest eigenvalue of $\mathcal{J}\psi_{\mathcal{S},\mathcal{S}}(x^\star)$ is exactly one, which leads to the constant rate observed in Figure 3.2. Despite the largest eigenvalue being exactly 1, a regime of locally linear convergence toward a (potentially non unique) minimizer is still observed. Linear convergence of non-strongly convex functions starts to be more and more understood (Necoara et al., 2019). Figure 3.2 with $\lambda = \lambda_{\max}/15$ for *rcv1* suggests extensions of Necoara et al. (2019) could be possible in the nonsmooth case.

4 Support Vector Regression with Linear Constraints

Contents

In this chapter, we propose an algorithm to solve the SVR optimization problem with added linear constraints on the estimator. This algorithm generalizes the SMO algorithm

used to solve the classical SVR optimization problem. In Section 4.2, we study the underlying optimization problem and prove that strong duality holds therefore we also provide the dual problem of the constrained SVR. Then in Section 4.3, we propose the generalized SMO algorithm and prove its convergence towards a solution of the constrained SVR. This algorithm has the particularity to combine separable and non-separable variables. This is an example where the separability condition on the coordinate descent can be weakened and a coordinate descent based strategy can be used to solve the optimization problem. Finally in Section 4.4, we show practical use of our proposed estimator to perform non-negative regression, regression onto the simplex and isotonic regression. We show that the constrained SVR can be a good estimator in certain settings in comparison to the Least Squares one.

4.1 Introduction

Regression analysis seeks to find a relation between explanatory variables and an outcome variable. The most known linear regression estimator is the Ordinary Least Squares (OLS) which is the best linear unbiased estimator if the noise is *i.i.d*, uncorrelated and homoscedastic. Additional constraints on the OLS estimator can be added to ensure that it follows specific properties. For example, penalized regression such as the Ridge regression improves the efficiency of the estimation by introducing a bias and reducing the variance of the estimator in presence of collinearity (Hoerl and Kennard, 1970). Another family of constrained regression is defined by the addition of hard constraints on the estimator, such as positivity constraints (Lawson and Hanson, 1995) which can improve the estimation performance. Some applications in biology, where the goal is to estimate cell proportions inside a tumor or the estimation of temperature in weather forecast, justify the use of hard constraints on the estimator. The constrained OLS has been well studied, but the literature around the Support Vector Regression with additional constraints is scarcer.

The Support Vector Machine (SVM) (Boser et al., 1992) is a class of supervised learning algorithms that have been widely used in the past 20 years for classification tasks and regression. These algorithms rely on two main ideas: the first one is the maximum margin hyperplane which consists in finding the hyperplane that maximizes the distance between the vectors that are to be classified and the hyperplane. The second idea is the kernel method that allows the SVM to be used to solve non-linear problems. The technique is to map the vectors in a higher dimensional space which is done by using a positive definite kernel, then a maximal margin hyperplane is computed in this space which gives a linear

classifier in the high dimensional space. In general, it leads to a non-linear classifier in the original input space.

From SVM to Support Vector Regression. Different implementations of the algorithms haven been proposed such as C-SVM, ν-SVM (Schölkopf et al., 1999), Least-Squares SVM (Suykens and Vandewalle, 1999), Linear Programming SVM (Friel and Harrison, 1998) among others. Each of these versions have their strengths and weaknesses depending on which application they are used. They differ in terms of constraints considered for the hyperplane (C-SVM and Least-Squares SVM), in terms of norm considered on the parameters (C-SVM and Linear Programming SVM) and in terms of optimization problem formulation (C-SVM and ν-SVM). Overall, these algorithms are a great tool for classification tasks and they have been used in many different applications like facial recognition (Jia and Martinez, 2009), image classification (Chapelle et al., 1999), cancer type classification (Haussler et al., 2000), text categorization (Joachims, 1998) to only cite a few examples. Even though, SVM was first developed for classification, an adaptation for regression estimation was proposed in Drucker et al. (1997) under the name Support Vector Regression (SVR). In this case, the idea of maximum margin hyperplane is slightly changed into finding a tube around the regressors. The size of the tube is controlled by a hyperparameter chosen by the user: ε. This is equivalent to using an ε-insensitive loss function, $|y - f(x)|_\varepsilon = \max\{0, |y - f(x)| - \varepsilon\}$ which only penalizes the error above the chosen ε level. As for the classification version of the algorithm, a ν-SVR method exists. In this version, the hyperparameter ε is computed automatically but a new hyperparameter ν has to be chosen by the user which controls asymptotically the proportions of support vectors (Schölkopf et al., 1999). SVR has proven to be a great tool in the field of function estimation for many different applications: predicting times series in stock trades (Van Gestel et al., 2001), travel-time prediction (C.-H. Wu et al., 2004) and for estimating the amount of cells present inside a tumor (Newman et al., 2015).

Incorporating priors. In this last example of application, the authors used SVR to estimate a vector of proportions, however the classical SVR estimator does not take into account the information known about the space in which the estimator lives. Adding this prior information on the estimator may lead to better estimation performance. Incorporating information in the estimation process is a wide field of studies in statistical learning (we refer to Figure 2 in Lauer and Bloch (2008) for a quick overview in the context of SVM). A growing interest in prior knowledge incorporated as regularization terms has emerged in the last decades. Lasso (Tibshirani, 1996), Ridge (Hoerl and Kennard, 1970),

elastic-net (Zou and Hastie, 2005) regression are examples of regularized problem where a prior information is used to fix an ill-posed problem or an overdetermined problem. The ℓ_1 regularization of the Lasso will force the estimator to be sparse and bring statistical guarantees of the Lasso estimator in high dimensional settings. Another common way to add prior knowledge on the estimator is to add constraints known a-priori on this estimator. The most common examples are the ones that constrain the estimator to live in a subspace such as Non Negative Least Squares Regression (NNLS) (Lawson and Hanson, 1995), isotonic regression (Barlow and Brunk, 1972). These examples belong to a more general type of constraints: linear constraints. Other types of constraints exist like constraints on the derivative of the function that is to be estimated, smoothness of the function for example. Adding those constraints on the Least Squares estimator has been widely studied (Barlow and Brunk, 1972; Liew, 1976; Bro and De Jong, 1997) and similar work has been done for the Lasso estimator in (Gaines et al., 2018). Concerning the SVR, inequality and equality constraints added as prior knowledge were studied in Lauer and Bloch (2008). In this paper, the authors described a method for adding linear constraints on the Linear Programming SVR (Friel and Harrison, 1998). This implementation of the algorithm considers the ℓ_1 norm of the parameters in the optimization problem instead of the classical ℓ_2 norm which leads to a linear programming optimization problem to solve instead of a quadratic programming problem. They also described a method for using information about the derivative of the function that is estimated.

Sequential Minimal Optimization. One of the main challenges of adding these constraints is that it often increases the difficulty of solving the optimization problem related to the estimator. For example, the Least Squares optimization problem has a closed-form solution whereas the NNLS uses sophisticated algorithms (Bro and De Jong, 1997) to approach the solution. SVM and SVR algorithms were extensively studied and used in practice because very efficient algorithms were developed to solve the underlying optimization problems. One of them is called Sequential Minimal Optimization (SMO) (Platt, 1998) and is based on a well known optimization technique called coordinate descent. The idea of the coordinate descent is to break the optimization problem into sub-problems selecting one coordinate at each step and minimizing the function only via this chosen coordinate. The development of parallel algorithms have increased the interest in these coordinate descent methods which show to be very efficient for large scale problems. One of the key settings for the coordinate descent is the choice of the coordinate at each step, the choice's strategy will affect the efficiency of the algorithm. There exists three families of strategies for coordinate descent: cyclic (Tseng, 2001), random (Nesterov, 2012) and

greedy. The SMO algorithm is a variant of a greedy coordinate descent (Wright, 2015) and is the algorithm implemented in LibSVM (Chang and Lin, 2011). It is very efficient to solve SVM/SVR optimization problems. In the context of linear kernel, other algorithm are used such as dual coordinate descent (Hsieh et al., 2008) or trust region newton methods (Lin et al., 2007).

Priors and SMO. In one of the application of SVR cited above, information a-priori about the estimator is not used in the estimation process and is only used in a post-processing step. This application comes from the cancer research field, where regression algorithms have been used to estimate the proportions of cell populations that are present inside a tumor (see Mohammadi et al. (2017) for a survey). Several estimators have been proposed in the biostatistics literature, most of them based on constrained least squares (Abbas et al., 2009; Qiao et al., 2012; Gong et al., 2011) but the gold standard is the estimator based on the Support Vector Regression (Newman et al., 2015). Our work is motivated by incorporating the fact that the estimator for this application belongs to the simplex: $\mathcal{S} = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1,\ x_i \geq 0\}$ in the SVR problem. We believe that for this application, it will lead to better estimation performance. From an optimization point of view, our motivation is to find an efficient algorithm that is able to solve the SVR optimization problem where generic linear constraints are added to the problem as prior knowledge, including simplex prior as described. This work follows the one from Lauer and Bloch (2008) except that in our case, we keep the ℓ_2 norm on the parameters in the optimization problem which is the most common version of the SVR optimization problem and we only focus on inequality and equality constraints as prior knowledge.

4.2 Constrained Support Vector Regression

First we introduce the optimization problem related to adding linear constraints to the SVR and discuss some interesting properties about this problem.

4.2.1 Previous work : ν-Support Vector Regression

The ν-SVR estimator (Schölkopf et al., 1999) is obtained solving the following quadratic optimization problem:

$$\begin{aligned} \min_{\beta,\beta_0,\xi_i,\xi_i^*,\varepsilon} \quad & \frac{1}{2}||\beta||^2 + C(\nu\epsilon + \frac{1}{n}\sum_{i=1}^{n}(\xi_i + \xi_i^*)) \\ \text{subject to} \quad & y_i - X_{i:}\beta - \beta_0 \leq \varepsilon + \xi_i \\ & X_{i:}\beta + \beta_0 - y_i \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \varepsilon \geq 0 \ . \end{aligned} \tag{SVR-P}$$

By solving Problem (SVR-P), we seek a linear function $f(x) = \beta^\top x + \beta_0$ where $\beta \in \mathbb{R}^p$ and $\beta_0 \in \mathbb{R}$, that is at most ε deviating from the response vector coefficient y_i. This function does not always exist which is why slack variables $\xi \in \mathbb{R}^n$ and $\xi^* \in \mathbb{R}^n$ are introduced in the optimization problem to allow some observations to break the condition given before. C and ν are two hyperparameters. $C \in \mathbb{R}$ controls the tolerated error and $\nu \in [0, 1]$ controls the number of observations that will lay inside the tube of size 2ε given by the two first constraints in Problem (SVR-P). It is an ε-insensitive loss function where a linear penalization is put on the observations that lay outside the tube and the observations that lay inside the tube are not penalized (see Smola and Schölkopf (2004) for more details).

The different algorithms proposed to solve Problem (SVR-P) often use its dual problem like in Platt (1998); Hsieh et al. (2008). The dual problem is also a quadratic optimization problem with linear constraints but its structure allows an efficient resolution as we will see in more details in Section 4.3. The dual problem of Problem (SVR-P) is the following optimization problem:

$$\begin{aligned} \min_{\alpha,\alpha^* \in \mathbb{R}^n} \quad & \frac{1}{2}(\alpha - \alpha^*)^\top Q(\alpha - \alpha^*) + y^\top(\alpha - \alpha^*) \\ \text{subject to} \quad & 0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{n} \\ & \mathbb{1}^\top(\alpha + \alpha^*) \leq C\nu \\ & \mathbb{1}^\top(\alpha - \alpha^*) = 0 \ , \end{aligned} \tag{SVR-D}$$

where $Q = XX^\top \in \mathbb{R}^{n \times n}$.

The equation link between Problem (SVR-P) and Problem (SVR-D) is given by the following formula: $\beta = -\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)X_{i:}$.

4.2.2 The constrained optimization problem

We propose a constrained version of Problem (SVR-P) that allows the addition of prior knowledge on the linear function f that we seek to estimate. The constrained estimator is obtained solving the optimization problem:

$$\begin{aligned} \min_{\beta,\beta_0,\xi_i,\xi_i^*,\varepsilon} \quad & \frac{1}{2}||\beta||^2 + C(\nu\epsilon + \frac{1}{n}\sum_{i=1}^{n}(\xi_i + \xi_i^*)) \\ \text{subject to} \quad & X_{i:}\beta + \beta_0 - y_i \leq \varepsilon + \xi_i \\ & y_i - X_{i:}\beta - \beta_0 \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \varepsilon \geq 0 \\ & A\beta \leq b \\ & \Gamma\beta = d \ , \end{aligned} \tag{LSVR-P}$$

where $A \in \mathbb{R}^{k_1 \times p}$, $\Gamma \in \mathbb{R}^{k_2 \times p}$, $\beta \in \mathbb{R}^p$, $\xi, \xi^* \in \mathbb{R}^n$ and $\beta_0 \in \mathbb{R}$, $\varepsilon > 0$.

The algorithm that we propose in Section 4.3 also uses the structure of the dual problem of Problem (LSVR-P). The next proposition introduces the dual problem and some of its properties.

Proposition 4.1. *If the set* $\{\beta \in \mathbb{R}^n, A\beta \leq b, \Gamma\beta = d\}$ *is not empty then,*

1. *Strong duality holds for Problem* (LSVR-P).

2. *The dual problem of Problem (LSVR-P) is*

$$
\begin{aligned}
\min_{\alpha,\alpha^*,\gamma,\mu} \quad & \frac{1}{2}\Big[(\alpha-\alpha^*)^\top Q(\alpha-\alpha^*) + \gamma^\top AA^\top\gamma \\
& + \mu^\top\Gamma\Gamma^\top\mu + 2\sum_{i=1}^{n}(\alpha_i-\alpha_i^*)\gamma^\top AX_{i:} \\
& - 2\sum_{i=1}^{n}(\alpha_i-\alpha_i^*)\mu^\top\Gamma X_{i:} - 2\gamma^\top A\Gamma^\top\mu\Big] \\
& + y^\top(\alpha-\alpha^*) + \gamma^\top b - \mu^\top d \\
\textit{subject to} \quad & 0 \le \alpha_i^{(*)} \le \frac{C}{n} \\
& \mathbb{1}^\top(\alpha+\alpha^*) \le C\nu \\
& \mathbb{1}^\top(\alpha-\alpha^*) = 0 \\
& \gamma_j \ge 0 \ ,
\end{aligned}
\tag{LSVR-D}
$$

3. *The equation link between primal and dual is*

$$\beta = -\sum_{i=1}^{n}(\alpha_i-\alpha_i^*)X_{i:} - A^\top\gamma + \Gamma^\top\mu \ .$$

The proof of the first statement of the proposition is given in the discussion below whereas the proofs for the two other statements are straightforward and derive from classical Lagrangian computation.

We have that α, $\alpha^* \in \mathbb{R}^n$, $\gamma \in \mathbb{R}^{k_1}$ are the vector of Lagrange multipliers associated the the inequality constraint $A\beta \le b$ which explains the non-negative constraints on its coefficients. The coefficients $\mu \in \mathbb{R}^{k_2}$ are the Lagrange multipliers associated to the equality constraint $\Gamma\beta = d$ which also explains that there are no constraints in the dual problem on μ. The objective function f which we will write in the stacked form as:

$$f(\theta) = \theta^\top\bar{Q}\theta + l^\top\theta,$$

where

$$\theta = \begin{bmatrix}\alpha\\ \alpha^*\\ \gamma\\ \mu\end{bmatrix}, \ l = \begin{bmatrix}y\\ -y\\ b\\ -d\end{bmatrix} \in \mathbb{R}^{2n+k_1+k_2},$$

$$\bar{Q} = \begin{bmatrix} Q & -Q & XA^\top & -X\Gamma^\top \\ -Q & Q & -XA^\top & X\Gamma^\top \\ AX^\top & -AX^\top & AA^\top & -A\Gamma^\top \\ -\Gamma X^\top & \Gamma X^\top & -\Gamma A^\top & \Gamma\Gamma^\top \end{bmatrix}$$

is a square matrix of size $2n + k_1 + k_2$.

An important observation is that this objective function is always convex. The matrix $\bar{Q}$ is the product of the matrix $\begin{bmatrix} X \\ -X \\ A \\ -\Gamma \end{bmatrix}$ and its transpose matrix. It means that $\bar{Q}$ is a Gramian matrix and it is positive semidefinite which implies that f is convex. Problem (LSVR-D) is then a quadratic programming optimization problem which meets Slater's condition if there exists a θ that belongs to the feasible domain which we will denote by $\mathcal{F}$. If there is such a θ we have strong duality holding between Problem (LSVR-P) and Problem (LSVR-D). The only condition to have the existence of a minimum on A and Γ is that they define a non-empty polyhedron in order to be able to solve the optimization problem.

Our second observation on Problem (LSVR-D) is that the inequality constraint:

$$\mathbb{1}^\top(\alpha + \alpha^*) \leq C\nu \ ,$$

is replaced by an equality constraint in the same way that it was suggested in Chang and Lin (2002) for the classical Problem (SVR-D).

Proposition 4.2. *If $\varepsilon > 0$, all optimal solutions of Problem (LSVR-D) satisfy*

1. $\alpha_i \alpha_i^* = 0, \forall i \in [n]$
2. $\mathbb{1}^\top(\alpha + \alpha^*) = C\nu$

Proof. To prove part 1., Let us recall that α_i, α_i^* are the Lagrange multipliers associated to the optimization Problem (LSVR-P) constraints:

$$\begin{aligned} X_{i:}\beta + \beta_0 - y_i &\leq \varepsilon + \xi_i \\ y_i - X_{i:}\beta - \beta_0 &\leq \varepsilon + \xi_i^* \ . \end{aligned}$$

The complementary optimality conditions leads to

$$\alpha_i(X_{i:}\beta + \beta_0 - y_i - \varepsilon - \xi_i) = 0$$
$$\alpha_i^*(y_i - X_{i:}\beta - \beta_0 - \varepsilon - \xi_i^*) = 0 \ .$$

Let us now suppose that $\alpha_i > 0$ and $\alpha_i^* > 0$ which implies that

$$X_{i:}\beta + \beta_0 - y_i - \varepsilon - \xi_i = 0$$
$$y_i - X_{i:}\beta - \beta_0 - \varepsilon - \xi_i^* = 0 \ .$$

It follows that $-2\varepsilon = \xi_i + \xi_i^*$ and $\xi_i, \xi_i^* \geq 0$ which implies $\xi_i = \xi_i^* = \varepsilon = 0$. This goes against our condition $\varepsilon > 0$.

To prove part 2., we need to remind the optimality conditions of Problem (LSVR-P);

$$(C\nu - \sum_{i=1}^{l} \alpha_i + \alpha_i^*)\varepsilon = 0 \ . \tag{4.1}$$

Thus, if $\varepsilon > 0$ we have that $\sum_{i=1}^{n} \alpha_i + \alpha_i^* = C\nu$. □

This observation will be important for the algorithm that we propose in Section 4.3.

4.3 Generalized Sequential Minimal Optimization

In this section we propose a generalization of the SMO algorithm (Platt, 1998) to solve Problem (LSVR-D) and present our main result on the convergence of the proposed algorithm to the solution of Problem (LSVR-D). The SMO algorithm is a variant of greedy coordinate descent taking into consideration non-separable constraints, which in our case are the two equality constraints. We start by describing the previous algorithm that solve Problem (SVR-D).

4.3.1 Previous work : Sequential Minimal Optimization

In this subsection, we define $f(\alpha, \alpha^*) = \frac{1}{2}(\alpha - \alpha^*)^\top Q(\alpha - \alpha^*) + y^\top(\alpha - \alpha^*)$ and we note $\nabla f \in \mathbb{R}^{2n}$ its gradient. Using Keerthi and Gilbert (2002), we rewrite the Karush-Kuhn-

Tucker (KKT) conditions in the following way:

$$\min_{i \in I_{\text{up}}} \nabla_{\alpha_i} f \geq \max_{j \in I_{\text{low}}} \nabla_{\alpha_j} f \ , \tag{4.2}$$

where $I_{\text{up}}(\alpha) = \{i \in [n] : \alpha_i < \frac{C}{l}\}$ and $I_{\text{low}}(\alpha) = \{i \in [n] : \alpha_i > 0\}$.

The same condition is written for the α^* variables replacing α_i by α_i^* above. These conditions lead to an important definition for the rest of this paper.

Definition 4.1. We will say that (i, j) is a violating pair of variables if one of these two conditions is satisfied:

$$i \in I_{\text{up}}(\alpha), j \in I_{\text{low}}(\alpha) \text{ and } \nabla_{\alpha_i} f < \nabla_{\alpha_j} f$$
$$i \in I_{\text{low}}(\alpha), j \in I_{\text{up}}(\alpha) \text{ and } \nabla_{\alpha_i} f > \nabla_{\alpha_j} f.$$

Because the algorithm SMO does not provide in general an exact solution in a finite number of steps there is a need to relax the optimality conditions which gives a new definition.

Definition 4.2. We will say that (i, j) is a τ-violating pair of variables if one of these two conditions is satisfied:

$$i \in I_{\text{up}}(\alpha), j \in I_{\text{low}}(\alpha) \text{ and } \nabla_{\alpha_i} f < \nabla_{\alpha_j} f - \tau$$
$$i \in I_{\text{low}}(\alpha), j \in I_{\text{up}}(\alpha) \text{ and } \nabla_{\alpha_i} f > \nabla_{\alpha_j} f + \tau.$$

The SMO algorithm will then choose at each iteration a pair of violating variables in the α block or in the α^* block. Once the choice is done, a subproblem of size two is solved, considering that only the two selected variables are to be minimized in Problem (SVR-D). The outline of the algorithm is presented in Algorithm 4.

The choice of the violating pair of variables presented in Keerthi et al. (2001) was to always work with the most violating pairs of variables, which means the variables that leads to the largest gap compared to the optimality conditions given in Equation (4.2). This choice is what makes a link with greedy coordinate descent, however greedy here is related to the largest gap with the optimality score and is not related to the largest decrease in the objective function.

The resolution of the subproblem of size two has a closed-form. The idea is to use the two equality constraints to go from a problem of size two to a problem of size one. Then, the goal is to minimize a quadratic function of one variable under box constraints which is

done easily. We will give more details of the resolution of these subproblems in Section 4.3.3 for our proposed algorithm.

The proof of convergence of SMO algorithm was given in Keerthi and Gilbert (2002) without convergence rate. The proof relies on showing that the sequence defined by the algorithm $f(\alpha^k, (\alpha^*)^k)$ is a decreasing sequence and that there cannot be the same violating pair of variables infinitely many times. The linear convergence rate was proved later by She and Schmidt (2017) as well as the identification of the support vectors in finite time.

Algorithm 4 SEQUENTIAL MINIMAL OPTIMIZATION (PLATT, 1998)

input : $\tau > 0$, $\alpha^{(0)}, (\alpha^*)^{(0)} \in \mathcal{F}$
$k = 0$
while $\Delta > \tau$ **do**
  $i \leftarrow \arg\min_{i \in I_{\text{up}}} \nabla_{\alpha_i} f$ $j \leftarrow \arg\min_{j \in I_{\text{low}}} \nabla_{\alpha_j} f$ // Most violating pair in the block α
  $i^* \leftarrow \arg\min_{i \in I_{\text{up}}} \nabla_{\alpha_i} f$ $j^* \leftarrow \arg\min_{j \in I_{\text{low}}} \nabla_{\alpha_j} f$ // Most violating pair in the block α^*
  $\Delta_1 \leftarrow \nabla_{\alpha_j} f - \nabla_{\alpha_i} f$ // KKT violation score in the block α
  $\Delta_2 \leftarrow \nabla_{\alpha_{j^*}} f - \nabla_{\alpha_{i^*}} f$ // KKT violation score in the block α^*
  $\Delta = \max(\Delta_1, \Delta_2)$
  if $\Delta = \Delta_1$ **then**
    $\alpha^{(k+1)} \leftarrow$ Solution of subproblem for variables α_i and α_j
  else
    $(\alpha^*)^{(k+1)} \leftarrow$ Solution of subproblem for variables α_{i^*} and α_{j^*}
  $k \leftarrow k + 1$
return $\alpha^{(k)}, (\alpha^*)^{(k)}$

4.3.2 Optimality conditions for the constrained SVR

In this subsection we define f as the objective function of Problem (LSVR-D) and $\nabla f \in \mathbb{R}^{2n+k_1+k_2}$ its gradient. We will now give the KKT conditions of Problem (LSVR-D) for the different block. The results derive from classical Lagrangian calculation.

The α block. The optimality conditions are satisfied if and only if

$$\min_{i \in I_{\text{up}}} \nabla_{\alpha_i} f \geq \max_{j \in I_{\text{low}}} \nabla_{\alpha_j} f \quad ,$$

where

$$I_{\text{up}}(\alpha) = \{i \in [n] : \alpha_i < \frac{C}{n}\},$$
$$I_{\text{low}}(\alpha) = \{i \in [n] : \alpha_i > 0\} \ .$$

The α^* block. In this block, the conditions are very similar to the ones given for the block α, which gives the following optimality condition:

$$\min_{i \in I^*_{\text{up}}} \nabla_{\alpha^*_i} f \geq \max_{j \in I^*_{\text{low}}} \nabla_{\alpha^*_j} f \ ,$$

where

$$I^*_{\text{up}}(\alpha^*) = \{i \in [n] : \alpha^*_i < \frac{C}{n}\},$$
$$I^*_{\text{low}}(\alpha) = \{i \in [n] : \alpha^*_i > 0\} \ .$$

The γ block. The optimality conditions in this block lead to the following conditions, for every $j \in [k_1]$:

$$\nabla_{\gamma_j} f \geq 0 \ .$$

Definition 4.3. We will say that j is a τ-violating variable for the block γ if $\nabla_{\gamma_j} f + \tau < 0$.

The μ block. The optimality conditions in this block lead to the following conditions, for every $j \in [k_2]$:

$$\nabla_{\mu_j} f = 0 \ .$$

Definition 4.4. We will say that j is a τ-violating variable for the block μ if $|\nabla_{\mu_j} f| > \tau$.

From these conditions on each block, we build an optimization strategy that follows the idea of the SMO described in Section 4.3.1. For each block of variables, we compute what we call a *violating optimality score* based on the optimality conditions given above. Once the scores are computed for each block, we select the block which has the largest score and solve an optimization subproblem in the block selected.

If the block α or the block α^* is selected, we will update a pair of variables by solving a

Algorithm 5 SEQUENTIAL MINIMAL OPTIMIZATION ALGORITHM

input : $\tau > 0$, $\alpha^{(0)}, (\alpha^*)^{(0)}, \gamma^{(0)}, \mu^{(0)} \in \mathcal{F}$
$k = 0$
while $\Delta > \tau$ **do**
  $i \leftarrow \arg\min_{i \in I_{\text{up}}} \nabla_{\alpha_i} f$ $j \leftarrow \arg\min_{j \in I_{\text{low}}} \nabla_{\alpha_j} f$ // Most violating pair in the block α
  $i^* \leftarrow \arg\min_{i \in I_{\text{up}}} \nabla_{\alpha_i} f$ $j^* \leftarrow \arg\min_{j \in I_{\text{low}}} \nabla_{\alpha_j} f$ // Most violating pair in the block α^*
  $\Delta_1 \leftarrow \nabla_{\alpha_j} f - \nabla_{\alpha_i} f$ // KKT violation score in the block α
  $\Delta_2 \leftarrow \nabla_{\alpha_{j^*}} f - \nabla_{\alpha_{i^*}} f$ // KKT violation score in the block α^*
  $\Delta_3 \leftarrow - \min_{j \in \{1,\dots,k_1\}} \nabla_{\gamma_j} f$ // KKT violation score in the block γ
  $\Delta_4 \leftarrow \max_{j \in \{1,\dots,k_2\}} |\nabla_{\mu_j} f|$ // KKT violation score in the block μ
  $\Delta = \max(\Delta_1, \Delta_2, \Delta_3, \Delta_4)$ // Block and index selection to update.
  if $\Delta = \Delta_1$ **then**
    $\alpha^{(k+1)} \leftarrow$ Solution of subproblem for variables α_i and α_j
  else if $\Delta = \Delta_2$ **then**
    $(\alpha^*)^{(k+1)} \leftarrow$ Solution of subproblem for variables α_{i^*} and α_{j^*}
  else if $\Delta = \Delta_3$ **then**
    $i = \arg\min_{i \in \{1,\dots,k_1\}} \nabla_{\gamma_i} f$ $\gamma^{k+1} \leftarrow$Solution of subproblem for γ_i
  else
    $i = \arg\max_{i \in \{1,\dots,k_2\}} |\nabla_{\mu_i} f|$ $\mu^{k+1} \leftarrow$ Solution of subproblem for μ_i
  $k \leftarrow k + 1$
return $\alpha^{(k)}, (\alpha^*)^{(k)}, \gamma^{(k)}, \mu^{(k)}$

minimization problem of size two. However if the block γ or the block μ is selected, we will update only one variable at a time.

This is justified by the fact that the variables α and α^* have non-separable equality constraints linking them together. The rest of this section will be dedicated to the presentation of our algorithm and to giving some interesting properties such as a closed-form for updates on each of the blocks and a convergence theorem.

4.3.3 Updates rules and convergence

The first definition describes the closed-form updates for the different blocks of variables.

Definition 4.5. The update between iterate k and iterate $k + 1$ of the generalized SMO algorithm has the following form:

1. if the block α is selected and (i, j) is the most violating pair of variable then the

update will be as follows:

$$\alpha_i^{k+1} = \alpha_i^k + t^*$$
$$\alpha_j^{k+1} = \alpha_j^k - t^*,$$

where $t^* = \min(\max(I_1, -\frac{(\nabla_{\alpha_i} f - \nabla_{\alpha_j} f)}{(Q_{ii} - 2Q_{ij} + Q_{jj})}), I_2)$ with $I_1 = \max(-\alpha_i^k, \alpha_j^k - \frac{C}{n})$ and $I_2 = \min(\alpha_j^k, \frac{C}{n} - \alpha_i^k)$.

2. if the block α^* is selected and (i^*, j^*) is the most violating pair of variable then the update will be as follows:

$$(\alpha_i^*)^{k+1} = (\alpha_i^*)^k + t^*$$
$$(\alpha_j^*)^{k+1} = (\alpha_j^*)^k - t^*,$$

where $t^* = \min(\max(I_1, -\frac{(\nabla_{\alpha_i^*} f - \nabla_{\alpha_j^*} f)}{(Q_{ii} - 2Q_{ij} + Q_{jj})}), I_2)$ with $I_1 = \max(-(\alpha_i^*)^k, (\alpha_j^*)^k - \frac{C}{n})$ and $I_2 = \min((\alpha_j^*)^k, \frac{C}{n} - (\alpha_i^*))^k$.

3. if the block γ is selected and i is the index of the most violating variable in this block then the update will be as follows:

$$\gamma_i^{k+1} = \max(-\frac{\nabla_{\gamma_i} f}{(AA^\top)_{ii}} + \gamma_i^k, 0).$$

4. if the block μ is selected and i is the index of the most violating variable in this block then the update will be as follows:

$$\mu_i^{k+1} = -\frac{\nabla_{\mu_i f}}{(\Gamma\Gamma^\top)_{ii}} + \mu_i^k.$$

This choice of updates comes from solving the optimization Problem (LSVR-D) considering that only one or two variables are updated at each step. One of the key elements of the algorithm is to make sure that at each step the iterate belongs to $\mathcal{F}$. Let us suppose that the block α is selected as the block in which the update will happen and let (i, j) be the most violating pair of variables. The update is the resolution of a subproblem of size 2, considering that only α_i and α_j are the variables, the rest remains constant. The two equality constraints in Problem (LSVR-D), $\sum_{i=1}^n \alpha_i - \alpha_i^* = 0$ and $\sum_{i=1}^n \alpha_i + \alpha_i^* = C\nu$, lead to the two following equalities: $\alpha_i^{k+1} + \alpha_j^{k+1} = \alpha_i^k + \alpha_j^k$. The later yields to using a parameter t for the update of the variables leading to $\alpha_i^{k+1} = \alpha_i^k + t$ and $\alpha_j^{k+1} = \alpha_j^k - t$.

Updating the variables in the block α this way will force the iterates of Algorithm 4 to meet the two equalities constraints at each step. We find t by solving Problem (LSVR-D) considering that we minimize only over t. Let $u \in \mathbb{R}^{2n+p+k_1+k_2}$ be the vector that contains

only zeros except at the i^{th} coordinate where it is equal to t and at j^{th} coordinate where it is equal to $-t$. Therefore, we find t by minimizing the following optimization problem:

$$\begin{aligned} \min_{t\in\mathbb{R}} \quad & \psi(t) = \frac{1}{2}\Big[(\theta^k + u)^\top \bar{Q}(\theta^k + u)\Big] + l^\top(\theta^k + u) \\ \text{subject to} \quad & 0 \leq \alpha_i^{k+1}, \alpha_j^{k+1} \leq \frac{C}{n}. \end{aligned}$$

First we minimize the objective function without the constraints and since it is a quadratic function of one variable we just clip the solution of unconstrained problem to have the solution of the constrained problem. We will use the term "clipped update" or "clipping" when the update is projected onto the constraints space and is not the result of the unconstrained optimization problem. As we only consider size one problem for the updates, it will mean that the update will be a bound of an interval. We will use the notation K as a term containing the terms that do not depend on t. We write that

$$\begin{aligned} \psi(t) &= \frac{1}{2}u^\top \bar{Q}u + u^\top \bar{Q}\theta^k + l^\top u + K \\ &= \frac{1}{2}t^2(\bar{Q}_{ii} + \bar{Q}_{jj} - 2\bar{Q}_{ij}) + u^\top \nabla f(\theta^k) + K \\ &= \frac{1}{2}t^2(\bar{Q}_{ii} + \bar{Q}_{jj} - 2\bar{Q}_{ij}) + t(\nabla_{\alpha_i} f(\theta^k) \\ &\quad - \nabla_{\alpha_j} f(\theta^k)) + K. \end{aligned}$$

It follows that the unconstrained minimum of $\psi(t)$ is $t_q = \frac{-(\nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k))}{(\bar{Q}_{ii} + \bar{Q}_{jj} - 2\bar{Q}_{ij})}$.

Taking the constraints into account we have that $0 \leq \alpha_i^k + t \leq \frac{C}{n}$ and $0 \leq \alpha_j^k - t \leq \frac{C}{n}$, which yields to $t^* = \min(\max(I_1, t_q), I_2)$ with $I_1 = \max(-\alpha_i, \alpha_j - \frac{C}{n})$ and $I_2 = \min(\alpha_j, \frac{C}{n} - \alpha_i)$. The definition of the updates for the block α^* relies on the same discussion.

Let us now make an observation that will explain the definition of the updates for the blocks γ and μ. Let i be the index of the variable that will be updated. Solving the problem; $\theta_i^{k+1} = \arg\min_{\theta_i} \frac{1}{2}\theta^\top \bar{Q}\theta + l^\top \theta$, leads to the following solution $\theta_i^{k+1} = \frac{-\nabla_i f(\theta^k)}{\bar{Q}_{ii}} + \theta_i^k$.

Let us recall that the update for the block γ has to keep the coefficient of γ positive to stay in $\mathcal{F}$ hence we have to perform the following clipped update with $i \in \{2n+p+1, \ldots, 2n+p+k_1\}$:

$$\theta_i^{k+1} = \max(\frac{-\nabla_{\gamma_i} f(\theta^k)}{\bar{Q}_{ii}} + \theta_i^k, 0).$$

Then noticing that $\bar{Q}_{ii} = AA_{ii}^\top$ for this block, we obtain the update for the block γ.

There are no constraints on the variables in the block μ, so the update comes from the fact that $\bar{Q}_{ii} = \Gamma\Gamma_{ii}^\top$ for $i \in \{2n + p + k_1 + 1, \dots, 2n + p + k_1 + k_2\}$ which corresponds to the indices of the block μ. It comes back to performing a projected coordinate descent step in the block γ and a simple coordinate descent step in the block μ.

From these updates we have to make sure that

$$Q_{ii} + Q_{jj} - 2Q_{ij} \neq 0,$$

let us recall that $Q_{ij} = \langle X_{i:}, X_{j:} \rangle$ which means that

$$Q_{ii} + Q_{jj} - 2Q_{ij} = ||X_{i:} - X_{j:}||^2.$$

This quantity is zero only when $X_{i:} = X_{j:}$ coordinatewise. It would mean that the same row appears two times in the design matrix which does not bring any new information for the regression and can be avoided easily. $(AA^\top)_{ii} = \langle A_{i:}, A_{i:} \rangle$ is zero if and only if $A_{i:} = 0$ which means that a row of the matrix A is zero, so there are no constraint on any variable of the optimization problem which will never happen. It is the same discussion for $(\Gamma\Gamma^\top)_{ii}$.

The next proposition makes sure that once a variable (resp. pair of variables) is updated, it cannot be a violating variable (resp. pair of variables) at the next step. This proposition makes sure, for the two blocks α and α^*, that the update t^* cannot be 0.

Proposition 4.3. *If (i, j) (resp.i) was the pair of most violating variable (resp. the most violating variable) in the block α or α^* (resp. block γ or μ) at iteration k then at iteration $k + 1$, (i, j) (resp. i) cannot be violating the optimality conditions.*

Proof. To prove Proposition 4.3, we start by giving a lemma that will be useful to prove the proposition for the blocks α and α^*.

Lemma 4.1. *If the update between iteration k and $k + 1$ happens in the block α (or α^*) and that (i, j) is the most violating pair of variables then*

$$\nabla_{\alpha_i} f(\theta^{k+1}) - \nabla_{\alpha_j} f(\theta^{k+1}) = \nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k) + t^*(Q_{ii} + Q_{jj} - 2Q_{ij}) \ .$$

Proof. Let us recall that the update in the block α (or α^*) has the following form ; $\alpha_i^{k+1} = \alpha_i^k + t^*$ and $\alpha_j^{k+1} = \alpha_j^k - t^*$, with t^* as defined in Definition 4.5. In a stacked form we have

that

$$\begin{aligned}\nabla_{\alpha_i} f(\theta^{k+1}) - \nabla_{\alpha_j} f(\theta^{k+1}) &= (Q\theta^{k+1})_i + l_i - (Q\theta^{k+1})_j - l_j \\ &= \sum_{s=1}^{2n+k_1+k_2} Q_{is}\theta_s^{k+1} + l_i - \sum_{s=1}^{2n+k_1+k_2} Q_{js}\theta_s^{k+1} - l_j \\ &= \nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k) + t^*(Q_{ii} - Q_{ij}) + t^*(Q_{jj} - Q_{ij}) \\ &= \nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k) + t^*(Q_{ii} + Q_{jj} - 2Q_{ij}) \ .\end{aligned}$$

□

This lemma is helpful for the blocks γ and μ.

Lemma 4.2. *If θ_i is the updated variable at iteration k, then it holds that:*

$$\nabla_i f(\theta^{k+1}) = \bar{Q}_{ii}(\theta_i^{k+1} - \theta_i^k) + \nabla_i f(\theta^k) \ .$$

Proof. The proof is straightforward,

$$\begin{aligned}\nabla_i f(\theta^{k+1}) =& (\bar{Q}\theta_i^{k+1})_i + l_i \\ =& \sum_{s=1}^{2n+k_1+k_2} \bar{Q}_{is}\theta_s^{k+1} + l_i \\ =& \sum_{s\neq i}^{2n+k_1+k_2} \bar{Q}_{is}\theta_s^{k+1} + l_i + \bar{Q}_{ii}\theta_i^{k+1} \\ =& \nabla_i f(\theta_k) + \bar{Q}_{ii}\theta_i^{k+1} - \bar{Q}_{ii}\theta_i^k \\ =& \bar{Q}_{ii}(\theta_i^{k+1} - \theta_i^k) + \nabla_i f(\theta^k) \ .\end{aligned}$$

□

We now consider that the update between iteration k and $k+1$ takes place in the block α. We will define (i, j) as the most violating pair of variables as defined in Section 4.3. From the discussion in Section 4.3.3, we know that minimizing the objective function of Problem (LSVR-D) considering that only the parameter t is a variable leads to minimizing the following function:

$$\psi(t) = \frac{1}{2}t^2(\bar{Q}_{ii} + \bar{Q}_{jj} - 2\bar{Q}_{ij}) + t(\nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k)) + K \ . \tag{4.3}$$

We recall that t is the parameter that will be used for the update of α_i and α_j and K is a

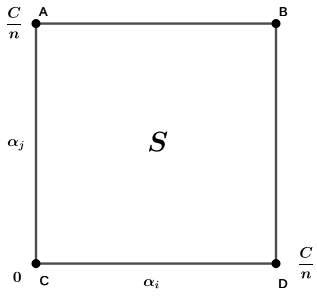

Figure 4.1 – Possible update for the block α or α^*

constant term. We also have the following result from Lemma 4.1:

$$\begin{aligned}\nabla_{\alpha_i} f(\theta^{k+1}) - \nabla_{\alpha_j} f(\theta^{k+1}) =& \nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k) \\ &+ (\bar{Q}_{ii} + \bar{Q}_{jj} - 2\bar{Q}_{ij}) t^* \ .\end{aligned} \tag{4.4}$$

The minimization update takes place in the square $S = [0, \frac{C}{n}] \times [0, \frac{C}{n}]$ illustrated in Figure 4.1.

At points B and C of the square S, (i, j) cannot be a τ-violating pair of variables because they belong to the same set of indices I_{up} (or I_{low}). Everywhere else, violation can take place.

- On $]CA]$, $\alpha_i = 0$ and $\alpha_j > 0$ so $i \in I_{\text{up}}$ and $j \in I_{\text{low}}$ which means that by definition of τ-violating pair of variable

$$\nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k) < -\tau < 0 \ ,$$

  which means $t_q = \frac{-(\nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k))}{(\bar{Q}_{ii} + \bar{Q}_{jj} - 2\bar{Q}_{ij})} > 0$. Let us remind that :

$$\max(-\alpha_i, \alpha_j - \frac{C}{n}) \leq t^* \leq \min(\frac{C}{n} - \alpha_i, \alpha_j) \ . \tag{4.5}$$

  It means that on $]CA]$, Equation (4.5) becomes : $0 \leq t^* \leq \alpha_j$ There are then two possibilities:

  - if $t_q \geq \alpha_j$, it implies because of the constraints on t^*, that $t^* = \alpha_j$. The update becomes then $\alpha_i^{k+1} = \alpha_i^k + \alpha_j^k$ and $\alpha_j^{k+1} = 0$. Then j belongs to the set of indices I_{up} and i belongs to I_{low}. From Equation (4.4), we deduce that $\nabla_{\alpha_i} f(\theta^{k+1}) - \nabla_{\alpha_j} f(\theta^{k+1}) \leq 0$ which proves that (i, j) is not a violating pair of

variable anymore and that $\alpha^{k+1} \neq \alpha^k$

- Second possibility is that $t_q \leq \alpha_j$ then $t^* = t_q$, then $(\alpha_i^{k+1}, \alpha_j^{k+1})$ belongs to int(S). From Equation (4.4), we deduce that $\nabla_{\alpha_i} f(\theta^{k+1}) - \nabla_{\alpha_j} f(\theta^{k+1}) = 0$, (i, j) is not a τ-violating pair of variables anymore and $\alpha^{k+1} \neq \alpha^k$.

The same reasoning can be done on each segment of the edge of the square and also for points that are inside it. Moreover, it stays true for the block α^* and the proof is similar.

Let us now prove that when the update takes place at index i in the block γ then i is not violating variable at iteration $k+1$. Then we need to show that $\nabla_{\gamma_i} f(\theta^{k+1}) \geq 0$. Let us start with the case where the update $\gamma_i^{k+1} = \frac{\nabla_{\gamma_i} f(\theta^k)}{\bar{Q}_{ii}} - \gamma_i^k$. Using Lemma 4.2, we have that $\nabla_{\gamma_i} f(\theta^{k+1}) = 0$. The second possible case is $\gamma_i^{k+1} = 0$ because $-\frac{\nabla_{\gamma_i} f(\theta^k)}{\bar{Q}_{ii}} + \gamma_i^k \leq 0$. If $\gamma_i^{k+1} = 0$ then $\nabla_{\gamma_i} f(\theta^{k+1}) = -\bar{Q}_{ii}\gamma_i^k + \nabla_{\gamma_i} f(\theta^k)$. $\bar{Q}_{ii}$ is positive because it is a diagonal element of a Gram matrix $(A^\top A)$ thus we get that $\nabla_{\gamma_i} f(\theta^{k+1}) \geq 0$, which proves that i is not a violating variable anymore.

The proof for the block μ relies on the same idea except that it is simpler because there is no clipped updates possible so $\nabla_{\mu_i} f(\theta^{k+1}) = 0$ if the updates takes place at μ_i which also proves that i is not a violating variable for this block of variables anymore.

□

Finally, we show that the algorithm converges to a solution of Problem (LSVR-D) and since strong duality holds it allows us to have a solution of Problem (LSVR-P).

Theorem 4.1. *For any given $\tau > 0$ the sequence of iterates $\{\theta^k\}$, defined by the generalized SMO algorithm, converges to an optimal solution of the optimization Problem* (LSVR-D)

The proof of this theorem relies on the same idea as the one proposed in Lopez and Dorronsoro (2012) for the classical SMO algorithm and is given in Section 4.5. We show that it can be extended to our algorithm with some new elements. The general idea of the proof is to see that the distance between the primal vector generated by the SMO-algorithm and the optimal solution of the primal is controlled by the following expression $\frac{1}{2}||\beta^k - \beta^{\text{opt}}|| \leq f(\theta^k) - f(\theta^{\text{opt}})$, where β^k is the k^{th} primal iterate obtained via the relationship primal-dual and θ^k and where β^{opt} is a solution of Problem (LSVR-P). From this observation, we show that we can find a subsequence of the SMO-algorithm θ^{k_j} that converges to some $\bar{\theta}$, solution of the dual problem. Using the continuity of the objective function of the dual problem, we have that $f(\theta^{k_j}) \to f(\bar{\theta})$. Finally, we show that the sequence $\{f(\theta^k)\}$ is decreasing and bounded which implies its convergence and from the

convergence monotone theorem we know that to $f(\theta^k)$ converges to $f(\bar{\theta})$ since one of its subsequence converges. This proves that $||\beta^k - \beta^{\text{opt}}|| \to 0$ and finishes the proof. The convergence rate for the SMO algorithm is difficult to obtain considering the greedy choice of the blocks and the greedy choice inside the blocks. A proof for the classical SMO exists but with uniformly at random choice of the block (She and Schmidt, 2017). Convergence rate for greedy algorithms in optimization can be found in Nutini et al. (2015) for example but the assumption that the constraints must be separable is a major issue for our case. The study of this convergence rate is left for future work.

4.4 Numerical experiments

The code for the different regression settings is available on a GitHub repository[1], each setting is wrapped up in a package and is fully compatible with scikit-learn (Pedregosa et al., 2011) `BaseEstimator` class.

In order to compare the estimators, we worked with the Mean Absolute Error:

$$\text{MAE} = \frac{1}{p}\sum_{i=1}^{p} |\beta_i^* - \hat{\beta}_i| \ ,$$

and the Root Mean Squared Error;

$$\text{RMSE} = \sqrt{\frac{1}{p}||\beta^* - \hat{\beta}||^2} \ ,$$

where β^* are the ground truth coefficients and $\hat{\beta}$ are the estimated coefficients. We also used the Signal-To-Noise Ratio (SNR) to control the level noise simulated in the data defined as

$$\text{SNR} = 10\log 10(\frac{\mathbb{E}(X\beta(X\beta)^\top)}{\mathbb{V}\text{ar}(\epsilon)}) \ .$$

4.4.1 Non Negative regression

First, the constraints are set to force the coefficient of β to be positive and we compare our constrained-SVR estimator with the NNLS (Lawson and Hanson, 1995) estimator which

[1] https://github.com/Klopfe/LSVR

is the result of the following optimization problem

$$\begin{aligned} \min_{\beta} \quad & \frac{1}{2}||y - X\beta|| \\ \text{subject to} \quad & \beta_j \geq 0, \forall j \in [p] \ . \end{aligned} \tag{NNLS}$$

In this special case of non-negative regression, $A = -I_p$, $b = 0$, $\Gamma = 0$, $d = 0$, the constrained-SVR optimization problem which we will call Non-Negative SVR (NNSVR) then becomes

$$\begin{aligned} \min_{\beta,\beta_0,\xi_i,\xi_i^*,\varepsilon} \quad & \frac{1}{2}||\beta||^2 + C(\nu\varepsilon + \frac{1}{n}\sum_{i=1}^{n}(\xi_i + \xi_i^*)) \\ \text{subject to} \quad & X_{i:}\beta + \beta_0 - y_i \leq \varepsilon + \xi_i \\ & y_i - X_{i:}\beta - \beta_0 \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \varepsilon \geq 0 \\ & \beta_j \geq 0, \forall j \in [p] \ . \end{aligned} \tag{NNSVR}$$

Synthetic data. We generated the design matrix X from a Gaussian distribution $\mathcal{N}(0, 1)$ with 500 samples and 50 features. The true coefficients to be found β^* were generated taking the exponential of a Gaussian distribution $\mathcal{N}(0, 2)$ in order to have positive coefficients. Y was simply computed as the product between X and β^*. We wanted to test the robustness of our estimator compared to NNLS and variant of SVR estimators. To do so, we simulated noise in the data using different types of distributions, we tested Gaussian noise and laplacian noise under different levels of noise. For this experiment, the noise distributions were generated to have a SNR equals to 10 and 20, for each type of noise we performed 50 repetitions. The noise was only added in the matrix Y the design matrix X was left noiseless. We compared different estimators NNLS, NNSVR, the Projected-SVR (P-SVR) which is simply the projection of the classical SVR estimator onto the positive orthant and also the classical SVR estimator without constraints. The results of this experiment are in Table 4.1. We see that for a low Gaussian noise level ($\mathrm{SNR} = 20$) the NNLS has a lower RMSE and lower MAE. However, we see that the differences between the four compared methods are small. When the level of noise increases ($\mathrm{SNR} = 10$), the NNSVR estimator is the one with the lowest RMSE and MAE. The NNLS estimator performs poorly in the presence of high level of noise in comparison to the SVR based estimator. When a laplacian noise is added to the data, the NNSVR is the estimator that has the lowest RMSE and MAE for low level of noise $\mathrm{SNR} = 20$ and high level of noise

Table 4.1 – Results for the Support Vector Regression (SVR), Projected Support Regression (P-SVR), Non-Negative Support Vector Regression (NNSVR) and Non-Negative Least Squares (NNLS) for simulated data with $n = 500$ and $p = 50$. The mean (standard deviation) of the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE) over 50 repetitions are reported. Different noise distribution (Gaussian and laplacian) and different Signal to Noise Ratio (SNR) values were tested.

Distribution	Estimator	RMSE	MAE
Gaussian noise	SVR	2.238 (0.081)	29.288 (2.452)
	P-SVR	2.178 (0.087)	27.248 (2.545)
SNR = 20	NNSVR	2.174 (0.089)	27.224 (2.480)
($\sigma = 773.1$)	NNLS	**2.120 (0.114)**	**25.226 (2.699)**
Gaussian noise	SVR	2.732 (0.099)	44.764 (4.230)
	P-SVR	2.584 (0.154)	39.687 (5.963)
SNR = 10	NNSVR	**2.536 (0.105)**	**37.740 (3.866)**
($\sigma = 2444.9$)	NNLS	3.478 (0.208)	60.553 (7.923)
Laplacian noise	SVR	2.086 (0.109)	25.538 (3.181)
	P-SVR	2.039 (0.109)	23.978 (3.059)
SNR = 20	NNSVR	**2.035 (0.115)**	**23.827 (3.146)**
(b = 546.7)	NNLS	2.115 (0.103)	25.028 (2.571)
Laplacian noise	SVR	2.665 (0.148)	42.245 (5.777)
	P-SVR	2.526 (0.198)	37.745 (7.271)
SNR = 10	NNSVR	**2.480 (0.157)**	**35.786 (5.761)**
(b = 1728.8)	NNLS	3.463 (0.230)	63.940 (8.375)

$\mathrm{SNR} = 10$.

4.4.2 Regression onto the simplex

In this subsection, we study the performance of our proposed estimator on simplex constraints Simplex Support Vector Regression (SSVR). In this case, $A = -I_p$, $b = 0$, $\Gamma = \mathbb{1}$ and $d = 1$. The optimization problem that we seek to solve is:

$$
\begin{aligned}
\min_{\beta,\beta_0,\xi_i,\xi_i^*,\varepsilon} \quad & \frac{1}{2}||\beta||^2 + C(\nu\varepsilon + \frac{1}{n}\sum_{i=1}^{n}(\xi_i + \xi_i^*)) \qquad \text{(SSVR)} \\
\text{subject to} \quad & X_{i:}\beta + \beta_0 - y_i \leq \varepsilon + \xi_i \\
& y_i - X_{i:}\beta - \beta_0 \leq \varepsilon + \xi_i^* \\
& \xi_i, \xi_i^* \geq 0, \varepsilon \geq 0 \\
& \beta_j \geq 0, \forall j \in [p] \\
& \sum_{j=1}^{p} \beta_j = 1 \ .
\end{aligned}
$$

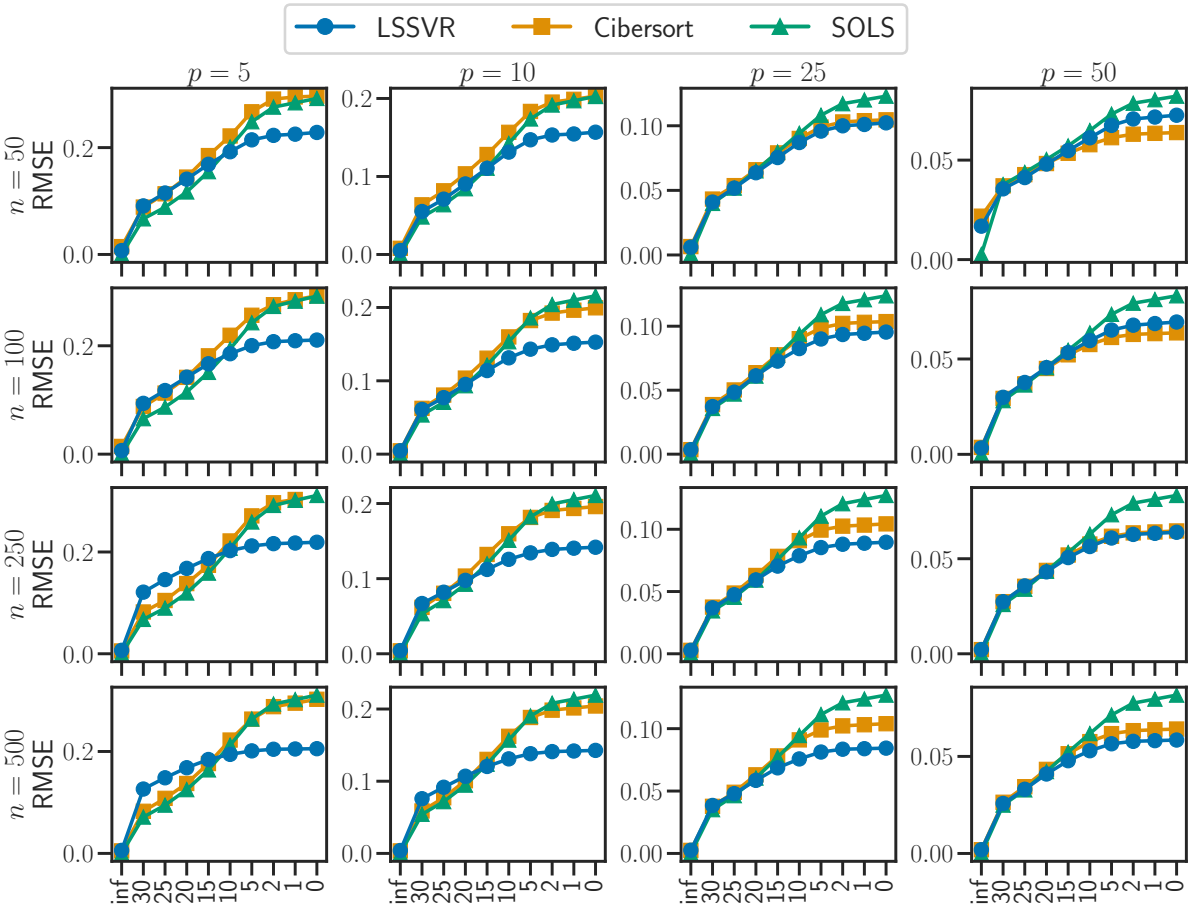

Figure 4.2 – The Root Mean Squared Error (RMSE) as a function of the Signal to Noise Ration (SNR) is presented. Different dimensions for the design matrix X and the response vector y were considered. n represents the number of rows of X and p the number of columns. For each plot, the blue line represents the RMSE for the Linear Simplex SVR (LSSVR) estimator, the green one the Simplex Ordinary Least Squares (SOLS) estimator and the orange on the Cibersort estimator. Each point of the curve is the mean RMSE of 50 repetitions. The noise in the data has a Gaussian distribution.

Synthetic data. We first tested on simulated data generated by the scikit-learn function `make_regression`. Once the design matrix X and the response vector y were generated using this function, we had access to the ground truth that we will write β^*. This function was not designed to generate data with a β^* that belongs to the simplex so we first projected β^* onto the simplex and then recomputed y multiplying the design matrix by the new projected ground truth vector. We added a centered Gaussian noise in the data with the standard deviation of the Gaussian was chosen such as the signal-to-noise ratio (SNR) was equal to a defined number, we used the following formula for a given SNR: $\sigma = \sqrt{\frac{\mathbb{V}\mathrm{ar}(y)}{10^{\mathrm{SNR}/10}}}$, where σ is the standard deviation used to simulate the noise in the data. The choice of the two hyperparameters C and ν was done using 5-folds cross validation

on a grid of possible pairs. The values of C were taken evenly spaced in the $log10$ base between $[-3, 3]$, we considered 10 different values. The values of ν were taken evenly spaced in the linear space between $[0.05, 1.0]$ and we also considered 10 possible values. We tested different size for the matrix $X \in \mathbb{R}^{n \times p}$ to check the potential effects of the dimensions on the quality of the estimation and we did 50 repetitions for each point of the curves. The measure that was used to compare the different estimators is the RMSE between the true β and the estimated $\hat{\beta}$.

We compared the RMSE of our estimator to the Simplex Ordinary Least Squares (SOLS) which is the result of the following optimization problem:

$$\begin{aligned} \min_{\beta} \quad & \frac{1}{2}||y - X\beta||^2 \\ \text{subject to} \quad & \beta_j \geq 0, \forall j \in [p] \\ & \sum_{j=1}^{p} \beta_i = 1 \ , \end{aligned} \tag{SOLS}$$

and to the estimator proposed in the biostatics literature that is called Cibersort. This estimator is simply the result of using the classical SVR and project the obtained estimator onto the simplex. The RMSE curves as a function of the SNR are presented in Figure Figure 4.2. We observe that the SSVR is generally the estimator with the lowest RMSE, this observation becomes clearer as the level of noise increases in the data. We notice that when there is a low level of noise and when n is not too large in comparison to p, the three compared estimator perform equally. However, there is a setting when n is large in comparison to p (in this experiment for $n = 250$ or 500 and $p = 5$) where the SSVR estimator has a higher RMSE than the Cibersort and SOLS estimator until a certain level of noise ($\mathrm{SNR} < 15$). Overall, this simulation shows that there is a significant improvement in the estimation performance of the SSVR mainly when there is noise in the data.

Real dataset. In the cancer research field, regression algorithms have been used to estimate the proportions of cell populations that are present inside a tumor. Indeed, a tumor is composed of different types of cells such as cancer cells, immune cells, healthy cells among others. Having access to the information of the proportions of these cells could be a key to understanding the interactions between the cells and the cancer treatment called immunotherapy (Couzin-Frankel, 2013). The modelization done is that the RNA extracted from the tumor is seen as a mixed signal composed of different pure signals coming from the different types of cells. This signal can be unmixed knowing the different pure RNA

signal of the different types of cells. In other words, y will be the RNA signal coming from a tumor and X will be the design matrix composed of the RNA signal from the isolated cells. The number of rows represent the number of genes that we have access to and the number of columns of X is the number of cell populations that we would like to quantify. The hypothesis is that there is a linear relationship between X and y. As said above, we want to estimate proportions which means that the estimator has to belong to the probability simplex $\mathscr{S} = \{x : x_i \geq 0\ ,\ \sum_i x_i = 1\}$.
Several estimators have been proposed in the biostatistics litterature most of them based on constrained least squares (Qiao et al., 2012; Gong et al., 2011; Abbas et al., 2009) but the gold standard is the estimator based on the SVR.

We compared the three same estimators on a real biological dataset where the real quantities of cells to obtain were known. The dataset can be found on the GEO website under the accession code GSE11103[2]. For this example $n = 584$ and $p = 4$ and we have access to 12 different samples that are our repetitions. Following the same idea than previous benchmark performed in this field of application, we increased the level of noise in the data and compared the RMSE of the different estimators. Gaussian and laplacian distributions of noise were added to the data. The choice of the two hyperparameters C and ν was done using 5-folds cross validation on a grid of possible pairs. The values of C were taken evenly spaced in the $\log_{10}$ base between $[-5, -3]$, we considered 10 different values. The interval of C is different than the simulated data because of the difference in the range value of the dataset. The values of ν were taken evenly spaced in the linear space between $[0.05, 1.0]$ and we also considered 10 possible values.

We see that when there is no noise in the data ($\mathrm{SNR} = \infty$) both Cibersort and SSVR estimator perform equally. The SOLS estimator already has a higher RMSE than the two others estimator probably due to the noise already present in the data. As the level of noise increases, the SSVR estimator remains the estimator with the lowest RMSE in both Gaussian and laplacian noise settings.

4.4.3 Isotonic regression

In this subsection, we will consider constraints that impose an order on the variables. This type of regression is usually called isotonic regression. Such constraints appear when prior knowledge are known on a certain order on the variables. This partial order on the variables can also be seen as an acyclic directed graph. More formally, we note $G = (V, E)$

[2]The dataset can be downloaded from the Gene Expression Omnibus website under the accession code GSE11103.

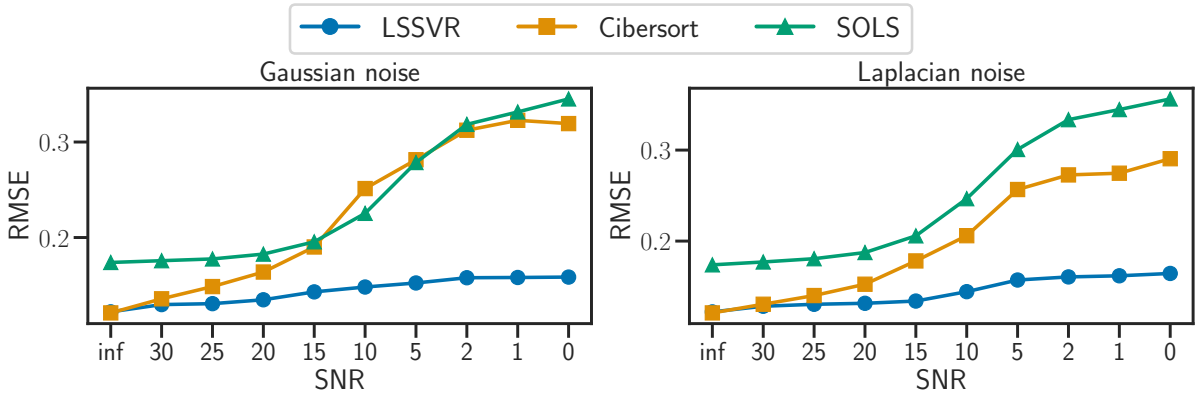

Figure 4.3 – The Root Mean Squared Error (RMSE) as a function of the Signal to Noise Ration (SNR) is presented on a real dataset where noise was manually added. Two different noise distribution were tested: Gaussian and laplacian. Each point of the curve is the mean RMSE of 12 different response vectors and we repeated the process four times for each level of noise. This would be equivalent to having 48 different repetitions.

a directed acyclic graph where V is the set of vertices and E is the set of nodes. On this graph, we define a partial order on the vertices. We will say for $u, v \in V$ that $u \leq v$ if and only if there is a path joining u and v in G. This type of constraints seems natural in different applications such as biology, medicine, weather forecast.

The most simple example of this type of constraints might be the monotonic regression where we force the variables to be in a increasing or decreasing order. It means that with our former notations that we would impose that $\beta_1 \leq \beta_2 \leq \ldots \leq \beta_p$ on the estimator. This type of constraints can be coded in a finite difference matrix (or more generally any incidence matrix of a graph)

$$A = \begin{bmatrix} 1 & -1 & 0 & \ldots & 0 \\ 0 & 1 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & 1 & -1 \end{bmatrix}$$

and $\Gamma = 0, b = 0, d = 0$ forming linear constraints as in the scope of this paper. The Isotonic Support Vector Regression (ISVR) optimization problem is written as follows:

$$\begin{aligned} \min_{\beta,\beta_0,\xi_i,\xi_i^*,\varepsilon} \quad & \frac{1}{2}||\beta||^2 + C(\nu\varepsilon + \frac{1}{n}\sum_{i=1}^{n}(\xi_i + \xi_i^*)) \\ \text{subject to} \quad & X_{i:}\beta + \beta_0 - y_i \leq \varepsilon + \xi_i \\ & y_i - X_{i:}\beta - \beta_0 \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \varepsilon \geq 0 \\ & \beta_1 \leq \beta_2 \leq \ldots \leq \beta_n \quad . \end{aligned} \tag{ISVR}$$

We compare our proposed ISVR estimator with the classical least squares isotonic regression (IR) (Barlow and Brunk, 1972) which is the solution of the following problem:

$$\begin{aligned} \min_{\beta\in\mathbb{R}^n} \quad & \frac{1}{2}||\beta - y||^2 \\ \text{subject to} \quad & \beta_1 \leq \beta_2 \leq \ldots \leq \beta_n \quad . \end{aligned} \tag{IR}$$

Table 4.2 – Results for the Isotonic Support Vector Regression (ISVR), and the Isotonic regression (IR) for simulated data with $p = 50$. The mean (standard deviation) of the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE) over 50 repetitions are reported. Different noise distribution (Gaussian and laplacian) and different Signal to Noise Ratio (SNR) values were tested.

Distribution	Estimator	RMSE	MAE
Gaussian noise	ISVR	0.212 (0.02)	0.254 (0.06)
SNR = 20	IR	**0.203 (0.02)**	**0.229 (0.04)**
Gaussian noise	ISVR	**0.284 (0.04)**	**0.446 (0.12)**
SNR = 10	IR	0.311 (0.04)	0.534 (0.12)
Laplacian noise	ISVR	**0.202 (0.03)**	0.223 (0.05)
SNR = 20	IR	0.203 (0.02)	**0.221 (0.04)**
Laplacian noise	ISVR	**0.276 (0.05)**	**0.414 (0.11)**
SNR = 10	IR	0.312 (0.05)	0.513 (0.13)

Synthetic dataset. We first generated data from a Gaussian distribution ($\mu = 0$, $\sigma = 1$) that we sorted and then added noise in the data following the same process as described in Section 4.4.2 with different SNR values (10 and 20). We tested Gaussian noise and laplacian noise. We compared the estimation quality of both methods using MAE and RMSE. In this experiment, the design matrix X is the identity matrix. We performed grid search selection via cross validation for the hyperparameters C and ν. C had 5 different possible

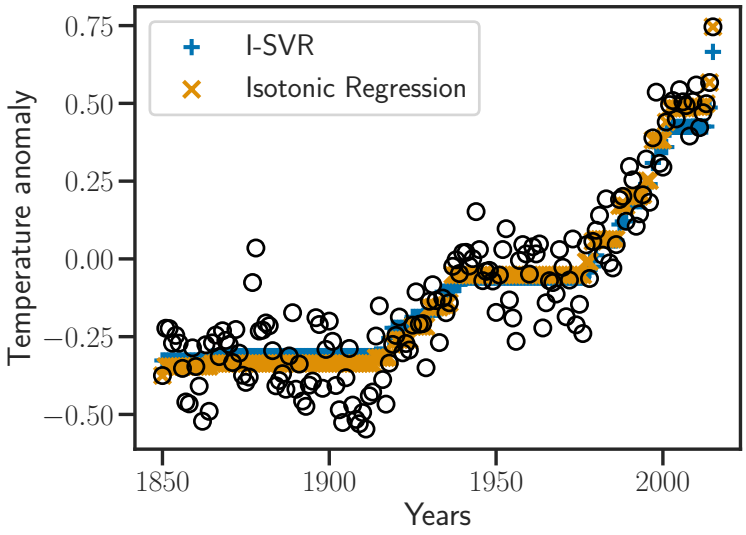

Figure 4.4 – Global warming dataset. Annual temperature anomalies relative to 1961-1990 average, with estimated trend using Isotonic Support Vector Regression (ISVR) and the classical Isotonic Regression (IR) estimator.

values taken on the logscale from 0 to 3, and ν had 5 different values taken between 0.05 and 1 on the linear scale. The dimension of the generated Gaussian vector was 50 and we did 50 repetitions. We present in Table Table 4.2 the results of the experiment, the value inside a cell is the mean RMSE or MAE over the 50 repetitions and the value between brackets is the standard deviation over the repetitions. Under a low level of Gaussian noise or laplacian noise, both methods are close in term of RMSE and MAE with a little advantage for the classical isotonic regression estimator. When the level of noise is important ($\mathrm{SNR} = 10$), our proposed ISVR has the lowest RMSE and MAE for the two noise distribution tested.

Real dataset. Isotonic types of constraints can be found in different applications such as biology, ranking and weather forecast for example. Focusing on global warming type of data, reserchers have studied the anomaly of the average temperature over a year in comparison to the years 1961-1990. These temperature anomalies have a monotenous trend and keep increasing since 1850 untill 2015. Isotonic regression estimator was used on this dataset[3] in Gaines et al. (2018) and we compared our proposed ISVR estimator for anomaly prediction. The hyperparameter for the ISVR were set manually for this simulation. Figure 4.4 shows the result for the two estimators. The classical isotonic regression

[3]This dataset can be downloaded from the https://cdiac.ess-dive.lbl.gov/trends/temp/jonescru/jones.htmlCarbon Dioxide Information Analysis Center at the Oak Ridge National Laboratory.

estimator perform better than our proposed estimator globally which is confirmed by the RMSE and MAE values of $\mathrm{RMSE}_{\mathrm{IR}} = 0.0067$ against $\mathrm{RMSE}_{\mathrm{ISVR}} = 0.047$ and $\mathrm{MAE}_{\mathrm{IR}} = 0.083$ against $\mathrm{MAE}_{\mathrm{ISVR}} = 0.29$. Howevever, we notice that in the portions where there is a significant change like between 1910-1940 and 1980-2005, the IR estimation looks like a step function whereas the ISVR estimation follows an increasing trend without these piecewise constant portions.

4.4.4 Performance of the GSMO versus SMO

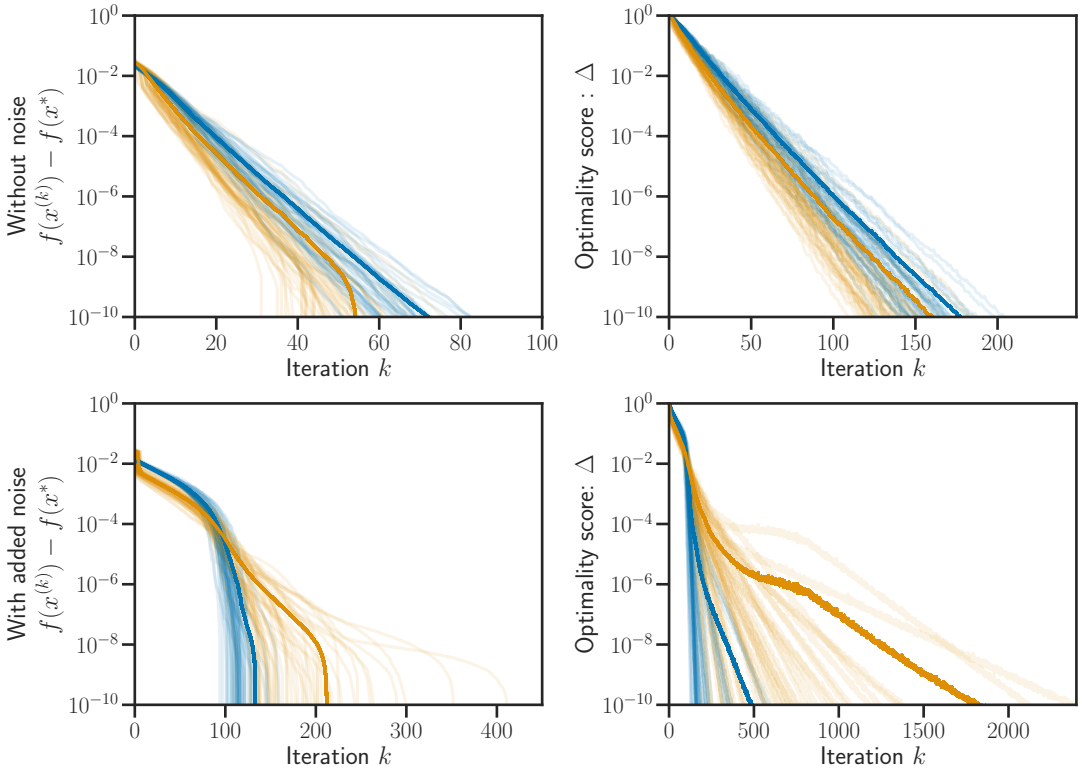

Figure 4.5 – Plots of 50 trajectories of the dual objective function value (first column) and the optimality score (second column) in function of the number of iterations for the classical SMO algorithm in blue and the proposed generalized SMO in red. Two settings were used, one without noise and another one with additive Gaussian noise.

We compared the efficiency of the SMO algorithm to solve the classical SVR optimization problem and the SSVR optimization problem in Figure 4.5. To do so, we used the same data simulation process described earlier in this subsection and set the number of rows of the matrix X, $n = 200$ and the number of columns $p = 25$. Two different settings were considered here, one without any noise in the data and another one with Gaussian noise added such that the SNR would be equal to 30. The transparent trajectories represent the decrease of the objective function or the optimality score Δ for the classical SMO in blue

and for the generalized SMO in red for the 50 repetitions considered. The average trajectory is represented in dense color. The first row of figures are the results for the noiseless setting and the second row for the setting with noise. When there is not noise in the data, the generalized SMO decreases faster than the classical SMO. It is important to remind that the true vector here belongs to the simplex so without any noise it is not surprising that our proposed algorithm goes faster than the classical SMO. However, when noise is adding to the data, it takes more iterations for the generalized SMO to find the solution of the optimization problem. Figure 4.5 illustrates the convergence towards a minimum the GSMO algorithm as stated in Theorem 4.1.

4.5 Proof of convergence.

We begin the proof of Theorem 4.1 by giving several preliminary results that will be helpful for giving the final proof. The first result gives a bound for controlling the distance of the primal iterates generated by the algorithm and the solution of Problem (LSVR-P).

Lemma 4.3. *For any generalized SMO iterative, we have:*

$$\beta^k = -\sum_{i=1}^{n}(\alpha_i^k - (\alpha_i^*)^k)X_{i:} - A^\top\gamma^k + \Gamma^\top\mu^k \quad .$$

Then let β^{opt} be a solution of Problem (LSVR-P) and θ^{opt} a solution of Problem (LSVR-D). It holds that $\frac{1}{2}||\beta^k - \beta^{opt}|| \leq f(\theta^k) - f(\theta^{opt})$.

Proof. A first observation is that the relationship between the primal optimization problem and the dual leads to this equality

$$f(\theta^k) = \frac{1}{2}||\beta^k||^2 + l^\top\theta^k \quad . \tag{4.6}$$

Replacing β^k by $-\sum_{i=1}^{n}(\alpha_i^k - (\alpha_i^*)^k)X_{i:} - A^\top\gamma^k + \Gamma^\top\mu^k$ leads to Equation (4.6). We have already seen that there is strong duality between both problems so the dual gap is zero at the solutions. Thus it means that for any primal optimal solution $(\beta^{\text{opt}}, \beta_0^{\text{opt}}, \xi^{\text{opt}}, \xi^{\text{opt}}, \epsilon^{\text{opt}})$

and any dual solution θ^{opt}, it holds true that

$$\begin{aligned}\frac{1}{2}||\beta^{\text{opt}}||^2 + C(\nu\epsilon^{\text{opt}} + \frac{1}{n}\sum_{i=1}^{n}\xi_i^{\text{opt}} + \xi_i^{\text{opt}}) &= -f(\theta^{\text{opt}}) \\ &= -\frac{1}{2}||\beta^{\text{opt}}||^2 - l^\top\theta^{\text{opt}} \ .\end{aligned}$$

Using the equation link between primal and dual yields to

$$\begin{aligned}\langle\beta, \beta^{\text{opt}}\rangle &= \langle -\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)X_{i:} - A^\top\gamma + \Gamma^\top\mu, \beta^{\text{opt}}\rangle \\ &= -\langle A^\top\gamma, \beta^{\text{opt}}\rangle - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\langle X_{i:}, \beta^{\text{opt}}\rangle + \langle\Gamma^\top\mu, \beta^{\text{opt}}\rangle \ .\end{aligned}$$

Since $\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$, we have that

$$\begin{aligned}\langle\beta, \beta^{\text{opt}}\rangle &= -\langle A^\top\gamma, \beta^{\text{opt}}\rangle - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\langle X_{i:}, \beta^{\text{opt}}\rangle + \langle\Gamma^\top\mu, \beta^{\text{opt}}\rangle - \beta_0^{\text{opt}}\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) \\ &= -\langle A^\top\gamma, \beta^{\text{opt}}\rangle - \sum_{i=1}^{n}\alpha_i(\langle X_{i:}, \beta^{\text{opt}}\rangle + \beta_0^{\text{opt}}) + \sum_{i=1}^{n}\alpha_i^*(\langle X_{i:}, \beta^{\text{opt}}\rangle + \beta_0^{\text{opt}}) \\ &+ \langle\Gamma^\top\mu, \beta^{\text{opt}}\rangle \ .\end{aligned}$$

Moreover, using the constraints of Problem (LSVR-P) and the fact that $\alpha \geq 0$ and $\alpha^* \geq 0$ it holds that:

$$\begin{aligned}\langle\beta, \beta^{\text{opt}}\rangle &\geq -\langle A^\top\gamma, \beta^{\text{opt}}\rangle + \sum_{i=1}^{n}\alpha_i(-y_i - \epsilon^{\text{opt}} - \xi_i^{\text{opt}}) + \sum_{i=1}^{n}\alpha_i^*(y_i - \epsilon^{\text{opt}} - (\xi_i^*)^{\text{opt}}) \\ &+ \langle\Gamma^\top\mu, \beta^{\text{opt}}\rangle \\ &= -\langle A^\top\gamma, \beta^{\text{opt}}\rangle - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)y_i - \epsilon^{\text{opt}}C\nu - \sum_{i=1}^{n}\alpha_i\xi_i^{\text{opt}} + \alpha_i^*(\xi_i^*)^{\text{opt}} + \langle\Gamma^\top\mu, \beta^{\text{opt}}\rangle \ .\end{aligned}$$

Finally we have

$$\begin{aligned}\frac{1}{2}||\beta^k - \beta^{\text{opt}}||^2 &= \frac{1}{2}||\beta^k||^2 - \langle \beta^k, \beta^{\text{opt}} \rangle + \frac{1}{2}||\beta^{\text{opt}}||^2 \\ &\leq \frac{1}{2}||\beta^k||^2 + \langle A^\top \gamma^k, \beta^{\text{opt}} \rangle + \epsilon^{\text{opt}} C\nu + \sum_{i=1}^{n} (\alpha_i^k - (\alpha_i^*)^k) y_i + \sum_{i=1}^{n} \alpha_i^k \xi_i^{\text{opt}} \\ &\quad + (\alpha_i^*)^k (\xi_i^*)^{\text{opt}} - \langle \Gamma^\top \mu^k, \beta^{\text{opt}} \rangle + \frac{1}{2}||\beta^{\text{opt}}||^2 \ .\end{aligned}$$

Since β^{opt} satisfies the constraints of the primal optimization problem, it holds that

$$\langle \Gamma^\top \mu, \beta^{\text{opt}} \rangle = \mu^\top d \ , \tag{4.7}$$

and since $\gamma \geq 0$ we have $\langle A^\top \gamma, \beta^{\text{opt}} \rangle \leq \gamma^\top b$, thus

$$\begin{aligned}\frac{1}{2}||\beta^k - \beta^{\text{opt}}||^2 \leq \frac{1}{2}||\beta^k||^2 + \gamma^\top b + \sum_{i=1}^{n} (\alpha_i^k - (\alpha_i^*)^k) y_i + \epsilon^{\text{opt}} C\nu + \sum_{i=1}^{n} \alpha_i^k \xi_i^{\text{opt}} \\ + (\alpha_i^*)^k (\xi_i^*)^{\text{opt}} - \mu^\top d + \frac{1}{2}||\beta^{\text{opt}}||^2 \ .\end{aligned}$$

The linear term that we wrote l in the objective function of Problem (LSVR-D) defines $l^\top \theta = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) X_{i:} + \gamma^\top b - \mu^\top d$ which in combination with Equation (4.6) gives

$$\frac{1}{2}||\beta^k - \beta^{\text{opt}}||^2 \leq \frac{1}{2} f(\theta^k) + \epsilon^{\text{opt}} C\nu + \sum_{i=1}^{n} \alpha_i^k \xi_i^{\text{opt}} + (\alpha_i^*)^k (\xi_i^*)^{\text{opt}} + \frac{1}{2}||\beta^{\text{opt}}||^2 \ .$$

Each α_i^k, $(\alpha_i^*)^k$ is bounded by $\frac{C}{n}$ which yields to

$$\frac{1}{2}||\beta^k - \beta^{\text{opt}}||^2 \leq f(\theta^k) + \epsilon^{\text{opt}} C\nu + \frac{C}{n} \sum_{i=1}^{n} \xi_i^{\text{opt}} + (\xi_i^*)^{\text{opt}} + \frac{1}{2}||\beta^{\text{opt}}||^2.$$

We recognize the objective function of the primal optimization problem and using that there is no dual gap at the optimum it follows that

$$\epsilon^{\text{opt}} C\nu + \frac{C}{n} \sum_{i=1}^{n} \xi_i^{\text{opt}} + (\xi_i^*)^{\text{opt}} + \frac{1}{2}||\beta^{\text{opt}}||^2 = -f(\theta^{\text{opt}}) \ ,$$

which finishes the proof. □

Before the next statement, we need to give a definition that we will use in the next proofs.

Definition 4.6. Let (i,j) ($i \in I_{\text{low}}$ and $j \in I_{\text{up}}$) be the most violating pair of variables in the block α, (i^*,j^*) ($i^* \in I^*_{\text{low}}$ and $j^* \in I^*_{\text{up}}$) for the block α^*. Let s_1 be the index of the most violating variable in the block γ and s_2 in the block μ. We will call "optimality score" at iteration k the quantity $\Delta^k = \max(\Delta^k_1, \Delta^k_2, \Delta^k_3, \Delta^k_4)$, where $\Delta^k_1 = \max(\nabla_{\alpha_j} f(\theta^k) - \nabla_{\alpha_i} f(\theta^k), 0)$, $\Delta^k_2 = \max(\nabla_{\alpha_{j^*}} f(\theta^k) - \nabla_{\alpha_{i^*}} f(\theta^k), 0)$, $\Delta^k_3 = \max(-\nabla_{\gamma_{s_1}} f(\theta^k), 0)$ and $\Delta^k_4 = \max(|\nabla_{\mu_{s_2}} f(\theta^k)|, 0)$.

The next result states that the sequence $\{f(\theta^k)\}$ is a decreasing sequence. This result already states the convergence to a certain value $\bar{f}$ because we know that the sequence is bounded by the existing global minimum of the function since f is a semidefinite positive quadratic function.

Lemma 4.4. *The sequence generated by the Generalized SMO algorithm $\{f(\theta^k)\}$ is a decreasing sequence. This sequence converges to a value $\bar{f}$.*

Proof. We first prove that $f(\theta^k) - f(\theta^{k+1}) \geq 0$ when minimization takes place in the block α. Let (i,j) be the indices of the variables selected to be optimized and let $u \in \mathbb{R}^{2n+k_1+k_2}$ be the vector with only zeros except at the i^{th} coordinate where it is equal to t^* as defined in Section 4.5 and at the j^{th} coordinate where it is equal to $-t^*$. We will also define $t_q = \frac{-(\nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k))}{Q_{ii}+Q_{jj}-2Q_{ij}}$, the unconstrained minimum for the update in α block. Let us compute

$$\begin{aligned}
f(\theta^k) - f(\theta^{k+1}) &= \frac{1}{2}(\theta^k)^\top \bar{Q}\theta^k + l^\top\theta^k - \frac{1}{2}(\theta^{k+1})^\top \bar{Q}\theta^{k+1} + l^\top\theta^{k+1} \\
&= \frac{1}{2}(\theta^k)^\top \bar{Q}\theta^k + l^\top\theta^k - \frac{1}{2}(\theta^k + U)^\top \bar{Q}(\theta^k + u) + l^\top(\theta^k + u) \\
&= -\frac{1}{2}u^\top \bar{Q}u - u^\top(Q\theta^k + l) \\
&= -\frac{1}{2}u^\top \bar{Q}u - u^\top(\nabla f(\theta^k)) \\
&= -\frac{(t^*)^2}{2}(Q_{ii} + Q_{jj} - 2Q_{ij}) - t^*(\nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k)) \ .
\end{aligned}$$

We first study the case when there is no clipping which means that $t^* = t_q$

1. No clipping. Replacing t^* by its expression leads to the following result:

$$\begin{aligned} f(\theta^k) - f(\theta^{k+1}) &= \frac{(\Delta_1^k)^2}{2(Q_{ii} + Q_{jj} - 2Q_{ij})} \\ &= \frac{(\Delta_1^k)^2}{2||X_{i:} - X_{j:}||^2} \geq 0 \ . \end{aligned}$$

2. Clipping takes place because $t_q \leq t^* = \max(-\alpha_i, \alpha_j - \frac{C}{n})$
We notice that $t_q \leq \max(-\alpha_i, \alpha_j - \frac{C}{n}) \leq 0$ which implies that $i \in I_{\text{low}}$ and $j \in I_{\text{up}}$. In that case $\Delta_1^k = \nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k)$. Replacing t_q by its expression leads to

$$\begin{aligned} -(\nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k)) &\leq t^*(Q_{ii} + Q_{jj} - 2Q_{ij}) \\ \frac{\Delta_1^k t^*}{2} &\leq \frac{-(t^*)^2}{2}(Q_{ii} + Q_{jj} - 2Q_{ij}) \\ \frac{\Delta_1^k t^*}{2} - t^* \Delta_1^k &\leq \frac{-(t^*)^2}{2}(Q_{ii} + Q_{jj} - 2Q_{ij}) - t^*(\nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k)) \\ -\frac{1}{2}\Delta_1^k t^* &\leq \frac{-(t^*)^2}{2}(Q_{ii} + Q_{jj} - 2Q_{ij}) - t^*(\nabla_{\alpha_i} f(\theta^k) - \nabla_{\alpha_j} f(\theta^k)) \ . \end{aligned}$$

Thus we have that if $t^* = -\alpha_i$, $f(\theta^k) - f(\theta^{k+1}) \geq \frac{1}{2}\Delta_1^k \alpha_i \geq 0$ and that if $t^* = \alpha_j - \frac{C}{n}$, $f(\theta^k) - f(\theta^{k+1}) \geq \frac{1}{2}\Delta_1^k(\frac{C}{n} - \alpha_j) \geq 0$.

3. Clipping takes place because $t_q \geq t^* = \min(\frac{C}{n} - \alpha_i, \alpha_j)$.
This time $t_q \geq \min(\frac{C}{n} - \alpha_i, \alpha_j) \geq 0$ which also implies that $i \in I_{\text{up}}$ and $j \in I_{\text{low}}$ and that $\Delta_1^k = \nabla_{\alpha_j} f(\theta^k) - \nabla_{\alpha_i} f(\theta^k)$. The only difference here is that multiplying by $-t^*$ will imply a change in the inequality.

$$\begin{aligned} -(\nabla_i f(\theta^k) - \nabla_j f(\theta^k)) &\geq t^*(Q_{ii} + Q_{jj} - 2Q_{ij}) \\ \frac{-\Delta_1^k t^*}{2} &\leq \frac{-(t^*)^2}{2}(Q_{ii} + Q_{jj} - 2Q_{ij}) \\ \frac{-\Delta_1^k t^*}{2} + t^* \Delta_1^k &\leq \frac{-(t^*)^2}{2}(Q_{ii} + Q_{jj} - 2Q_{ij}) - t^*(\nabla_i f(\theta^k) - \nabla_j f(\theta^k)) \\ \frac{1}{2}\Delta_1^k t^* &\leq \frac{-(t^*)^2}{2}(Q_{ii} + Q_{jj} - 2Q_{ij}) - t^*(\nabla_i f(\theta^k) - \nabla_j f(\theta^k)) \ . \end{aligned}$$

Thus we have that if $t^* = \frac{C}{n} - \alpha_i$

$$f(\theta^k) - f(\theta^{k+1}) \geq \frac{1}{2}\Delta_1^k(\frac{C}{n} - \alpha_i) \geq 0 \ ,$$

and if $t^* = \alpha_j$,

$$f(\theta^k) - f(\theta^{k+1}) \geq \frac{1}{2}\Delta_1^k \alpha_j \geq 0 \ .$$

To prove that $f(\theta^k) - f(\theta^{k+1}) \geq 0$ when the update takes place in the block γ and μ we first need to observe that when only one variable is updated between iteration k and $k+1$ it follows that

$$f(\theta^k) - f(\theta^{k+1}) = \frac{1}{2}\bar{Q}_{ii}(\theta_i^k - \theta_i^{k+1})^2 \ .$$

Therefore, we now prove the result for the block γ. If the update is not a clipped update and i is the index of the updated variable, it holds that $\gamma_i^k - \gamma_i^{k+1} = \frac{\nabla_{\gamma_i} f(\theta^k)}{(AA^\top)_{ii}}$, which gives the following bound

$$f(\theta^k) - f(\theta^{k+1}) = \frac{1}{2(AA^\top)_{ii}}(\nabla_{\gamma_i} f(\theta^k))^2 \geq 0 \ . \tag{4.8}$$

Moreover, if a clipped update takes place in this block, we know that it happens when $0 \leq \gamma_i^k \leq \frac{\nabla_{\gamma_i} f(\theta^k)}{(AA^\top)_{ii}}$. It yields to the following bound

$$f(\theta^k) - f(\theta^{k+1}) = \frac{1}{2}(AA^\top)_{ii}(\gamma_i^k)^2 \geq 0 \ .$$

The result for the block μ is obtained using the same arguments except that there is no clipped updates. □

Lemma 4.5. *There exists a subsequence $\{\theta^{k_j}\}$ of iterations generated by the generalized SMO where clipping does not take place.*

Proof. Let us suppose the contrary, which means that there exists an iteration K such that for all $k \geq K$ we only perform clipped updates. The number of variables N_B^k that belong to the boundary of its constraints (0 or $\frac{C}{n}$ for the blocks α or α^* and 0 for the block γ) is non-decreasing for all $k \geq K$ and it is bounded thus it must converge to another integer N^*.

This convergence implies that there exists k^* such that for all $k \geq k^*$, $N_B^k = N^*$ since N_B^k and N^* are integers. This observation allows us to conclude that for all $k \geq k^*$ clipped updates only take place in the blocks α or α^* since the updates in the block γ are made on only one variable and that the number of clipped variables has reached its maximum value. An update in the block γ would strictly increase the number of clipped variables which is not possible for all $k \geq k^*$ or the update would not change the value of θ and we showed before that this situation is not possible (Proposition 4.3).

For all $k \geq k^*$, we have that updates in the block α (resp. α^*) have this necessary scheme: α_i^k or α_j^k is equal to 0 or $\frac{C}{n}$ thus after the update, one of them will leave the boundary and the other one goes to it in order to keep the number of clipped variables equals to N^*. The different possibilities are then the following:

- if $\alpha_i^k = 0$ and $0 < \alpha_j^k \leq \frac{C}{l}$ the only possible update following the Definition 4.5 is

$$\begin{aligned} \alpha_i^{k+1} &= \alpha_i^k + \alpha_j^k = \alpha_j^k \\ \alpha_j^{k+1} &= \alpha_j^k - \alpha_j^k = 0 \ . \end{aligned}$$

- if $\alpha_j^k = \frac{C}{l}$ and $0 \leq \alpha_i^k < \frac{C}{l}$ the only possible update following the Definition 4.5 is

$$\begin{aligned} \alpha_i^{k+1} &= \alpha_i^k + (\frac{C}{l} - \alpha_i^k) = \frac{C}{l} \\ \alpha_j^{k+1} &= \alpha_j^k - (\frac{C}{l} - \alpha_i^k) = \alpha_i^k \ . \end{aligned}$$

It stays true for the block α^* and the discussion is similar. It is clear that from the description of the updates made above that there is only a finite number of ways to shuffle the values which means that there exists $k_1, k_2 \geq k^*$ such as $\theta^{k_1} = \theta^{k_2}$ and with $k_1 < k_2$. Therefore $f(\theta^{k_1}) = f(\theta^{k_2})$ which contradicts the decrease of the sequence $f(\theta^k)$ (Lemma 4.4). □

Lemma 4.6. *Let $\{\theta^{k_j}\}$ be a subsequence generated by the Generalized SMO algorithm where clipping does not take place. We then have that $\Delta^{k_j} \to 0$.*

Proof. We have that

$$f(\theta^{k_j}) - f(\theta^{k_j+1}) \geq \frac{(\nabla_{\alpha_i} f(\theta^{k_j}) - \nabla_{\alpha_j} f(\theta^{k_j}))^2}{2D^2} = \frac{(\Delta^{k_j})^2}{2D^2} \ ,$$

where $D = \max_{p,q} ||X_{p:} - X_{q:}||$ when the update happens in the blocks α or α^*. When it happens in the block γ with no clipping we have the following inequality

$$f(\theta^{k_j}) - f(\theta^{k_j+1}) \geq \frac{(\nabla_{\gamma_i} f(\theta^{k_j}))^2}{2} = \frac{(-\Delta^{k_j})^2}{2} = \frac{(\Delta^{k_j})^2}{2} \ .$$

When the update takes place in the block μ, we have that

$$f(\theta^{k_j}) - f(\theta^{k_j+1}) \geq \frac{(\nabla_{\mu_i} f(\theta^{k_j}))^2}{2}) = \frac{(\Delta^{k_j})^2}{2} \ .$$

We then define a sequence

$$u^{k_j} = \begin{cases} \frac{1}{2D^2}(\Delta^{k_j})^2 & \text{update in the blocks } \alpha \text{ or } \alpha^*. \\ \frac{1}{2}(\Delta^{k_j})^2 & \text{update in the blocks } \gamma \text{ or } \mu. \end{cases}$$

The sequence $\{u^{k_j}\} \to 0$ because of the bound given above and the fact that $f(\theta^{k_j}) - f(\theta^{k_j+1}) \to 0$ too (Lemma 4.4). This implies that $\Delta^{k_j} \to 0$ as well. □

A consequence of the lemma above is that $\Delta_1^{k_j} \to 0$, $\Delta_2^{k_j} \to 0$, $\Delta_3^{k_j} \to 0$ and $\Delta_4^{k_j} \to 0$ because Δ^{k_j} is defined as the maximum of those four positive values.

Lemma 4.7. *Let $\{\theta^{k_j}\}$ be a subsequence generated by the generalized SMO algorithm where clipping does not take place. This subsequence is bounded.*

Proof. To prove the statement, we will show that $||\theta^{k_j} - \theta^{\text{opt}}||^2$ is bounded where θ^{opt} belongs to the set of solution of Problem (LSVR-D). Since each α_i and α_i^* is belongs to $[0, \frac{C}{n}]$, we have that

$$\begin{aligned} ||\theta^{k_j+1} - \theta^{\text{opt}}||^2 &= ||\alpha^{k_j+1} - \alpha^{\text{opt}}||^2 + ||(\alpha^*)^{k_j+1} - (\alpha^*)^{\text{opt}}||^2 \\ &\quad + ||\gamma^{k_j+1} - \gamma^{\text{opt}}||^2 + ||\mu^{k_j+1} - \mu^{\text{opt}}||^2 \\ &\leq \frac{2C^2}{n} + ||\gamma^{k_j+1} - \gamma^{\text{opt}}||^2 + ||\mu^{k_j+1} - \mu^{\text{opt}}||^2 \ . \end{aligned}$$

We will work on the bound for the quantity $||\mu^{k_j+1} - \mu^{\text{opt}}||^2$ first. If the update happens in the block μ at coordinate μ_j, we have the following

$$\begin{aligned} ||\mu^{k_j+1} - \mu^{\text{opt}}||^2 &= ||\mu^{k_j} - e_j \frac{\nabla_{\mu_j} f(\theta^{k_j})}{(\Gamma\Gamma^\top)_{jj}} - \mu^{\text{opt}}||^2 \\ &= ||\mu^{k_j} - \mu^{\text{opt}}||^2 - 2\langle \mu^{k_j} - \mu^{\text{opt}}, e_j \frac{\nabla_{\mu_j} f(\theta^{k_j})}{(\Gamma\Gamma^\top)_{jj}} \rangle + ||e_j \frac{\nabla_{\mu_j} f(\theta^{k_j})}{(\Gamma\Gamma^\top)_{jj}}||^2 \\ &= ||\mu^{k_j} - \mu^{\text{opt}}||^2 + \frac{\nabla_{\mu_j} f(\theta^{k_j})^2}{(\Gamma\Gamma^\top)_{jj}^2} - 2\frac{\nabla_{\mu_j} f(\theta^{k_j})}{(\Gamma\Gamma^\top)_{jj}}(\mu_j^{k_j} - \mu_j^{\text{opt}}) \ . \end{aligned}$$

We then have that

$$\begin{aligned}
-2\frac{\nabla_{\mu_j} f(\theta^{k_j})}{(\Gamma\Gamma^\top)_{jj}}(\mu_j^{k_j} - \mu_j^{\text{opt}}) &= 2(\mu_j^{k_j+1} - \mu_j^{k_j})(\mu_j^{k_j} - \mu_j^{\text{opt}}) \\
&= 2\langle \mu^{k_j+1} - \mu^{k_j}, \mu^{k_j} - \mu^{\text{opt}} \rangle \\
&\leq 2||\mu^{k_j+1} - \mu^{k_j}|| \cdot ||\mu^{k_j} - \mu^{\text{opt}}|| \\
&\leq 2\frac{|\nabla_{\mu_j} f(\theta^{k_j})|}{(\Gamma\Gamma^\top)_{jj}}||\mu^{k_j} - \mu^{\text{opt}}|| \\
&\leq 2\frac{\Delta_4^{k_j}}{(\Gamma\Gamma^\top)_{jj}}||\mu^{k_j} - \mu^{\text{opt}}|| \ .
\end{aligned}$$

From Lemma 4.6, we have that $\Delta_4^{k_j} \to 0$ then it can be bounded by a constant M_0. We know from Equation (4.8) that

$$\frac{\nabla_{\mu_j} f(\theta^{k_j})^2}{(\Gamma\Gamma^\top)_{jj}^2} = \frac{2}{(\Gamma\Gamma^\top)_{jj}}(f(\theta^{k_j}) - f(\theta^{k_j+1})).$$

From Lemma 4.4, we know that $f(\theta^{k_j}) - f(\theta^{k_j+1}) \to 0$ then it can be bounded by a constant M_1. Overall we have that

$$||\mu^{k_j+1} - \mu^{\text{opt}}||^2 \leq ||\mu^{k_j} - \mu^{\text{opt}}||^2 + 2\frac{M_0}{(\Gamma\Gamma^\top)_{jj}}||\mu^{k_j} - \mu^{\text{opt}}|| + \frac{2}{(\Gamma\Gamma^\top)_{jj}}M_1 \ .$$

By recursion we have

$$\begin{aligned}
||\mu^{k_j+1} - \mu^{\text{opt}}||^2 &\leq ||\mu^0 - \mu^{\text{opt}}||^2 + 2\frac{M_0}{(\Gamma\Gamma^\top)_{jj}}||\mu^0 - \mu^{\text{opt}}|| + \frac{2}{(\Gamma\Gamma^\top)_{jj}}M_1 \\
&< \infty \ .
\end{aligned}$$

Since there is no clipped update on the subsequence $\{\theta^{k_j}\}$, the proof for the block γ is similar which proves that $||\theta^{k_j} - \theta^{\text{opt}}||$ is bounded. □

Lemma 4.8. *Let $\{\theta^{k_j}\}$ be a subsequence generated by the generalized SMO algorithm where clipping does not take place. There exists a sub-subsequence that converges to $\bar{\theta}$, with $\bar{\theta}$ being a solution of Problem (LSVR-D).*

Proof. From Lemma 4.7, we have that $\{\theta^{k_j}\}$ is a bounded sequence, it means that we can extract a converging subsequence that we will write $\{\theta^{k_j}\}$ not to complicate the notations. Since $\mathcal{F}$ is closed, $\bar{\theta}$ meets the constraints of the dual optimization problem and belongs

to $\mathcal{F}$. We now want to prove that it belongs to the set of solution of Problem (LSVR-D) by showing that $\bar{\Delta}_1(\bar{\theta}) \leq 0$, $\bar{\Delta}_2(\bar{\theta}) \leq 0$, $\bar{\Delta}_3(\bar{\theta}) \leq 0$ and $\bar{\Delta}_4(\bar{\theta}) \leq 0$. Let us make two observations that will be used for the following proof. The first one comes from the continuity of the gradient which implies that for all ϵ there exists K_1 such that for all $k_j \geq K_1$, $|\nabla_i f(\theta^{k_j}) - \nabla_i f(\bar{\theta})| < \epsilon$ for all i. The second observation is that it is possible too chose an ϵ small enough such that there exists K_2 such that for all $k_j \geq K_2$: if $\bar{\alpha}_i > 0$, we have $\alpha_i^{k_j} > 0$ and if $\bar{\alpha}_i < \frac{C}{n}$ we have $\alpha_i^{k_j} < \frac{C}{n}$. In other words, we say that all the indices in the set $I_{\text{low}}(\bar{\alpha})$(resp. I_{up}) are also in $I_{\text{low}}(\alpha_{k_j})$(resp. I_{up}). The same argument holds for indices in the block α^*.

Let us assume that $\bar{\Delta}_1 > 0$, it means that there exists at least one violating pair of variables that we will note $(\bar{i}, \bar{j})$ at $\bar{\theta}$. From the discussion above, we know that $\bar{i} \in I_{\text{low}}$ for all $k_j \geq K_2$ and that $\bar{j} \in I_{\text{up}}$ for all $k_j \geq K_2$. We then have that for all $\epsilon > 0$, there exists K_1 such as for all $k_j \geq \max(K_1, K_2)$,

$$\begin{aligned}
\Delta_1^{k_j} &= \min_{i \in I_{\text{up}}} \nabla_i f(\theta^{k_j}) - \max_{i \in I_{\text{low}}} \nabla_i f(\theta^{k_j}) \\
&\geq \nabla_{\bar{i}} f(\theta^{k_j}) - \nabla_{\bar{j}} f(\theta^{k_j}) \\
&\geq (\nabla_{\bar{i}} f(\bar{\theta}) - \epsilon) - (\nabla_{\bar{j}} f(\bar{\theta}) + \epsilon) \\
&= \bar{\Delta}_1 - 2\epsilon \ .
\end{aligned}$$

We choose $\epsilon = \frac{\bar{\Delta}_1}{2} - \epsilon'$ where $0 < \epsilon' < \frac{\bar{\Delta}_1}{2}$ which leads to

$$\Delta_1^{k_j} \geq \bar{\Delta}_1 - 2\epsilon' = 2\epsilon' > 0 \ .$$

This inequality is true for all $k_j \geq \max(K_1, K_2)$ which contradicts the fact that $\Delta_1^{k_j} \to 0$. The proof is similar to show that $\bar{\Delta}_2 \leq 0$.

Let us now suppose that $\bar{\Delta}_3 > 0$ it means that there exists an index $\bar{i}$ such that $\nabla_{\gamma_i} f(\bar{\theta}) < 0$. For all $\epsilon > 0$, there exists K_1, K_2 such as for all $k_j > \max(K_1, K_2)$

$$\begin{aligned}
\Delta_3^{k_j} &= - \min_{i \in \{1, \ldots, k_1\}} \nabla_{\gamma_i} f(\theta^{k_j}) \\
&\geq -\nabla_{\gamma_{\bar{i}}} f(\theta^{k_j}) \\
&\geq -(\nabla_{\gamma_{\bar{i}}} f(\bar{\theta}) + \epsilon) \\
&= \bar{\Delta}_3 - \epsilon \ .
\end{aligned}$$

We choose $\epsilon = \bar{\Delta}_3 - \epsilon'$ where $0 < \epsilon' < \bar{\Delta}_3$ which leads to

$$\Delta_3^{k_j} \geq \bar{\Delta}_3 - \epsilon = \epsilon' > 0 \ .$$

This inequality is true for all $k_j \geq \max(K_1, K_2)$ which contradicts the fact that $\Delta_3^{k_j} \to 0$.

Finally Let us assume that $\Delta_4^{k_j} > 0$, it means that $|\nabla_{\mu_i} f(\theta^{k_j})| \neq 0$. Using the continuity of the gradient we write that for all $\epsilon > 0$ there exists K_1 such that for all $k_j \geq K_1$ we have $|\nabla_{\mu_i} f(\theta^{k_j}) - \nabla_{\mu_i} f(\bar{\theta}| < \epsilon$. Using triangle inequality we get that

$$\Big| |\nabla_{\mu_i} f(\theta^{k_j})| - |\nabla_{\mu_i} f(\bar{\theta})| \Big| \leq |\nabla_{\mu_i} f(\theta^{k_j}) - \nabla_{\mu_i} f(\bar{\theta})| < \epsilon \ .$$

Thus $-\epsilon \leq |\nabla_{\mu_i} f(\theta^{k_j})| - |\nabla_{\mu_i} f(\bar{\theta})| \leq \epsilon$, which means that

$$|\nabla_{\mu_i} f(\bar{\theta})| - \epsilon \leq |\nabla_{\mu_i} f(\theta^{k_j})| \ .$$

Then we have the following:

$$\begin{aligned} \Delta_4^{k_j} &= \max_{i \in \{1,\dots,k_2\}} |\nabla_{\mu_i} f(\theta^{k_j})| \\ &\geq \nabla_{\mu_{\bar{i}}} f(\theta^{k_j})| \\ &\geq |\nabla_{\sigma} f(\bar{\theta})| - \epsilon \\ &= \bar{\Delta}_4 - \epsilon \ . \end{aligned}$$

We choose $\epsilon = \bar{\Delta}_4 - \epsilon'$ where $0 < \epsilon' < \bar{\Delta}_4$ which leads to

$$\Delta_4^{k_j} \geq \bar{\Delta}_4 - \epsilon = \epsilon' > 0 \ .$$

This inequality is true for all $k_j \geq \max(K_1, K_2)$ which contradicts the fact that $\Delta_4^{k_j} \to 0$. □

Proof. We are now able to give the proof of Theorem 4.1. From Lemma 4.3, we have that $\frac{1}{2}||\beta^k - \beta^{\text{opt}}|| \leq f(\theta^k) - f(\theta^{\text{opt}})$. Moreover, from Lemma 4.6 we know that there is a subsequence $\{\theta^{k_j}\}$ generated by the Generalized SMO algorithm where clipping does not take place and that converges to $\bar{\theta}$, with $\bar{\theta}$ a solution of Problem (LSVR-D). The continuity of the objective function f allows us to say that $f(\theta^{k_j}) \to f(\bar{\theta})$. From Lemma 4.4, we know that $\{f(\theta^{k_j})\}$ is decreasing and bounded so the monotone convergence theorem implies that the whole sequence $f(\theta^k) \to f(\bar{\theta})$ and it follows that $\frac{1}{2}||\beta^k - \beta^{\text{opt}}|| \to 0$ and finally that $\beta^k \to \beta^{\text{opt}}$. □

Part II

Hyperparameters selection for non-smooth convex models

5 Introduction to hyperparameter optimization

Contents

In this chapter, we introduce the second part of this thesis dedicated to hyperparameter selection or hyperparameter optimization. We present the common tools used to select hyperparameters in machine learning such as grid-search or bayesian optimization. We introduce the fact that choosing hyperparameters given a performance criterion can be cast as a bilevel optimization. We present the literature around first order optimization to solve the bilevel problem and the advantages of using these methods.

Table 5.1 – Examples of non-smooth lower problems.

lower problem	$f(\beta)$	$g_j(\beta_j, \lambda)$	$e^{\lambda_{\max}}$
Lasso	$\frac{1}{2n}\|y - X\beta\|^2$	$e^{\lambda}\|\beta_j\|$	$\frac{1}{n}\|X^\top y\|_\infty$
elastic net	$\frac{1}{2n}\|y - X\beta\|^2$	$e^{\lambda_1}\|\beta_j\| + \frac{1}{2}e^{\lambda_2}\beta_j^2$	$\frac{1}{n}\|X^\top y\|_\infty$
sparse logistic regr.	$\frac{1}{n}\sum_{i=1}^n \ln(1 + e^{-y_i X_{i:}\beta})$	$e^{\lambda}\|\beta_j\|$	$\frac{1}{2n}\|X^\top y\|_\infty$
dual SVM	$\frac{1}{2}\|(y \odot X)^\top \beta\|^2 - \sum_{j=1}^p \beta_j$	$\iota_{[0, e^{\lambda}]}(\beta_j)$	$-$

5.1 Hyperparameter selection

Almost all models in machine learning require at least one hyperparameter, the tuning of which drastically affects accuracy. This is the case for a large number of popular machine learning estimators, where the regularization hyperparameter controls the trade-off between a data fidelity term and a regularization term. Popular estimators, including Ridge regression (Hoerl and Kennard, 1970), Lasso (Tibshirani, 1996; Chen et al., 1998), elastic net (Zou and Hastie, 2005), (sparse) logistic regression (McCullagh and Nelder, 1989; Hastie and Tibshirani, 1990), support-vector machine (Boser et al., 1992; Platt, 1999) can all be cast as an optimization problem (see Table 5.1):

$$\hat{\beta}^{(\lambda)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min}\, \Phi(\beta, \lambda) \triangleq f(\beta) + \underbrace{\sum_{j=1}^p g_j(\beta_j, \lambda)}_{\triangleq g(\beta, \lambda)} \;, \tag{5.1}$$

with a design matrix $X \in \mathbb{R}^{n \times p}$, a function $f : \mathbb{R}^n \to \mathbb{R}$ with Lipschitz gradient, proper closed convex (possibly non-smooth) functions $g_j(\cdot, \lambda)$, and a regularization hyperparameter $\lambda \in \mathbb{R}^r$. For a fixed λ, the question of solving efficiently Problem (5.1) has been largely explored. If the functions g_j are smooth, one can resort to powerful solvers such as L-BFGS (Liu and Nocedal, 1989), SVRG (Johnson and Zhang, 2013; Zhang et al., 2013), or SAGA (Defazio et al., 2014). When the functions g_j are non-smooth Problem (5.1) can be tackled efficiently using working set methods (Fan and Lv, 2008; Tibshirani et al., 2012) combined with (block) coordinate descent (Tseng and Yun, 2009; Wright, 2015; Shi et al., 2016), see Massias et al. (2020) for an overview. The question of *model selection*, *i.e.*, how to select the hyperparameter $\lambda \in \mathbb{R}^r$ (potentially multidimensional), is a more open one, especially for non-smooth optimization problems, and when the dimension of the regularization hyperparameter λ is large.

Selecting λ in Problem (5.1) can be cast as a *statistical* problem. For example, for the Lasso, and under strong statistical assumptions on the design matrix X, a wide literature has

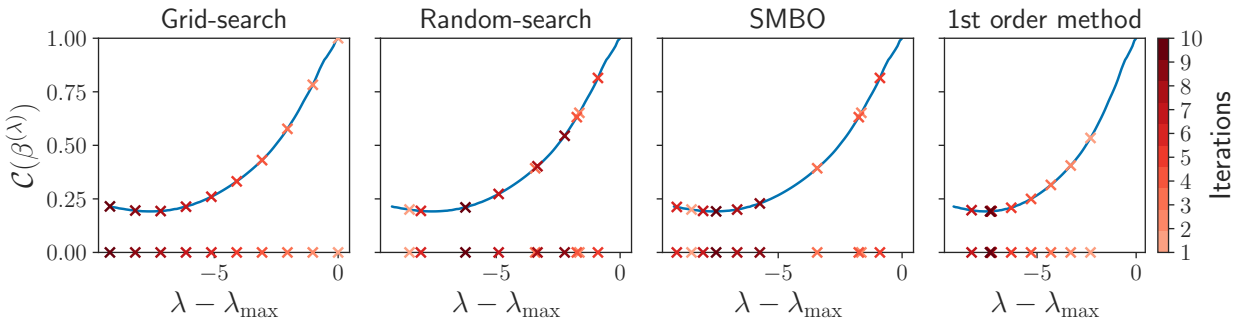

Figure 5.1 – **Lasso CV, *real-sim* dataset.** 5-fold cross-validation error $\mathcal{C}(\beta^{(\lambda)})$ as a function of λ for multiple hyperparameter optimization methods for the Lasso $\arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + e^{\lambda}\|\beta\|_1$. Crosses represents 10 first errors evaluation by each method.

been devoted to model selection, leading to a closed-form formula for the regularization parameter λ (Lounici, 2008; Bickel et al., 2009; Belloni et al., 2011). Unfortunately, this formula relies on quantities which are unknown in practice, and Lasso-users still have to resort to other techniques to select the hyperparameter λ. A popular approach for hyperparameter selection is to cast such a tuning as *hyperparameter optimization* (Kohavi and John, 1995; Hutter et al., 2015; Feurer and Hutter, 2019), *i.e.*, selecting the hyperparameter λ such that the solution of the optimization problem also minimizes a given criterion $\mathcal{C} : \mathbb{R}^p \to \mathbb{R}$. This criterion is typically chosen to promote good generalization error, *e.g.*, the held-out loss (Devroye and Wagner, 1979), the cross-validation loss (CV, Stone and Ramer 1965, see Arlot and Celisse 2010 for a survey), or to reduce model complexity, *e.g.*, AIC (Akaike, 1974), BIC (Schwarz, 1978) or SURE (Stein, 1981) criteria (see Table 5.2 for some common examples). More formally the hyperparameter optimization problem can be cast as a bilevel optimization problem (Colson et al., 2007)

$$\begin{aligned} &\arg\min_{\lambda \in \mathbb{R}^r} \left\{ \mathcal{L}(\lambda) \triangleq \mathcal{C}\left(\hat{\beta}^{(\lambda)}\right) \right\} \\ &s.t.\ \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \Phi(\beta, \lambda) \ . \end{aligned} \tag{5.2}$$

Criterion	Problem type	Criterion $\mathcal{C}(\beta)$
Held-out mean squared error	Regression	$\frac{1}{n}\|y^{\text{val}} - X^{\text{val}}\beta\|^2$
Stein unbiased risk estimate (SURE)[1]	Regression	$\|y - X\beta\|^2 - n\sigma^2 + 2\sigma^2 \text{dof}(\beta)$
Held-out logistic loss	Classification	$\frac{1}{n}\sum_{i=1}^n \log(1 + e^{-y_i^{\text{val}} X_{i:}^{\text{val}}\beta})$
Held-out smoothed Hinge loss[2]	Classification	$\frac{1}{n}\sum_{i=1}^n \ell(y_i^{\text{val}}, X_{i:}^{\text{val}}\beta)$

Table 5.2 – Examples of outer criteria used for hyperparameter selection

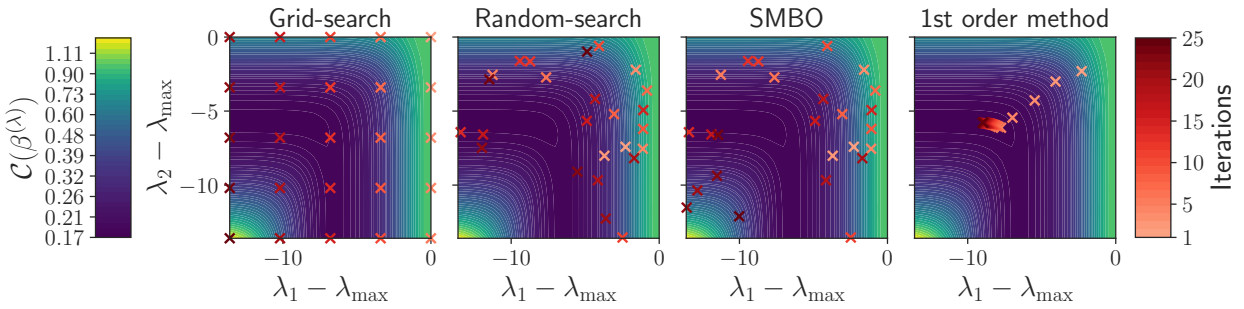

Figure 5.2 – **Elastic net CV, *rcv1* dataset.** Level sets of a 5-fold cross-validation error $\mathcal{C}(\beta^{(\lambda)})$ as a function of λ_1 and λ_2 for multiple hyperparameter optimization methods for the elastic net $\arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + e^{\lambda_1}\|\beta\|_1 + e^{\lambda_2}\|\beta\|^2/2$. The crosses represent 25 first errors evaluation by each method.

Popular approaches to address Problem (5.2) include zero-order optimization (gradient-free) techniques such as *grid-search*, *random-search* (Rastrigin, 1963; Bergstra and Bengio, 2012; Bergstra et al., 2013), or *Bayesian optimization* (Brochu et al., 2010; Snoek et al., 2012). Grid-search consists in evaluating the outer function $\mathcal{L}$ on a user-chosen grid of hyperparameters, solving one inner optimization Problem (5.1) for each λ in the grid (see Figures 5.1 and 5.2). For each solution of the lower-problem $\hat{\beta}^{(\lambda)}$, the criterion of success $\mathcal{C}(\hat{\beta}^{(\lambda)})$ is evaluated, and the selected model is the one achieving the lowest value. This can be interpreted as a naive discretization of Problem (5.2). Other examples of gradient-free methods include *Bayesian optimization* (Brochu et al., 2010; Snoek et al., 2012) such as Sequential Model-Based Optimization (SMBO). The core idea is to model the objective function $\mathcal{L}$ via a Gaussian process. Iteratively, *good* candidates will be tested by the algorithm as potential minimizers of $\mathcal{L}$. These candidates are chosen to maximize a given function called the expected improvement as described in Bergstra et al. (2011). These zero-order methods share a common drawback: they scale exponentially with the dimension of the search space (Nesterov, 2004, Sec. 1.1.2).

On the other hand, when the hyperparameter space is continuous and the (regularization path) function $\lambda \mapsto \hat{\beta}^{(\lambda)}$ is well-defined and (almost everywhere) differentiable, first-order optimization methods are well suited to solve the bilevel optimization Problem (5.2). Using the chain rule, the gradient of $\mathcal{L}$ *w.r.t.* λ, also referred to as the *hypergradient*, evaluates to

$$\nabla_\lambda \mathcal{L}(\lambda) = \hat{\mathcal{J}}_{(\lambda)}^\top \nabla \mathcal{C}(\hat{\beta}^{(\lambda)}) \ , \tag{5.3}$$

[1]For a linear model $y = X\beta + \varepsilon$, with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, the degree of freedom (dof, Efron 1986) is defined as $\text{dof}(\beta) = \sum_{i=1}^n \text{cov}(y_i, (X\beta)_i)/\sigma^2$.

[2]The smoothed Hinge loss is given by $\ell(x) = \frac{1}{2} - x$ if $x \leq 0$, $\frac{1}{2}(1-x)^2$ if $0 \leq x \leq 1$, 0 else .

with $\hat{\mathcal{J}}_{(\lambda)} \in \mathbb{R}^{p \times r}$ the *Jacobian* of the function $\lambda \mapsto \hat{\beta}^{(\lambda)}$. The main challenge of applying first-order methods to solve Problem (5.2) is evaluating the hypergradient in Equation (5.3). There are three main algorithms to compute the hypergradient $\nabla_\lambda \mathcal{L}(\lambda)$: *implicit differentiation* (Larsen et al., 1996; Bengio, 2000), *automatic differentiation* using the *backward mode* (Linnainmaa, 1970) or *forward mode* (Wengert, 1964; Deledalle et al., 2014; Franceschi et al., 2017). As illustrated in Figures 5.1 and 5.2, once the hypergradient in Equation (5.3) has been computed, one can solve Problem (5.2) by using a first-order optimization scheme, for instance, gradient descent, with a step size $\rho > 0$: $\lambda^{(t+1)} = \lambda^{(t)} - \rho \nabla_\lambda \mathcal{L}(\lambda^{(t)})$. Note however that the function $\mathcal{L}$ may be non-convex and convergence towards a global minimum is not guaranteed.

5.2 Bilevel optimization with smooth lower problems

The main challenge to evaluate the hypergradient $\nabla_\lambda \mathcal{L}(\lambda)$ is the computation of the Jacobian $\mathcal{J}_{(\lambda)}$. We first focus on the case where $\Phi(\cdot, \lambda)$ is convex and smooth for any fixed λ.

5.2.1 Hypergradient computation

Implicit differentiation. In this paragraph, we recall how the *implicit differentiation*[3] formula of the gradient $\nabla_\lambda \mathcal{L}(\lambda)$ is obtained. This formula is established for smooth lower optimization problem, we will provide a generalization to non-smooth optimization problems in Chapter 6.

Theorem 5.1 (Bengio 2000). *Let $\hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \Phi(\beta, \lambda)$ be a solution of Problem (5.1). Assume that for all $\lambda > 0$, $\Phi(\cdot, \lambda)$ is a convex smooth function, $\nabla^2_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda) \succ 0$, and that for all $\beta \in \mathbb{R}^p$, $\Phi(\beta, \cdot)$ is differentiable over $(0, +\infty)$. Then the hypergradient $\nabla_\lambda \mathcal{L}(\lambda)$ has a closed-form expression:*

$$\underbrace{\nabla_\lambda \mathcal{L}(\lambda)}_{\in \mathbb{R}^r} = \underbrace{-\nabla^2_{\beta,\lambda} \Phi(\hat{\beta}^{(\lambda)}, \lambda)}_{\in \mathbb{R}^{r \times p}} \underbrace{\left(\nabla^2_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda) \right)^{-1}}_{\in \mathbb{R}^{p \times p}} \underbrace{\nabla \mathcal{C}(\hat{\beta}^{(\lambda)})}_{\in \mathbb{R}^p} . \tag{5.4}$$

We recall the proof for completeness and pedagogical purpose: it provides insights for the formula in the non-smooth case (Chapter 6).

Proof. With a convex smooth function $\beta \mapsto \Phi(\beta, \lambda)$ the first order optimality condition

[3] Note that *implicit* refers to the implicit function theorem, but leads to an *explicit* formula of the gradient.

writes:

$$\nabla_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda) = 0 \ , \tag{5.5}$$

for any $\hat{\beta}^{(\lambda)}$ solution of the lower problem. Moreover, if $\lambda \mapsto \nabla_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda)$ is also smooth, differentiating Equation (5.5) *w.r.t.* λ leads to:

$$\nabla^2_{\beta,\lambda} \Phi(\hat{\beta}^{(\lambda)}, \lambda) + \hat{\mathcal{J}}^\top_{(\lambda)} \nabla^2_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda) = 0 \ . \tag{5.6}$$

The Jacobian $\hat{\mathcal{J}}^\top_{(\lambda)}$ can then be computed solving the following linear system:

$$\hat{\mathcal{J}}^\top_{(\lambda)} = -\nabla^2_{\beta,\lambda} \Phi(\hat{\beta}^{(\lambda)}, \lambda) \underbrace{\Big(\nabla^2_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda) \Big)^{-1}}_{\in \mathbb{R}^{p \times p}} \ . \tag{5.7}$$

Plugging Equation (5.7) into Equation (5.3) leads to:

$$\nabla_\lambda \mathcal{L}(\lambda) = -\nabla^2_{\beta,\lambda} \Phi(\hat{\beta}^{(\lambda)}, \lambda) \Big(\nabla^2_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda) \Big)^{-1} \nabla \mathcal{C}(\hat{\beta}^{(\lambda)}) \ .$$

□

The computation of the gradient via implicit differentiation (Equation (5.4)) involves the resolution of a $p \times p$ linear system (Bengio, 2000, Sec. 4). The linear system (potentially large) can be solved using different algorithms such as conjugate gradient (Hestenes and Stiefel 1952, as in Pedregosa 2016) or fixed point methods (Lions and Mercier 1979; Tseng and Yun 2009, as in Grazzi et al. 2020). Implicit differentiation has been used for model selection of multiple estimators with smooth regularization term: kernel-based models (Chapelle et al., 2002; Seeger, 2008), weighted Ridge estimator (with one λ_j per feature, Foo et al. 2008), neural networks (Lorraine et al., 2019) or meta-learning (Franceschi et al., 2018; Rajeswaran et al., 2019).

Note that Problem (5.1) is typically solved using iterative solvers providing an approximation of the exact solution $\hat{\beta}$. Practically, a high precision solution is expensive to compute, and solvers usually return only solutions with low precision. Thus, Equation (5.5) is not exactly satisfied and the linear system to solve Equation (5.4) does not lead to the exact gradient $\nabla_\lambda \mathcal{L}(\lambda)$. However Pedregosa (2016) showed that one can resort to *approximated gradients* when the lower problem is smooth, justifying that implicit differentiation can be applied using an approximation of $\hat{\beta}$. Interestingly, this approximation scheme was shown to yield significant practical speedups when solving Problem (5.2), while preserv-

ing theoretical properties of convergence toward the optimum.

Iterative differentiation. Iterative differentiation computes the gradient $\nabla_\lambda \mathcal{L}(\lambda)$ capitalizing on the iterative algorithms used to solve Problem (5.1). Iterative differentiation can be applied using the forward mode (Wengert, 1964) or the backward mode (Linnainmaa, 1970). Both techniques rely on the chain rule, the gradient being decomposed as a large product of matrices, either computed in a forward way, or a backward way. Note that forward and backward differentiation are algorithm dependent: in this part we present iterative differentiation for proximal gradient descent (PGD, Lions and Mercier 1979; Combettes and Wajs 2005).

The most popular method in automatic differentiation is the backward iterative differentiation, which is the cornerstone of modern optimization for deep learning (Goodfellow et al., 2016, Chap. 8). Iterative differentiation in the field of hyperparameter optimization can be traced back to Domke (2012), who derived a backward differentiation algorithm for gradient descent, heavy ball and L-BFGS algorithms applied to smooth loss functions. It first computes the solution $\hat{\beta}$ of the optimization Problem (5.1) using an iterative solver. It requires to store the intermediate iterates along the computation in order to compute the hypergradient in a backward way. Forward iterative differentiation computes jointly the coefficients $\hat{\beta}$ along with the gradient $\nabla_\lambda \mathcal{L}(\lambda)$. It is memory efficient (iterates are not stored) but computationally expensive when the number of parameters is large; see Baydin et al. (2018) for a survey.

5.2.2 Resolution of the bilevel optimization

From a practical point of view, once the hypergradient has been computed, one does not solve easily Problem (5.2): in addition to being potentially non-convex, the smoothness constant (if it exists) of the function $\lambda \mapsto \mathcal{L}(\lambda)$ may be hard to estimate. Solving Problem (5.2) with gradient methods thus relies on optimization algorithms which do not require the knowledge of the Lipschitz constant: L-BFGS (as in Deledalle et al. 2014), heuristic line search (as in Pedregosa 2016) or gradient descent with heuristic step sizes (as in Frecon et al. 2018; Ji et al. 2020). The optimization challenge is the same as in deep learning (Goodfellow et al., 2016, Chap. 8): optimizing a non-convex function with only access to the gradient.

Solving Problem (5.2) using gradient-based methods is also very challenging from a theoretical point of view, and results in the literature are quite scarce. Kunisch and Pock (2013) studied the convergence of a semi-Newton algorithm to solve Problem (5.2) where both

the outer and lower problems are smooth. Franceschi et al. (2018) gave similar results with weaker assumptions in a work that aimed at unifying hyperparameter optimization and meta-learning, written as bilevel optimization. They require the lower problem to have a unique solution for all $\lambda > 0$ but do not have the assumption on the second derivative of Φ. Recent results (Ghadimi and Wang, 2018; Ji et al., 2020) provided quantitative convergence toward a global solution of Problem (5.2), but under global joint convexity assumption, and require the knowledge of the Lipschitz constant.

5.3 Automatic differentiation

We now give a quick overview of automatic differentiation and its two modes: forward and backward which is a crucial aspect of modern deep-learning to quickly and accurately compute gradients. For a more complete survey on these techniques, we refer to Baydin et al. (2018).

Automatic differentiation aims at automatically compute the derivatives of numerical functions that can be written as computer programs. Numerical computations are composition of several elementary operations for which it is easy to compute the derivatives. These operations include the binary arithmetic operations, the unary sign switch and outside functions for which the derivative can be computed explicitly such as the exponential, the logarithm, the trigonometric functions, the polynomial ones or eventually proximal operators (see Chapters 6 and 7). Using the chain rule, we can obtain the derivative of the overall composition using the elementary derivatives. One of the main advantage of automatic differentiation is that it allows the differentiation of functions that can involves loops, recursion, etc. for which it would be impossible to compute the derivatives explicitly.

In our case, automatic differentiation will be used to compute the Jacobian of the function $\lambda \mapsto \hat{\beta}^{(\lambda)}$ where $\hat{\beta}^{(\lambda)}$ is a solution of Problem (5.1). We consider that a model depending of hyperparameters can be seen as a function of these hyperparameters; for each value of the hyperparameters corresponds one (or several) solution of the optimization problem. We discuss in more details in Chapter 6 the potential multi-valued mapping (when the optimization problem has several possible solutions for a fixed λ).

In the non-smooth case, Problem (5.1) can be solved using iterative algorithms involving the proximal operator of the non-smooth function g such as the proximal gradient descent or the proximal coordinate descent described in Chapter 2. The iterative algorithms stops in practice after a finite number of iterations and can be seen as a finite number of

composition of operations for which we can compute the derivatives.

Consider for example that the proximal gradient descent stops after N_{iter} iterations then the solution given by the algorithm denoted by $\beta^{(N_{\text{iter}})}$ can be written as:

$$\beta^{(N_{\text{iter}})} = h \circ \ldots \circ h(\beta^{(0)}) \ , \tag{5.8}$$

where $\beta^{(0)}$ is an initial points and $h(x) \triangleq \text{prox}_{\gamma g}\left(x - \gamma \nabla f(x)\right)$. The idea is then to use the chain rule to compute the derivative of the composition *i.e.,* the derivative of the approximated solution of Problem (5.1) by the iterative algorithm.

Before diving in the details on how to compute the derivatives of proximal operators and how to use it for automatic hyperparameter selection, we describe the two modes of automatic differentiation. To explain these concepts, we use the notation of Griewank and Walther (2008) called the three parts notation. We consider a function $f : \mathbb{R}^n \to \mathbb{R}^p$ that is constructed via intermediate variables v_i such that:

- $v_{i-n} = x_i, i = 1, \ldots, n$ are the input variables
- $v_i, i = 1, \ldots, l$ are the intermediate variables
- $y_{p-i} = v_{l-i}, i = p-1, \ldots, 0$ are the output variables.

5.3.1 Forward differentiation

Forward differentiation can be traced back to Wengert (1964) and has gain a lot of interest in the past decade due to improvements in computer programming languages and in the computer hardware. To present the idea of forward differentiation, we use a simple example which will hopefully help the readers to apprehend this computer programming method.

Let us first consider the function $f : \mathbb{R}^3 \to \mathbb{R}^2$ with

$$f(x_1, x_2, x_3) = \begin{pmatrix} x_1 x_2 - \exp(x_3) \\ x_1 x_2 + x_3^2 \end{pmatrix} \ .$$

Computing the Jacobian of this function at a point (x_1, x_2, x_3) can be done manually and we easily obtain that

$$\mathcal{J} f(x_1, x_2, x_3) = \begin{pmatrix} x_2 & x_1 & -\exp(x_3) \\ x_2 & x_1 & 2x_3 \end{pmatrix} \ . \tag{5.9}$$

Algorithm 6 FUNCTION f

input : $x_1, x_2, x_3 \in \mathbb{R}$

def $f(x_1, x_2, x_3)$: // $(1, 2, 0)$

$v_{-2} = x_1$	// $= 1$
$v_{-1} = x_2$	// $= 2$
$v_0 = x_3$	// $= 0$
$v_1 = v_{-2}v_{-1}$	// $= 2$
$v_2 = \exp(v_0)$	// $= 1$
$v_3 = v_0^2$	// $= 0$
$v_4 = v_1 - v_2$	// $= 1$
$v_5 = v_1 + v_3$	// $= 2$
return $(y_1, y_2) = (v_4, v_5)$	// $(1, 2)$

Algorithm 7 FORWARD DERIVATIVE OF f

input : $\dot{x}_1, \dot{x}_2, \dot{x}_3 \in \mathbb{R}$

def $df(\dot{x}_1, \dot{x}_2, \dot{x}_3)$: // $(1, 0, 0)$

$\dot{v}_{-2} = \dot{x}_1$	// $= 1$
$\dot{v}_{-1} = \dot{x}_2$	// $= 0$
$\dot{v}_0 = \dot{x}_3$	// $= 0$
$\dot{v}_1 = v_{-2}\dot{v}_{-1} + v_{-1}\dot{v}_{-2}$	// $= 2$
$\dot{v}_2 = \dot{v}_0 \exp(v_0)$	// $= 0$
$\dot{v}_3 = 2\dot{v}_0 v_0$	// $= 0$
$\dot{v}_4 = \dot{v}_1 - \dot{v}_2$	// $= 2$
$\dot{v}_5 = \dot{v}_1 + \dot{v}_3$	// $= 2$
return $(\dot{y}_1, \dot{y}_2) = (\dot{v}_4, \dot{v}_5)$	// $(2, 2)$

Our goal is to illustrate how the Jacobian of f (or directional derivatives) can be computed using the forward differentiation. The function f can be seen as a computer program given in Algorithm 6. This computer program can be broken in elementary operations involving intermediate variables denoted by v_i. As an example, we could put as an input of the function the values $(1, 2, 0)$ and the computer program would return the values $(1, 2)$ corresponding to f evaluated at the point $(1, 2, 0)$. To compute the derivative of f *w.r.t.* x_1 at the point $(1, 2, 0)$, we first associate to each intermediate v_i its derivative with respect to x_1 *i.e.*,

$$\dot{v}_i = \frac{\partial v_i}{\partial x_1} \ .$$

Applying the chain rule to each step in the computer program, leads to the computation of the directional derivative *w.r.t.* x_1 in Algorithm 7. To obtain the directional derivative with respect to x_1, the input values for the variables $\dot{x}_1, \dot{x}_2, \dot{x}_3$ has to be the vector $e_1 = (1, 0, 0)$. We want to draw the attention of the readers here on the fact that the function df cannot be dissociated from the function that computes the value of the functions. The forward mode is computed along side with the function itself, we dissociated the two algorithms for readability and to ease the explanation.

Coming back to our example, entering the values $(1, 0, 0)$ in df gives the expected results, the directional derivative of f at the point $(1, 2, 0)$ is $(2, 2)$ as it can be read on the first column of the Jacobian matrix Equation (5.9). Thus, the complete Jacobian can be evaluated in our example in three runs of the function df, initializing the function with the three vectors of the canonical base of $\mathbb{R}^3$. As we will see in Part II of this manuscript, one often needs to compute the product between a Jacobian and a vector. The forward mode allows

Algorithm 8 FUNCTION f

input : $x_1, x_2, x_3 \in \mathbb{R}$

def $f(x_1, x_2, x_3)$: // $(1, 2, 0)$

$v_{-2} = x_1$ // $= 1$

$v_{-1} = x_2$ // $= 2$

$v_0 = x_3$ // $= 0$

$v_1 = v_{-2}v_{-1}$ // $= 2$

$v_2 = \exp(v_0)$ // $= 1$

$v_3 = v_0^2$ // $= 0$

$v_4 = v_1 - v_2$ // $= 1$

$v_5 = v_1 + v_3$ // $= 2$

return $(y_1, y_2) = (v_4, v_5)$ // $(1, 2)$

Algorithm 9 BACKWARD DERIVATIVE OF f

input : $\bar{y}_1, \bar{y}_2 \in \mathbb{R}$

def $df(\bar{y}_1, \bar{y}_2)$: // $(1, 0)$

$\bar{v}_5 = \bar{y}_1$ // $= 1$

$\bar{v}_4 = \bar{y}_1$ // $= 0$

$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$ // $= 0$

$\bar{v}_1 = \bar{v}_5 \frac{\partial v_5}{\partial v_1}$ // $= 0$

$\bar{v}_1 = \bar{v}_1 + \bar{v}_4 \frac{\partial v_4}{\partial v_1}$ // $= 1$

$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$ // $= -1$

$\bar{v}_0 = \bar{v}_3 \frac{\partial v_3}{\partial v_0}$ // $= 0$

$\bar{v}_0 = \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0}$ // $= -1$

$\bar{v}_{-1} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-1)}}$ // $= 1$

$\bar{v}_{-2} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-2}}$ // $= 2$

$(\bar{x_1}, \bar{x_2}, \bar{x_3}) = (\bar{v}_{-2}, \bar{v}_{-1}, \bar{v}_0)$ // $(2, 1, -1)$

return $(\bar{x_1}, \bar{x_2}, \bar{x_3})$ // $(2, 1, -1)$

us to compute this product in only one forward pass. By initializing the function df with a vector $r \in \mathbb{R}^3$, the output of the forward derivative program is the product $\mathcal{J}_f v$. It gives a matrix-free way to compute this product which is very efficient in practice.

The forward differentiation only requires one call to the differentiated algorithm to compute the p partial derivatives of a function $f : \mathbb{R} \to \mathbb{R}^p$. On the other hand, computing the gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ requires n calls to the differentiated algorithm. Generally, for functions $f : \mathbb{R}^n \to \mathbb{R}^p$, with $n \gg p$, a different technique is often preferred. However we want to address here that in the particular cases described in Part II, the forward differentiation can benefit from the cheap updates described for coordinate descent in Equation (2.5) leading to lower computation costs which can sometimes make it more suitable even when a gradient is computed.

5.3.2 Backward differentiation

Backward differentiation, also known as backpropagation, is an essential element of modern deep-learning but can be traced back to Linnainmaa (1970). It corresponds to using the chain rule in reverse mode from the output of a computer program to the beginning. This time the computation of the value of f and its derivative are dissociated. We first run the function f in Algorithm 6 and store all the intermediate variables v_i. Once it is done,

we can associate to each variable v_i the following:

$$\bar{v_i} = \frac{\partial y_j}{\partial v_i} \ ,$$

where y_j represents the output of the computer program computing the value of f. This value $\bar{v_i}$ represents the sensitivity of the output y_j with respect to changes in v_i.

Coming back to our example, we can obtain the derivatives of y_1 *w.r.t.* x_1, x_2 and x_3 in one pass *i.e.,* the first row of the Jacobian Equation (5.9). Algorithm 9 presents the backward differentiation to compute the derivatives $\frac{\partial y_1}{\partial x_1}$, $\frac{\partial y_1}{\partial x_2}$ and $\frac{\partial y_1}{\partial x_3}$ at the point $(1, 2, 0)$. To do so, one has to initialize the values $\bar{y}_1$ and $\bar{y}_2$ at $(1, 0)$. To explain the computation of the intermediate variables $\bar{v}_i$, we take the variable v_0 as an example, we see in Algorithm 8 that the only way it can affect y_1 is through v_2 and v_3, so the change in y_1 by v_0 is given by:

$$\frac{\partial y}{\partial v_0} = \frac{\partial y}{\partial v_2}\frac{\partial v_2}{\partial v_0} + \frac{\partial y}{\partial v_3}\frac{\partial v_3}{\partial v_0} \ ,$$

or equivalently

$$\bar{v}_0 = \bar{v}_2\frac{\partial v_2}{\partial v_0} + \bar{v}_3\frac{\partial v_3}{\partial v_0} \ .$$

We see in Algorithm 9 that it is computed in two steps $\bar{v}_0 = \bar{v}_3\frac{\partial v_3}{\partial v_0}$ and $\bar{v}_0 = \bar{v}_0 + \bar{v}_2\frac{\partial v_2}{\partial v_0}$.

Similarly to the forward mode, the backward mode gives a very efficient way to compute the product between the transpose of a Jacobian and a vector *i.e.,* $\mathcal{J}_f^\top v$. This product can be obtained by initializing the reverse mode with the vector v. The biggest advantage of the backward mode is to compute gradients very efficiently. The counter part is that it requires the storage of the intermediate variables used to compute the output of a computer program.

6 HYPERGRADIENT COMPUTATION IN NON-SMOOTH CONVEX LEARNING

Contents

In this chapter, we focus on the computation of the hypergradient, *i.e.*, the quantity $\nabla_\lambda \mathcal{L}(\lambda)$. First, we lay the theoretical foundations of our work in Section 6.1 where we discuss the differentiability of proximal operators and the potential problems of the mapping $\lambda \mapsto \hat{\beta}^{(\lambda)}$. Then, we present how implicit differentiation (Section 6.2) and iterative differentiation (Section 6.3) can be used to compute the desired hypergradient. In Section 6.4, we prove a stability result when one only has acces to an approximated solution of the lower problem. Finally, we describe our proposed method for the computation of the hypergradient in the non-smooth case comparing it to classical other methods (Section 6.5).

Solving Problem (6.1) with non-smooth lower problem has been a question of interest in the literature. Momma and Bennett (2002) used a pattern search method (Dennis and Torczon, 1994) which is derivative free to select the hyperparameters of the SVM/SVR model based on the cross-validation criterion. Later, Moore et al. (2011) proposed a subgradient

method to select the best hyperparameters of SVM/SVR writting the problem as a bilevel optimization problem. Peyré and Fadili (2011) proposed to smooth the lower optimization problem, Ochs et al. (2015); Frecon et al. (2018) relied on the forward differentiation combined with Bregman iterations to get differentiable steps. For non-smooth optimization problems, implicit differentiation has been considered for (constrained) convex optimization problems (Gould et al., 2016; Amos and Kolter, 2017; Agrawal et al., 2019), Lasso-type problems (Mairal et al., 2012; Bertrand et al., 2020), total variation penalties (Cherkaoui et al., 2020) and generalized to strongly monotone operators (Winston and Kolter, 2020).

6.1 Theoretical framework

We recall that our goal is to use a first order method to select hyperparameters. We saw in Chapter 5 that solving this problem boils down to the resolution of a bilevel optimization problem which writes:

$$\begin{aligned} &\arg\min_{\lambda \in \mathbb{R}^r} \left\{ \mathcal{L}(\lambda) \triangleq \mathcal{C}\left(\hat{\beta}^{(\lambda)}\right) \right\} \\ &s.t.\ \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \Phi(\beta, \lambda) \ , \end{aligned} \tag{6.1}$$

where $\mathcal{C}$ is a criterion that measures the performance of the model (see Chapter 5). Moreover, we consider the class of estimators obtained from solving the following generic optimization problem:

$$\hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \Phi(\beta, \lambda) \triangleq f(\beta) + \sum_{j=1}^{p} g_j(\beta_j, \lambda) \ . \tag{6.2}$$

The difficulty comes from the fact that we suppose that the functions g_j are possibly non-differentiable hence one cannot compute the hypergradient directly using for example the implicit formula of Theorem 5.1. However, we suppose that f is L-smooth. The rest of this chapter is dedicated to present results on the computation of the hypergradient for problems of the form Problem (6.2).

Differentiability of the regularization path. Before applying first-order methods to tackle Problem (6.1), one must ensure that the regularization path $\lambda \mapsto \hat{\beta}^{(\lambda)}$ is almost everywhere differentiable (as in Figure 6.1). This is the case for the Lasso (Mairal and Yu, 2012) and the SVM (Hastie et al., 2004; Rosset and Zhu, 2007) since solution paths are piecewise differentiable (see Figure 6.1). Results for nonquadratic datafitting terms are scarcer: Friedman

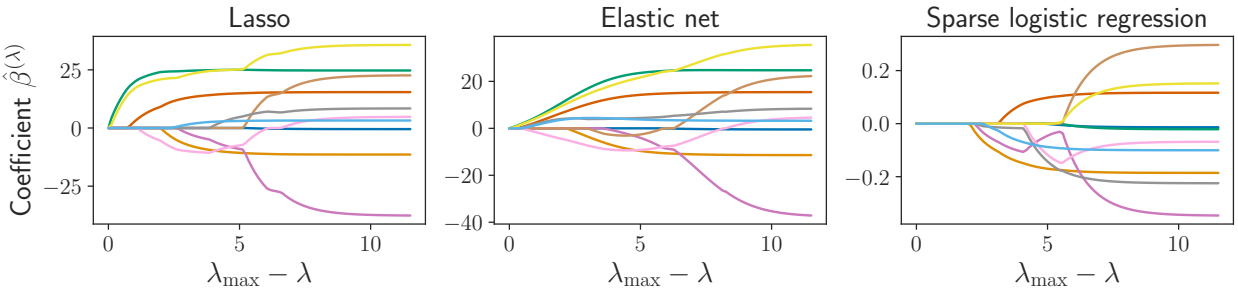

Figure 6.1 – **Regularization path, *diabetes* and *breast cancer* dataset.** Value of the coefficients as a function of λ for the Lasso, the elastic net and the sparse logistic regression. It illustrates the weak differentiability of the paths. We used *diabetes* for the Lasso and the elastic net, and the 10 first features of *breast cancer* for the sparse logistic regression.

et al. (2010) address the practical resolution of sparse logistic regression, but stay evasive regarding the differentiability of the regularization path. In the general case for problems of the form Problem (6.2), we believe it is an open question and leave it for future work.

Differentiability of proximal operators. The key point to obtain an implicit differentiation formula for non-smooth lower problems is to differentiate the fixed point equation of proximal gradient descent. Let $\hat{\beta}^{(\lambda)}$ be a solution of Problem (6.2) for a fixed λ then the following equation is satisfied for any $\gamma > 0$:

$$\hat{\beta}^{(\lambda)} = \operatorname{prox}_{\gamma g(\cdot,\lambda)}\left(\hat{\beta}^{(\lambda)} - \gamma \nabla f(\hat{\beta}^{(\lambda)})\right) \quad . \tag{6.3}$$

From a theoretical point of view, ensuring this differentiability at the optimum is non-trivial: Poliquin and Rockafellar (1996a, Thm. 3.8) showed that under a *twice epi-differentiability* condition the proximal operator is differentiable at optimum. For the convergence of forward and reverse modes in the non-smooth case, one has to ensure that, after enough iterations, the updates of the algorithms become differentiable. Deledalle et al. (2014) justified (weak) differentiability of proximal operators as they are non-expansive. However this may not be a sufficient condition, see Bolte and Pauwels (2020a,b). Bareilles et al. (2020, Thm. 3.2) shows that proximal operators are locally $\mathcal{C}^1$ under the assumptions that g is a partly smooth function (Definition 3.2), f is $\mathcal{C}^2$ with a L-Lipschitz gradient and the non-degeneracy assumption (Assumption 2.4). In our case, we show differentiability after *support identification* of the algorithms: active constraints are identified after a finite number of iterations by proximal gradient descent (Liang et al., 2014; Vaiter et al., 2018) and proximal coordinate descent, see Nutini (2018, Sec. 6.2) or see Theorem 3.1 in Chap-

ter 3.

For the rest of this chapter, we consider the bilevel optimization Problem (6.1) with the assumptions that the lower composite minimization Problem (6.2) is regular (Definition 2.16). To simplify the notation, we will denote by $\hat{\mathcal{S}} \triangleq \hat{\mathcal{S}}_{\hat{\beta}}$, the generalized support of $\hat{\beta}$ where $\hat{\beta}$ is a solution of Problem (6.2). Moreover, to prove local convergence properties of the Jacobian we assume regularity and strong convexity on the generalized support.

Assumption 6.1 (Locally $\mathcal{C}^2$ and $\mathcal{C}^3$)**.** The map $\beta \mapsto f(\beta)$ is locally $\mathcal{C}^3$ around $\hat{\beta}$. For all $\lambda \in \mathbb{R}^r$, for all $j \in \hat{\mathcal{S}}$ the map $g_j(\cdot, \lambda)$ is locally $\mathcal{C}^2$ around $\hat{\beta}_j$.

Assumption 6.2 (Restricted injectivity)**.** Let $\hat{\beta}$ be a solution of Problem (6.2) and $\hat{\mathcal{S}}$ its generalized support. The solution $\hat{\beta}$ satisfies the following restricted injectivity condition:

$$\nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\hat{\beta}) \succ 0 \ .$$

Assumptions 6.1 and 6.2 are classical for the analysis (Liang et al., 2017) and sufficient to derive rates of convergence for the Jacobian of the lower problem once the generalized support has been identified.

The next lemma guarantees uniqueness of the solution of Problem (6.2) under Assumptions 2.4 and 6.2.

Lemma 6.1 (Liang et al. 2017, Prop. 4.1)**.** *Assume that there exists a neighborhood Λ of λ such that Assumptions 2.4 and 6.2 are satisfied for every $\lambda \in \Lambda$ such that $\hat{\beta}^{(\lambda)}$ is a solution of Problem (6.2). Then for every $\lambda \in \Lambda$, Problem (6.2) has a unique solution, and the map $\lambda \mapsto \hat{\beta}^{(\lambda)}$ is well-defined on Λ.*

In the next two sections, we show how implicit and iterative differentiation can be used with a non-smooth lower problem taking into account the *structure* of the solution carried out by the notion of generalized support. Our work suppose that the non-smooth function is separable which may seem restrictive but already encompasses various examples found in machine learning.

6.2 Hypergradient computation using implicit differentiation

We use the fixed point iteration given in Equation (6.3) to derive an implicit differentiation formula for the hypergradient. The main theoretical challenge is to show the dif-

ferentiability of the function $\beta \mapsto \operatorname{prox}_{\gamma g}(\beta - \gamma \nabla f(\beta))$. Besides, taking advantage of the generalized sparsity of the regression coefficients $\hat{\beta}^{(\lambda)}$, one can show that the Jacobian $\hat{\mathcal{J}}$ is row-sparse, leading to substantial computational benefits when computing the hypergradient $\nabla_\lambda \mathcal{L}(\lambda)$) for Problem (6.2),

Theorem 6.1 (Non-smooth implicit formula)**.** *Let $0 < \gamma \leq 1/L$. Suppose Problem (6.2) is regular (Definition 2.16) and Assumption 6.1 hold. Let $\lambda \in \mathbb{R}^r$, Λ be a neighborhood of λ, and $\Gamma^\Lambda \triangleq \left\{ \hat{\beta}^{(\lambda)} - \gamma \nabla f(\hat{\beta}^{(\lambda)}) \,:\, \lambda \in \Lambda \right\}$. In addition,*

(H1) Suppose Assumptions 2.4 and 6.2 hold on Λ.

(H2) Suppose $\lambda \mapsto \hat{\beta}^{(\lambda)}$ is continuously differentiable on Λ.

(H3) Suppose for all $z \in \Gamma^\Lambda$, $\lambda \mapsto \operatorname{prox}_{\gamma g(\cdot,\lambda)}(z)$ is continuously differentiable on Λ.

(H4) Suppose $\partial_1 \operatorname{prox}_{\gamma g}$ and $\partial_2 \operatorname{prox}_{\gamma g}$ are Lipschitz continuous on $\Gamma^\Lambda \times \Lambda$.

Let $\hat{\beta} \triangleq \hat{\beta}^{(\lambda)}$ be the solution of Problem (6.2), $\hat{\mathcal{S}}$ its generalized support of cardinality $\hat{s}$. Then the Jacobian $\hat{\mathcal{J}}$ of the lower Problem (6.2) is given by the following formula,

$\hat{z} = \hat{\beta} - \gamma \nabla f(\hat{\beta})$, and $A \triangleq \operatorname{Id}_{|\hat{\mathcal{S}}|} - \partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}} \left(\operatorname{Id}_{|\hat{\mathcal{S}}|} - \gamma \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\hat{\beta}) \right)$:

$$\hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:} = \partial_2 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}^c} \quad , \tag{6.4}$$

$$\hat{\mathcal{J}}_{\hat{\mathcal{S}}:} = A^{-1} \left(\partial_2 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}} - \gamma \partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}} \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}^c} f(\hat{\beta}) \hat{\mathcal{J}}_{\hat{\mathcal{S}}^c} \right) \quad . \tag{6.5}$$

Proof. According to Lemma 6.1, Assumptions 2.4 and 6.2 ensure Problem (6.2) has a unique minimizer and $\lambda \mapsto \hat{\beta}^{(\lambda)}$ is well-defined on Λ. We consider the proximal gradient descent fixed point equation (Equation (6.3)). Together with the conclusion of Lemma 6.1, Assumptions 2.1 and 6.1, and given (H2), (H3) and (H4), we have that $\lambda \mapsto \psi \left(\beta^{(\lambda)} - \gamma \nabla f(\hat{\beta}^{(\lambda)}), \lambda \right) \triangleq \operatorname{prox}_{\gamma g(\cdot,\lambda)} \left(\hat{\beta}^{(\lambda)} - \gamma \nabla f(\hat{\beta}^{(\lambda)}) \right)$ is differentiable at λ. One can thus differentiate Equation (6.3) *w.r.t.* λ, which leads to:

$$\hat{\mathcal{J}} = \partial_1 \operatorname{prox}_{\gamma g}(\hat{z}) \left(\operatorname{Id} - \gamma \nabla^2 f(\hat{\beta}) \right) \hat{\mathcal{J}} + \partial_2 \operatorname{prox}_{\gamma g}(\hat{z}) \quad , \tag{6.6}$$

with $\hat{z} = \hat{\beta} - \gamma \nabla f(\hat{\beta})$. In addition to $0 < \gamma < 1/L \leq 1/L_j$, the separability of g, the regularity of Problem (6.2) (see Definition 2.16) and Assumptions 2.4 and 6.1 ensure (see Lemmas 3.2 and 3.3 in Chapter 3) that for any $j \in \hat{\mathcal{S}}^c$,

$$\partial_1 \operatorname{prox}_{\gamma g_j} \left(\hat{\beta}_j - \gamma \nabla_j f(\hat{\beta}) \right) = 0 \quad . \tag{6.7}$$

Plugging Equation (6.7) into Equation (6.6) ensures Equation (6.4) for all $j \in \hat{\mathcal{S}}^c$:

$$\hat{\mathcal{J}}_{j:} = \partial_2 \operatorname{prox}_{\gamma g_j}\left(\hat{\beta}_j - \gamma \nabla_j f(\hat{\beta})\right) \quad . \tag{6.8}$$

Plugging Equations (6.7) and (6.8) into Equation (6.6) shows that the Jacobian restricted on the generalized support $\hat{\mathcal{S}}$ satisfies the following linear system:

$$\left(\mathrm{Id}_{|\hat{\mathcal{S}}|} - \partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}}\left(\mathrm{Id}_{|\hat{\mathcal{S}}|} - \gamma \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\hat{\beta})\right)\right)\hat{\mathcal{J}}_{\hat{\mathcal{S}}:} = \\ -\gamma \partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}} \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}^c} f(\hat{\beta}) \hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:} + \partial_2 \operatorname{prox}_g(\hat{z})_{\hat{\mathcal{S}}:} \quad .$$

Since $0 < \gamma \leq 1/L$,

$$\|\partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}}(\mathrm{Id}_{|\hat{\mathcal{S}}|} - \gamma \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\hat{\beta}))\|_2 \leq \|\partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}}\|_2 \|\mathrm{Id}_{|\hat{\mathcal{S}}|} - \gamma \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\hat{\beta})\|_2 \\ < 1 \quad . \tag{6.9}$$

Since Equation (6.9) hold, $A \triangleq \mathrm{Id}_{|\hat{\mathcal{S}}|} - \partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}}(\mathrm{Id}_{|\hat{\mathcal{S}}|} - \gamma \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\hat{\beta}))$ is invertible, which leads to Equation (6.5). □

Remark 6.2. In the smooth case a $p \times p$ linear system is needed to compute the Jacobian (see in Chapter 5, Equation (5.7)). For non-smooth problems this is reduced to an $|\hat{\mathcal{S}}| \times |\hat{\mathcal{S}}|$ linear system ($|\hat{\mathcal{S}}| \leq p$ being the size of the generalized support, *e.g.*, the number of non-zero coefficients for the Lasso). This leads to significant speedups in practice, especially for very sparse vector $\hat{\beta}^{(\lambda)}$.

Remark 6.3. To obtain Theorem 6.1 we differentiated the fixed point equation of proximal gradient descent, though one could differentiate other fixed point equations (such as the one from proximal coordinate descent). The value of the Jacobian $\hat{\mathcal{J}}$ obtained with different fixed point equations would be the same, yet the associated systems could have different numerical stability properties. We leave this analysis to future work.

Example for the Lasso. We illustrate the result of Theorem 6.1 on the Lasso (Tibshirani, 1996) as the formula obtained is *simple* and is consistant with the existing literature. The Lasso is obtained as the solution of the following optimization problem:

$$\Phi(\beta, \lambda) = \frac{1}{2n}\|y - X\beta\|_2^2 + e^{\lambda}\|\beta\|_1 \quad , \tag{6.10}$$

where $X \in \mathbb{R}^{n \times p}$ is the design matrix, $y \in \mathbb{R}^n$ the observation vector and $\lambda \in \mathbb{R}$ the regularization parameter. Note that we adopt the hyperparameter parametrization of Pedregosa

(2016), *i.e.,* we write the regularization parameter as e^λ. This avoids working with a positivity constraint in the optimization process. It is also coherent with the usual choice of a geometric grid for grid-search (Friedman et al., 2010).

Solving the Lasso using the proximal gradient descent involves the proximal operator of the ℓ_1-norm, which is the soft-thresholding (ST). It is given by the following formula: $\mathrm{ST}(t, \tau) = \mathrm{sign}(t) \cdot (|t| - \tau)_+$ for any $t \in \mathbb{R}$ and $\tau \geq 0$ (extended for vectors componentwise). In the case of the Lasso, the function $f(\beta) = \|y - X\beta\|_2^2$ is L-smooth with the constant $L = \frac{\|X\|_2^2}{n}$ and is $\mathcal{C}^3$ on its whole domain. The functions $g_j(\beta_j) = e^\lambda|\beta_j|$ are convex, closed and proper. On the support, $\hat{\beta}_j \neq 0$ and for all $j \in \hat{\mathcal{S}}$, g_j is $\mathcal{C}^\infty$ around $\hat{\beta}_j$.

The non-degeneracy assumptions is key to ensure identification of the support as stated in Liang et al. (2017). As already said in Lemma 6.1, the assumptions Assumption 6.2 and Assumption 2.4 ensure the unicity of the solution of the Lasso for a fixed λ, hence the mapping $\lambda \mapsto \hat{\beta}^{(\lambda)}$ is well defined. The non-degeneracy assumption Assumption 2.4 has been well studied for the Lasso under the name of non-degenerate dual certificate (Candès et al., 2006) and the restricted injectivity Assumption 6.2 leads to a unique solution for the problem of Basis Pursuit (Chen et al., 1998).

To apply Theorem 6.1, we need to differentiate the soft-thresholding operator.

Lemma 6.2. *The soft-thresholding* $\mathrm{ST} : \mathbb{R} \times \mathbb{R}^+ \mapsto \mathbb{R}$ *defined by* $\mathrm{ST}(t, \tau) = \mathrm{sign}(t) \cdot (|t| - \tau)_+$ *is weakly differentiable with derivatives*

$$\partial_1 \mathrm{ST}(t, \tau) = \mathbb{1}_{\{|t|>\tau\}} \ , \tag{6.11}$$

and

$$\partial_2 \mathrm{ST}(t, \tau) = -\mathrm{sign}(t) \cdot \mathbb{1}_{\{|t|>\tau\}} \ , \tag{6.12}$$

where

$$\mathbb{1}_{\{|t|>\tau\}} = \begin{cases} 1, & \text{if } |t| > \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{6.13}$$

Proof. See (Deledalle et al., 2014, Proposition 1) □

Consider $\hat{\beta} \triangleq \hat{\beta}^{(\lambda)}$ a solution of Problem (6.10) and $\hat{\mathcal{S}}$ its support *i.e.,* the set of non-zero

coefficients. Then, Lemma 6.2 tells us that for any $j \in [p]$

$$\partial_1 \operatorname{ST}\left(\hat{\beta} - \frac{1}{L}X^\top(X\hat{\beta} - y), \frac{e^\lambda}{L}\right)_j = \begin{cases} 1, \text{ if } j \in \hat{\mathcal{S}} \\ 0, \text{ otherwise } . \end{cases} \tag{6.14}$$

and

$$\partial_2 \operatorname{ST}\left(\hat{\beta} - \frac{1}{L}X^\top(X\hat{\beta} - y), \frac{e^\lambda}{L}\right)_j = \begin{cases} -\operatorname{sign}(\hat{\beta}_j)\frac{e^\lambda}{L}, \text{ if } j \in \hat{\mathcal{S}} \\ 0, \text{ otherwise } . \end{cases} \tag{6.15}$$

This leads to an implicit formula for the Lasso that could already be derived from the solution of the Lasso given in Dossal et al. (2013).

Corollary 6.1 (Implicit formula for the Lasso)**.** *The Jacobian* $\hat{\mathcal{J}} = \hat{\mathcal{J}}_{(\lambda)}$ w.r.t. λ *of the Lasso Problem* (6.10) *writes:*

$$\hat{\mathcal{J}}_{\hat{\mathcal{S}}} = -ne^\lambda \left(X_{\hat{\mathcal{S}}}^\top X_{\hat{\mathcal{S}}}\right)^{-1} \operatorname{sign}(\hat{\beta}_{\hat{\mathcal{S}}}), \tag{6.16}$$

$$\hat{\mathcal{J}}_{\hat{\mathcal{S}}^c} = 0 \ . \tag{6.17}$$

Interestingly, the sparsity pattern of the Jacobian is the same than the solution in the case of the Lasso. Solving the linear system *only* requires to inverse a matrix of size $|\hat{\mathcal{S}}| \times |\hat{\mathcal{S}}|$, instead of the linear systemp of size $p \times p$ in the smooth case (see Section 5.2).

Example for the weighted Lasso. Taking advantage of the sparsity becomes even more advantageous when considering the weighted Lasso (wLasso, Zou 2006) which was introduced to reduce the bias of the Lasso. This model has one hyperparameter by feature which means p hyperparameters to select and reads:

$$\Phi(\beta, \lambda) = \frac{1}{2n}\|y - X\beta\|_2^2 + \sum_{j=1}^{p} e^{\lambda_j}|\beta_j| \ . \tag{6.18}$$

Applying Theorem 6.1 involves the derivatives of the soft-thresholding as described in the previous section.

Corollary 6.2 (Implicit formula for the weighted Lasso)**.** *The Jacobian* $\hat{\mathcal{J}} = \hat{\mathcal{J}}_{(\lambda)}$ w.r.t. $\lambda \in \mathbb{R}^p$

Algorithm 10 FORWARD-MODE PGD

input : $X \in \mathbb{R}^{n \times p}, y \in \mathbb{R}^n, \lambda \in \mathbb{R}^r, n_{\text{iter}} \in \mathbb{N}$, $\beta^{(0)} \in \mathbb{R}^p$, $\mathcal{J}^{(0)} \in \mathbb{R}^{p \times r}, \gamma > 0$

// jointly compute coef. & Jacobian

for $k = 1, \dots, n_{\text{iter}}$ **do**

  // update the regression coefficients

  $z^{(k)} = \beta^{(k-1)} - \gamma \nabla f(\beta^{(k-1)})$ // GD step

  $\mathrm{d}z^{(k)} = \mathcal{J}^{(k-1)} - \gamma \nabla^2 f(\beta^{(k-1)}) \mathcal{J}^{(k-1)}$

  $\beta^{(k)} = \text{prox}_{\gamma g}(z^{(k)})$ // prox. step $\mathcal{O}(p)$

  // update the Jacobian

  $\mathcal{J}^{(k)} = \partial_1 \text{prox}_{\gamma g}(z^{(k)}) \mathrm{d}z^{(k)}$ // $\mathcal{O}(pr)$

  $\mathcal{J}^{(k)} \mathrel{+}= \partial_2 \text{prox}_{\gamma g}(z^{(k)})$ // $\mathcal{O}(pr)$

$v = \nabla \mathcal{C}(\beta^{n_{\text{iter}}})$

return $\beta^{n_{\text{iter}}}, {\mathcal{J}^{n_{\text{iter}}}}^\top v$

Algorithm 11 REVERSE-MODE PGD

input : $X \in \mathbb{R}^{n \times p}, y \in \mathbb{R}^n, \lambda \in \mathbb{R}^r, n_{\text{iter}} \in \mathbb{N}$, $\beta^{(0)} \in \mathbb{R}^p, \gamma > 0$

// computation of $\hat{\beta}$

for $k = 1, \dots, n_{\text{iter}}$ **do**

  $z^{(k)} = \beta^{(k-1)} - \gamma \nabla f(\beta^{(k-1)})$ // GD step

  $\beta^{(k)} = \text{prox}_{\gamma g}\left(z^{(k)}\right)$ // proximal step

// backward computation of the gradient g

$v = \nabla \mathcal{C}(\beta^{(n_{\text{iter}})}), h = 0_{\mathbb{R}^r}$

for $k = n_{\text{iter}}, n_{\text{iter}} - 1, \dots, 1$ **do**

  $h \mathrel{+}= v^\top \partial_2 \text{prox}_{\gamma g}(z^{(k)})$ // $\mathcal{O}(pr)$

  $v \leftarrow \partial_1 \text{prox}_{\gamma g}(z^{(k)}) \odot v$ // $\mathcal{O}(p)$

  $v \leftarrow (\text{Id} - \gamma \nabla^2 f(\beta^{(k)}))v$ // $\mathcal{O}(np)$

return $\beta^{n_{\text{iter}}}, h$

of the weighted Lasso Problem (6.18) writes:

$$\hat{\mathcal{J}}_{\hat{\mathcal{S}}, \hat{\mathcal{S}}} = -\left(X_{\hat{\mathcal{S}}}^\top X_{\hat{\mathcal{S}}}\right)^{-1} \text{diag}\left(n e^{\lambda_{\hat{\mathcal{S}}}} \odot \text{sign}\, \hat{\beta}_{\hat{\mathcal{S}}}\right) \tag{6.19}$$

$$\hat{\mathcal{J}}_{j_1, j_2} = 0 \quad \textit{if } j_1 \notin \hat{\mathcal{S}} \textit{ or } \textit{if } j_2 \notin \hat{\mathcal{S}} \ . \tag{6.20}$$

Corollary 6.2 shows that the Jacobian of the weighted Lasso $\hat{\mathcal{J}}_{(\lambda)} \in \mathbb{R}^{p \times p}$ is row and column sparse. This is key for algorithmic efficiency. Indeed, *a priori*, one has to store a possibly dense $p \times p$ matrix, which is prohibitive when p is large. It leads to a *cheaper* way to compute the Jacobian as it *only* requires storing and inverting an $|\hat{\mathcal{S}}| \times |\hat{\mathcal{S}}|$ matrix.

Implicit differentiation can suffer from the fact that the linear system to solve requires the exact solution which is often not accessible to in practice. Another possibility for computing the Jacobian of the lower problem *w.r.t.* to the hyperparameters is to turn towards automatic differentiation which computes the derivatives based on the iterations of the solver.

6.3 Hypergradient computation using iterative differentiation

We now focus on iterative differentiation for proximal gradient descent and proximal coordinate descent (Algorithms 12 and 13). Iterative differentiation in the field of hyperpa-

rameter setting can be traced back to Domke (2012) who derived a backward differentiation algorithm for gradient descent, heavy ball and L-BFGS algorithms applied to smooth loss functions. Agrawal et al. (2019) generalized it to a specific subset of convex programs. Maclaurin et al. (2015) derived a backward differentiation for stochastic gradient descent. On the other hand Deledalle et al. (2014) used forward differentiation of (accelerated) proximal gradient descent for hyperparameter optimization with non-smooth penalties. Franceschi et al. (2017) proposed a benchmark of forward mode versus backward mode, varying the number of hyperparameters to learn. Frecon et al. (2018) cast the problem of inferring the groups in a group-Lasso model as a bilevel optimization problem and solved it using backward differentiation.

For coordinate descent, the computation of the iterative Jacobian in a forward way involves differentiating the following update, for $\gamma_j > 0$:

$$\begin{aligned} z_j &\leftarrow \beta_j - \gamma_j \nabla_j f(\beta) \\ \beta_j &\leftarrow \operatorname{prox}_{\gamma_j g_j}\left(\beta_j - \gamma_j \nabla_j f(\beta)\right) \\ \mathcal{J}_{j:} &\leftarrow \partial_1 \operatorname{prox}_{\gamma_j g_j}(z_j)\left(\mathcal{J}_{j:} - \gamma_j \nabla^2_{j:} f(\beta)\mathcal{J}\right) + \partial_2 \operatorname{prox}_{\gamma_j g_j}(z_j) \ . \end{aligned}$$

We address now the convergence of the iterative Jacobian scheme, a question which remained open in Deledalle et al. (2014, Section 4.1). We show that the forward differentiation converges to the Jacobian in the non-smooth separable setting. Moreover, we prove that the iterative Jacobian convergence is linear after support identification.

Theorem 6.4 (Local linear convergence of the Jacobian)**.** *Let $0 < \gamma \leq 1/L$. Suppose Problem (6.2) is regular (Definition 2.16) and Assumption 6.1 hold. Let $\lambda \in \mathbb{R}^r$, Λ be a neighborhood of λ, and $\Gamma^\Lambda \triangleq \left\{\hat{\beta}^{(\lambda)} - \gamma \nabla f(\hat{\beta}^{(\lambda)}) \ : \ \lambda \in \Lambda\right\}$. In addition, suppose hypotheses (H1) to (H4) from Theorem 6.1 are satisfied and the sequence $(\beta^{(k)})_{k\in\mathbb{N}}$ generated by Algorithm 10 (respectively by Algorithm 12) converges toward $\hat{\beta}$.*

Then, the sequence of Jacobians $(\mathcal{J}^{(k)})_{k\geq 0}$ generated by the forward differentiation of proximal gradient descent (Algorithm 10) (respectively by the forward differentiation of proximal coordinate descent, Algorithm 12) converges locally linearly towards $\hat{\mathcal{J}}$.

Proof. We start the proof with a result on an asymptotic vector autoregressive sequence, with an error term vanishing localy linearly to 0, then it converges linearly to its limit. In a more formal way:

Algorithm 12 FORWARD-MODE PCD

input : $X \in \mathbb{R}^{n\times p}, y \in \mathbb{R}^n, \lambda \in \mathbb{R}^r, n_{\text{iter}} \in \mathbb{N}$, $\beta \in \mathbb{R}^p$, $\mathcal{J} \in \mathbb{R}^{p\times r}, \gamma_1, \dots, \gamma_p$

// jointly compute coef. & Jacobian

for $k = 1, \dots, n_{\text{iter}}$ **do**

  for $j = 1, \dots, p$ **do**

    // update the regression coefficients

    $z_j \leftarrow \beta_j - \gamma_j \nabla_j f(\beta)$ // CD step

    $\mathrm{d}z_j \leftarrow \mathcal{J}_{j:} - \gamma_j \nabla^2_{j:} f(\beta)\mathcal{J}$

    $\beta_j \leftarrow \operatorname{prox}_{\gamma_j g_j}(z_j)$ // proximal step

    // update the Jacobian

    // diff. w.r.t. λ

    $\mathcal{J}_{j:} \leftarrow \partial_1 \operatorname{prox}_{\gamma_j g_j}(z_j)\mathrm{d}z_j$

    $\mathcal{J}_{j:}$ += $\partial_2 \operatorname{prox}_{\gamma_j g_j}(z_j)$

  $\beta^{(k)} = \beta$; $\mathcal{J}^{(k)} = \mathcal{J}$

$v = \nabla C(\beta)$

return $\beta^{n_{\text{iter}}}, \mathcal{J}^\top v$

Algorithm 13 REVERSE-MODE PCD

input : $X \in \mathbb{R}^{n\times p}, y \in \mathbb{R}^n, \lambda \in \mathbb{R}^r, n_{\text{iter}} \in \mathbb{N}$, $\beta \in \mathbb{R}^p, \gamma_1, \dots, \gamma_p$

// compute coef.

for $k = 1, \dots, n_{\text{iter}}$ **do**

  for $j = 1, \dots, p$ **do**

    // update the regression coefficients

    $z_j \leftarrow \beta_j - \gamma_j \nabla_j f(\beta)$ // CD step

    $\beta_j \leftarrow \operatorname{prox}_{\gamma_j g_j}(z_j)$ // proximal step

    $\beta^{(k,j)} = \beta; z_j^{(k)} = z_j$ // store iterates

// compute gradient g in a backward way

$v = \nabla C(\beta^{n_{\text{iter}}})$, $h = 0_{\mathbb{R}^r}$

for $k = n_{\text{iter}}, n_{\text{iter}} - 1, \dots, 1$ **do**

  for $j = p, \dots, 1$ **do**

    h −= $\gamma_j v_j \partial_2 \operatorname{prox}_{\gamma_j g_j}\left(z_j^{(k)}\right)$ // $\mathcal{O}(r)$

    v_j *= $\partial_1 \operatorname{prox}_{\gamma_j g_j}\left(z_j^{(k)}\right)$ // $\mathcal{O}(1)$

    v −= $\gamma_j v_j \nabla^2_{j:} f(\beta^{(k,j)})$ // $\mathcal{O}(np)$

return $\beta^{n_{\text{iter}}}, h$

Lemma 6.3. *Let $A \in \mathbb{R}^{p\times p}, b \in \mathbb{R}$ with $\rho(A) < 1$. Let $(\mathcal{J}^{(t)})_{t\in\mathbb{N}}$ be a sequence of $\mathbb{R}^p$ such that:*

$$\mathcal{J}^{(t+1)} = A\mathcal{J}^{(t)} + b + \epsilon^{(t)} \ , \tag{6.21}$$

with $(\epsilon^{(t)})_{t\in\mathbb{N}}$ a sequence which converges locally linearly to 0, then $(\mathcal{J}^{(t)})_{t\in\mathbb{N}}$ converges locally linearly to its limit $\hat{\mathcal{J}} \triangleq (\mathrm{Id} - A)^{-1}b$.

Proof. $(\epsilon^{(t)})_{t\in\mathbb{N}}$ converges locally linearly: there exists $c_1 > 0, 0 < \nu < 1$ such that:

$$\|\epsilon^{(t)}\| \leq c_1 \nu^t \ .$$

Polyak (1987, Chapter 2, Lemma 1) yields a bound on $\|A^k\|_2$, more precisely for every $\delta > 0$ there is a $c_2(\delta) = c_2$ such that

$$\|A^k\|_2 \leq c_2(\rho(A) + \delta)^k \ .$$

Since $\hat{\mathcal{J}} = (\mathrm{Id} - A)^{-1} b$ the limit $\hat{\mathcal{J}}$ of the sequence satisfies:

$$\hat{\mathcal{J}} = A\hat{\mathcal{J}} + b \ . \tag{6.22}$$

Taking the difference between Equations (6.21) and (6.22) yields:

$$\mathcal{J}^{(t+1)} - \hat{\mathcal{J}} = A(\mathcal{J}^{(t)} - \hat{\mathcal{J}}) + \epsilon^{(t)} \ . \tag{6.23}$$

Unrolling Equation (6.23) yields:

$$\begin{aligned}
\mathcal{J}^{(t+1)} - \hat{\mathcal{J}} &= A^{t+1}(\mathcal{J}^{(0)} - \mathcal{J}) + \sum_{k=0}^{t} A^k \epsilon^{t-k} \\
\|\mathcal{J}^{(t+1)} - \hat{\mathcal{J}}\|_2 &\leq \|A^{t+1}(\mathcal{J}^{(0)} - \mathcal{J})\|_2 + \sum_{k=0}^{t} \|A^k\|_2 \|\epsilon^{t-k}\| \\
&\leq \|A^{t+1}\|_2 \cdot \|\mathcal{J}^{(0)} - \hat{\mathcal{J}}\|_2 + c_1 \sum_{k=0}^{t} \|A^k\| \nu^{t-k} \\
&\leq c_2(\rho(A) + \delta)^{t+1} \cdot \|\mathcal{J}^{(0)} - \hat{\mathcal{J}}\|_2 + c_1 \sum_{k=0}^{t} c_2(\rho(A) + \delta)^k \nu^{t-k} \\
&\leq c_2(\rho(A) + \delta)^{t+1} \cdot \|\mathcal{J}^{(0)} - \hat{\mathcal{J}}\|_2 + c_1 c_2 \sum_{k=0}^{t/2} (\rho(A) + \delta)^k \nu^{t-k} + c_1 c_2 \sum_{k=t/2}^{t} (\rho(A) + \delta)^k \nu^{t-k} \\
&\leq c_2(\rho(A) + \delta)^{t+1} \cdot \|\mathcal{J}^{(0)} - \hat{\mathcal{J}}\|_2 + \frac{c_1 c_2 (\rho(A) + \delta)}{1 - \rho(A) - \delta} \sqrt{\nu}^t + \frac{c_1 c_2 \nu}{1 - \nu} \sqrt{(\rho(A) + \delta)}^t \ .
\end{aligned}$$

Thus, $(\mathcal{J}^{(t)})_{t \in \mathbb{N}}$ converges locally linearly towards its limit $\hat{\mathcal{J}}$. □

Now we prove Theorem 6.4 for proximal gradient descent.

Sketch of proof:

- The first step of the proof is to derive the forward update of the Jacobian applying the proximal gradient descent algorithm.
- Then, we show that once the support is identified, the updates lead to a vector autoregressive sequence.
- Finally, we use Lemma 6.3 to prove our results on the local linear convergence.

Proximal gradient descent case.

Solving Problem (6.2) with proximal gradient descent leads to the following updates:

$$\beta^{(k+1)} = \operatorname{prox}_{\gamma g}(\underbrace{\beta^{(k)} - \gamma \nabla f(\beta^{(k)})}_{z^{(k)}}) \ . \tag{6.24}$$

Consider the following sequence $(\mathcal{J}^{(k)})_{k\in\mathbb{N}}$ defined by:

$$\mathcal{J}^{(k+1)} = \partial_1 \operatorname{prox}_{\gamma g}(z^{(k)}) \left(\mathrm{Id} - \gamma \nabla^2 f(\beta^{(k)})\right) \mathcal{J}^{(k)} + \partial_2 \operatorname{prox}_{\gamma g}(z^{(k)}) \ . \tag{6.25}$$

Note that if $\operatorname{prox}_{\gamma g}$ is not differentiable with respect to the first variable at $z^{(k)}$ (respectively with respect to the second variable λ), any weak Jacobian can be used. When (H3) holds, differentiating Equation (6.24) *w.r.t.* λ yields exactly Equation (6.25).

The regularity of Problem (6.2) (Definition 2.16), Assumptions 2.4 and 6.1 and the convergence of $(\beta^{(k)})$ toward $\hat{\beta}$ ensure proximal gradient descent algorithm has finite identification property (Liang et al., 2014, Thm. 3.1): we note K the iteration when identification is achieved. As before, the separability of g, the regularity of Problem (6.2) (see Definition 2.16) and Assumptions 2.4 and 6.1 ensure (see Lemma 3.2 in Chapter 3) $\partial_1 \operatorname{prox}_{\gamma g}(z^k)_{\hat{\mathcal{S}}^c} = 0$, for all $k \geq K$. Thus, for all $k \geq K$,

$$\mathcal{J}^{(k)}_{\hat{\mathcal{S}}^c:} = \hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:} = \partial_2 \operatorname{prox}_{\gamma g}(z^{(k)})_{\hat{\mathcal{S}}^c:} \ .$$

The updates of the Jacobian then become:

$$\mathcal{J}^{(k+1)}_{\hat{\mathcal{S}}:} = \partial_1 \operatorname{prox}_{\gamma g}(z^{(k)})_{\hat{\mathcal{S}}:} \left(\mathrm{Id} - \gamma \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\beta^{(k)})\right) \mathcal{J}^{(k)}_{\hat{\mathcal{S}}:} + \partial_2 \operatorname{prox}_{\gamma g}(z^{(k)})_{\hat{\mathcal{S}}:} \ .$$

From Assumption 6.1, we have that f is locally $\mathcal{C}^3$ at $\hat{\beta}$, $g(\cdot, \lambda)$ is locally $\mathcal{C}^2$ at $\hat{\beta}$ hence $\operatorname{prox}_{g(\cdot,\lambda)}$ is locally $\mathcal{C}^2$. The function $\beta \mapsto \partial_1 \operatorname{prox}_{\gamma g}(\beta - \gamma \nabla f(\beta))_{\hat{\mathcal{S}}:}(\mathrm{Id} - \gamma \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\beta))$ is differentiable at $\hat{\beta}$. Using (H4) we have that $\beta \mapsto \partial_2 \operatorname{prox}_{\gamma g}(\beta - \gamma \nabla f(\beta))_{\hat{\mathcal{S}}:}$ is also differentiable at $\hat{\beta}$. Using the Taylor expansion of the previous functions yields:

$$\mathcal{J}^{(k+1)}_{\hat{\mathcal{S}}:} = \underbrace{\partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}} \left(\mathrm{Id} - \gamma \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\hat{\beta})\right)}_{A} \mathcal{J}^{(k)}_{\hat{\mathcal{S}}:} + \underbrace{\partial_2 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}:}}_{b} + \underbrace{o(\|\beta^{(k)} - \hat{\beta}\|)}_{\epsilon^{(k)}} \ . \tag{6.26}$$

Thus, for $0 < \gamma \leq 1/L$,

$$\rho(A) \leq \|A\|_2 \leq \underbrace{\|\partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{\mathcal{S}}}\|_2}_{\leq 1 \text{ (non-expansiveness)}} \underbrace{\|\mathrm{Id} - \gamma \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\hat{\beta})\|_2}_{<1 \text{ (Assumption 6.2)}} < 1 \ . \tag{6.27}$$

The inequality on the derivative of the proximal operator comes from the non-expansiveness of proximal operators. The second inequality comes from Assumption 6.2 and $0 < \gamma \leq 1/L$.

The regularity of Problem (6.2) (see Definition 2.16), Assumptions 2.4, 6.1 and 6.2 and the convergence of $(\beta^{(k)})$ toward $\hat{\beta}$ ensure $(\beta^{(k)})_{k\in\mathbb{N}}$ converges locally linearly (Liang et al., 2014, Thm. 3.1).

The asymptotic autoregressive sequence in Equation (6.26), $\rho(A) < 1$, and the local linear convergence of $(\epsilon^{(k)})_{k\in\mathbb{N}}$, yield our result using Lemma 6.3.

We now prove Theorem 6.4 for proximal coordinate descent.

Sketch of proof:

- The first step of the proof is to derive the forward update of the Jacobian applying the proximal coordinate descent algorithm.
- Then, we consider a whole epoch of the coordinate descent algorithm and show that once the support is identified, the sequence of Jacobian is a vector autoregressive sequence. The matrix appearing in the vector autoregressive sequence is the same as the one studied for the iterates in Chapter 3.
- Finally, we use Lemma 6.3 to prove our results on the local linear convergence.

Proximal coordinate descent. Compared to proximal gradient descent, the analysis of coordinate descent requires studying functions defined as a the composition of p applications, each of them only modifying one coordinate.

Coordinate descent updates read as follows:

$$\beta_j^{(k,j)} = \operatorname{prox}_{\gamma_j g_j}\underbrace{\left(\beta_j^{(k,j-1)} - \gamma_j \nabla_j f(\beta^{(k,j-1)})\right)}_{\triangleq z_j^{(k,j-1)}} \,. \tag{6.28}$$

We consider the following sequence:

$$\mathcal{J}_{j:}^{(k,j)} = \partial_1 \operatorname{prox}_{\gamma_j g_j}(z_j^{(k,j-1)}) \left(\mathcal{J}_{j:}^{(k,j-1)} - \gamma_j \nabla^2_{j:} f(\beta^{(k,j-1)}) \mathcal{J}^{(k,j-1)}\right) + \partial_2 \operatorname{prox}_{\gamma_j g_j}(z_j^{(k,j-1)}) \,. \tag{6.29}$$

Note that if $\operatorname{prox}_{\gamma g}$ is not differentiable with respect to the first variable at $z^{(k)}$ (respectively with respect to the second variable λ), any weak Jacobian can be used. When (H3) holds,

differentiating Equation (6.28) *w.r.t.* λ yields exactly to Equation (6.29).

The regularity of Problem (6.2), Assumptions 2.4 and 6.1 and the convergence of $(\beta^{(k)})_{k\in\mathbb{N}}$ toward $\hat{\beta}$ ensure proximal coordinate descent has finite identification property (Theorem 3.1 in Chapter 3): we note K the iteration when identification is achieved. Once the generalized support $\hat{\mathcal{S}}$ (of cardinality $\hat{s}$) has been identified, we have that for all $k \geq K$, $\beta^{(k)}_{\hat{\mathcal{S}}^c} = \hat{\beta}_{\hat{\mathcal{S}}}$ and for any $j \in \hat{\mathcal{S}}^c$, $\partial_1 \operatorname{prox}_{\gamma_j g_j}(z_j^{(k,j-1)}) = 0$. Thus $\mathcal{J}_{j:}^{(k,j)} = \partial_2 \operatorname{prox}_{\gamma_j g_j}(z_j^{(k,j-1)})$.

We then have that for any $j \in \hat{\mathcal{S}}$ and for all $k \geq K$:

$$\begin{aligned}\mathcal{J}_{j:}^{(k,j)} = \partial_1 \operatorname{prox}_{\gamma_j g_j}(z_j^{(k,j-1)}) \left(\mathcal{J}_{j:}^{(k,j-1)} - \gamma_j \nabla^2_{j,\hat{\mathcal{S}}} f(\beta^{(k,j-1)}) \mathcal{J}_{\hat{\mathcal{S}}:}^{(k,j-1)}\right) \\ + \partial_2 \operatorname{prox}_{\gamma_j g_j}(z_j^{(k,j-1)}) - \gamma_j \partial_1 \operatorname{prox}_{\gamma_j g_j}(z_j^{(k,j-1)}) \nabla^2_{j,\hat{\mathcal{S}}^c} f(\beta^{(k,j-1)}) \mathcal{J}_{\hat{\mathcal{S}}^c:}^{(k,j-1)} \quad .\end{aligned}$$

We can consider the applications

$$\beta \mapsto \partial_1 \operatorname{prox}_{\gamma_j g_j}\left(\beta_j - \gamma_j \nabla_j f(\beta)\right) \left(e_j - \gamma_j \nabla^2_{j:} f(\beta)\right) \quad ,$$

and

$$\beta \mapsto \partial_2 \operatorname{prox}_{\gamma_j g_j}\left(\beta_j - \gamma_j \nabla_j f(\beta)\right) - \gamma_j \partial_1 \operatorname{prox}_{\gamma_j g_j}\left(\beta_j - \gamma_j \nabla_j f(\beta)\right) \nabla^2_{j,\hat{\mathcal{S}}^c} f(\beta) \hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:} \quad ,$$

which are both differentiable at $\hat{\beta}$ using Assumption 6.1 and (H4). The Taylor expansion of the previous functions yields:

$$\begin{aligned}\mathcal{J}_{j:}^{(k,j)} = \partial_1 \operatorname{prox}_{\gamma_j g_j}\left(\hat{z}_j\right) \left(e_j - \gamma_j \nabla^2_{j,\hat{\mathcal{S}}} f(\hat{\beta})\right) \mathcal{J}_{\hat{\mathcal{S}}:}^{(k,j-1)} \\ + \partial_2 \operatorname{prox}_{\gamma_j g_j}\left(\hat{z}_j\right) - \gamma_j \partial_1 \operatorname{prox}_{\gamma_j g_j}\left(\hat{z}_j\right) \nabla^2_{j,\hat{\mathcal{S}}^c} f(\hat{\beta}) \mathcal{J}_{\hat{\mathcal{S}}^c:}^{(k,j-1)} \\ + o(||\beta^{(k,j-1)} - \hat{\beta}||) \quad .\end{aligned}$$

When considering a full epoch of coordinate descent, the Jacobian is obtained as the product of matrices of the form

$$A_j^\top = \left(\begin{array}{c|c|c|c|c|c|c} e_1 & \dots & e_{j-1} & v_j & e_{j+1} & \dots & e_{\hat{s}} \end{array} \right) \quad ,$$

where $v_j = \partial_1 \operatorname{prox}_{\gamma_j g_j}\left(\hat{z}_j\right) \left(e_j - \gamma_j \nabla^2_{j,\hat{\mathcal{S}}} f(\hat{\beta})\right)$. A full epoch can then be written

$$\mathcal{J}_{\hat{\mathcal{S}}:}^{(k+1)} = \underbrace{A_{j_{\hat{s}}} A_{j_{\hat{s}-1}} \dots A_{j_1}}_{A} \mathcal{J}_{\hat{\mathcal{S}}:}^{(k)} + b + \epsilon^{(k)} \quad ,$$

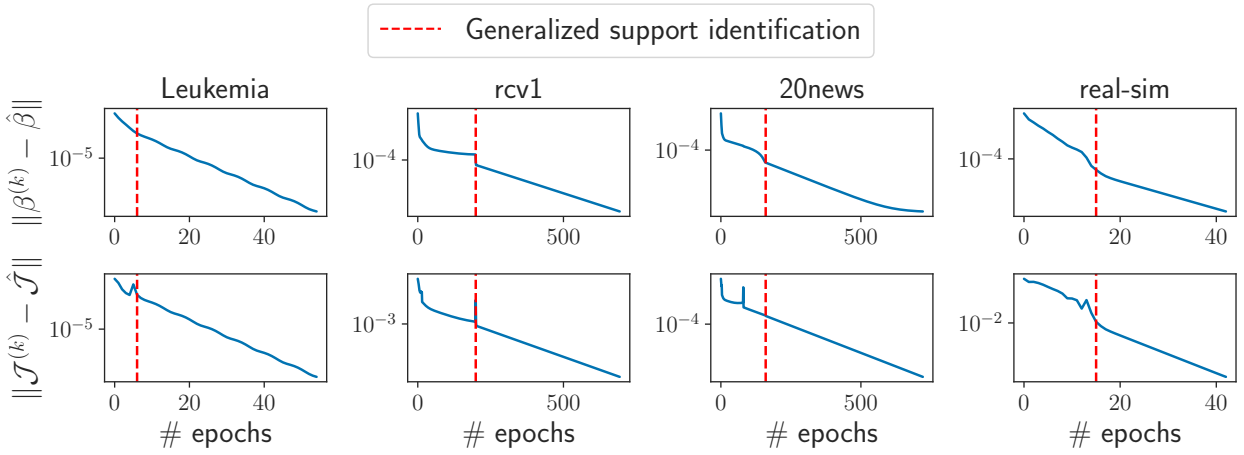

Figure 6.2 – **Local linear convergence of the Jacobian for the SVM.** Distance to optimum for the coefficients β (top) and the Jacobian $\mathcal{J}$ (bottom) of the forward differentiation of proximal coordinate descent (Algorithm 12) on multiple datasets. One epoch corresponds to one pass over the data, *i.e.*, one iteration with proximal gradient descent.

for a certain $b \in \mathbb{R}^{|\hat{\mathcal{S}}|}$.

The spectral radius of A is strictly bounded by 1 (Lemma 3.8): $\rho(A) < 1$. The regularity of Problem (6.2), Assumptions 2.4 and 6.1 and the convergence of $(\beta^{(k)})_{k\in\mathbb{N}}$ toward $\hat{\beta}$ ensure local linear convergence of $(\beta^{(k)})_{k\in\mathbb{N}}$ (Theorem 3.2). Hence, we can write the update for the Jacobian after an update of the coordinates from 1 to p:

$$\mathcal{J}^{(k+1)}_{\hat{\mathcal{S}}:} = A\mathcal{J}^{(k)}_{\hat{\mathcal{S}}:} + b + \epsilon^{(k)} \quad , \tag{6.30}$$

with $(\epsilon^{(k)})_{k\in\mathbb{N}}$ converging locally linearly to 0.

The asymptotic autoregressive sequence in Equation (6.30), $\rho(A) < 1$, and the local linear convergence of $(\epsilon^{(k)})_{k\in\mathbb{N}}$, yield our result using Lemma 6.3. □

Illustration of Theorem 6.4. We illustrate the results of the previous theorem on the SVM (Figure 6.2), on the Lasso (Figure 6.3) and on the sparse logistic regression (Figure 6.4) for multiple datasets (*leukemia*, *rcv1*, *news20* and *real-sim*[1]). The values of the hyperparameters λ are summarized in Table 6.1. Regression coefficients $\hat{\beta}^{(\lambda)}$ were computed to machine precision (up to duality gap smaller than 10^{-16}) using a state-of-the-art coordinate descent solver implemented in `Lightning` (Blondel and Pedregosa, 2016) for the SVM or `Celer` (Massias et al., 2020) for the Lasso and sparse logistic regression. The exact Ja-

[1]Data available on the *libsvm* website: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

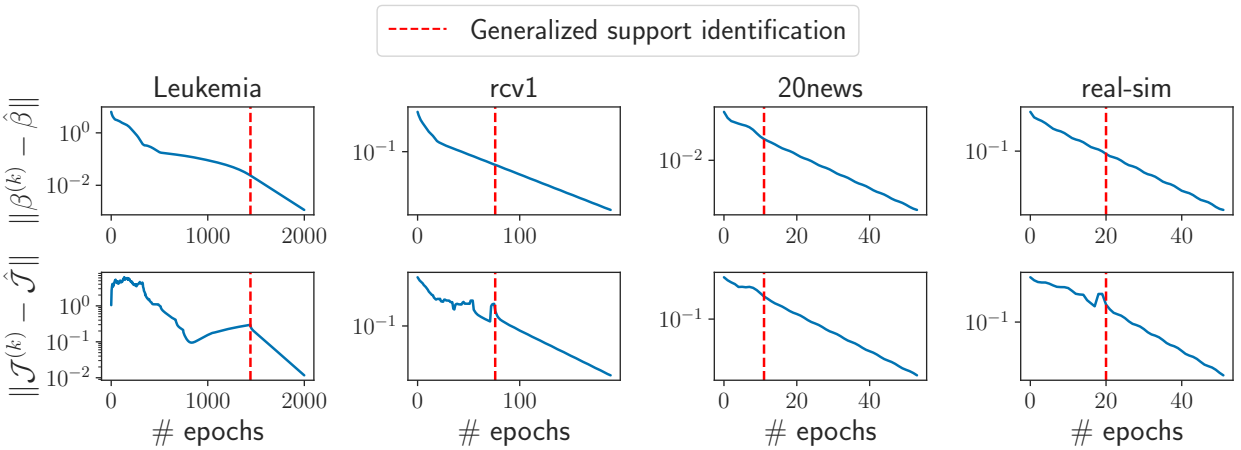

Figure 6.3 – **Local linear convergence of the Jacobian for the Lasso.** Distance to optimum for the coefficients β (top) and the Jacobian $\mathcal{J}$ (bottom) of the forward differentiation of proximal coordinate descent (Algorithm 12) on multiple datasets.

Table 6.1 – Dataset characteristics and regularization parameters used in Figure 6.2.

Datasets # samples # features	*leukemia* $n = 38$ $p = 7129$	*rcv1* $n = 20{,}242$ $p = 19{,}959$	*news20* $n = 19{,}996$ $p = 632{,}982$	*real-sim* $n = 72{,}309$ $p = 20{,}958$
Lasso	$e^{\lambda} = 0.01\, e^{\lambda_{\max}}$	$e^{\lambda} = 0.075\, e^{\lambda_{\max}}$	$e^{\lambda} = 0.3\, e^{\lambda_{\max}}$	$e^{\lambda} = 0.1\, e^{\lambda_{\max}}$
Logistic regression	$e^{\lambda} = 0.1\, e^{\lambda_{\max}}$	$e^{\lambda} = 0.25\, e^{\lambda_{\max}}$	$e^{\lambda} = 0.8\, e^{\lambda_{\max}}$	$e^{\lambda} = 0.15\, e^{\lambda_{\max}}$
SVM	$e^{\lambda} = 10^{-5}$	$e^{\lambda} = 3 \times 10^{-2}$	$e^{\lambda} = 10^{-3}$	$e^{\lambda} = 5 \times 10^{-2}$

cobian was computed via implicit differentiation (Equation (6.5)). Once these quantities were obtained, the forward differentiation was used of proximal coordinate descent (Algorithm 12) and monitored the distance between the iterates of the regression coefficients $\beta^{(k)}$ and the exact solution $\hat{\beta}$. We also monitored the distance between the iterates of the Jacobian $\mathcal{J}^{(k)}$ and the exact Jacobian $\hat{\mathcal{J}}$. The red vertical dashed line represents the iteration number where support identification happens. Once the support is identified, Figures 6.2 to 6.4 illustrate the linear convergence of the Jacobian. However, the behavior of the iterative Jacobian before support identification is more erratic and not even monotone.

6.4 Stability of the hypergradient

We now adress the question of stability for the computation of the hypergradient via implicit differentiation. Relying on iterative algorithms to solve Problem (6.2), only gives access to an approximation of $\hat{\beta}^{(\lambda)}$: this may lead to numerical errors when computing the gradient in Theorem 6.1. Extending the result of Pedregosa (2016, Thm. 1), which states

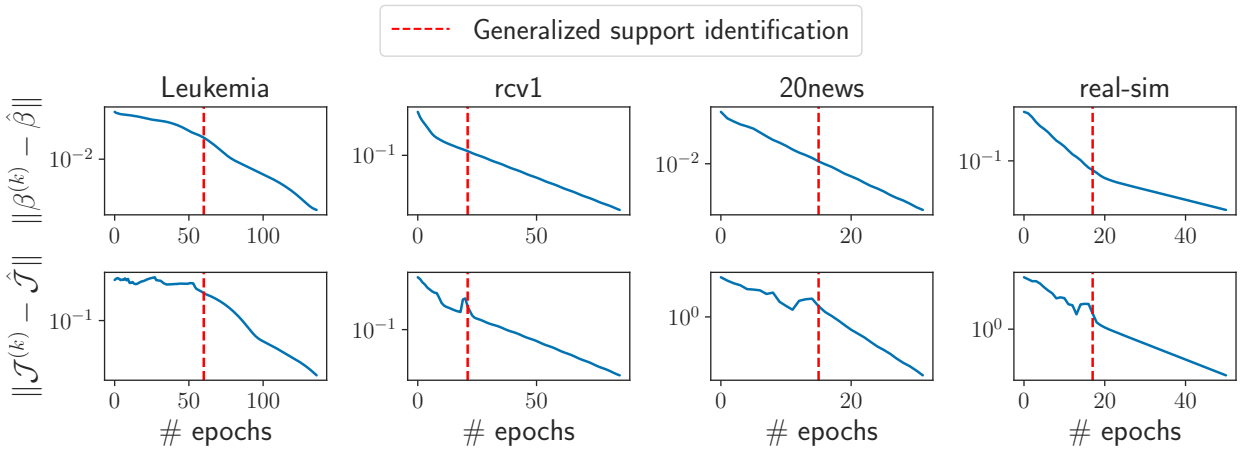

Figure 6.4 – **Local linear convergence of the Jacobian for sparse logistic regression.** Distance to optimum for the coefficients β (top) and the Jacobian $\mathcal{J}$ (bottom) of the forward differentiation of proximal coordinate descent (Algorithm 12) on multiple datasets.

that hypergradients can be computed approximately, we give a stability result for the computation of approximate hypergradients in the case of non-smooth lower problems.

Theorem 6.5 (Bound on the error of approximate hypergradient). *For $\lambda \in \mathbb{R}^r$, let $\hat{\beta}^{(\lambda)} \in \mathbb{R}^p$ be the exact solution of the lower Problem (6.2), and $\hat{\mathcal{S}}$ its generalized support. Suppose Problem (6.2) is regular (see Definition 2.16) and Assumption 6.1 hold. Let Λ be a neighborhood of λ, and $\Gamma^\Lambda \triangleq \left\{\hat{\beta}^{(\lambda)} - \gamma\nabla f(\hat{\beta}^{(\lambda)}) \;:\; \lambda \in \Lambda\right\}$. Suppose hypotheses (H1) to (H4) from Theorem 6.1 are satisfied. In addition suppose*

(H5) The application $\beta \mapsto \nabla^2 f(\beta)$ is Lipschitz continuous.

(H6) The criterion $\beta \mapsto \nabla\mathcal{C}(\beta)$ is Lipschitz continuous.

(H7) Both optimization problems in Algorithm 14 are solved up to precision ϵ with support identification: $\|\beta^{(\lambda)} - \hat{\beta}^{(\lambda)}\| \leq \epsilon$, $A^\top$ is invertible, and $\|A^{-1\top}\nabla_{\hat{\mathcal{S}}}\mathcal{C}(\beta^{(\lambda)}) - v\| \leq \epsilon$.

Then the error on the approximate hypergradient h returned by Algorithm 14 is of the order of magnitude of the error ϵ on $\beta^{(\lambda)}$ and v:

$$\|\nabla\mathcal{L}(\lambda) - h\| = \mathcal{O}(\epsilon) \quad .$$

Proof. **Overview of the proof.** Our goal is to bound the error between the approximate hypergradient h returned by Algorithm 14 and the true hypergradient $\nabla\mathcal{L}(\lambda)$. Following the analysis of Pedregosa (2016), two sources of approximation errors arise when comput-

ing the hypergradient:

- One from the inexact computation of $\hat{\beta}$. We denote β the approximate solution and suppose the problem is solved to precision ϵ with support identification (H7):

$$\begin{aligned} \beta_{\hat{\mathcal{S}}^c} &= \hat{\beta}_{\hat{\mathcal{S}}^c} \\ \|\beta_{\hat{\mathcal{S}}} - \hat{\beta}_{\hat{\mathcal{S}}}\| &\leq \epsilon \ . \end{aligned}$$

- One from the approximate resolution of the linear system, using (H7) yields:

$$\|A^{-1\top} \nabla_{\hat{\mathcal{S}}} \mathcal{C}(\beta) - v\| \leq \epsilon \ .$$

  The exact solution of the exact linear system $\hat{v}$ satisfies:

$$\hat{v} = \hat{A}^{-1\top} \nabla_{\hat{\mathcal{S}}} \mathcal{C}(\hat{\beta}) \ ,$$

  with

$$\begin{aligned} A &\triangleq \mathrm{Id}_{|\hat{\mathcal{S}}|} - \underbrace{\partial_1 \operatorname{prox}_{\gamma g}(\beta - \gamma \nabla f(\beta))_{\hat{\mathcal{S}}}}_{\triangleq C} \underbrace{\left(\mathrm{Id}_{|\hat{\mathcal{S}}|} - \gamma \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\beta)\right)}_{\triangleq D} \ , \\ \hat{A} &\triangleq \mathrm{Id}_{|\hat{\mathcal{S}}|} - \underbrace{\partial_1 \operatorname{prox}_{\gamma g}\left(\hat{\beta} - \gamma \nabla f(\hat{\beta})\right)_{\hat{\mathcal{S}}}}_{\triangleq \hat{C}} \underbrace{\left(\mathrm{Id}_{|\hat{\mathcal{S}}|} - \gamma \nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}} f(\hat{\beta})\right)}_{\triangleq \hat{D}} \ . \end{aligned}$$

- Using the last two points, the goal is to bound the difference between the exact hypergradient and the approximate hypergradient, $\|\nabla \mathcal{L}(\lambda) - h\|$. Following Algorithm 14, the exact hypergradient writes

$$\nabla \mathcal{L}(\lambda) = \hat{B}\hat{v} + \hat{\mathcal{J}}^{\top}_{\hat{\mathcal{S}}^c:} \nabla_{\hat{\mathcal{S}}^c} \mathcal{C}(\hat{\beta}) \ ,$$

  and the approximate hypergradient writes

$$h = Bv + \mathcal{J}^{\top}_{\hat{\mathcal{S}}^c:} \nabla_{\hat{\mathcal{S}}^c} \mathcal{C}(\beta) \ ,$$

  with

$$\begin{aligned} B &\triangleq \partial_2 \operatorname{prox}_{\gamma g}(\beta - \gamma \nabla f(\beta))_{\hat{\mathcal{S}}:} - \gamma \partial_1 \operatorname{prox}_{\gamma g}(\beta - \gamma \nabla f(\beta))_{\hat{\mathcal{S}}} \left(\nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}^c} f(\beta)\right) \hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:} \\ \hat{B} &\triangleq \partial_2 \operatorname{prox}_{\gamma g}\left(\hat{\beta} - \gamma \nabla f(\hat{\beta})\right)_{\hat{\mathcal{S}}:} - \gamma \partial_1 \operatorname{prox}_{\gamma g}\left(\hat{\beta} - \gamma \nabla f(\hat{\beta})\right)_{\hat{\mathcal{S}}} \left(\nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}^c} f(\hat{\beta})\right) \hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:} \ . \end{aligned}$$

We can exploit these decompositions to bound the difference between the exact hypergradient and the approximate hypergradient:

$$\begin{aligned}\|\nabla\mathcal{L}(\lambda) - h\| &= \|\hat{B}\hat{v} - Bv + \hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:}^\top \nabla_{\hat{\mathcal{S}}^c}\mathcal{C}(\hat{\beta}) - \hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:}^\top \nabla_{\hat{\mathcal{S}}^c}\mathcal{C}(\beta)\| \\ &\leq \|\hat{B}\hat{v} - Bv\| + \|\hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:}^\top \nabla_{\hat{\mathcal{S}}^c}\mathcal{C}(\hat{\beta}) - \hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:}^\top \nabla_{\hat{\mathcal{S}}^c}\mathcal{C}(\beta)\| \\ &\leq \|\hat{B}\hat{v} - B\hat{v} + B\hat{v} - Bv\| + \|\hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:}^\top (\nabla_{\hat{\mathcal{S}}^c}\mathcal{C}(\hat{\beta}) - \nabla_{\hat{\mathcal{S}}^c}\mathcal{C}(\beta))\| \\ &\leq \|\hat{v}\| \cdot \|\hat{B} - B\| + \|B\| \cdot \|\hat{v} - v\| + L_{\mathcal{C}}\|\hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:}^\top\| \cdot \|\beta - \hat{\beta}\| \ .\end{aligned}$$

Bounding $\|\hat{v} - v\|$ and $\|\hat{B} - B\|$ yields the desired result.

Bound on $\|\hat{v} - v\|$. We first prove that $\|A - \hat{A}\| = \mathcal{O}(\epsilon)$. Let L_H be the Lipschitz constant of the application $\beta \mapsto \nabla^2 f(\beta)$, then we have:

$$\begin{aligned}\|A - \hat{A}\|_2 &= \|CD - \hat{C}\hat{D}\|_2 \\ &\leq \|CD - C\hat{D}\|_2 + \|C\hat{D} - \hat{C}\hat{D}\|_2 \\ &\leq \underbrace{\|C\|_2}_{\leq 1 \text{ (non-expansiveness)}} \underbrace{\|D - \hat{D}\|_2}_{\leq L_H\|\beta-\hat{\beta}\| \text{ using (H5)}} + \underbrace{\|\hat{D}\|_2}_{\leq 1} \underbrace{\|C - \hat{C}\|_2}_{\mathcal{O}(\|\beta-\hat{\beta}\|) \text{ using (H4)}} \\ &\leq L_H\|\beta - \hat{\beta}\| + \mathcal{O}(\|\beta - \hat{\beta}\|) \\ &= \mathcal{O}(\|\beta - \hat{\beta}\|) \ . \end{aligned} \tag{6.31}$$

Let $\tilde{v}$ be the exact solution of the approximate system $A^\top \tilde{v} \triangleq \nabla_{\hat{\mathcal{S}}}\mathcal{C}(\beta)$. The following conditions are met:

- $\hat{v}$ is the exact solution of the exact linear system and $\tilde{v}$ is the exact solution of the approximate linear system

$$\begin{aligned}\hat{A}^\top \hat{v} &\triangleq \nabla_{\hat{\mathcal{S}}}\mathcal{C}(\hat{\beta}) \\ A^\top \tilde{v} &\triangleq \nabla_{\hat{\mathcal{S}}}\mathcal{C}(\beta) \ .\end{aligned}$$

- One can control the difference between the exact matrix in the linear system $\hat{A}$ and the approximate matrix A.

$$\|A - \hat{A}\|_2 \leq \delta\|\beta - \hat{\beta}\| \ ,$$

for a certain $\delta > 0$ (Equation (6.31)).

- One can control the difference between the two right-hand side of the linear systems

$$\|\nabla_{\hat{\mathcal{S}}}\mathcal{C}(\beta) - \nabla_{\hat{\mathcal{S}}}\mathcal{C}(\hat{\beta})\| \leq L_{\mathcal{C}}\|\beta - \hat{\beta}\| \ ,$$

  since $\beta \mapsto \nabla\mathcal{C}(\beta)$ is $L_{\mathcal{C}}$-Lipschitz continuous (H6).

- One can control the product of the perturbations

$$\delta \cdot \|\beta - \hat{\beta}\| \cdot \|\hat{A}^{-1}\|_2 \leq \rho < 1 \ .$$

Conditions are met to apply the normwise analysis result from Higham (2002, Thm 7.2), which leads to

$$\begin{aligned}
\|\tilde{v} - \hat{v}\| &\leq \frac{\epsilon}{1 - \epsilon\|\hat{A}^{-1}\|\delta}\left(L_{\mathcal{C}}\|\hat{A}^{-1}\| + \|\hat{v}\| \cdot \|\hat{A}^{-1}\|\delta\right) \\
&\leq \frac{\epsilon}{1 - \rho}\left(L_{\mathcal{C}}\|\hat{A}^{-1}\| + \|\hat{v}\| \cdot \|\hat{A}^{-1}\|\delta\right) \\
&= \mathcal{O}(\epsilon) \ . \qquad (6.32)
\end{aligned}$$

The bound on $\|\tilde{v} - \hat{v}\|$ finally yields a bound on $\|v - \hat{v}\|$:

$$\begin{aligned}
\|v - \hat{v}\| &= \|v - \tilde{v} + \tilde{v} - \hat{v}\| \\
&\leq \|v - \tilde{v}\| + \|\tilde{v} - \hat{v}\| \\
&\leq \|A^{-1}A(v - \tilde{v})\| + \|\tilde{v} - \hat{v}\| \\
&\leq \|A^{-1}\|_2 \times \underbrace{\|A(v - \tilde{v})\|}_{\leq\epsilon \text{ (H7)}} + \underbrace{\|\tilde{v} - \hat{v}\|}_{\mathcal{O}(\epsilon) \text{ (Equation (6.32))}} \\
&= \mathcal{O}(\epsilon) \ .
\end{aligned}$$

Bound on $\|B - \hat{B}\|_2$.

$$\begin{aligned}
\|B - \hat{B}\|_2 \leq\ & \|\partial_2 \operatorname{prox}_{\gamma g}(\beta - \gamma\nabla f(\beta))_{\hat{\mathcal{S}}:} - \partial_2 \operatorname{prox}_{\gamma g}(\hat{\beta} - \gamma\nabla f(\hat{\beta}))_{\hat{\mathcal{S}}:}\|_2 \\
&+ \gamma\|\partial_1 \operatorname{prox}_{\gamma g}(\hat{\beta} - \gamma\nabla f(\hat{\beta}))_{\hat{\mathcal{S}}}\nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}^c}f(\hat{\beta})\hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:} - \partial_1 \operatorname{prox}_{\gamma g}(\beta - \gamma\nabla f(\beta))_{\hat{\mathcal{S}}}\nabla^2_{\hat{\mathcal{S}},\hat{\mathcal{S}}^c}f(\beta)\hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:}\|_2 \\
&\leq L_1\|\beta - \gamma\nabla f(\beta)_{\hat{\mathcal{S}}:} - \hat{\beta} + \gamma\nabla f(\hat{\beta})\| \text{ using (H4)} \\
&+ L_2\|\hat{\beta} - \beta\| \cdot \|\hat{\mathcal{J}}_{\hat{\mathcal{S}}^c:}\| \text{ using (H4) and Assumption 6.1} \\
&= \mathcal{O}(\|\hat{\beta} - \beta\|) \ .
\end{aligned}$$

□

Remark 6.6. The Lipschitz continuity of the proximity operator with respect to λ (H4) is satisfied for usual proximal operators, in particular all the operators in Table 6.3. The Lipschitz continuity of the Hessian and the criterion, hypotheses (H5) and (H6), are satisfied for usual machine learning loss functions and criteria, such as the least squares and the logistic loss.

Remark 6.7. To simplify the analysis, we used the same tolerance for the resolution of the lower Problem (6.2) and the resolution of the linear system. Theorem 6.5 gives intuition on the fact that the lower problem does not need to be solved at high precision to lead to good hypergradients estimation. Note that in practice one does not easily control the distance between the approximate solution and the exact one $\|\beta^{(k)} - \hat{\beta}\|$: most softwares provide a solution up to a given duality gap (sometimes even other criteria), not $\|\beta^{(k)} - \hat{\beta}\|$.

6.5 Proposed method for the computation of the hypergradient

Algorithm 14 IMPLICIT DIFFERENTIATION

input : $X \in \mathbb{R}^{n \times p}, y \in \mathbb{R}^n, \lambda \in \mathbb{R}, \epsilon > 0$

init : $\gamma > 0$

`// compute the solution of lower problem`

Find β such that: $\Phi(\beta, \lambda) - \Phi(\hat{\beta}, \lambda) \leq \epsilon$

`// compute the gradient`

Compute the generalized support S of β

$z = \beta - \gamma \nabla f(\beta)$

$\mathcal{J}_{S^c:} = \partial_2 \operatorname{prox}_{\gamma g}(z)_{S^c}$

$A = \mathrm{Id}_s - \partial_1 \operatorname{prox}_{\gamma g}(z)_S (\mathrm{Id}_s - \gamma \nabla^2_{S,S} f(\beta))$

Find v such that $\|A^{-1\top} \nabla_S \mathcal{C}(\beta) - v\| \leq \epsilon$

$B = \partial_2 \operatorname{prox}_{\gamma g}(z)_S$

$\quad - \gamma \partial_1 \operatorname{prox}_{\gamma g}(z)_S \nabla^2_{S,S^c} f(\beta) \mathcal{J}_{S^c}$

$\nabla \mathcal{L}(\lambda) = \mathcal{J}^\top_{S^c:} \nabla_{S^c} \mathcal{C}(\beta) + v^\top B$

return $\mathcal{L}(\lambda) \triangleq \mathcal{C}(\beta), \nabla \mathcal{L}(\lambda)$

We now describe our proposed method to compute the hypergradient of Problem (6.1). In order to take advantage of the sparsity induced by the generalized support, we propose an implicit differentiation algorithm for non-smooth lower problem that can be found in

Algorithm 14. First, a solution of the lower Problem (6.2) is computed using a solver that identifies the generalized support. Then, the hypergradient is computed by solving the linear system in Equation (6.5). This linear system can be solved using multiple algorithms, including conjugate gradient or fixed point methods. Table 6.2 summarizes the computational complexity in space and time of the described algorithms.

Table 6.2 – Cost in time and space for each method: p is the number of features, n the number of samples, r the number of hyperparameters, and $|\hat{\mathcal{S}}|$ is the size of the generalized support (Definition 2.17, $|\hat{\mathcal{S}}| \leq p$ and usually $|\hat{\mathcal{S}}| \ll p$). The number of iterations of the lower solver is noted n_{iter}, the number of iterations of the solver of the linear system is noted n_{sys}.

Differentiation	Algorithm	Space	Time
Forward-mode PGD	Algorithm 10	$\mathcal{O}(p\,r)$	$\mathcal{O}(n\,p\,r\,n_{\text{iter}})$
Reverse-mode PGD	Algorithm 11	$\mathcal{O}(p\,n_{\text{iter}})$	$\mathcal{O}(n\,p\,n_{\text{iter}} + n\,p\,n_{\text{iter}})$
Forward-mode PCD	Algorithm 12	$\mathcal{O}(p\,r)$	$\mathcal{O}(n\,p\,r\,n_{\text{iter}})$
Reverse-mode PCD	Algorithm 13	$\mathcal{O}(p\,n_{\text{iter}})$	$\mathcal{O}(n\,p\,n_{\text{iter}} + n\,p^2\,n_{\text{iter}})$
Implicit differentiation	Algorithm 14	$\mathcal{O}(p + \lvert\hat{\mathcal{S}}\rvert)$	$\mathcal{O}(n\,p\,n_{\text{iter}} + n\,\lvert\hat{\mathcal{S}}\rvert\,n_{\text{sys}})$

To compute the hypergradient, one needs to compute the derivatives of the related proximal operator. We provide in Table 6.3 formula for the proximal operators used in this chapter.

Comparison with alternative approaches (Figure 6.5). First, we compare different methods to compute the hypergradient:

- Forward differentiation of proximal coordinate descent (Algorithm 12).
- Backward differentiation of proximal coordinate descent (Algorithm 13).
- `cvxpylayers` (Agrawal et al., 2019), a software based on `cvxpy` (Diamond and Boyd, 2016), solving *disciplined parametrized programming* and providing derivatives with respect to the parameters of the program. It is thus possible to use `cvxpylayers`

Table 6.3 – Partial derivatives of proximal operators used.

$g_j(\beta_j, \lambda)$	$\operatorname{prox}_{g_j(\cdot,\lambda)}(z_j)$	$\partial_1 \operatorname{prox}_{g_j(\cdot,\lambda)}(z_j)$	$\partial_2 \operatorname{prox}_{g_j(\cdot,\lambda)}(z_j)$
$e^\lambda \beta_j^2/2$	$z_j/(1+e^\lambda)$	$1/(1+e^\lambda)$	$-z_j e^\lambda/(1+e^\lambda)^2$
$e^\lambda \lvert\beta_j\rvert$	$\operatorname{ST}(z_j, e^\lambda)$	$\lvert\operatorname{sign}(\operatorname{ST}(z_j, e^\lambda))\rvert$	$-e^\lambda \operatorname{sign}(\operatorname{ST}(z_j, e^\lambda))$
$e^{\lambda_1}\lvert\beta_j\rvert + \frac{1}{2}e^{\lambda_2}\beta_j^2$	$\frac{\operatorname{ST}(z_j, e^{\lambda_1})}{1+e^{\lambda_2}}$	$\frac{\lvert\operatorname{sign}(\operatorname{ST}(z_j, e^{\lambda_1}))\rvert}{1+e^{\lambda_2}}$	$\left(\frac{-e^{\lambda_1}\operatorname{sign}(\operatorname{ST}(z_j, e^{\lambda_1}))}{1+e^{\lambda_2}}, \frac{-\operatorname{ST}(z_j, e^{\lambda_1})e^{\lambda_2}}{(1+e^{\lambda_2})^2}\right)$
$\iota_{[0,e^\lambda]}(\beta_j)$	$\max(0, \min(z_j, e^\lambda))$	$\mathbb{1}_{]0,e^\lambda[}(z_j)$	$e^\lambda \mathbb{1}_{z_j > e^\lambda}$

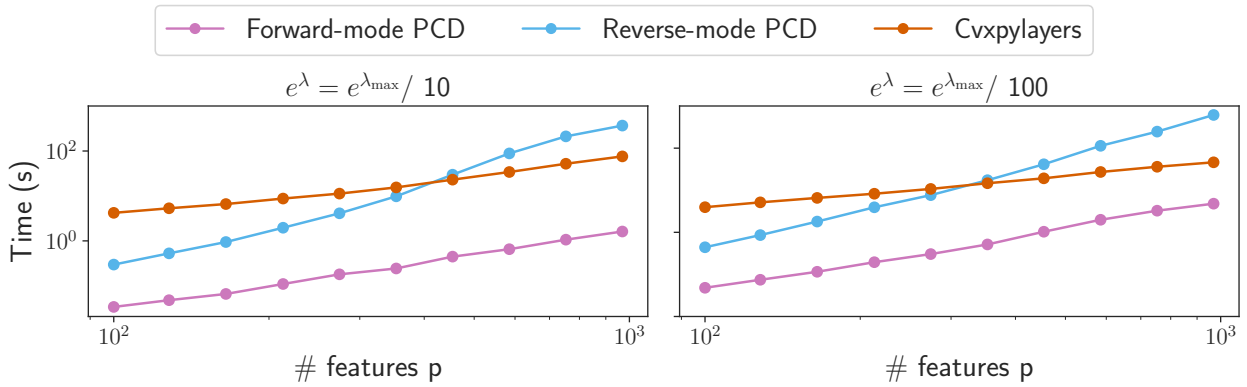

Figure 6.5 – **Lasso held-out, time to compute the hypergradient, *gina* dataset.** Time comparison to compute a single hypergradient as a function of the number of features, for multiple values of λ: $e^\lambda = e^{\lambda_{\max}}/10$ (left), and $e^\lambda = e^{\lambda_{\max}}/100$ (right).

to compute gradients with respect to the regularization parameters.

Figure 6.5 compares the time taken by multiple methods to compute a single hypergradient $\nabla\mathcal{L}(\lambda)$ for the Lasso (see Table 5.1), for multiple values of λ. It shows the time taken to compute the regression coefficients and the hypergradient, as a function of the number of columns, sampled from the design matrix from the *gina* dataset. The columns were selected at random and 10 repetitions were performed for each point of the curves. In order to aim for good numerical precision, problems were solved up to a duality gap of 10^{-6} for the forward and the backward differentiation. `cvxpylayers` relies on `cvxpy`, solving Problem (6.2) using a splitting conic solver (O'Donoghue et al., 2019). Since the termination criterion of the splitting conic solver is not exactly the duality gap (O'Donoghue et al., 2016, Sec. 3.5), we used the default tolerance of 10^{-4}. The hypergradient $\nabla\mathcal{L}(\lambda)$ was computed for held-out mean squared error (see Table 5.2).

The forward differentiation of proximal coordinate descent is one order of magnitude faster than `cvxpylayers` and two orders of magnitude faster than the backward differentiation of proximal coordinate descent. The larger the value of λ, the more the coefficients β are sparse, leading to significant speedups in this regime. This performance is in accordance with the lower time cost of the forward mode in Table 6.2.

Combining implicit differentiation with state-of-the art solvers (Figures 6.6 and 6.7). We now compare the different approaches described in this chapter:

- Forward differentiation of proximal coordinate descent (Algorithm 12).
- Implicit differentiation (Algorithm 14) with proximal coordinate descent to solve the

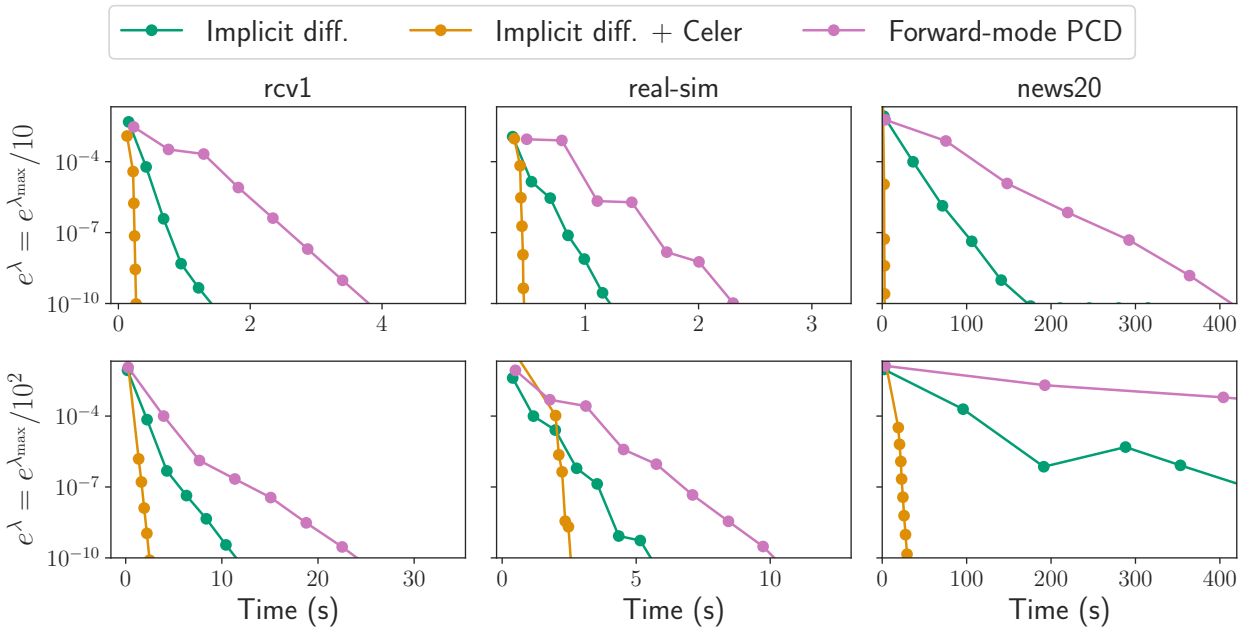

Figure 6.6 – **Lasso held-out, time to compute one hypergradient.** Absolute difference between the exact hypergradient (using $\hat{\beta}$) and the iterate hypergradient (using $\beta^{(k)}$) of the Lasso as a function of time. Results are for three datasets and two different regularization parameters. For the implicit differentiation, the lower problem is solved using proximal coordinate descent (Implicit diff.) or `Celer` (Massias et al. 2020, Implicit diff. + `Celer`).

lower problem. For efficiency, this solver was coded in `Numba` (Lam et al., 2015).

- Implicit differentiation (Algorithm 14) with state-of-the-art algorithm to solve the lower problem: we used `Celer` (Massias et al., 2020) for the Lasso, and `Lightning` (Blondel and Pedregosa, 2016) for the SVM.

Figure 6.6 shows for three datasets and two values of regularization parameters the absolute difference between the exact hypergradient and the approximate hypergradient obtained via multiple algorithms as a function of time. Figure 6.7 reports similar results for the SVM, on the same datasets, except *news20*, which is not well suited for SVM, due to limited number of samples.

First, it demonstrates that implicit differentiation methods are faster than the forward differentiation of the proximal coordinate descent (pink). This illustrates the benefits of restricting the gradient computation to the support of the Jacobian. Second, thanks to the flexibility of our approach, we obtain additional speed-ups by combining implicit differentiation with a state-of-the-art solver, `Celer`. The resulting method (orange) significantly improves over implicit differentiation using a vanilla proximal coordinate descent (green).

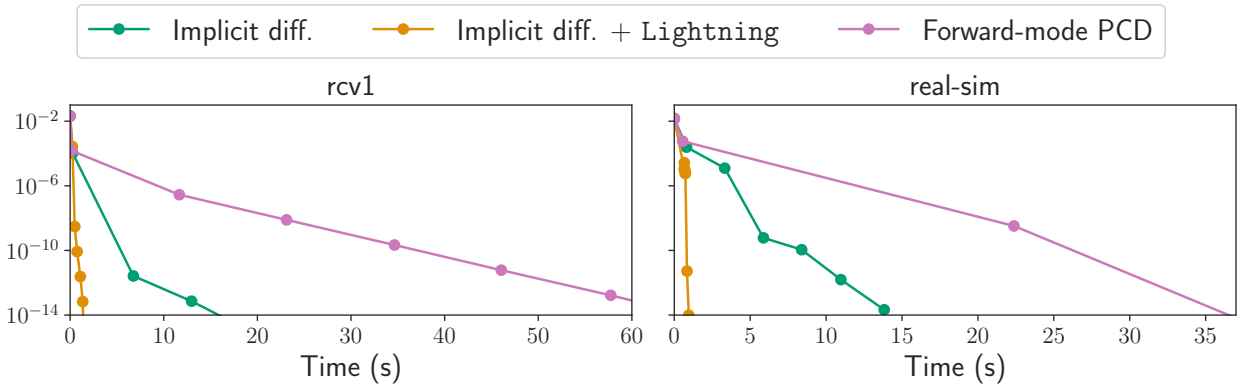

Figure 6.7 – **SVM held-out, time to compute one hypergradient.** Absolute difference between the exact hypergradient (using $\hat{\beta}$) and the iterate hypergradient (using $\beta^{(k)}$) of the SVM as a function of time. For the implicit differentiation, the lower problem is solved using proximal coordinate descent (Implicit diff.) or `Lightning` (Blondel and Pedregosa 2016, Implicit diff. + `Lightning`).

7 HYPERPARAMETER OPTIMIZATION IN NON-SMOOTH CONVEX LEARNING

Contents

In this chapter, we capitalise on the results from the previous chapter to propose an efficient algorithm for hyperparameter optimization. We begin with a discussion on the resolution of the related bilevel optimization in Section 7.1. Then we illustrate the efficiency and the benefits of using our proposed methods to select the hyperparameters of the Lasso (Section 7.2), the elastic net (Section 7.3), the multiclass logistic regression (Section 7.4) and finally the weighted Lasso (Section 7.5).

Multiple datasets will be used in this chapter. We describe in Table 7.1 the characteristics of these datasets coming from *libsvm* or *openML*.

We will compare multiple methods to find the optimal hyperparameters for the Lasso, elastic net and multiclass sparse logistic regression. The following methods are compared:

name	# samples n	# features p	# classes q	density
breast cancer	569	30	–	1
diabetes	442	10	–	1
leukemia	72	7129	–	1
gina agnostic	3468	970	–	1
rcv1	20,242	19,960	–	3.7×10^{-3}
real-sim	72,309	20,958	–	2.4×10^{-3}
news20	19,996	632,983	–	6.1×10^{-4}
mnist	60.000	683	10	2.2×10^{-1}
usps	7291	256	10	1
rcv1 multiclass	15,564	16,245	53	4.0×10^{-3}
aloi	108,000	128	1000	2.4×10^{-1}

Table 7.1 – Characteristics of the datasets.

- **Grid-search**: for the Lasso and the elastic net, the number of hyperparameters is small, and grid-search is tractable. For the Lasso we chose a grid of 100 hyperparameters λ, uniformly spaced between $\lambda_{\max} - \ln(10^4)$ and $\lambda_{\max}$. For the elastic net we chose for each of the two hyperparameters a grid of 10 values uniformly spaced between $\lambda_{\max}$ and $\lambda_{\max} - \ln(10^4)$. The product grid thus has 100 points.

- **Random-search**: we chose 30 values of λ sampled uniformly between $\lambda_{\max}$ and $\lambda_{\max} - \ln(10^4)$ for each hyperparameter. For the elastic net we chose 30 points sampled uniformly in $[\lambda_{\max} - \ln(10^4), \lambda_{\max}]^2$

- **SMBO**: this algorithm is SMBO using as criterion expected improvement (EI) and the Tree-structured Parzen Estimator (TPE) as model. First it evaluates $\mathcal{L}$ using 5 values of λ, chosen uniformly at random between $\lambda_{\max}$ and $\lambda_{\max} - \ln(10^4)$. Then a TPE model is fitted on the data points $(\lambda^{(1)}, \mathcal{L}(\lambda^{(1)})), \ldots, (\lambda^{(5)}, \mathcal{L}(\lambda^{(5)}))$. Iteratively, the EI is used to choose the next point to evaluate $\mathcal{L}$ at, and this value is used to update the model. We used the `hyperopt` implementation (Bergstra et al., 2013).

- **1st order**: first-order method with exact gradient (Algorithm 15 with constant tolerances $\epsilon_i = 10^{-6}$), with $\lambda_{\max} - \ln(10^2)$ as a starting point.

- **1st order approx**: a first-order method using approximate gradient (Algorithm 15 with tolerances ϵ_i, geometrically decreasing from 10^{-2} to 10^{-6}), with $\lambda_{\max} - \ln(10^2)$ as a starting point.

Algorithm 15 HEURISTIC GRADIENT DESCENT WITH APPROXIMATE GRADIENT

input : $X \in \mathbb{R}^{n \times p}, y \in \mathbb{R}^n, \lambda \in \mathbb{R}^r, (\epsilon_i)$
init : use_adaptive_step_size $=$ True
for $i = 1, \dots, iter$ **do**
  $\lambda^{\text{old}} \leftarrow \lambda$
  `// compute the value and the gradient`
  $\mathcal{L}(\lambda), \nabla\mathcal{L}(\lambda) \leftarrow$ Algorithm 14$(X, y, \lambda, \epsilon_i)$
  if use_adaptive_step_size **then**
    $\alpha = 1/\|\nabla\mathcal{L}(\lambda)\|$
  $\lambda \mathrel{-}= \alpha\nabla\mathcal{L}(\lambda)$ `// gradient step`
  if $\mathcal{L}(\lambda) > \mathcal{L}(\lambda^{\text{old}})$ **then**
    use_adaptive_step_size $=$ False
    $\alpha \mathrel{/}= 10$
return λ

7.1 Resolution of the bilevel optimization problem

We recall that choosing the best hyperparameters for a given criterion can be written as:

$$\begin{aligned} &\underset{\lambda \in \mathbb{R}^r}{\arg\min} \left\{ \mathcal{L}(\lambda) \triangleq \mathcal{C}\left(\hat{\beta}^{(\lambda)}\right) \right\} \\ &s.t.\ \hat{\beta}^{(\lambda)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min}\, \Phi(\beta, \lambda) \ , \end{aligned} \tag{7.1}$$

We saw in the previous chapters that one could rely on first order methods to solve Problem (7.1) which would require the computation of the gradient of $\mathcal{L}$ *w.r.t.* λ. In Chapter 6, we presented new results on the computation of this gradient when the lower problem considered is non-smooth. We now use the previous results to propose an algorithm to solve Problem (5.2) and hence, dive into the field of hyperparameter optimization.

From a practical point of view, once the hypergradient has been computed, first-order methods require the definition of a step size to solve the non-convex Problem (7.1). As the Lipschitz constant is not available for the outer problem, first-order methods need to rely on other strategies, such as:

- Gradient descent with manually adjusted fixed step sizes (Frecon et al., 2018; Ji et al., 2020). The main disadvantage of this technique is that it requires a careful tuning of the step size for each experiment. In addition to being potentially tedious, it does not lead to an automatic procedure.
- L-BFGS (as in Deledalle et al. 2014). L-BFGS is a quasi-Newton algorithm that exploits past iterates to approximate the Hessian and propose a better descent direc-

tion, which is combined with some line search (Nocedal and Wright, 2006). Yet, due to the approximate gradient computation, we observed that L-BFGS did not always converge.

- ADAM (Kingma and Ba, 2014). It turned out to be inappropriate to the present setting. ADAM was very sensitive to the initial step size and required a careful tuning for each experiment.

- Iteration specific step sizes obtained by line search (Pedregosa, 2016). While the approach from Pedregosa (2016) requires no tuning, we observed that it could diverge when close to the optimum. The adaptive step size strategy proposed in Algorithm 15, used in all the experiments, turned out to be robust and efficient across problems and datasets.

Remark 7.1 (Uniqueness). The solution of the lower problem may be non-unique, leading to a multi-valued regularization path $\lambda \mapsto \hat{\beta}^{(\lambda)}$ (Liu et al., 2020) and requiring tools such as *optimistic gradient* (Dempe et al., 2015, Chap. 3.8). Though it is not possible to ensure uniqueness in practice, we did not face experimental issues due to potential non-uniqueness. For the Lasso, this experimental observation can be theoretically justified (Tibshirani, 2013): when the design matrix is sampled from a continuous distribution, the solution of the Lasso is almost surely unique.

Remark 7.2 (Initialization). One advantage of the non-smooth case with the ℓ_1 norm is that one can find a good initialization point: there exists a value $\lambda_{\max}$ (see Table 5.1) such that the solution of Problem (6.2) vanishes for $\lambda \geq \lambda_{\max}$. Hence, a convenient and robust initialization value can be chosen as $e^{\lambda} = e^{\lambda_{\max}}/100$. This is in contrast with the smooth case, where finding a good initialization heuristic is hard: starting in flat zones can lead to poor performance for gradient-based methods (Pedregosa, 2016).

Remark 7.3 (Resolution of the bilevel problem). We do not have theoretical results proving that there is convergence towards a solution of Problem (5.2). From our experiment, the function that we are trying to minimize is often non-convex and we have to deal with the presence of local minima. In the case of smooth lower problems, Pedregosa (2016) proved that their proposed algorithm, *HOAG*, converges towards a stationnary point of $\mathcal{L}$. However, when considering a non-smooth lower problem, deriving a similar result seems to be challenging and would be a topic on its own. We then use the term *heuristic* to describe the proposed algorithm to *solve* the bilvel optimization problem. In practice, the results show that it is an efficient way to select hyperparameters for a given model but the theoretical results derived in Chapter 6 focused only on the hypergradient computation

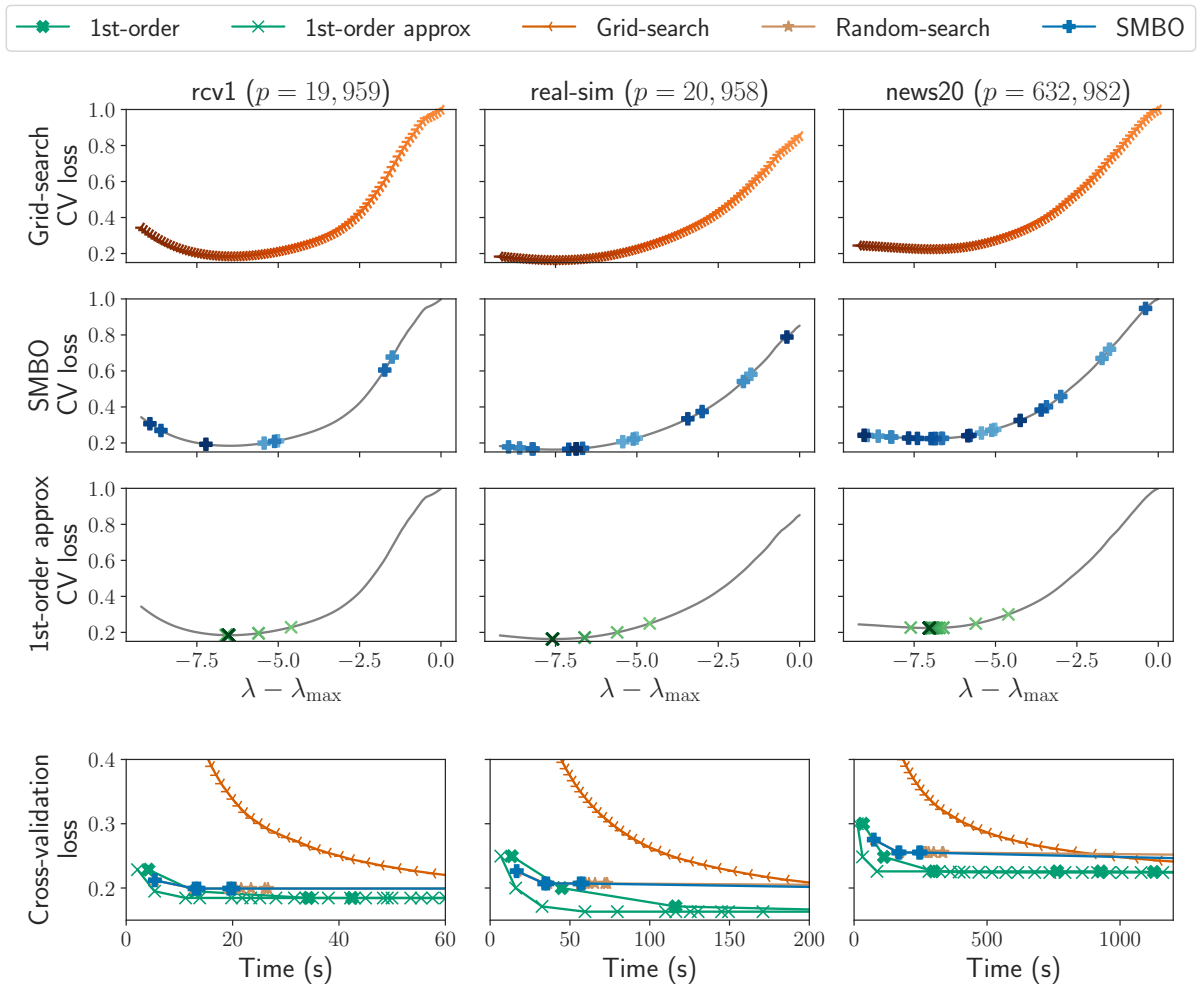

Figure 7.1 – **Lasso cross-validation, time comparison (1 hyperparameter).** Cross-validation loss as a function of λ (black line, top) and as a function of time (bottom).

step.

7.2 Hyperparameter selection for the Lasso

We start our experiments with selecting a single parameter for the Lasso. We pick a K-fold cross-validation (CV) loss as outer criterion[1]. The dataset (X, y) is partitioned into K held-out datasets $(X^{\text{train}_k}, y^{\text{train}_k}), (X^{\text{val}_k}, y^{\text{val}_k})$. The bilevel optimization problem then writes:

[1]In our experiments the default choice is $K = 5$.

$$\begin{aligned} &\arg\min_{\lambda \in \mathbb{R}} \mathcal{L}(\lambda) = \frac{1}{K} \sum_{k=1}^{K} \| y^{\text{val}_k} - X^{\text{val}_k} \hat{\beta}^{(\lambda,k)} \|_2^2 \\ &s.t.\ \hat{\beta}^{(\lambda,k)} \in \arg\min_{\beta \in \mathbb{R}^p} \tfrac{1}{2n} \left\| y^{\text{train}_k} - X^{\text{train}_k} \beta \right\|_2^2 + e^{\lambda} \|\beta\|_1 \ , \end{aligned} \tag{7.2}$$

Figure 7.1 represents the cross-validation loss in Lasso CV as a function of the regularization parameter λ (black curve, three top rows) and as a function of time (bottom). Each point corresponds to the evaluation of the cross-validation criterion for one λ value. The top rows show cross-validation loss as a function of λ, for the grid-search, the SMBO optimizer and the first-order method. The lightest crosses correspond to the first iterations of the algorithm and the darkest, to the last ones. For instance, Lasso grid-search starts to evaluate the cross-validation function with $\lambda = \lambda_{\max}$ and then decreases to $\lambda = \lambda_{\max} - \ln(10^4)$. On all the datasets, first-order methods are faster to find the optimal regularization parameter, requiring only 5 iterations.

7.3 Hyperparameter selection for the elastic net

The elastic net proposed by Zou (2006) interpolates between the ℓ_1 regularization and the ℓ_2 regularization using two hyperparameters: λ_1 and λ_2. Once again we use the cross-validation loss as the criterion to perform the selection of the hyperparameters. The bilvel optimization can be written:

$$\begin{aligned} &\arg\min_{\lambda=(\lambda_1,\lambda_2) \in \mathbb{R}^2} \mathcal{L}(\lambda) = \frac{1}{K} \sum_{k=1}^{K} \| y^{\text{val}_k} - X^{\text{val}_k} \hat{\beta}^{(\lambda,k)} \|_2^2 \\ &s.t.\ \hat{\beta}^{(\lambda,k)} \in \arg\min_{\beta \in \mathbb{R}^p} \tfrac{1}{2n} \left\| y^{\text{train}_k} - X^{\text{train}_k} \beta \right\|_2^2 + e^{\lambda_1} \|\beta\|_1 + \frac{e^{\lambda_2}}{2} \|\beta\|_2^2, \quad \forall k \in [K] \ , \end{aligned} \tag{7.3}$$

Figure 7.1 represents the cross-validation loss in Lasso CV as a function of the regularization parameter λ (black curve, three top rows) and as a function of time (bottom). Each point corresponds to the evaluation of the cross-validation criterion for one λ value. The top rows show cross-validation loss as a function of λ, for the grid-search, the SMBO optimizer and the first-order method. The lightest crosses correspond to the first iterations of the algorithm and the darkest, to the last ones. For instance, Lasso grid-search starts to evaluate the cross-validation function with $\lambda = \lambda_{\max}$ and then decreases to $\lambda = \lambda_{\max} - \ln(10^4)$. On all the datasets, first-order methods are faster to find the optimal regularization parameter, requiring only 5 iterations.

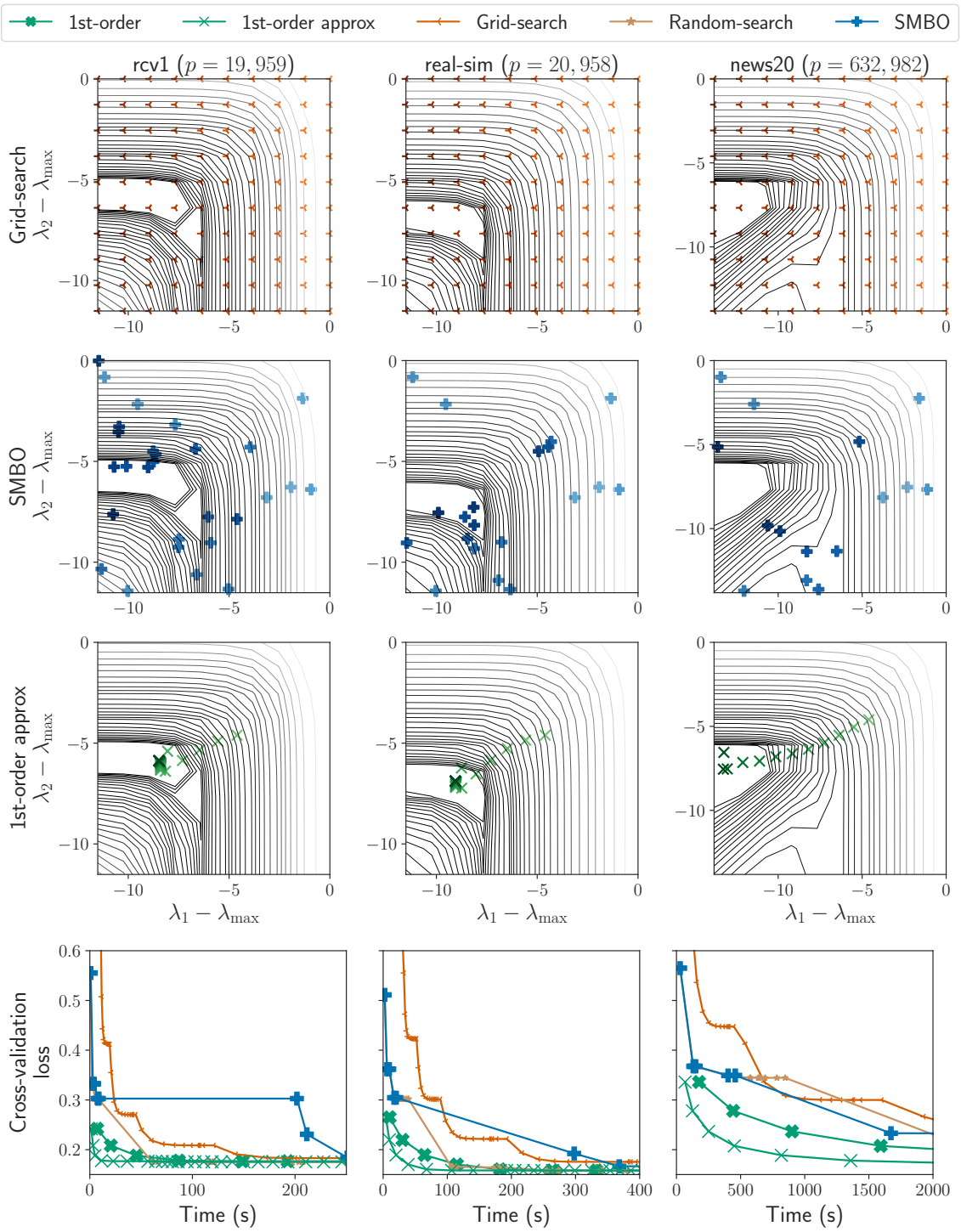

Figure 7.2 – **Elastic net cross-validation, time comparison (2 hyperparameters).** Level sets of the cross-validation loss (black lines, top) and cross-validation loss as a function of time (bottom) on *rcv1*, *real-sim* and *news20* datasets.

Figure 7.2 represents the level sets of the cross-validation loss for the elastic net (three top rows) and the cross-validation loss as a function of time (bottom). One can see that

after 5 iterations the SMBO algorithm (blue crosses) suddenly slows down (bottom) as the hyperparameter suggested by the algorithm leads to a costly optimization problem to solve, while first-order methods converge quickly as for Lasso CV. In the present context, lower problems are slower to solve for low values of the regularization parameters.

7.4 Hyperparameter selection for the multiclass sparse logistic regression

We now consider a multiclass classification problem with q classes. The design matrix is noted $X \in \mathbb{R}^{n \times p}$, and the target variable $y \in \{1, \dots, q\}^n$. We chose to use a one-versus-all model with q regularization parameters to illustrate the advantages of using a first order method to set a large number of hyperparameters. We use a binary cross-entropy for the lower loss:

$$\psi^k(\beta, \lambda_k; X, y) \triangleq -\frac{1}{n} \sum_{i=1}^{n} \left(\mathbb{1}_{y_i=k} \ln(\sigma(X_{i:}\beta)) + (1 - \mathbb{1}_{y_i=k}) \ln(1 - \sigma(X_{i:}\beta)) \right) + e^{\lambda_k} \|\beta\|_1 \ ,$$

and a multiclass cross-entropy for the outer criterion:

$$\mathcal{C}\left(\hat{\beta}^{(\lambda_1)}, \dots, \hat{\beta}^{(\lambda_q)}; X, y\right) \triangleq -\sum_{i=1}^{n} \sum_{k=1}^{q} \ln\left(\frac{e^{X_{i:}\hat{\beta}^{(\lambda_k)}}}{\sum_{l=1}^{q} e^{X_{i:}\hat{\beta}^{(\lambda_l)}}} \right) \mathbb{1}_{y_i=k} \ . \tag{7.4}$$

With a single train/test split, the bilevel problem to solve writes:

$$\begin{aligned} \underset{\lambda \triangleq (\lambda_1, \dots, \lambda_q) \in \mathbb{R}^q}{\arg\min} \quad & \mathcal{C}\left(\hat{\beta}^{(\lambda_1)}, \dots, \hat{\beta}^{(\lambda_q)}; X^{\text{test}}, y^{\text{test}}\right) \\ s.t. \ \hat{\beta}^{(\lambda_k)} & \in \underset{\beta \in \mathbb{R}^p}{\arg\min}\, \psi^k(\beta, \lambda_k; X^{\text{train}}, y^{\text{train}}) \quad \forall k \in [q] \ . \end{aligned} \tag{7.5}$$

Figure 7.3 represents the multiclass cross-entropy (top), the accuracy on the validation set (middle) and the accuracy on the test set (unseen data, bottom). When the number of hyperparameter is moderate ($q = 10$, on *mnist* and *usps*), the multiclass cross-entropy reached by SMBO and random techniques is as good as first-order techniques. This is expected and follows the same conclusion as Bergstra and Bengio (2012); Frazier (2018): when the number of hyperparameters is moderate, SMBO and random techniques can be used efficiently. However, when the number of hyperparameters increases (*rcv1*, $q = 53$ and *aloi*, $q = 1000$), the hyperparameter space is too large: zero-order solvers simply fail. On the contrary, first-order techniques manage to find hyperparameters leading to

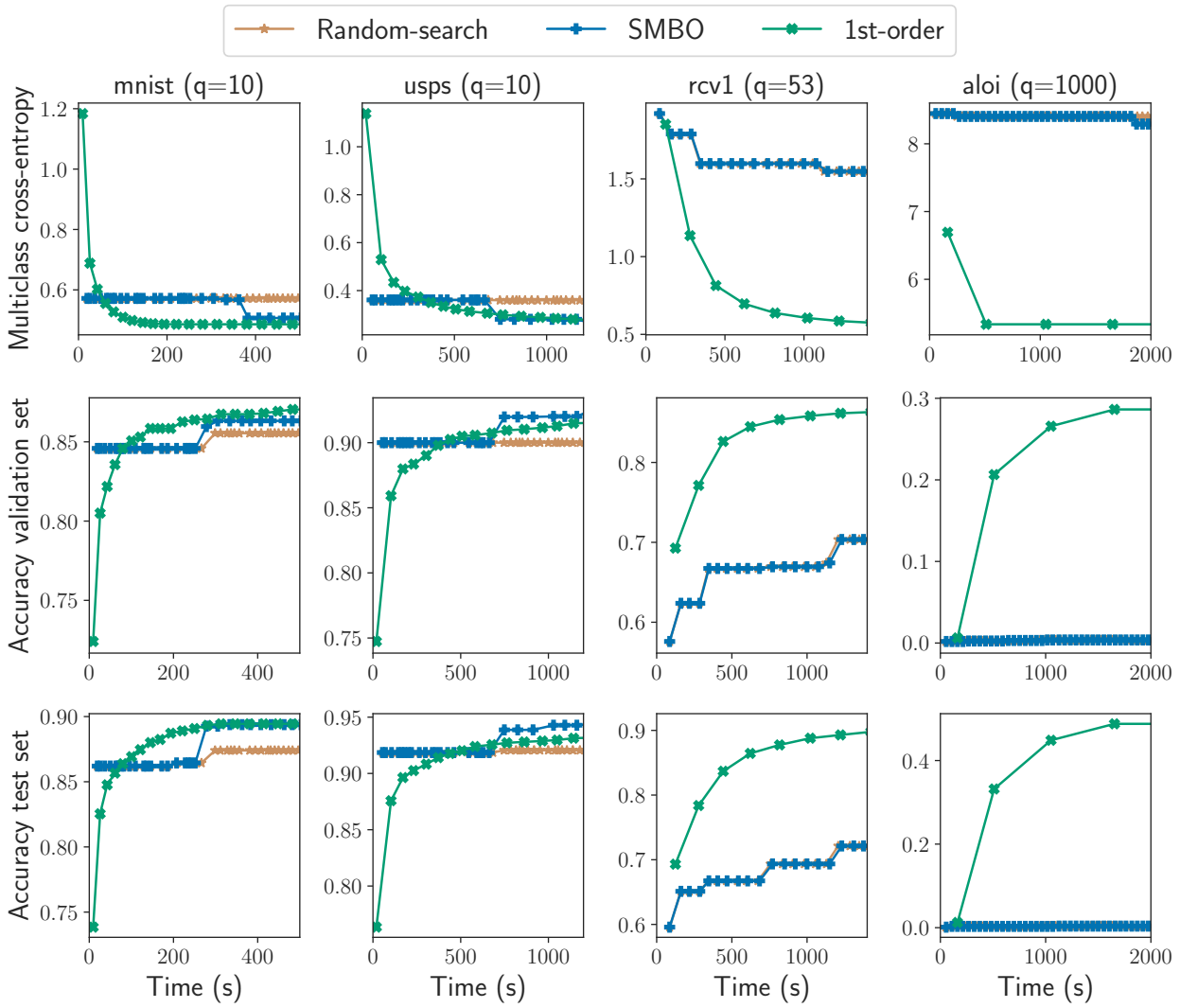

Figure 7.3 – **Multiclass sparse logistic regression held-out, time comparison (# classes hyperparameters).** Multiclass cross-entropy (top), accuracy on the validation set (middle), and accuracy on the test set (bottom) as a function of time on *mnist*, *usps* ($q = 10$ classes), *rcv1* ($q = 53$ classes), *aloi* ($q = 1000$ classes).

significantly better accuracy.

Remark 7.4 (Modelization). We believed at first that considering one hyperparameter per class could lead to an increased performance for the sparse logistic regression. However, practically we did not find an increase in the accuracy or an improvement in the minimization of the cross-entropy loss between considering the same regularization parameter for all the one-versus-all models or considering one for each class. Our experiment is still interesting when considering the resolution of the bilevel optimization of the form Problem (7.5) but the practical use of the model itself still has to be showed.

7.5 Hyperparameter selection for the weighted Lasso

Evaluating models on held-out or cross-validation data makes sense if the design is formed from random samples as it is often considered in supervised learning. However, this assumption does not hold for certain kinds of applications in signal or image processing. For these applications, the held-out loss cannot be used as the criterion for optimizing the hyperparameters of a given model. In this case, one may use a proxy of the prediction risk, like the Stein Unbiased Risk Estimation (SURE, Stein (1981)). The SURE is an unbiased estimator of the prediction risk under weak differentiable conditions. The drawback of this criterion is that it requires the knowledge of the variance of the noise (σ^2). The SURE is defined as follows: $\text{SURE}(\lambda) = \|y - X\hat{\beta}^{(\lambda)}\|^2 - n\sigma^2 + 2\sigma^2 \text{dof}(\hat{\beta}^{(\lambda)})$, where the degrees of freedom (dof Efron 1986) is defined as $\text{dof}(\hat{\beta}^{(\lambda)}) = \sum_{i=1}^{n} \text{cov}(y_i, (X\hat{\beta}^{(\lambda)})_i)/\sigma^2$. The dof can be seen as a measure of the complexity of the model, for instance for the Lasso $\text{dof}(\hat{\beta}^{(\lambda)}) = |\hat{\mathcal{S}}|$, see Zou et al. (2007). The SURE can thus be seen as a criterion trading data-fidelity against model complexity. However, the dof is not differentiable (not even continuous in the Lasso case), yet it is possible to construct a weakly differentiable approximation of it based on Finite Differences Monte-Carlo (see Deledalle et al. 2014 for full details), with $\epsilon > 0$ and $\delta \sim \mathcal{N}(0, \text{Id}_n)$:

$$\text{dof}_{\text{FDMC}}(y, \lambda, \delta, \epsilon) = \tfrac{1}{\epsilon}\langle X\hat{\beta}^{(\lambda)}(y + \epsilon\delta) - X\hat{\beta}^{(\lambda)}(y), \delta\rangle \ .$$

In order to go one step further in the validation of our proposed algorithm, we considered the weighted Lasso model. The weighted Lasso (wLasso, Zou 2006) has p hyperparameters and was introduced to reduce the bias of the Lasso. Its underlying optimization problem was given in Problem (6.18). However setting the p hyperparameters is impossible with grid-search.

We use the smooth approximation of the SURE in the bilevel optimization problem to find the best hyperparameters of the wLasso. The bilevel optimization problem then reads:

$$\begin{aligned} &\arg\min_{\lambda \in \mathbb{R}^p} \|y - X\hat{\beta}^{(\lambda)}\|^2 + 2\sigma^2 \text{dof}_{\text{FDMC}}(y, \lambda, \delta, \epsilon) \\ &s.t.\ \hat{\beta}^{(\lambda)}(y) \in \arg\min_{\beta \in \mathbb{R}^p} \tfrac{1}{2n}\|y - X\beta\|_2^2 + \sum_{j=1}^{p} e_j^{\lambda}|\beta_j| \\ &\hat{\beta}^{(\lambda)}(y + \epsilon\delta) \in \arg\min_{\beta \in \mathbb{R}^p} \tfrac{1}{2n}\|y + \epsilon\delta - X\beta\|_2^2 + \sum_{j=1}^{p} e_j^{\lambda}|\beta_j| \end{aligned} \tag{7.6}$$

Note that solving this problem requires the computation of two (instead of one for the held-out loss) Jacobians *w.r.t.* λ of the solution $\hat{\beta}^{(\lambda)}$ at the points y and $y + \epsilon\delta$.

To compare the estimation performance of the Lasso and the wLasso with automatic hyperparameter selection, we used as a metric the (normalized) Mean Squared Error (MSE) defined as MSE $\triangleq \|\hat{\beta} - \beta^*\|^2 / \|\beta^*\|^2$. The entries of the design matrix $X \in \mathbb{R}^{n \times p}$ are i.i.d. random Gaussian variables $\mathcal{N}(0, 1)$. The number of rows is fixed to $n = 100$. Then, we generated β^* with 5 non-zero coefficients equals to 1. The vector y was computed by adding to $X\beta^*$ additive Gaussian noise controlled by the Signal-to-Noise Ratio: SNR $\triangleq \|X\beta^*\| / \|y - X\beta^*\|$ (here SNR $= 3$). Following Deledalle et al. (2014), we set $\epsilon = 2\sigma / n^{0.3}$. We varied the number of features p between 200 and 10,000 on a linear grid of size 10. For a fixed number of features, we performed 50 repetitions and each point of the curves represents the average of these repetitions. We compared the computation time of different methods used to compute the hypergradient: the Forward mode, the Backward mode and the implicit differentiation solved with a conjugate gradient algorithm and the Implicit forward mode where the linear system is solved via a coordinate descent algorithm.

Figure 7.4 shows the estimation MSE and the running time of the different methods to obtain the hyperparameter values as a function of the number of features used to simulate the data. Problem (7.6) is not convex for the weighted Lasso and a descent algorithm like ours can be trapped in local minima, crucially depending on the starting point λ_{init}. To alleviate this problem, we introduced a regularized version of Problem (5.2):

$$\begin{aligned} \underset{\lambda \in \mathbb{R}}{\arg\min} \quad & \mathcal{C}\left(\hat{\beta}^{(\lambda)}\right) + \gamma \sum_{j}^{p} \lambda_j^2 \\ & s.t.\ \hat{\beta}^{(\lambda)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \triangleq \Phi(\beta, \lambda) \ . \end{aligned} \tag{7.7}$$

The solution obtained by solving Equation (7.7) is then used as the initialization $\lambda^{(0)}$ for our algorithm. In this experiment the regularization term is constant $\gamma = C(\beta^{(\lambda_{\max})})/10$. We see in Figure 7.4 that the weighted Lasso gives a lower MSE than the Lasso and allows for a better recovery of β^*. This experiment shows that the amount of time needed to obtain the vector of hyperparameters of the weighted Lasso via our algorithm is in the same range as for obtaining the unique hyperparameter of the Lasso problem. It also shows that our proposed method is much faster than the *naive* way of computing the Jacobian using forward or backward iterative differentiation. The implicit differentiation method stays competitive for the wLasso due to the small support of the solution and

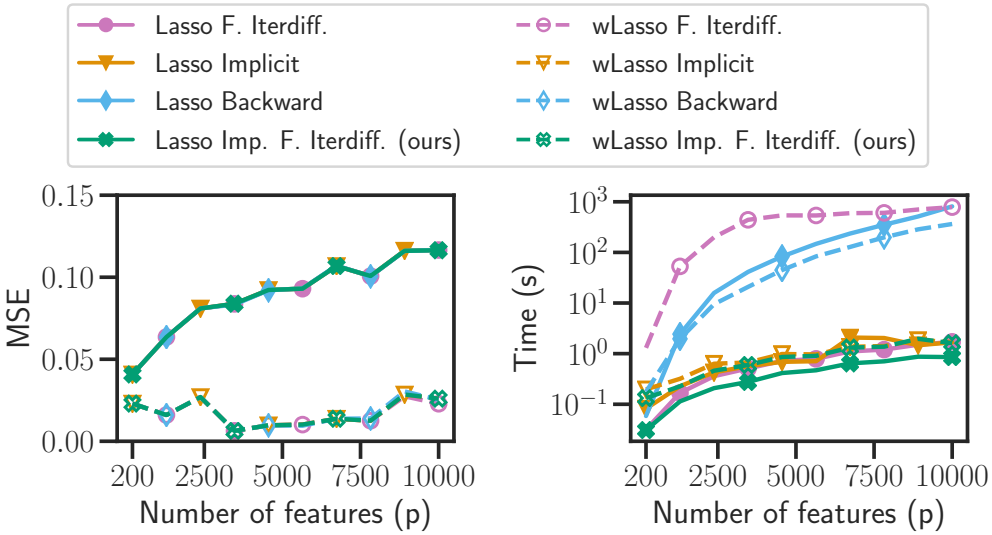

Figure 7.4 – **Lasso vs wLasso.** Estimation Mean Squared Error (left) and running (right) of competitors as a function of the number of features for the weighted Lasso and Lasso models.

hence a small matrix to inverse. A maximum running time threshold was used for this experiment checking the running time at each line-search iteration, explaining why the forward differentiation and backward differentiation of the wLasso does not explode in time on Figure 7.4.

Remark 7.5 (Limits). This last experiment on the weighted Lasso revealed some of the limits of our work. The objective function of Problem (7.6) is probably non-convex and very difficult to optimize in comparison to the other examples in smaller dimensions. The results obtained in Figure 7.4 were dependent on the initialization point hence the introduction of the heuristic regularization Equation (7.7) to find a starting point. This experiment on the wLasso is exploratory and shows the difficulty to set p hyperparameters when $p \gg n$.

Part III

Estimating cells proportions with developed tools

8 VALIDATION OF OUR METHOD

Contents

In this chapter, we use the algorithms described in the previous chapters to propose a new method to estimate the proportions of cells inside a tumor. In Section 8.1, we start with the description of the estimator used, namely the ε-SVR with additional constraints and show how the automatic hyperparameters selection can be applied to it. Then we validate the use of our proposed method by comparing it to Cibersort and the Simplex Ordinary Least Squares on different types of data: microarray (Section 8.2) and RNA-seq (Section 8.3). Finally, we show how this method was used to estimate the cells proportions of 15 different types of cells to improve the risk of relapse estimation for patients suffering from breast cancer in Section 8.4.

8.1 Simplex ε-SVR

The start of this work was to propose a method that could improve the resolution of the inverse problem related to the proportions of cells inside a tumor as described in Chapter 1. As a result of the previous chapters, we now propose a new estimator with an automatic selection of its related hyperparameters. A first natural candidate would be the constrained ν-SVR described in Chapter 4. One of the difficulties that comes with using this estimator is the choice of the two hyperparameters $\nu \in [0, 1]$ and $C > 0$. Avoiding a grid search and relying on automatic differentiation techniques as proposed in Chapter 7 would greatly improve the usability of this estimator. However, the ν-SVR underlying

dual optimization problem involves a non-separable equality constraint. This constraint prevents us from using the results obtained in Chapter 6 to set the hyperparameters using a first order method. The results on the support identification property and the convergence of the Jacobian is only obtained with the assumption that the non-smooth term is separable. For these reasons, we consider the Simplex ε-SVR without bias instead which writes

$$\begin{aligned} \hat{\beta} \in \arg\min_{\beta \in \mathbb{R}^p} \quad & \frac{1}{2} \|\beta\|^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i + \xi_i^* \\ s.t. \quad & y_i - X_{i:}\beta \leq \varepsilon + \xi_i, \\ & X_{i:}\beta - y_i \leq \varepsilon + \xi_i^*, \\ & \xi_i \geq 0, \xi_i^* \geq 0, \\ & \beta_j \geq 0, \sum_{i=1}^{p} \beta_j = 1 \ . \end{aligned} \tag{8.1}$$

The ε-SVR and the ν-SVR are equivalent in the sense that if the ν-SVR has the solution $\beta^\star$ and $\varepsilon^\star$ then computing the solution of the ε-SVR with the same value of C and setting $\varepsilon = \varepsilon^\star$ leads to the same solution $\beta^\star$ (Schölkopf and Smola, 2002, Prop. 9.3). The choice of the ν-SVR estimator at first can be explained by the fact that the hyperparameter $\nu \in [0, 1]$ has an easy interpretation; it roughly represents the proportions of support vectors (Schölkopf et al., 1999) and is easier to tune. Since the problem of hyperparameters selection is more simple and straightforward with the first order methods, these arguments on the hyperparameters ν do not hold anymore. The separability of the ε-SVR dual problem motivates the choice of the ε-SVR for the rest of this chapter.

Deriving the dual problem of Problem (8.1) is very similar to the work done in Chapter 4. The dual optimization problem is as follows

$$\begin{aligned} \min_{\alpha, \alpha^*, \gamma, \mu} \frac{1}{2} \Big[& (\alpha - \alpha^*)^\top X X^\top (\alpha - \alpha^*) + \gamma^\top \gamma + p\mu^2 \\ & + 2 \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \gamma^\top X_{i:} + 2\mu \sum_{i=1}^{n} \mathbb{1}^\top X_{i:} + 2\mu \mathbb{1}^\top \gamma \Big] \\ & + \varepsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) - \sum_{i=1}^{n} y_i (\alpha_i - \alpha_i^*) - \mu \\ s.t. \ & 0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{n} \\ & \gamma_j \geq 0 \ . \end{aligned} \tag{8.2}$$

The primal-dual relation at the optimum is given by the following formula:

$$\beta = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) X_{i:} + \gamma + \mu \mathbb{1} \ . \tag{8.3}$$

Since the non-smooth functions are separable, we can use the cyclic proximal coordinate descent algorithm to solve the dual optimization problem with theoretical guarantees (Tseng and Yun, 2009) to converge towards a solution of Problem (8.2). The cyclic coordinate descent algorithm obtained to solve Problem (8.2) is given in Algorithm 16. As suggested in Ho and Lin (2012) for the classical ε-SVR without constraints, we can avoid storing the matrix $XX^\top \in \mathbb{R}^{n \times n}$ which appears in the quadratic objective function of the dual problem. To do so, the primal variables are updated at each iterations of the cyclic coordinate descent based on the dual variables using Equation (8.3). In addition, keeping β updated at each iteration reduces the computation cost of the gradient required for the coordinate descent updates.

For example, the gradient of the objective function of Problem (8.2) denoted f *w.r.t.* to α is given by

$$\nabla_\alpha f(\alpha, \alpha^*, \gamma, \mu) = X\beta + \varepsilon \mathbb{1} - y \ . \tag{8.4}$$

Similar *tricks* can be used in the variables resulting from the additional linear constraints in the primal. The gradient of f with respect to γ writes

$$\nabla_\gamma f(\alpha, \alpha^*, \gamma, \mu) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) X_{i:} + \gamma + \mu \mathbb{1} = \beta \ . \tag{8.5}$$

Identically, we have that

$$\nabla_\mu f(\alpha, \alpha^*, \gamma, \mu) = \mathbb{1}^\top \beta - 1 \ . \tag{8.6}$$

To select the hyperparameters $\varepsilon > 0$ and $C > 0$, we use the implicit differentiation algorithm Algorithm 15 proposed in Chapter 7. The Mean Squared Error (MSE) was chosen as the outer criterion for the hyperparameters selection, obtained as

$$\frac{1}{2n} \|y - X\hat{\beta}\|^2 \ , \tag{8.7}$$

with $\hat{\beta} \in \mathbb{R}^p$ being a solution of Problem (8.1). The updates to compute the Jacobian

Algorithm 16 SIMPLEX ε-SVR

input : $X \in \mathbb{R}^{n \times p}, y \in \mathbb{R}^n, \varepsilon \in \mathbb{R}, C \in \mathbb{R}, n_{\text{iter}} \in \mathbb{N}$,
$\alpha = 0, \alpha^* = 0, \gamma = 0, \mu = 0$ // potentially warm started
$\beta = 0$ // Primal variables for cheap updates
// Perform cyclic coordinate descent
for $k = 1, \dots, n_{iter}$ **do**
  for $i = 1, \dots, n$ **do**
    // Update in the block α
    $F = X_{i:}\beta + \varepsilon - y_i$ // Gradient w.r.t. α_i
    $\alpha_i^{\text{old}} = \alpha_i$
    $\alpha_i = \mathcal{P}_{[0,C]}\left(\alpha_i - \frac{1}{\|X_{i:}\|^2} F\right)$
    $\beta = \beta + (\alpha_i - \alpha_i^{old}) X_{i:}$ // Update primal variables
  for $i = 1, \dots, n$ **do**
    // Update in the block α^*
    $F = -X_{i:}\beta + \varepsilon + y_i$ // Gradient w.r.t. α_i^*
    $(\alpha_i^*)^{\text{old}} = \alpha_i^*$
    $\alpha_i^* = \mathcal{P}_{[0,C]}\left(\alpha_i^* - \frac{1}{\|X_{i:}\|^2} F\right)$
    $\beta = \beta - (\alpha_i - \alpha_i^{old}) X_{i:}$ // Update primal variables
  for $j = 1, \dots, p$ **do**
    // Update in the block γ
    $F = \beta_j$ // Gradient w.r.t. γ_j
    $\gamma_j^{\text{old}} = \gamma_j$
    $\gamma_j = \max(\gamma_j - F, 0)$
    $\beta_j = \beta_j + (\gamma_j - \gamma_j^{\text{old}})$ // Update primal variables
  // Update of the variable μ
  $F = \mathbb{1}^\top \beta - 1$ // Gradient w.r.t. μ
  $\mu^{\text{old}} = \mu$
  $\mu = \mu - \frac{1}{p} F$
  $\beta = \beta + (\mu - \mu^{\text{old}})\mathbb{1}$ // Update primal variables
return β

involve the differentiation of the indicator function $\iota_{[0,C]}$ and the non-smooth function $\max(0, \cdot)$.

In Figure 8.1, we illustrate the difference between using a grid-search method to select the hyperparameters C and ε and the first order method. The first order method converges towards the minimum of the MSE on this toy example and only requires an initial points. The initialization has to be done so that the gradient is not zero, this happens when the parameter C is chosen too small or ε too large. One can see that the grid search *misses* the minimum of the MSE requiring a finer grid to select the best hyperparameters.

Equipped with these tools we can now turn towards applying these methods on real

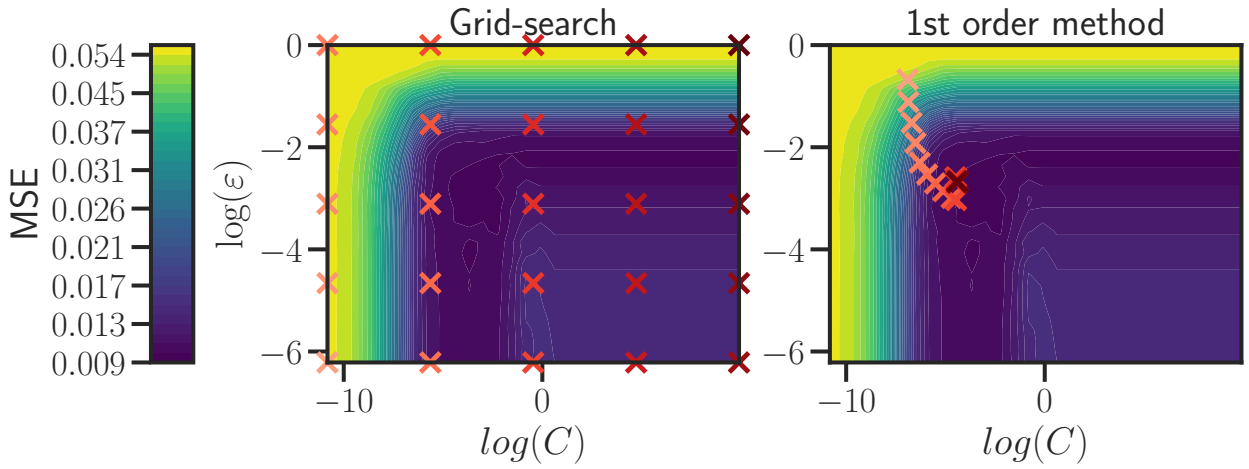

Figure 8.1 – **Illustration of first order method for hyperparameter selection.** Level set of Mean Squared Error as a function of C and ε for the grid-search method and the first order method based on implicit differentiation. The crosses represent 25 evaluations of the Mean Squared Error by each method.

datasets where the goal is to estimate proportions of cells inside a tumor.

8.2 Validation on microarray data

First, we will use microarray datasets as validation for our new method. On this type of dataset, the state-of-the-art method is called Cibersort (Newman et al., 2015) and is based on the SVR estimator as described in Chapter 1. The other method that will be considered is the Simplex Ordinary Least Squares (SOLS), which the OLS estimator with the positivity and sum-to-one constraints proposed for this application by Gong et al. (2011).

The validation process requires to have access to the ground truth proportions of cells present inside a tumor. Unfortunately these datasets are rare and a first validation step consists in using pure transcriptomes from different types of cells and create a semi-simulated mixture y by using a linear combination of these pure transcriptomes. We will use the dataset introduced by Abbas et al. (2009) that can be found on the Gene Expression Omnibus (GEO) (Edgar et al., 2002) under the accession name *GSE11103*. In this dataset, there are four different types of cells (Jurkat, IM-9, Raji and THP-1) and 12 mixed samples with known cells proportions.

As a first experiment, we wanted to assess the robustness of the estimator as it was one of the main claim about the SVR estimator of Cibersort. We simulated 100 different mixtures with known proportions of cells in each of the samples. We added log-Gaussian

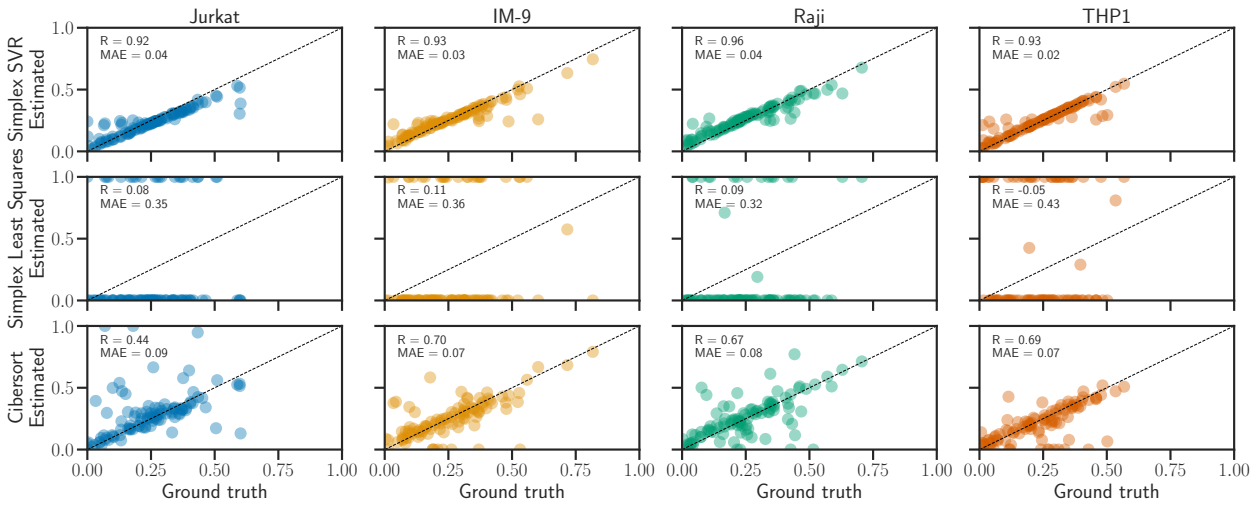

Figure 8.2 – **Robustness to noise.** Scatter plot of the estimated proportions of cells for four different types of cells (Jurkat, IM-9, Raji and THP1) as a function of the true proportions. The dashed black line represents the line $x = y$ which would be perfect estimation. We compared three different estimators the Simplex Ordinary Least Squares, the one proposed by Cibersort and our proposed method (Simplex SVR) with automatic hyperparameters selection.

with $\sigma = 9$ to the data as proposed by Newman et al. (2015) and compared the estimation performance of the SOLS, SVR and our proposed Simplex SVR estimator with automatic hyperparameters selection. The initial point for the selection of C and ε was set at $C_0 = 1$ and $\varepsilon_0 = 0.1$ with 10 outer iterations for the gradient descent to find the *best* hyperparameters. The step size for the gradient descent was chosen as suggested in Algorithm 15 *i.e.*, $\gamma = \frac{1}{\|\nabla_\lambda \mathcal{L}(\lambda)\|}$. The data was normalized following the method described in Nadel et al. (2021): each row of the concatenated matrix composed of $X \in \mathbb{R}^{n \times p}$ and $y \in \mathbb{R}^n$ is scaled to have coefficients between 0 and 1. It is done by taking the maximum and the minimum of each row and then performing the classical *min-max* scaler on a single row. The interpretation of this normalization is that each gene has now a comparable influence on the linear inverse problems, it does not depend on the magnitude of expression anymore.

Figure 8.2 illustrates the benefits of our method. It represents the scatter plots of the estimated proportions as a function of the true proportions for the four different cell types. If the estimation was perfect, all the points would lie on the line $x = y$ depicted as a dashed black line on the figure. To compare the different estimators we rely on three different measures:

- The correlation coefficient (R) between the estimated and the true coefficients.

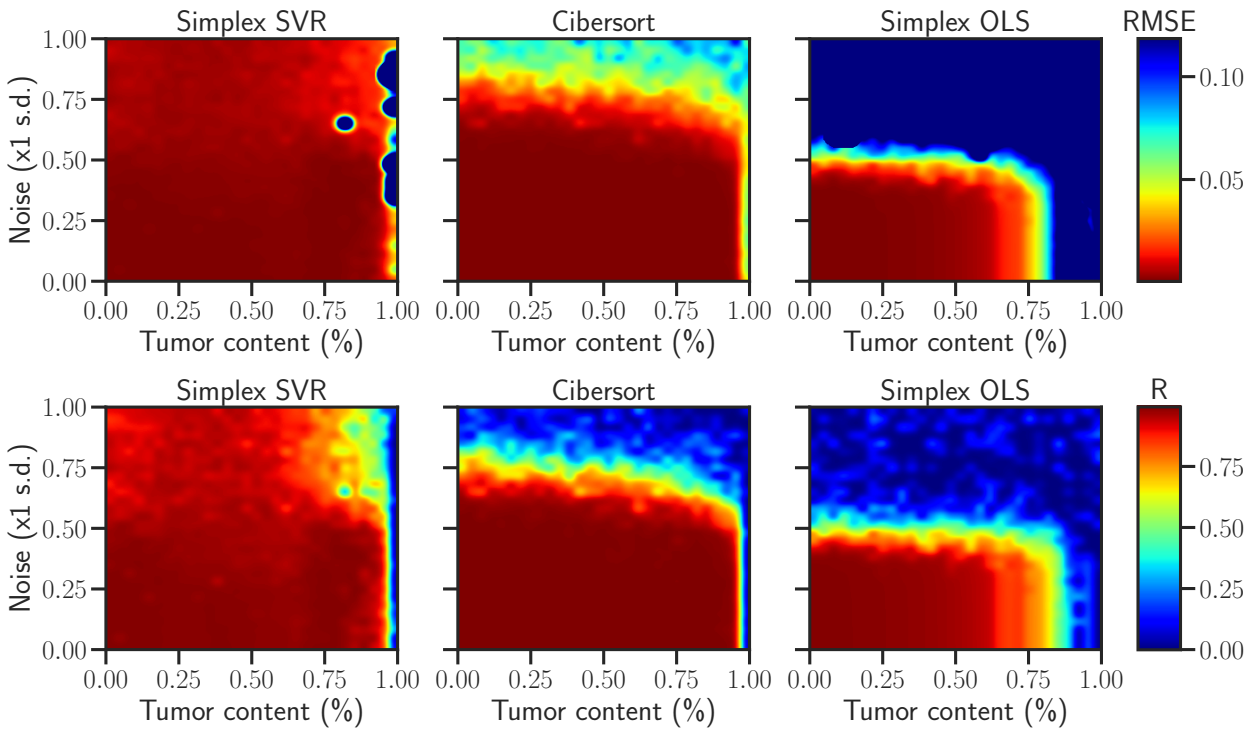

Figure 8.3 – **Robustness to noise and tumor content.** Heatmap representing the Root Mean Squared Error (RMSE) or the correlation coefficient (R) between the true proportions of cells and the estimated ones as a function of the tumor content percentage (x-axis) and the level of noise (y-axis). We compared three different estimator the Simplex Support Vector Regression, Cibersort and the Simplex Ordinary Least Squares.

- The Maximum Absolute Error (MAE) which is obtained as:

$$\text{MAE} = \frac{1}{p}\sum_{i=1}^{p} |\hat{\beta}_j - \beta_j^\star| \ ,$$

  where $\hat{\beta} \in \mathbb{R}^p$ are the estimated proportions and $\beta^\star$ the true proportions

- The Root Mean Squared Error given by:

$$\text{RMSE} = \sqrt{\frac{1}{p}\|\hat{\beta} - \beta^\star\|^2} \ .$$

It can be noticed that in the presence of heavy-tailed noise, the estimator based on OLS performs very poorly as it is sensitive to outliers. Our proposed method shows to be more robust than the method based on the classical SVR estimator with a better correlation coefficient over the 4 different types of cells and an improvement on the MAE.

To further the validation, an experiment from Newman et al. (2015) was replicated. It aims at benchmarking different estimators for robustness to noise and unknown content. Different level of noise was added to the dataset *GSE11103*. The noise follows a log-Gaussian distribution $2^{\mathcal{N}(0,\sigma^2)}$ where σ is chosen as a percentage of $\sigma_{\max} = 11.6$ as given in Newman et al. (2015). We used 30 different percentage on a linear scale between 0 and 1. To simulate the presence of tumor content mixed with the immune cells, we used a tumor cell line found under the accession name GSM269529 and GSM269530. We added a percentage of tumor content by taking a convex combination between the real mixture and the tumor content. We increased the coefficient related to the tumor content from 0 to 1 taking 30 different coefficients, equally spaced on a linear scale. This simulation process lead to 900 different datasets, for each of them the ground-truth proportions were known and we used 25 different mixtures for which the estimation process has to be performed.

Figure 8.3 represents the heatmaps of the RMSE (top) and the correlation coefficient (R, bottom) calculated between the true proportions and the estimated ones for the 900 datasets. The increased tumor content is represented on the x-axis and the increased level of noise on the y-axis. As it can be seen, our proposed method allows a better estimation when the level of noise increases. The SOLS estimator does not seem to be able to perform well when the level of noise is above 50% of $\sigma_{\max}$; for Cibersort it is 80%. Our estimator is able to give good estimation even when the level of noise is above the 80% of $\sigma_{\max}$ threshold, this is certainly due to the two facts that have been taken into account in the estimation process: the natural constraints of being a vector of proportions and the hyperparameters selection process.

8.3 Validation on RNAseq/sc-RNAseq data

Other methods based on high throughput sequencing, slowly replaced the microarrays even if this type of data stays largely available in public repositories. The new sequencing method, called RNA sequencing (RNA-seq), has several advantages over the microarrays. One of them is the ability of the RNA-seq to detect genes expressed at low level or very high level, unlike the microarrays which lack sensitivity (Wang et al., 2009). A second advantage is that the noise level inside the data is significantly reduced in the RNA-seq data in comparison to microarrays (Zheng, 2019). Moreover, the cost of performing one RNA-seq greatly decreased in the past decade and RNA-seq has become the preferred technology to obtain the transcriptome of a cell or a tumor.

A more recent technique, named single cell RNA sequencing (scRNA-seq), is of interest

for our problem of quantifying cells inside a tumor. This new technology provides the transcriptome of a single cell in place of the transcriptome of a bulk of cells. This type of sequencing method leads to a better description of the immune cells that we wish to quantify. It allows the construction of signature matrices with a better precision because one can have access to the cells of interest in their environment (like in a tumor for example).

Validation on blood samples. Even if the start of this work was the development of a method for microarray data, we present in this section the developed tools applied on RNA-seq and scRNA-seq data. Our first validation step involves the quantification of immune cells in blood samples. Typically, the quantification of cells is easier than in a tumor for several biological reasons. We used the datasets available on the Gene Expression Omnibus repository under the accession number *GSE127813*. It is also available on the website `https://cibersortx.stanford.edu/`. In this study, five types of cells were quantified using flow cytometry for 12 blood samples along side with the RNA-seq transcriptomes. We used the signature matrix provided on the `https://cibersortx.stanford.edu/` website built from scRNA-seq.

Figure 8.4 presents the scatter plots of the estimated proportions of immune cells for our proposed method (Simplex SVR) and Cibersort. As it can be seen, the MAE is lower for Simplex SVR and the estimation is more accurate than Cibersort. The correlation coefficient (R) is often used in the biostatistic literature to compare methods for the estimation of cells, here we have an example that shows a method that has a better precision in terms of absolute error (MAE) but the linear correlation between the estimated proportions and the ground-truth proportions is lower for our estimator. The improvement is small but it can be seen that for cells present in low proportions, our algorithm allows a better quantification in comparison to Cibersort on this dataset.

Remark 8.1. The estimation was performed using the batch correction method proposed by the team that developped Cibersort in Newman et al. (2019). The corrected signature matrix was obtained using `Cibersortx` with the S-mode batch correction.

Remark 8.2. We are aware that the comparison for RNA-seq data should be extended to other methods. The recent literature on this topic has thrived with a large number of proposed algorithms: Scaden (Menden et al., 2020), Cibersortx (Newman et al., 2019), Gedit (Nadel et al., 2021), DeconRNAseq (Gong and Szustakowski, 2013), Music (Wang et al., 2019) is only a short list of available tools to perform the estimation of cells inside a tumor. Here, our main claim is not the outperformance of all these methods but to

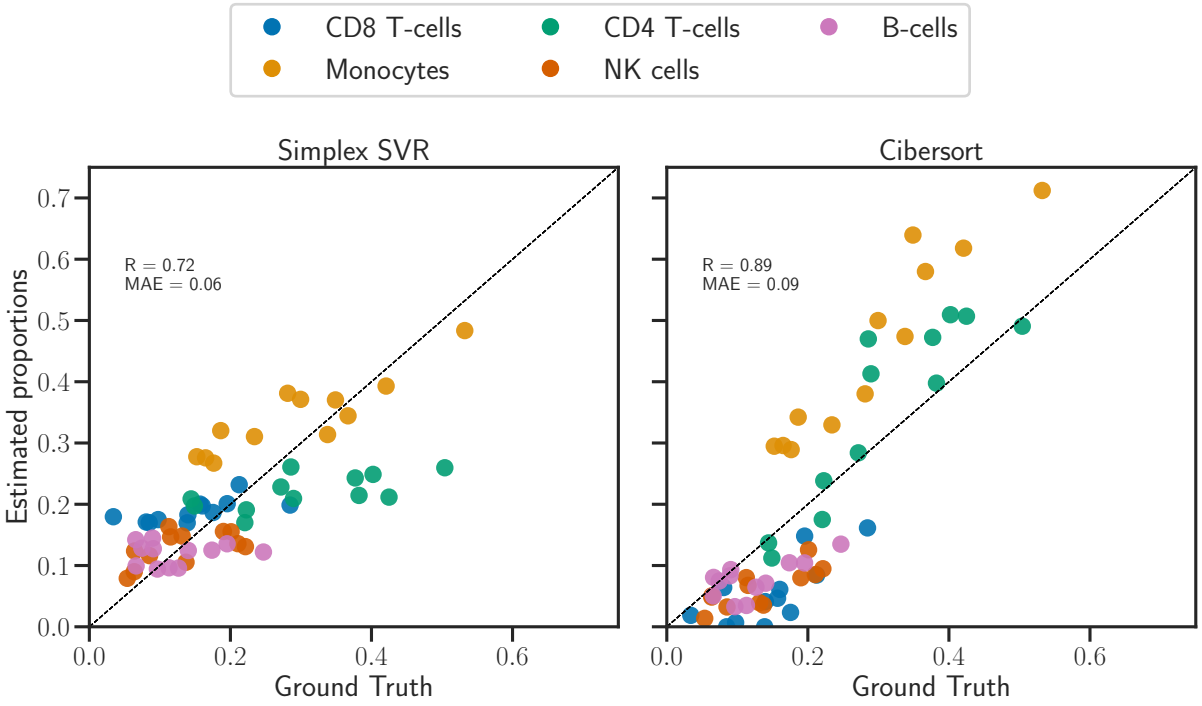

Figure 8.4 – Scatter plot of the estimated cells proportions obtained by the Simplex Support Vector Regression method (left) and Cibersort (right) as a function of the ground-truth proportions on the PBMC dataset (GSE127813). The estimation process was performed using the S-mode batch correction method implemented in `Cibersortx`. The correlation coefficient (R) and the Maximum Absolute Error (MAE) are two measures of performance to compare the two methods.

show that there is room for improvement in taking mathematical theory into account for a specific application.

The promising results on blood samples will have to be validated on tumor samples for which we have access to the RNA-seq transcriptome of the tumors and the ground-truth proportions of immune cells of interest. This is a current project with a cancer institute where we designed a study to validate the estimation methods on samples coming from patients suffering from a colon cancer. For each patient, we will have access to

- the transcriptomes of single cells of interest to construct the signature matrix via scRNA-seq,
- the transcriptomes of the tumors on which the estimation process will be performed (RNA-seq),
- the proportions of cells measured by cytometry which will be considered as the

ground-truth.

This study will allow us to have the final stage of validation for our proposed method and to see how it compares to the large number of tools mentioned in Remark 8.2.

We want to mention that when using our estimator on mixture reconstituted from scRNA-seq data as studied in Newman et al. (2019), the estimation performance of our proposed method was worse than Cibersort. We tested the two methods on the datasets coming from head and neck squamous cell carcinomas (Puram et al., 2017) which includes18 patients and the melanoma datasets (Tirosh et al., 2016) with 19 patients. The bad performance on these types of mixtures could be explained by different facts: the construction of the signature matrix, the normalization techniques used on the signature and on the mixture matrices or the selection of hyperparameters that fails at finding a good estimator in terms of estimation performance.

Nevertheless, the end-point application of this methodology would be to build signature matrices from scRNA-seq for each tumor types as we assume that the transcriptome of an immune cells will not be too different from one patient to another when they are located in the same organ. Then, a RNA-seq would be performed for each patient and the estimation process would be used on this RNA-seq transcriptome. The price of a RNA-seq remains significantly lower than scRNA-seq, this has to be taken into account to imagine that the whole process could be used for clinical purposes and not only for research. In the end, the goal is to estimate the proportions of cells coming from bulk RNA-seq.

8.4 Clinical application

This section is dedicated to showing how the estimation of cells proportions can be used for clinical purposes. We want to draw the attention of the readers towards the fact that the following results were obtained without the automatic hyperparameters selection described in Chapter 7 and the hyperparameters were chosen using grid-search. This work has lead to an article that is currently under review.

From public datasets repositories, we had access to 2800 microarray transcriptomes of patients suffering from breast cancer. Our goal was to study the impact of different immune cells on the progression of the cancer and see if the estimation of cells proportions within the tumor could bring any valuable information. We estimated the cells proportions of 15 different immune cells that we will not be detailed here but their names are given at the top of Figure 8.5.

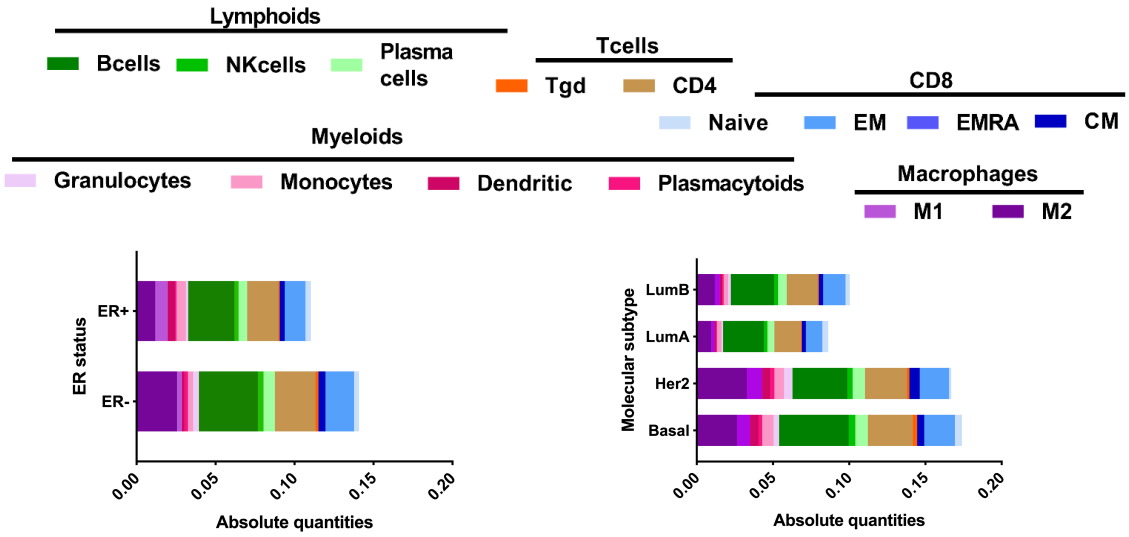

Figure 8.5 – **Estimated cells proportions of 15 immune cells.** Bar graph representing the average estimation of cells proportions in the different groups of ER status (left) and PAM50 (right).

An important element for the following is that there exists different classification of breast cancer that will give a direction for the oncologists to choose a treatment. The pam50 classification (Parker et al., 2009) divides the patients in four different subgroups: the Luminal A, Luminal B, Basal and Her2. Typically, the Luminal B and Basal are the tumors with the worst prognosis and the Her2 tumors are treated by hormone therapy. Another classification exists and split the patients in two groups: ER+ or ER- based on the estrogen receptor.

Figure 8.5 represents the average cells proportions estimated within each of the different tumor subgroups. The total of the immune cells does not sum-up to one because the tumor content was taken into account and not represented on the figure. It can be seen that the HER2 and Basal tumors are generally more infiltrated with immune cells than the Luminal tumors. These results corroborate the existing literature (Gao et al., 2020) on the infiltration of immune cells in breast cancer subtypes. An important observation is that within each subtypes there were differences between patients with the same tumor type. This fact has driven us to consider the estimation of cells proportions as a valuable piece of information to estimate the prognosis of each patient which is normally based only the classification and some clinical variables.

Then we used a Cox regression model (Cox, 1972) including clinical variables, the pam50 classification and the information about the different immune cells proportions estimated via the Simplex SVR estimator. From this Cox model, we could estimate for each patient

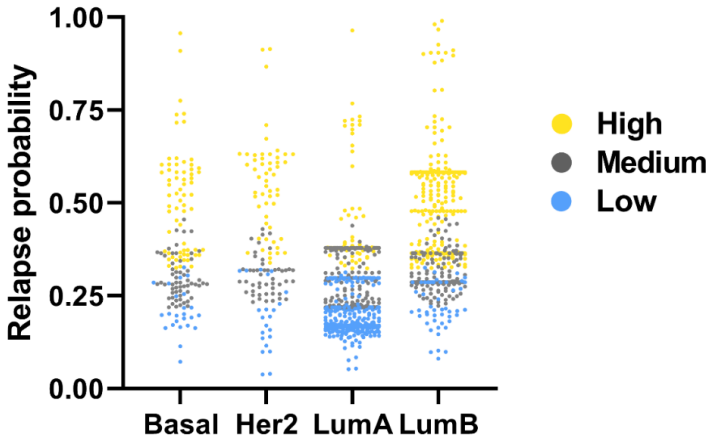

Figure 8.6 – **Estimation of the relapse probability score.** Scatter plot of the Relapse probability at 5 years of each patient of the whole cohort in each of the PAM50 groups. The points were colored based on the classification obtained via our model combining all three types of available information (clinical + PAM50 + cells proportions). The patients with low, medium and high risk of relapse according to our final model (using tertiles as the threshold) appears respectively in blue, grey and yellow, distinguishing different outcomes in each PAM50 tumor group.

the probability of relapse. Figure 8.6 represents the probability estimated for each patient and for the four different tumor subgroups. The score was split in three groups using tertiles: high group, medium and low. It is important to see that our model is able to find patients in the Luminal A group that have a probability of relapse equals or higher to patients in the Basal group. Typically, the Luminal A group was considered as the best prognosis and only light treatment was proposed to the patients. These results highlight the fact that the pam50 classification does not catch all the specificities of the tumors and that the immune cells proportions help distinguishing the patients that have more chance to relapse and refine the classical classification.

To illustrate and confirm these results, we represented in Figure 8.7 the estimation of the Kaplan Meier curve for the relapse free survival probability as a function of the time (in years). The left Kaplan-Meier curve was obtained on the training cohort (836 patients), the one used for the estimation of the Cox regression model and the right curves are the results obtained on a independent cohort (399 patients) used as a validation set. The total of patients does not reach the 2800 mentioned earlier for the descriptive results in Figure 8.5 because there was missing data in the clinical information available. This curve shows the relapse free survival (RFS) for the three different groups built from the Cox model including clinical variables, pam50 classification and proportions of cells. We can

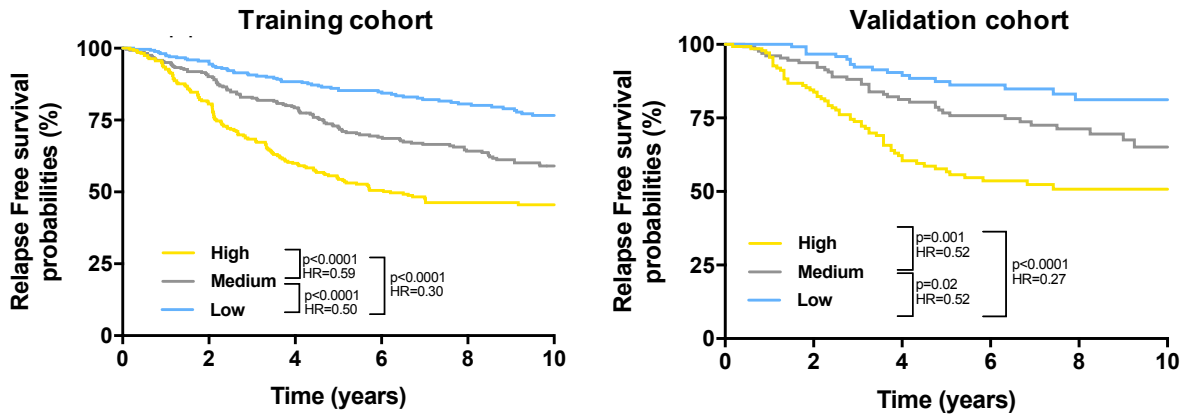

Figure 8.7 – **Relapse Free Survival Kaplan-Meier curves.** Kaplan-Meier estimates for relapse free survival; patients from the training cohort were stratified according to the score obtained from the Cox model combining clinical variables, PAM50 classification and immune cell estimations, using tertiles as thresholds. Graphs are presented for all patients of the training cohort (left) and validation cohort (right).

see that the differences between the different groups created are significant. The logrank test compares two survival curves and tests if they are significantly different, the p-values on Figure 8.7 are the results of logrank tests comparing each group 2 by 2. The hazard ratio (HR) between the different groups is also shown. Note that a hazard ratio smaller than one means that the considered group has a better prognosis than the group against which it is compared. These results illustrate the fact that immune cells proportions bring information for the classification of patients suffering from breast cancer.

9 Conclusion

This thesis focused in developing new methods to estimate the proportions of immune cells present inside a tumor from genomic data. The first part of it was dedicated to studying several convergence properties of an optimization algorithm in a non-smooth setting: the coordinate descent. We proved that the vanilla cyclic coordinate descent enjoys a finite model identification property and thanks to this property we could prove local linear convergence property once the model is identified (Chapter 3). These two theoretical results were illustrated on popular estimators (Lasso, sparse logistic regression and SVM dual) and popular machine learning datasets. Then, we proposed a variant of the coordinate descent algorithm to solve a linearly constrained Support Vector Regression optimization problem and proved its convergence towards a solution (Chapter 4). This algorithm uses a coordinate descent strategy where a closed-form of the updates were defined. The proposed algorithm is easy to implement and shows good performance in practice.

The second part studies the problem of setting hyperparameters in non-smooth machine learning models. We first studied the computation of the gradient with respect to the hyperparameters for which we derived an implicit formula using implicit differentiation (Chapter 6). This implicit differentiation formula takes advantage of the sparsity of the estimator to speed-up the computation of the gradient. We showed that our proposed algorithm outperforms generic differentiation packages or iterative differentiation techniques. In Chapter 7, we showed the interest of first-order techniques to solve the bilevel optimization problem related to the setting of hyperparameters on a wide range of estimators (ℓ_1 penalized methods, SVM, etc.) and datasets. We showed that our first order method can lead to significant benefits mainly when the number of hyperparameters is large. Finally, we proposed a new method to estimate the proportions of cells inside a tumor using the results obtained in previous chapters leading to a better estimation performance in comparison to the state-of-the-art methods on microarray data (Chapter 8).

The results obtained have given a theoretical background to the method proposed for the

estimation of cells proportions. The natural constraints arising from the estimation of proportions can now be directly taken into account in the estimation process thanks to our work on the constrained Support Vector Regression. Another important challenge alleviated by our work was the hyperparameters selection for the Support Vector Regression problem. This process was eased by the proposed first order method to set the hyperparameters of generic machine learning models. We believe that the questions coming from the biomedical application have lead us to answer more generic questions. The results of this thesis can be applied and useful to various problems and machine learning users. For example the proposed algorithm to solve the linearly constrained Support Vector optimization problem can be used to solve generic constraints that can be written as a polyhedron. Furthermore, the theory and practical methodology developed around hyperparameters setting can be used for a wide range of non-smooth models and have showed to be a fast and efficient way to select hyperparameters.

Several questions of interest remain in the continuity of our work.

Results for block coordinate descent. Our result on local linear convergence of the coordinate descent algorithm does not cover the case of block coordinate descent. In this case the non-smooth optimization problem considered writes:

$$x^\star \in \arg\min_{x \in \mathbb{R}^p} f(x) + \sum_{b \in \mathcal{B}} g_b(x_b) \ , \tag{9.1}$$

where $\mathcal{B} = (b_1, \dots, b_K)$ is a disjoint partition of $[p]$. The group Lasso (Yuan and Lin, 2006) is an example of an optimization problem that can be written as in Equation (9.1) with $f(x) = \|Ax - y\|^2$ and $g_b(x_b) = \lambda \|x_b\|$, with $A \in \mathbb{R}^{n \times p}$, $y \in \mathbb{R}^n$ and $\lambda > 0$. In this case, the non-smooth term of the composite minimization problem is block separable. We believe that our results on support identification and local linear convergence can be extended to the block separable case. The group Lasso penalty is a partly smooth function (Vaiter et al., 2018) which would allow us to use the identification theorem from Hare and Lewis (2004, Thm. 5.3) to prove model identification. However, proving local linear convergence requires to study the differentiability of the proximal operator for a group of variables and not for a single variable which would require a more careful analysis.

Resolution of the bilevel optimization problem. In Chapter 7, we used the fact that the hyperparameter selection with respect to a criterion could be written as a bilevel opti-

mization problem:

$$\begin{aligned} &\underset{\lambda \in \mathbb{R}^r}{\arg\min} \left\{ \mathcal{L}(\lambda) \triangleq \mathcal{C}\left(\hat{\beta}^{(\lambda)}\right) \right\} \\ &s.t.\ \hat{\beta}^{(\lambda)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min}\, \Phi(\beta, \lambda) \ . \end{aligned} \tag{9.2}$$

As already mentioned in Chapter 7, we have no convergence guarantees towards a solution of the optimization problem when the lower optimization problem is not smooth. This remains an opened question which appears to be very difficult to answer for the moment.

Applying our method for clinical purposes. To extend the validation of our method on the estimation of immune cells, we would need genomic data with ground-truth proportions. An ongoing work with the cancer institute Georges François Leclerc in Dijon will try to further the validation on a study that was especially designed for this purpose. This study will provide samples with ground truth immune cells proportions from patients that are being cured from colon cancer in the cancer institute. The true proportions of cells will be assessed by flow cytometry and the transcriptome of the same sample will be sequenced to obtain the gene expression of the tumor. After the estimation process, we will be able to compare the proportions obtained by flow cytometry (considered as the ground truth) and the estimated ones from the genomic data.

Moreover, the end of Chapter 8 shows the potential clinical use of our proposed method. Our analysis reveals that the information about the proportions of immune cells can be valuable to assess the risk of relapse in breast cancer. It was also shown to be a valuable information to predict the prognosis of patients suffering from a brain cancer (glioblastoma) in Klopfenstein et al. (2019). A natural extension would be to consider other types of cancer such as colon cancer tumors, which are known to be highly infiltrated in immune cells.

Automatic signature matrix construction. As stated in Chapter 1, this thesis was focused on the estimation process for the linear inverse problem arising from a biological quantification problem. For this particular application, the design matrix, containing the *pure* signals of the cells that we wish to quantify, has to be manually built as a preprocessing step. Currently, this step involves a differential genes expression analysis to select the genes that are overexpressed in a cell in comparison to all the other cells. The started assumption was to consider that this matrix was known and fixed.

The gene selection process is decoupled from the estimation step which means that the

genes selected are not the genes that would lead to the *best* estimation in some sense but only the significant genes remaining from the differential gene expression analysis. We can imagine a model where we select the genes of the matrix X at the same time as performing the estimation of the quantity of cells.

A first idea would be to control the condition number of the matrix X. This is done manually as suggested by Newman et al. (2015). A constraint on the condition number of the design matrix can possibly be added in the optimization problem as considered for the covariance matrix estimation process by Oh et al. (2015).

RÉSUMÉ DES TRAVAUX

1 Contexte

Le cancer est un problème majeur de notre société avec de nouvelles statistiques donnant plus de 17 millions de nouveaux cas au niveau mondial en 2020 (Sung et al., 2021). De récentes recherches dans le domaine médical suggèrent que chaque cancer est unique et qu'il faut donc trouver le traitement le mieux adapté à chaque patient au lieu de traiter les différents types de cancer de la même manière, c'est ce qu'on appelle la médecine personnalisée.

L'immunothérapie est un type de traitement contre le cancer qui existe depuis une dizaine d'années et qui a pour but d'utiliser le système immunitaire du patient pour combattre le cancer. Ce type de traitement est très prometteur mais malheureusement une faible proportion de patients répond favorablement au traitement. Une des pistes de recherche actives est de mieux comprendre pourquoi ces patients répondent ou ne répondent pas à l'immunothérapie et de pouvoir identifier en amont du traitement les patients qui répondront ou non. Une des clés pour répondre à ce problème est de pouvoir décrire l'environnement tumoral de manière précise et de pouvoir comprendre les intéractions entre les cellules immunitaires qui pourraient influencer l'efficacité du traitement.

Une tumeur est un amas de différents types de cellules, dont parfois des cellules de notre système immunitaires. Il est important d'avoir accès à l'information concernant la quantité de ces cellules immunitaires au sein de la tumeur et de pouvoir quantifier les différentes familles de cellules présentes. Une des manières de quantifier ces cellules est d'utiliser des données génomiques.

La modélisation est la suivante: le signal ARN venant de la tumeur est vu comme un signal brouillé composé de différents signaux provenant des différents types de cellules qui la composent. L'hypothèse est que la relation entre ces signaux purs et le signal de la

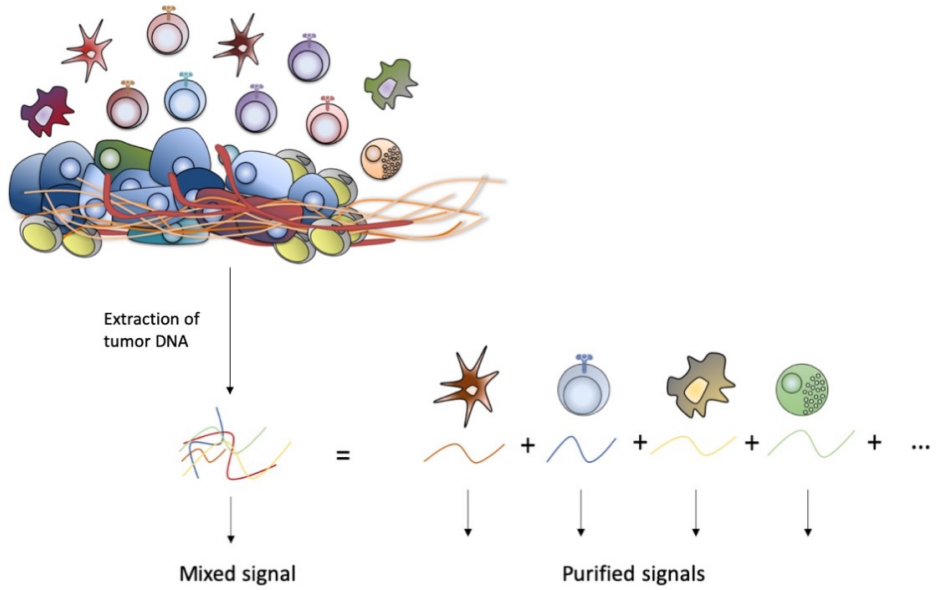

Figure 1 – **Problème inverse linéaire.** Schéma représentant la dépendance linéaire entre le signal ADN venant de la tumeur et les signaux pures provenant des cellules qui la composent.

tumeur est une relation linéaire. Trouver les proportions de cellules contenues au sein de la tumeur revient alors à un problème inverse linéaire comme l'illustre la Figure 1.

Nous nous intéressons à l'estimation de ces proportions une fois les signaux purs et le signal de la tumeur obtenue.

2 État de l'art

Une des méthodes les plus utilisées pour cette estimation est une méthode appelée Cibersort (Newman et al., 2015). Le processus d'estimation est basé sur la Support Vector Regression (Schölkopf et al., 1999) qui est le résultat du problème d'optimisation suivant

:

$$\hat{\beta}_{\text{SVR}} \in \underset{\beta,\beta_0,\xi_i,\xi_i^*,\varepsilon}{\arg\min} \quad \frac{1}{2}||\beta||^2 + C(\nu\varepsilon + \frac{1}{n}\sum_{i=1}^{n}(\xi_i + \xi_i^*)) \tag{3}$$

$$\text{sujet à} \quad y_i - X_{i:}\beta - \beta_0 \leq \varepsilon + \xi_i$$

$$X_{i:}\beta + \beta_0 - y_i \leq \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0, \varepsilon \geq 0 \ .$$

Cet estimateur dépend de deux hyperparamètres $C > 0$ et $\nu \in [0, 1]$. Nous considérons que ce processus d'estimation pourrait être amélioré pour deux raisons principales. La première, Cibersort ne tient pas compte des contraintes sous-jacentes à l'estimation de proportions, c'est-à-dire, la positivité des coefficients et que leur somme est égale à un. Nous proposons d'étudier l'ajout de ces contraintes dans la Support Vector Regression. La deuxième piste d'amélioration concerne la sélection des hyperparamètres pour l'estimation, pour le moment le paramètre C est fixe et toujours égale à un et ν peut prendre trois valeurs différentes $0.25, 0.5, 0.75$, la valeur retenue est celle qui minimise l'erreur quadratique moyenne.

Ces questions résument les axes de recherche principaux de cette thèse :

- Peut-on résoudre la Support Vector Regression avec les contraintes de proportions avec un algorithme rapide et avoir des guaranties de convergence vers une solution ?
- Peut-on simplifier le processus de sélection d'hyperparamètres pour éviter la *grid-search* qui est coûteuse pour deux hyperparamètres ?
- Finalement, est-ce que la prise en compte des contraintes et le fait de choisir les hyperparamètres de manière plus précise permet d'améliorer la qualité de l'estimation ?

3 Résolution du problème d'optimisation

Le problème d'optimisation de la Support Vector Regression (SVR) est souvent résolu en utilisant un algorithme qui s'appelle la descente par coordonnées. Généralement, la descente par coordonnées est utilisée dans le problème d'optimisation dual comme proposé par Ho and Lin (2012). La version proximale de cet algorithme permet de résoudre effi-

cacement les problèmes de minimisation composés qui ont la forme suivante :

$$x^\star \in \arg\min_{x \in \mathbb{R}^p} f(x) + \sum_{j=1}^{p} g_j(x_j) \ , \tag{4}$$

où f est une fonction lisse, c'est-à-dire que son gradient est Lipschitz, et les fonctions g_j sont convexes (possiblement non-différentiables).

Plusieurs estimateurs utilisés pour des tâches de régression ou de classification sont obtenus en résolvant un problème d'optimisation qui s'écrit sous la forme de Equation (4): le Lasso (Tibshirani, 1996), le problème de elastic net (Zou and Hastie, 2005), la régression logistique parcimonieuse (Koh et al., 2007) et les problèmes de Support Vector Machine (SVM) et SVR (Boser et al., 1992; Schölkopf et al., 1999).

3.1 Convergence linéaire local pour la descente par coordonnées

Ces problèmes d'optimisation non-lisses provenant du domaine de l'apprentissage automatique génèrent des solutions qui sont *structurées*. Par exemple, la régularisation parcimonieuse de la norme ℓ_1, où tous les $g_j = |\cdot|$, donne des solutions qui ont seulement quelques coefficients non-nuls. La structure induite par la norme ℓ_1 est portée par la notion de support qui est l'ensemble des indices correspondant aux coefficients non-nuls de la solution. Cette notion de support peut être généralisée pour des fonctions g_j convexes.

Définition 1 (Support généralisé). *Le support généralisé $\mathcal{S}_x \subseteq \{1, \dots, p\}$ est l'ensemble des indices $j \in \{1, \dots, p\}$ où g_j est différentiable en x_j :*

$$\mathcal{S}_x \triangleq \{j \in \{1, \dots, p : \partial g_j(x_j) \textit{ est un singleton.}\} \ .$$

La descente par coordonnées est un algorithme itératif ce qui implique que le problème d'optimisation est résolu à une certaines précision. Une question importante est de savoir si cet algorithme itératif est capable d'identifier le bon support après un nombre fini d'itérations ? En d'autres termes, est-ce qu'il existe une itération $K > 0$ telle que pour tout $k \geq K$, nous ayons $x^{(k)}_{\mathcal{S}_{x^\star}} = x^\star_{\mathcal{S}_{x^\star}}$ avec $x^\star$ qui est une solution de Equation (4). La propriété d'identification du support après un nombre fini d'itérations pour la descente par coordonnées a été prouvée par Nutini et al. (2017, Lemme 3) et un résultat similaire avec une technique de preuve différente est prouvé dans le Chapitre 3. De plus, la descente de gradient proximale a une convergence linéaire une fois que le support a été identifié

(Liang et al., 2014). Nous prouvons que cela est le cas également pour la descente par coordonnées.

Contribution principale. Afin d'étudier la convergence locale de la descente par coordonnées, nous considérons l'équation de point fixe d'une époque (une époque est une mise à jour de chacune des coordonnées). Une époque de descente par coordonnées peut s'écrire :

$$x^{(k+1)} = \psi(x^{(k)}) \triangleq \mathcal{P}_p \circ \ldots \circ \mathcal{P}_1(x^{(k)}) \ , \tag{5}$$

où les $\mathcal{P}_j$ sont les applications coordonnées par coordonnées de l'opérateur proximal $\mathcal{P}_j : \mathbb{R}^p \to \mathbb{R}^p$ and $\gamma_j > 0$:

$$x \mapsto \begin{pmatrix} x_1 \\ \vdots \\ x_{j-1} \\ \mathrm{prox}_{\gamma_j g_j}\left(x_j - \gamma_j \nabla_j f(x)\right) \\ x_{j+1} \\ \vdots \\ x_p \end{pmatrix} .$$

Le théorème suivant montre qu'après l'identification du support l'algorithme de descente par coordonnées converge linéairement vers la solution du problème d'optimisation Equation (4).

Théorème 1 (Convergence linéaire locale)**.** *Considérons une solution $x^\star$ de Equation (4) et $\mathcal{S} = \mathcal{S}_{x^\star}$. Supposons que*

1. *La solution est non-dégénérée* i.e., $-\nabla f(x^\star) \in \mathrm{ri}(\partial g(x^\star))$ *où* $\mathrm{ri}(C)$ *est l'intérieur relatif*[1] *d'un ensemble convexe C et ∂g est le sous-différentiel de g.*

2. *Pour tout $j \in \mathcal{S}$, g_j est localement $\mathcal{C}^2$ et f est localement $\mathcal{C}^2$ autour de $x^\star$.*

3. *La condition d'injectivité restreinte est vérifiée* i.e., $\nabla^2_{\mathcal{S},\mathcal{S}} f(x^\star) \succ 0$ *(la Hessienne est restreinte aux lignes et aux colonnes dont les indices sont dans $\mathcal{S}$.)*

4. *La suite $(x^{(k)})_{k\geq 0}$ générée par l'algorithme de descente par coordonnées converge vers $x^\star$.*

[1] voir Definition 2.10 dans le Chapitre 2.

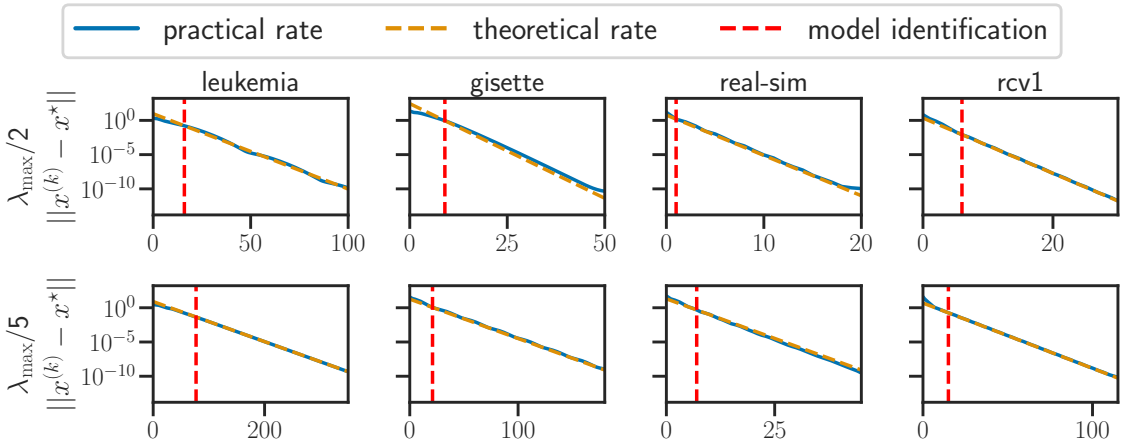

Figure 2 – **Lasso, convergence linéaire.** Distance à l'optimum, $\|x^{(k)} - x^\star\|$, en fonction du nombre d'itérations k, sur 4 jeux de données: *leukemia*, *gisette*, *rcv1*, et *real-sim*. Le paramètre de régularisation a été choisi proportionnellement à $\lambda_{\max} = \frac{\|A^\top b\|}{2n}$.

5. *Le support a été identifié* i.e., *il existe* $K \geq 0$ *tel que pour tout* $k \geq K$

$$x^{(k)}_{\mathcal{S}^c} = x^\star_{\mathcal{S}^c} \ .$$

Alors $(x^{(k)})_{k \geq K}$ *converge linéairement vers* $x^\star$. *Plus précisément, pour tout* $\nu \in [\rho(\mathcal{J}\psi_{\mathcal{S},\mathcal{S}}(x^\star)), 1[$, *il existe une constante* C *tels que pour tout* $k \geq K$,

$$\|x^{(k)}_{\mathcal{S}} - x^\star_{\mathcal{S}}\| \leq C\nu^{(k-K)}\|x^{(K)}_{\mathcal{S}} - x^\star_{\mathcal{S}}\| \ .$$

On désigne par $\mathcal{J}f(x)$ la Jacobienne d'une fonction f au point x et $\rho(M)$ correspond au rayon spectral de la matrice M.

Nous avons montré sur plusieurs jeux de données réels que le taux de convergence obtenu théoriquement dans le Théorème 1 correspond au taux de convergence empirique comme montré sur la Figure 2 pour le cas du Lasso où $f(x) = \frac{1}{2n}\|Ax - b\|^2$, avec $A \in \mathbb{R}^{n \times p}$ et $b \in \mathbb{R}^n$, et $g_j(x_j) = \lambda|x_j|$ pour $\lambda > 0$. Ce théorème est prouvé au Chapitre 3.

Relation avec la littérature. La convergence linéaire locale a été prouvée pour les algorithmes ISTA et FISTA pour le problème du Lasso (Tao et al., 2016) en étudiant les propriétés spectrales de la matrice de récurrence utilisée pour les mises à jour des deux algorithmes. Ce résultat a ensuite été étendu à l'algorithme générique de la descente de gradient proximal par Liang et al. (2014) en supposant que la fonction non-différentiable g était partiellement lisse. La classe des fonctions partiellement lisses a été définis par

Lewis (2002) et regroupe la plupart des fonctions non-lisses utilisées en apprentissage automatique. Elle définit rigoureusement la structure dont nous avons parlé dans le cas des problèmes d'optimisation non-lisses. Dans notre travail, nous avons considéré l'algorithme de descente par coordonnées avec l'hypothèses que g est séparable *i.e.*, $g(x) = \sum_{i=1}^{p} g_j(x_j)$. Pour une solution $x^\star$ de Equation (4), nous avons montré que la classe des fonctions séparables qui sont $\mathcal{C}^2$ sur le support $\mathcal{S}_{x^\star}$ en $x^\star$ est équivalent au fait que ces fonctions sont partiellement lisses. Théorème 1 montre que la descente par coordonnées a une convergence linéaire une fois le support identifié tout comme la descente de gradient proximale. La convergence linéaire locale a également été montrée pour les algorithmes SAGA et prox-SVRG par Poon et al. (2018) avec le même cadre de fonctions partiellement lisses.

Ce comportement de convergence localement linéaire soulève la question de la vitesse d'identification de cette structure induite par la fonction non-lisse g. Cette notion est aussi connue comme étant la complexité de l'ensemble actif défini dans Nutini et al. (2019). C'est le nombre d'itération nécessaire pour identifier le support. Sous l'hypothèse que g est séparable, Nutini et al. (2019) ont donné une borne sur le nombre d'itération nécessaire pour la descente de gradient proximale pour identifier la structure. Ce résultat a ensuite était étendu pour les problèmes de minimisation avec f fortement convexe et g séparable pour la descente par coordonnées cyclique et gloutonne. Dans notre travail, nous montrons seulement que le régime de convergence linéaire local arrive après l'identification du support mais nous ne montrons pas que ce régime débute immédiatement après l'identification même si cela semble être le cas en pratique.

3.2 Algorithme pour la résolution de la Support Vector Regression sous contraintes

La Support Vector Regression (SVR) est très utilisée pour l'estimation de fonctions linéaires ou non-linéaires. Nous souhaitons étudier l'ajout de contraintes linéaires à l'estimateur ν-SVR (Schölkopf et al., 1999) afin notamment de pouvoir ajouter les contraintes liées à l'estimation de proportions. Le problème d'optimisation que nous cherchons à résoudre, pour une matrice de design $X \in \mathbb{R}^{n \times p}$ et un vecteur d'observations $y \in \mathbb{R}^n$, s'écrit :

$$\begin{aligned} \min_{\beta,\beta_0,\xi_i,\xi_i^*,\varepsilon} \quad & \frac{1}{2}||\beta||^2 + C\left(\nu\varepsilon + \frac{1}{n}\sum_{i=1}^{n} L_\varepsilon(y_i, X_{i:}\beta)\right) \\ \text{sujet à} \quad & \xi_i, \xi_i^* \geq 0, \varepsilon \geq 0 \\ & A\beta \leq b \\ & \Gamma\beta = d \ , \end{aligned} \tag{LSVR-P}$$

où $A \in \mathbb{R}^{k_1 \times p}$, $\Gamma \in \mathbb{R}^{k_2 \times p}$, $\beta \in \mathbb{R}^p$, $\xi, \xi^* \in \mathbb{R}^n$ et $\beta_0 \in \mathbb{R}$, $\varepsilon > 0$.

Par exemple, si $A = -\operatorname{Id}_p$, $b = 0$, $\Gamma = 0$ et $d = 0$, une contrainte de positivité est ajoutée sur les coefficients de β. Cet estimateur peut être vu comme une alternative de l'estimateur des moindres carrés avec des contraintes de positivités (Lawson and Hanson, 1995). En considérant $A = -\operatorname{Id}_p$, $b = 0$, $\Gamma = (1, \ldots, 1)$ et $d = 1$, nous obtenons la Simplex SVR avec des contraintes de positivité et que la somme des coefficients soit égale à 1.

Le problème d'optimisation de la SVR sans contraintes linéaires générales est souvent résolu dans son dual en utilisant la descente par coordonnées (Ho and Lin, 2012) ou un algorithme appelé *Sequential Minimal Optimization* (SMO) (Platt, 1999) qui est une variante de la descente par coordonnées. Le problème dual de Equation (LSVR-P) sera décrit explicitement dans le Chapitre 4 mais peut s'écrire :

$$\begin{aligned} \min_{\theta \in \mathbb{R}^{2n+k_1+k_2}} \quad & f(\theta) = \frac{1}{2}\theta^\top Q\theta + l^\top\theta \\ \text{sujet à} \quad & 0 \leq \theta_i \leq \frac{C}{n}, \forall i \in \{1, \ldots, 2n\} \\ & \sum_{i=1}^{2n} \theta_i = C\nu \\ & \sum_{i=1}^{n} \theta_i - \theta_{i+n} = 0 \\ & \theta_i \geq 0, \forall i \in \{2n+1, \ldots, 2n+k_1\} \ , \end{aligned} \tag{LSVR-D}$$

avec Q une matrice semi-définie positive.

Les deux contraintes d'égalité dans Equation (LSVR-D) relient des variables entre elles, ce qui donne un problème d'optimisation non-séparable. L'algorithme SMO met à jour deux variables à chaque itération en s'assurant que les deux contraintes d'égalité sont satisfaites

après chaque itération.

Contribution principale. Nous prouvons que le problème d'optimisation de la ν-SVR sous contraintes linéaires est un problème convexe qui mélange des blocs de variables séparables et non-séparables. Nous proposons dans le Chapitre 4 une généralisation de l'algorithme SMO proposé par Platt (1999) afin de résoudre Equation (LSVR-D). Nous prouvons également le théorème de convergence suivant :

Théorème 2. *Supposons que* $\{x \in \mathbb{R}^p : Ax \leq b, \Gamma x = d\}$ *définisse un polyèdre non-vide. Alors la suite des itérées* $(\theta^k)_{k\geq 0}$*, définie par le SMO généralisé, converge vers une solution du problème d'optimisation Equation* (LSVR-D)*.*

Les détails de l'algorithme proposé et la preuve de convergence sont donnés dans le Chapitre 4.

Relation avec la littérature. La convergence de l'algorithme SMO a été prouvé par Keerthi and Gilbert (2002). Plus tard, Lopez and Dorronsoro (2012) ont proposé une preuve de convergence plus simple. Dans le Chapitre 4, nous montrons que la technique de preuve utilisée dans Lopez and Dorronsoro (2012) peut être généralisée aux problèmes avec les contraintes linéaires en utilisant de nouveaux arguments. Concernant la vitesse de convergence, She and Schmidt (2017) ont montré que la convergence était linéaire lorsque la sélection des blocs est faite aléatoirement avec la même probabilité pour chacun des blocs. L'algorithme que nous proposons est un algorithme *glouton* qui sélectionne une paire de variables (ou une seule variable) à mettre à jour. Le choix est basé sur un score qui est lié aux conditions d'optimalité de Karush–Kuhn–Tucker (KKT).

Les garanties de convergence pour la descente par coordonnées sont généralement données sous l'hypothèse que la fonction non-lisse est séparable. Le SMO de Platt (1999) est un exemple de variante de la descente par coordonnées qui peut être utilisé sans l'hypothèse de séparabilité. Pour la descente par coordonnées aléatoire, Necoara and Patrascu (2014) ont proposé une variante de la descente par coordonnées qui peut être utilisée pour résoudre un problème de minimisation composée Equation (4) avec l'addition d'une seule contrainte d'égalité liant les variables entre elles. L'algorithme proposé est très similaire au SMO mais permet de résoudre une classe plus large de problèmes et cet algorithme utilise une stratégie de sélection de coordonnées aléatoire. Ce résultat a ensuite été étendue par Reddi et al. (2014) avec un système linéaire qui relie les variables entre elles. Notre algorithme est un exemple de variante de la descente par coordonnées qui résout un problème avec des variables séparables et non-séparables. A notre connaissance, une

preuve générale de convergence avec une sélection de coordonnées gloutonne n'existe pas pour ce type de problème.

4 Sélection d'hyperparamètres pour des problèmes non-lisses

Après avoir porté notre attention sur des outils d'optimisation afin de résoudre le problème d'optimisation sous-jacent à notre application de quantification de cellules immunitaires, nous nous tournons vers le problème de la sélection d'hyperparamètres. Choisir les hyperparamètres pour les modèles d'apprentissage automatique peut être une tâche difficile surtout lorsque le nombre d'hyperparamètres à choisir est grand. En reprenant l'exemple de la SVR, choisir les deux hyperparamètres C et ν est souvent fait en utilisant un algorithme de recherche sur une grille, *grid-search*. La performance du modèle est testée sur chacun des points de la grille, ce qui donnerait 100 problèmes d'optimisation à résoudre pour une grille de taille 10 pour chacun des deux paramètres.

La sélection d'hyperparamètre requiert une mesure de performance pour un estimateur donné appelé un critère de sélection. Formellement, un critère est une fonction $\mathcal{C} : \mathbb{R}^p \to \mathbb{R}$, qui est choisi pour avoir une bonne erreur de généralisation, *e.g.*, la fonction held-out (Devroye and Wagner, 1979), la validation croisée (CV, Stone and Ramer 1965, voir Arlot and Celisse 2010 pour plus de détails), ou pour réduire la complexité du modèle *e.g.*, les critères AIC (Akaike, 1974), BIC (Schwarz, 1978) ou SURE (Stein, 1981) (voir dans le Tableau 1 pour des exemples classiques).

Choisir des hyperparamètres étant donné un critère peut être écrit comme un problème d'optimisation à deux niveaux (Colson et al., 2007) :

$$\begin{aligned} &\underset{\lambda \in \mathbb{R}^r}{\arg\min} \left\{ \mathcal{L}(\lambda) \triangleq \mathcal{C}\left(\hat{\beta}^{(\lambda)}\right) \right\} \\ &\text{sujet à} \hat{\beta}^{(\lambda)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min}\, \Phi(\beta, \lambda) \ . \end{aligned} \tag{6}$$

Utiliser de la *grid-search* pour résoudre le problème d'optimisation peut être vu comme l'utilisation d'une méthode à l'ordre zéro *i.e.*, le problème est résolu en utilisant simplement l'évaluation de la fonction.

Cependant, si l'espace dans lequel vivent les hyperparamètres est continu et que le chemin de régularisation, *i.e.*, la fonction $\lambda \mapsto \hat{\beta}^{(\lambda)}$ est bien définie et presque partout différentiable,

[2]For a linear model $y = X\beta + \varepsilon$, with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, the degree of freedom (dof, Efron 1986) is defined as $\text{dof}(\beta) = \sum_{i=1}^{n} \text{cov}(y_i, (X\beta)_i)/\sigma^2$.

[3]The smoothed Hinge loss is given by $\mathcal{L}(x) = \frac{1}{2} - x$ if $x \leq 0$, $\frac{1}{2}(1-x)^2$ if $0 \leq x \leq 1, 0$ else .

Criterion	Problem type	Criterion $\mathcal{C}(\beta)$
Held-out mean squared error	Regression	$\frac{1}{n}\|y^{\text{val}} - X^{\text{val}}\beta\|^2$
Stein unbiased risk estimate (SURE)[2]	Regression	$\|y - X\beta\|^2 - n\sigma^2 + 2\sigma^2 \text{dof}(\beta)$
Held-out logistic loss	Classification	$\frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i^{\text{val}} X_{i:}^{\text{val}}\beta})$
Held-out smoothed Hinge loss[3]	Classification	$\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y_i^{\text{val}}, X_{i:}^{\text{val}}\beta)$

Table 1 – Exemples de critères utilisés pour la sélection d'hyperparamètres.

alors les méthodes du premier ordre peuvent être utilisées pour la résolution du problème à deux niveaux. En utilisant la formule de dérivation composée, le gradient de $\mathcal{L}$ par rapport à λ, aussi appelé *hypergradient*, s'obtient par :

$$\nabla_\lambda \mathcal{L}(\lambda) = \hat{\mathcal{J}}_{(\lambda)}^\top \nabla \mathcal{C}(\hat{\beta}^{(\lambda)}) \ , \tag{7}$$

avec $\hat{\mathcal{J}}_{(\lambda)} \in \mathbb{R}^{p \times r}$ la *Jacobienne* de la fonction $\lambda \mapsto \hat{\beta}^{(\lambda)}$. La principale difficulté est d'évaluer cet hypergradient. Pour cela, Il existe trois algorithmes principaux pour le calculer: la différentiation implicite (Larsen et al., 1996), la différentiation automatique avec le *mode backward* (Linnainmaa, 1970) ou le *mode forward* (Wengert, 1964). Une fois l'hypergradient calculé, Equation (6) peut être résolu en utilisant un schéma du premier ordre, par exemple une descente de gradient avec un pas $\rho > 0$: $\lambda^{(t+1)} = \lambda^{(t)} - \rho \nabla_\lambda \mathcal{L}(\lambda^{(t)})$.

4.1 Calcul de l'hypergradient pour des problèmes non-lisses

Nous souhaitons proposer une méthode pour le calcul de l'hypergradient $\nabla \mathcal{L}$ afin d'utiliser une méthode du premier ordre pour résoudre Equation (6) quand le sous-problème est non-lisse.

Nous considérons pour cela, des estimateurs qui sont le résultat du problème d'optimisation suivant :

$$\hat{\beta}^{(\lambda)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min}\, \Phi(\beta, \lambda) = f(\beta) + \underbrace{\sum_{i=1}^{n} g_j(\beta_j, \lambda)}_{= g(\beta, \lambda)} \ , \tag{8}$$

Ce type de problèmes d'optimisation peut être résolu avec des algorithmes proximaux. Une solution $\hat{\beta}^{(\lambda)}$ satisfait l'équation de point fixe suivante, pour tout $\gamma > 0$:

$$\hat{\beta}^{(\lambda)} = \text{prox}_{\gamma g}\left(\hat{\beta}^{(\lambda)} - \gamma \nabla f(\hat{\beta}^{(\lambda)})\right) \ . \tag{9}$$

Contribution principale. Nous donnons une formule explicite de la différentiation implicite pour les problèmes qui s'écrivent comme Equation (8) en utilisant la notion de support généralisé (Définition 1). L'ensemble des hypothèses utilisées pour ce théorème n'est pas détaillé ici mais est donné dans le Chapitre 6.

Théorème 3 (Formule implicite non-lisse)**.** *Soit $\lambda \in \mathbb{R}^r$. Soit $\hat{\beta} \triangleq \hat{\beta}^{(\lambda)}$ une solution de Equation (1.19) et $\hat{S}$ son support généralisé. Sous un ensemble d'hypothèses qui assurent que l'opérateur proximal est différentiable au point $\hat{\beta} - \gamma \nabla f(\hat{\beta})$, la Jacobienne $\hat{\mathcal{J}}$ de Equation (8) est donnée par la formule, avec $\hat{z} = \hat{\beta} - \gamma \nabla X^\top f(X\hat{\beta})$, et $A \triangleq \mathrm{Id}_{\hat{s}} - \partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{S}} \left(\mathrm{Id}_{\hat{s}} - \gamma X_{:\hat{S}}^\top \nabla^2 f(X\hat{\beta}) X_{:\hat{S}}\right)$:*

$$\begin{aligned}
\hat{\mathcal{J}}_{\hat{S}^c} &= \partial_2 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{S}^c} \quad , \\
\hat{\mathcal{J}}_{\hat{S}} &= A^{-1} \left(\partial_2 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{S}} - \gamma \partial_1 \operatorname{prox}_{\gamma g}(\hat{z})_{\hat{S}} X_{:\hat{S}}^\top \nabla^2 f(X\hat{\beta}) X_{:\hat{S}^c} \hat{\mathcal{J}}_{\hat{S}^c}\right) \quad .
\end{aligned}$$

Ici l'opérateur proximal de $g(\cdot, \lambda)$ est vu comme une fonction ψ de β et λ :

$$\begin{aligned}
\mathbb{R}^p \times \mathbb{R}^r &\to \mathbb{R}^p \\
(\beta, \lambda) &\mapsto \operatorname{prox}_{g(\cdot,\lambda)}(\beta) = \psi(\beta, \lambda) \quad .
\end{aligned}$$

On note $\partial_1 \operatorname{prox}_g \triangleq \partial_1 \psi$ et $\partial_2 \operatorname{prox}_g \triangleq \partial_2 \psi$ où $\partial_1 \psi$ est la Jacobienne par rapport à la première variable et $\partial_2 \psi$ la Jacobienne par rapport à la deuxième variable. Ce théorème est prouvé dans le Chapitre 6.

Comme énoncé dans le théorème, la parcimonie induite par le support généralisé peut être prise en compte pour accélérer le calcul de l'hypergradient. De plus, l'utilisation de la différentiation implicite nous permet de résoudre le sous-problème Equation (8) avec l'algorithme de notre choix, du moment qu'il identifie le support après un nombre fini d'itérations.

Nous avons comparé trois méthodes différentes pour calculer l'hypergradient Equation (6) en choisissant le Lasso (Tibshirani, 1996) comme sous-problème et l'erreur quadratique moyenne sur un ensemble de validation comme critère de sélection :

- **Le mode Forward** : C'est l'algorithme classique de différentiation *forward* utilisé sur la descente par coordonnées proximale pour résoudre le Lasso. Cet algorithme ne tient pas compte de la parcimonie de la Jacobienne pour le calcul de l'hypergradient.
- **La différentiation implicite** : Premièrement, le Lasso est résolu en utilisant une descente par coordonnées proximale et dans un deuxième temps, nous résolvons le système linéaire donné dans le Théorème 3 pour calculer l'hypergradient.

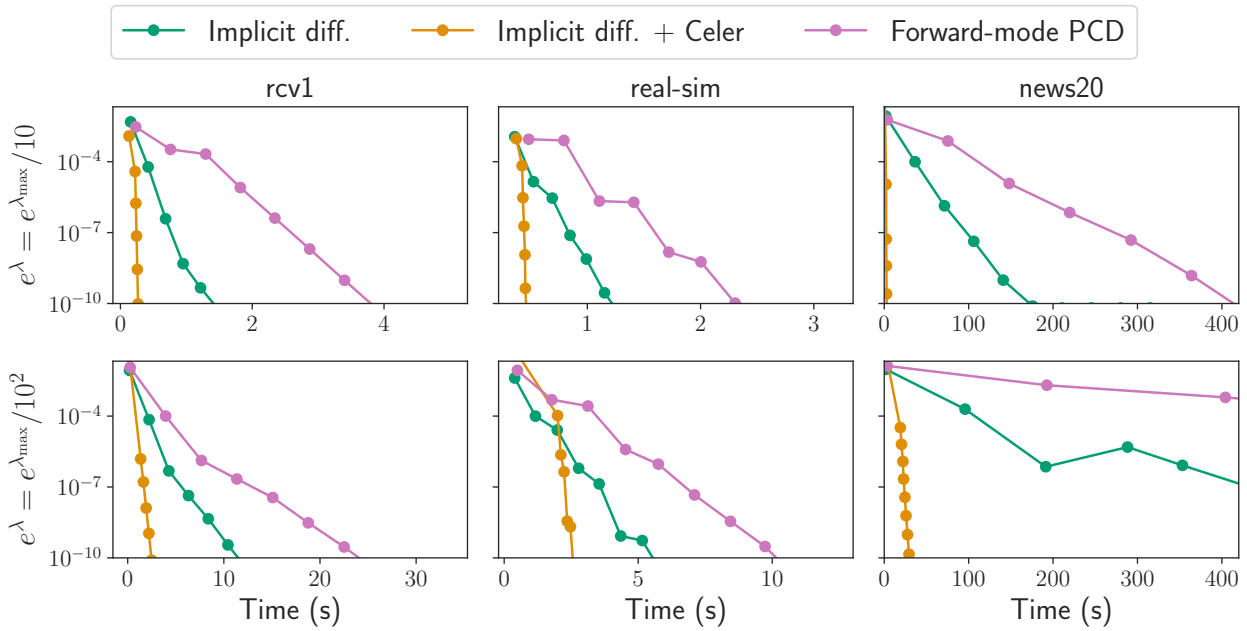

Figure 3 – **Lasso held-out, time to compute one hypergradient.** Absolute difference between the exact hypergradient (using $\hat{\beta}$) and the iterate hypergradient (using $\beta^{(k)}$) of the Lasso as a function of time. Results are for three datasets and two different regularization parameters. For the implicit differentiation, the lower problem is solved using proximal coordinate descent (Implicit diff.) or `Celer` (Massias et al. 2020, Implicit diff. + `Celer`).

- **La différentiation implicite + Celer** : Nous changons l'algorithme de résolution du Lasso pour utiliser une version accélérée de la descente par coordonnées proximale appelée `Celer`. Après avoir résolu le Lasso, nous calculons l'hypergradient à partir du système linéaire du Théorème 3.

Comme on peut le voir sur la Figure 3, tenir compte de la parcimonie de la Jacobienne permet d'obtenir un gain significatif en temps de calcul pour l'hypergradient. De plus, grâce à la différentiation implicite, nous pouvons utiliser les algorithmes état de l'art pour la résolution du sous-problème. La combinaison de l'algorithme `Celer` avec la formule de différentiation implicite du Théorème 3 augmente grandement la vitesse de calcul.

Relation avec la littérature. La différentiation implicite pour la sélection d'hyperparamètres remonte à Bengio (2000). Ils ont considéré des fonctions Φ deux fois différentiables et la différentiation implicite mène à la résolution d'un système linéaire de taille $p \times p$:

$$\nabla_\lambda \mathcal{L}(\lambda) = -\nabla^2_{\beta,\lambda}\Phi(\hat{\beta}^{(\lambda)}, \lambda) \left(\nabla^2_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda)\right)^{-1} \nabla \mathcal{C}(\hat{\beta}^{(\lambda)}) \ . \tag{10}$$

Notre résultat montre que pour les modèles de type Lasso, la Jacobienne a le même support que les itérées. Nous pouvons utiliser cela pour accélérer le calcul de la Jacobienne en résolvant un système linéaire de taille $|\hat{S}| \times |\hat{S}|$.

Une des hypothèses importante est que l'algorithme itératif utilisé pour résoudre le Lasso identifie le bon support. Cela est vérifié pour la descente de gradient proximale (Hale et al., 2008) et la descente par coordonnées proximale (Massias et al., 2020). La différentiation d'algorithmes proximaux a également été considéré par Deledalle et al. (2014) dans le cadre de la différentiation faible. Cependant, ce travail considère seulement de la différentiation *forward*. La grande différence avec notre approche est que nous utilisons la structure induite par le support pour restreindre le calcul de la Jacobienne sur ce support, une fois la solution du Lasso obtenue.

Ochs et al. (2015) ont considéré des problèmes non-lisses qui peuvent être résolus avec des algorithmes itératifs dont chaque itération est différentiable. Ils ont proposé de calculer le gradient du problème d'optimisation à deux niveaux en calculant les dérivées d'opérateurs proximaux de Bregman qui sont supposés être différentiables. En utilisant la géométrie du problème d'optimisation non-lisse via le support généralisé, nous avons pu prouver que les opérateurs proximaux classiques peuvent être différentiés à l'optimum sous quelques hypothèses. Ces algorithmes itératifs comme la descente par coordonnées sont état de l'art pour les problèmes incluant la norme ℓ_1 par exemple, ce qui permet une résolution plus rapide du problème et un calcul plus rapide du gradient pour la méthode du premier ordre.

4.2 Optimisation d'hyperparamètres pour des problèmes non-lisses

Grâce au résultat de la section précédente pour le calcul de l'hypergradient, nous pouvons maintenant nous tourner vers la résolution du problème à deux niveaux afin de faire de l'optimisation d'hyperparamètres.

Algorithm 17 DESCENTE DE GRADIENT HEURISTIQUE

input : $X \in \mathbb{R}^{n \times p}, y \in \mathbb{R}^n, \lambda \in \mathbb{R}^r, (\epsilon_i)$
init : use_adaptive_step_size = True
for $i = 1, \dots, iter$ **do**
  $\lambda^{\text{old}} \leftarrow \lambda$
  `// compute the value and the gradient`
  $\mathcal{L}(\lambda), \nabla\mathcal{L}(\lambda) \leftarrow$ Solution du système linéaire donné dans le Theorem 1.3
  if use_adaptive_step_size **then**
    $\alpha = 1/\|\nabla\mathcal{L}(\lambda)\|$
  $\lambda \mathrel{-}= \alpha\nabla\mathcal{L}(\lambda)$ `// gradient step`
  if $\mathcal{L}(\lambda) > \mathcal{L}(\lambda^{\text{old}})$ **then**
    use_adaptive_step_size = False
    $\alpha \mathrel{/}= 10$
return λ

Contribution principale. Nous proposons un algorithme *heuristique* (Algorithm 17) pour la résolution de Equation (6) basé sur la descente de gradient. Nous avons comparé notre méthode basée sur le Théorème 3 à d'autres méthodes de sélection d'hyperparamètres comme la *grid-search*, la *random search* ou une méthode Bayésienne appelée *SMBO*. Figure 4 montre la trajectoire de notre méthode du premier ordre sur les lignes de niveaux de la fonction de perte de la validation croisée. Cela illustre également que notre méthode est plus rapide que ses concurrents pour trouver les deux hyperparamètres de l'elastic net qui minimisent le critère de validation croisée.

Relation avec la littérature. Pedregosa (2016) a étudié l'optimisation d'hyperparamètres via la différentiation implicite dans le cas de problèmes à deux niveaux lisses. Ils ont prouvé que leur algorithme, *HOAG*, converge vers un point stationnaire de Equation (6). Notre algorithme est heuristique parce que nous n'avons pas de garanties théoriques sur la convergence vers une solution de Equation (6) quand le sous-problème est non-lisse. Etendre ce résultat au cas non-lisse est une question difficile qui est laissée pour un travail futur.

Développement de package Python. Le travail présenté dans le Chapitre 6 et le Chapitre 7 ont abouti sur le développement d'un package disponible en Python appelé **sparse-ho**. Ce package suit la même interface de programmation que *scikit-learn* (Pedregosa et al., 2011).

Le package est construit pour suivre la structure du problème d'optimisation à deux

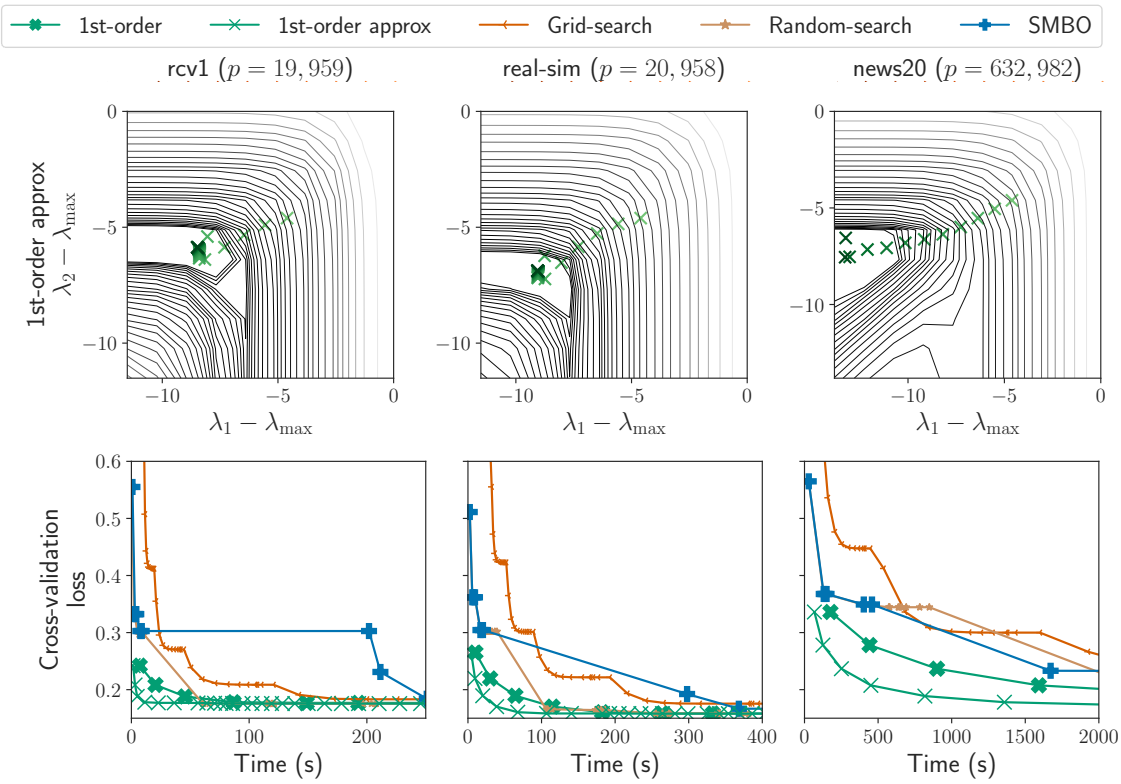

Figure 4 – **Validation croisée pour Elastic net, comparaison en temps (2 hyperparamètres).** Lignes de niveaux de la fonction de perte de la validation croisée (lignes noires) et la valeur de la fonction de perte en fonction du temp sur différent jeux de données. *rcv1*, *real-sim* et *news20*

niveaux. Premièrement, un modèle doit être choisi; plusieurs modèles très utilisés en apprentissage automatique sont déjà implémentés pour une sélection d'hyperparamètres automatique comme le Lasso, le Lasso à poids, l'elastic net, la régression logistique parcimonieuse ou encore la SVM et SVR. Ensuite, un critère doit être choisi pour mesurer la performance du modèle, il y a plusieurs options pour la régression l'erreur quadratique moyenne ou le critère SURE (Stein, 1981). L'un des désavantages des méthodes du premier ordre est que le critère doit être *régulier* et au moins continu, donc le taux de bien classé pour la classification ne peut pas être utilisé ici. Cependant, nous avons implémenté la fonction de perte logistique, la version lisse de la fonction de perte de Hinge et la version multiclasse de la fonction de perte logistique. Finalement, il faut choisir une méthode du premier ordre pour résoudre le problème d'optimisation à deux niveaux qui peut être de la descente de gradient avec un pas constant, un algorithme de *line-search* ou encore l'algorithme ADAM (Kingma and Ba, 2014).

Des exemples, de la documentation et la description du package sont disponibles sur

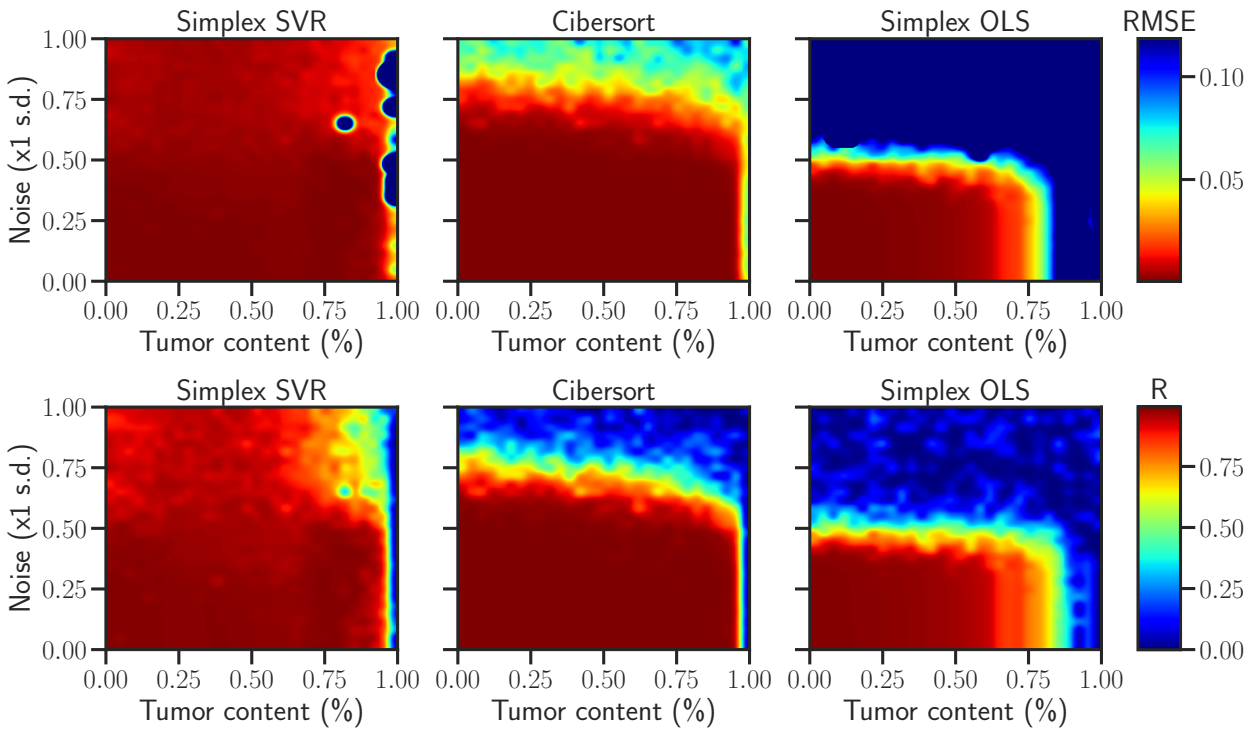

Figure 5 – **Robustesse au bruit et à l'environnement tumoral.** Heatmap représentant la racine carrée de l'erreur quadratique moyenne (RMSE) ou le coefficient de corrélation (R) entre les vrais proportions de cellules et les proportions estimées en fonction de pourcentage de tumeur (axe x) et le niveau de bruit (axe y). Nous avons comparé trois estimateurs différents le Simplex SVR, Cibersort et les moindres carrés sous contraintes de Simplex.

https://qb3.github.io/sparse-ho/.

5 Estimation de la proportion de cellules immunitaires

Les résultats des deux sections précédentes ont permis de proposer une nouvelle méthode pour l'estimation de proportions de cellules au sein d'une tumeur. Ce nouvel estimateur utilise la SVR sous contraintes qui peut être résolu en utilisant l'algorithme SMO généralisé et la sélection d'hyperparamètres automatique. Nous avons pu appliquer cette méthode sur des jeux de données réels et comparer nos résultats avec des travaux antérieurs.

Figure 5 compare notre méthode à la méthode état de l'art appelée Cibersort (Newman et al., 2015) et les moindres carrés sous contraintes de simplexe (Gong et al., 2011). Nous avons testé la capacité des différentes méthodes à être robuste par rapport au bruit et à la présence de cellules tumorales. Nous avons ajouté un bruit log-Gaussien dans les données, $\mathcal{N}(0, \sigma^2)$, en choisissant σ comme un pourcentage de $\sigma_{\max} = 11.6$ (valeur prise de Newman et al. (2015)). Le pourcentage a été choisi entre 0 et 1 avec 30 valeurs différentes.

Nous avons artificiellement ajouté des cellules tumorales dans les données pour répliquer les conditions pour les cellules au sein d'une tumeur. Ce pourcentage de contenu tumoral a été choisi entre 0% et 100% en prenant 30 pourcentages différents. La performance en estimation de ces trois estimateurs a été comparée en utilisant la racine carrée de l'erreur quadratique moyenne (RMSE) et le coefficient de corrélation entre les vraies proportions et les proportions estimées. On peut voir que notre estimateur a une meilleure performance en estimation alors que le bruit augmente. Il est plus robuste au bruit que les deux autres estimateurs.

Pendant la thèse, j'ai travaillé une journée par semaine pendant un an dans un centre spécialisé contre le cancer pour appliquer nos résultats à des buts cliniques. Cette collaboration a permis l'écriture de plusieurs articles déjà publiés (Klopfenstein et al., 2019; Reichling et al., 2020).

Ce premier article décrit comment l'information de la proportion de cellules à l'intérieur d'un glioblastome peut aider à affiner le pronostique du patient. Le second article utilise une approche de forêts aléatoires pour détecter automatiquement les cellules d'intérêts sur une lame d'immunohistochimie.

Un autre article est actuellement en révision. Dans ce travail, nous montrons que l'information à propos des cellules est importante et peut être utilisée cliniquement pour estimer le risque de rechute de patientes atteintes de cancer du sein. Quelques résultats et figures seront présentés dans le Chapitre 8 de cette thèse.

BIBLIOGRAPHY

Alexander R. Abbas, Kristen Wolslegel, Dhaya Seshasayee, Zora Modrusan, and Hilary F. Clark. Deconvolution of blood microarray data identifies cellular activation patterns in systemic lupus erythematosus. *PLOS ONE*, 4(7):1–16, 07 2009.

A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. Differentiable convex optimization layers. In *Advances in neural information processing systems*, pages 9558–9570, 2019.

H. Akaike. A new look at the statistical model identification. *IEEE Trans. Autom. Control*, AC-19:716–723, 1974.

B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *ICML*, volume 70, pages 136–145, 2017.

S. M. Ansell and R. H. Vonderheide. Cellular composition of the tumor microenvironment. *American Society of Clinical Oncology Educational Book*, 33(1):e91–e97, 2013.

S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

Gilles Bareilles, Franck Iutzeler, and Jérôme Malick. Newton acceleration on manifolds identified by proximal-gradient methods. *arXiv preprint arXiv:2012.12936*, 2020.

R. E. Barlow and H. D. Brunk. The isotonic regression problem and its dual. *Journal of the American Statistical Association*, 67(337):140–147, 1972.

T. A Barnes and E. Amir. Hype or hope: the prognostic value of infiltrating immune cells in cancer. *British journal of cancer*, 117(4):451–460, 2017.

H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, New York, 2011.

A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18(153):1–43, 2018.

A. Beck and L. Tetruashvili. On the convergence of block coordinate type methods. *SIAM J. Imaging Sci.*, 23(4):651–694, 2013.

A. Belloni, V. Chernozhukov, and L. Wang. Square-root Lasso: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98(4):791–806, 2011.

Y. Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8): 1889–1900, 2000.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012.

J. Bergstra, D. Yamins, and D. D. Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in science conference*, pages 13–20, 2013.

J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

Q. Bertrand, Q. Klopfenstein, M. Blondel, S. Vaiter, A. Gramfort, and J. Salmon. Implicit differentiation of lasso-type models for hyperparameter optimization. *ICML*, 2020.

D. P. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Trans. Autom. Control*, 21(2):174–184, 1976.

D. P. Bertsekas. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

P. J. Bickel, Y. Ritov, and A. B. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Ann. Statist.*, 37(4):1705–1732, 2009.

J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot. Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE journal of selected topics in applied earth observations and remote sensing*, 5(2):354–379, 2012.

M. Blondel and F. Pedregosa. Lightning: large-scale linear classification, regression and ranking in Python, 2016.

J. Bolte and E. Pauwels. Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. *Mathematical Programming*, pages 1–33, 2020a.

J. Bolte and E. Pauwels. A mathematical model for automatic differentiation in machine learning. *arXiv preprint arXiv:2006.02080*, 2020b.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

Rasmus Bro and Sijmen De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401, 1997.

E. Brochu, V. M. Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

J. V. Burke and J. J. Moré. On the identification of active constraints. *SIAM J. Numer. Anal.*, 25(5):1197–1211, 1988.

C.-H. Wu, J-.M. Ho, and D. T. Lee. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):276–281, Dec 2004.

E. J. Candès and B. Recht. Simple bounds for recovering low-complexity models. *Math. Program.*, pages 1–13, 2012.

E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theory*, 52(2): 489–509, 2006.

C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

C. Chang and C. Lin. Training v-support vector regression: Theory and algorithms. *Neural Comput.*, 14(8):1959–1977, August 2002. ISSN 0899-7667.

O. Chapelle, P. Haffner, and V. N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, Sep. 1999. ISSN 1045-9227.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1-3):131–159, 2002.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20(1):33–61, 1998.

H. Cherkaoui, J. Sulam, and T. Moreau. Learning to solve tv regularised problems with unrolled algorithms. *Advances in Neural Information Processing Systems*, 33, 2020.

W. G. Cochran. The role of mathematics in the medical sciences. *New England Journal of Medicine*, 265(4):176–176, 1961.

B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, volume 49 of *Springer Optim. Appl.*, pages 185–212. Springer, New York, 2011.

P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

Jennifer Couzin-Frankel. Cancer immunotherapy. *Science*, 342(6165):1432–1433, 2013. ISSN 0036-8075.

D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220, 1972. ISSN 00359246.

C. Criscitiello, A. Esposito, D. Trapani, and G. Curigliano. Prognostic and predictive value of tumor infiltrating lymphocytes in early breast cancer. *Cancer treatment reviews*, 50: 205–207, 2016.

A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

C.-A. Deledalle, S. Vaiter, J. Fadili, and G. Peyré. Stein Unbiased GrAdient estimator of the Risk (SUGAR) for multiple parameter selection. *SIAM J. Imaging Sci.*, 7(4):2448–2487, 2014.

S. Dempe, V. Kalashnikov, G. A. Pérez-Valdés, and N. Kalashnykova. Bilevel programming problems. *Energy Systems. Springer, Berlin*, 2015.

J Dennis and Virginia Torczon. Derivative-free pattern search methods for multidisciplinary design problems. In *5th Symposium on Multidisciplinary Analysis and Optimization*, page 4349, 1994.

L. Devroye and T. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604, 1979.

S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.*, 17(83):1–5, 2016.

J. Domke. Generic methods for optimization-based modeling. In *AISTATS*, volume 22, pages 318–326, 2012.

Charles Dossal, Maher Kachour, MJ Fadili, Gabriel Peyré, and Christophe Chesneau. The degrees of freedom of the lasso for general design matrix. *Statistica Sinica*, pages 809–828, 2013.

H. Drucker, C.J. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. *Advances in neural information processing systems*, 9:155–161, 1996.

H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 155–161. MIT Press, 1997.

R. Edgar, M. Domrachev, and A. E. Lash. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. *Nucleic acids research*, 30(1):207–210, 2002.

B. Efron. How biased is the apparent error rate of a prediction rule? *J. Amer. Statist. Assoc.*, 81(394):461–470, 1986.

L. C. Evans and R. F. Gariepy. *Measure theory and fine properties of functions*. CRC Press, 1992.

J. Fadili, J. Malick, and G. Peyré. Sensitivity analysis for mirror-stratifiable convex functions. *SIAM J. Optim.*, 28(4):2975–3000, 2018.

J. Fan and J. Lv. Sure independence screening for ultrahigh dimensional feature space. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 70(5):849–911, 2008.

O. Fercoq and P. Richtárik. Accelerated, parallel and proximal coordinate descent. *SIAM J. Optim.*, 25(3):1997 – 2013, 2015.

M. Feurer and F. Hutter. Hyperparameter optimization. In *Automated Machine Learning*, pages 3–33. Springer, Cham, 2019.

C. S. Foo, C. B. Do, and A. Y. Ng. Efficient multiple hyperparameter learning for log-linear models. In *Advances in neural information processing systems*, pages 377–384, 2008.

L. Franceschi, M. Donini, P. Frasconi, and M. Pontil. Forward and reverse gradient-based hyperparameter optimization. In *ICML*, pages 1165–1173, 2017.

L. Franceschi, P. Frasconi, S. Salzo, and M. Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, pages 1563–1572, 2018.

P.I. Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

J. Frecon, S. Salzo, and M. Pontil. Bilevel learning of the group lasso structure. In *Advances in Neural Information Processing Systems*, pages 8301–8311, 2018.

J. Friedman, T. J. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Ann. Appl. Stat.*, 1(2):302–332, 2007.

J. Friedman, T. J. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *J. Stat. Softw.*, 33(1):1–22, 2010.

T.-T. Friel and R. Harrison. Linear programming support vector machines for pattern classification and regression estimation: and the sr algorithm: Improving speed and tightness of vc bounds in sv algorithms. Research report, Department of Automatic Control and Systems Engineering, February 1998.

J.-J. Fuchs. On sparse representations in arbitrary redundant bases. *IEEE Trans. Inf. Theory*, 50(6):1341–1344, 2004.

B. R. Gaines, J. Kim, and H. Zhou. Algorithms for fitting the constrained lasso. *Journal of Computational and Graphical Statistics*, 27(4):861–871, 2018.

F. Galton. Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:246–263, 1886.

Guoxuan Gao, Zihan Wang, Xiang Qu, and Zhongtao Zhang. Prognostic value of tumor-infiltrating lymphocytes in patients with triple-negative breast cancer: a systematic review and meta-analysis. *BMC cancer*, 20(1):1–15, 2020.

S. Ghadimi and M. Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

M. A. Glaire, E. Domingo, A. Sveen, J. Bruun, A. Nesbakken, G. Nicholson, M. Novelli, K. Lawson, D. Oukrif, W. Kildal, et al. Tumour-infiltrating cd8+ lymphocytes and colorectal cancer recurrence by tumour and nodal stage. *British journal of cancer*, 121(6): 474–482, 2019.

T. Gong, N. Hartmann, I. S. Kohane, V. Brinkmann, F. Staedtler, M. Letzkus, S. Bongiovanni, and J. D. Szustakowski. Optimal deconvolution of transcriptional profiling data using quadratic programming with application to complex clinical blood samples. *PLOS ONE*, 6(11):1–11, 11 2011.

Ting Gong and Joseph D Szustakowski. Deconrnaseq: a statistical framework for deconvolution of heterogeneous tissue samples based on mrna-seq data. *Bioinformatics*, 29(8): 1083–1085, 2013.

I. Goodfellow, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

S. Gould, B. Fernando, A. Cherian, P. Anderson, R. S. Cruz, and E. Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447.*, 2016.

R. Grazzi, L. Franceschi, M. Pontil, and S. Salzo. On the iteration complexity of hypergradient computation. *arXiv preprint arXiv:2006.16218*, 2020.

Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

E. Hale, W. Yin, and Y. Zhang. Fixed-point continuation for ℓ_1-minimization: Methodology and convergence. *SIAM J. Optim.*, 19(3):1107–1130, 2008.

W. L. Hare. Identifying active manifolds in regularization problems. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 261–271. Springer, 2011.

W. L. Hare and A. S. Lewis. Identifying active constraints via partial smoothness and prox-regularity. *Journal of Convex Analysis*, 11(2):251–266, 2004.

W. L. Hare and A. S. Lewis. Identifying active manifolds. *Algorithmic Operations Research*, 2(2):75–75, 2007.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *J. Mach. Learn. Res.*, 5(Oct):1391–1415, 2004.

T. J. Hastie and R. J. Tibshirani. *Generalized additive models*, volume 43. CRC press, 1990.

D. Haussler, D. W. Bednarski, M. Schummer, N. Cristianini, N. Duffy, and T. S. Furey. Support vector machine classification and validation of cancer tissue samples using microarray expression data . *Bioinformatics*, 16(10):906–914, 10 2000. ISSN 1367-4803.

M.R. Hestenes and E. Stiefel. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC, 1952.

N. J. Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms. I*, volume 305. Springer-Verlag, Berlin, 1993a.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms. II*, volume 306. Springer-Verlag, Berlin, 1993b.

C-H Ho and C-J Lin. Large-scale linear support vector regression. *The Journal of Machine Learning Research*, 13(1):3323–3348, 2012.

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

M. Hong, X. Wang, M. Razaviyayn, and Z-Q. Luo. Iteration complexity analysis of block coordinate descent methods. *Mathematical Programming*, 163(1-2):85–114, 2017.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. SS Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 408–415, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

F. Hutter, J. Lücke, and L. Schmidt-Thieme. Beyond manual tuning of hyperparameters. *KI-Künstliche Intelligenz*, 29(4):329–337, 2015.

K. Ji, J. Yang, and Y. Liang. Provably faster algorithms for bilevel optimization and applications to meta-learning. *arXiv preprint arXiv:2010.07962*, 2020.

H. Jia and A. M. Martinez. Support vector machines in face recognition with occlusions. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 136–141, June 2009.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, pages 137–142, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-69781-7.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

S. S. Keerthi and E. G. Gilbert. Convergence of a generalized smo algorithm for svm classifier design. *Mach. Learn.*, 46(1-3):351–360, March 2002. ISSN 0885-6125.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural Comput.*, 13(3):637–649, March 2001. ISSN 0899-7667.

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Quentin Klopfenstein, Caroline Truntzer, Julie Vincent, and Francois Ghiringhelli. Cell lines and immune classification of glioblastoma define patient's prognosis. *British journal of cancer*, 120(8):806–814, 2019.

K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *J. Mach. Learn. Res.*, 8(8):1519–1555, 2007.

R. Kohavi and G. H. John. Automatic parameter selection by minimizing estimated error. In *Machine Learning Proceedings 1995*, pages 304–312. Elsevier, 1995.

P. Krzyszczyk, A. Acevedo, E. J. Davidoff, L. M Timmins, I. Marrero-Berrios, M. Patel, C. White, C. Lowe, J. J. Sherba, C. Hartmanshenn, et al. The growing role of precision and personalized medicine for cancer treatment. *Technology*, 6(03n04):79–100, 2018.

K. Kunisch and T. Pock. A bilevel optimization approach for parameter learning in variational models. *SIAM J. Imaging Sci.*, 6(2):938–983, 2013.

S. K. Lam, A. Pitrou, and S. Seibert. Numba: A LLVM-based Python JIT Compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6. ACM, 2015.

J. Larsen, L. K. Hansen, C. Svarer, and M. Ohlsson. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, pages 62–71, 1996.

F. Lauer and G. Bloch. Incorporating prior knowledge in support vector regression. *Machine Learning*, 70, 01 2008.

C. Lawson and R. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995.

D. Leventhal and A. S. Lewis. Randomized methods for linear constraints: convergence rates and conditioning. *Mathematics of Operations Research*, 35(3):641–654, 2010.

A. S. Lewis. Active sets, nonsmoothness, and sensitivity. *SIAM Journal on Optimization*, 13 (3):702–725, 2002.

X. Li, T. Zhao, R. Arora, H. Liu, and M. Hong. On faster convergence of cyclic block coordinate descent-type methods for strongly convex minimization. *Journal of Machine Learning Research*, 18(1):6741–6764, 2017.

J. Liang, J. Fadili, and G. Peyré. Local linear convergence of forward–backward under partial smoothness. In *Advances in neural information processing systems*, pages 1970–1978, 2014.

J. Liang, J. Fadili, and G. Peyré. Activity Identification and Local Linear Convergence of Forward–Backward-type Methods. *SIAM J. Optim.*, 27(1):408–437, 2017.

C. L. Liew. Inequality constrained least-squares estimation. *Journal of the American Statistical Association*, 71(355):746–751, 1976.

Chih-Jen Lin, Ruby C Weng, and S Sathiya Keerthi. Trust region newton methods for large-scale logistic regression. In *Proceedings of the 24th international conference on Machine learning*, pages 561–568. ACM, 2007.

S. Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7, 1970.

P.-L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

R. Liu, P. Mu, X. Yuan, S. Zeng, and J. Zhang. A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton. *ICML*, 2020.

J. Lopez and J. R. Dorronsoro. Simple proof of convergence of the smo algorithm for different svm variants. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7): 1142–1147, July 2012. ISSN 2162-237X.

J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. *arXiv preprint arXiv:1911.02590*, 2019.

K. Lounici. Sup-norm convergence rate and sign concentration property of Lasso and Dantzig estimators. *Electron. J. Stat.*, 2:90–102, 2008.

P. Lu, A. Nakorchevskiy, and E. M. Marcotte. Expression deconvolution: a reinterpretation of dna microarray data reveals dynamic changes in cell populations. *Proceedings of the National Academy of Sciences*, 100(18):10370–10375, 2003.

J. Luo, M. Wu, D. Gopukumar, and Y. Zhao. Big data application in biomedical research and health care: a literature review. *Biomedical informatics insights*, 8:BII–S31559, 2016.

Z-Q. Luo and P. Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.

D. Maclaurin, D. Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, volume 37, pages 2113–2122, 2015.

J. Mairal and B. Yu. Complexity analysis of the lasso regularization path. In *ICML*, pages 353–360, 2012.

J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(4):791–804, 2012.

M. Massias, A. Gramfort, and J. Salmon. Celer: a fast solver for the lasso with dual extrapolation. In *ICML*, volume 80, pages 3315–3324, 2018.

M. Massias, S. Vaiter, A. Gramfort, and J. Salmon. Dual extrapolation for sparse generalized linear models. *arXiv preprint arXiv:1907.05830*, 2019.

M. Massias, S. Vaiter, A. Gramfort, and J. Salmon. Dual extrapolation for sparse generalized linear models. *J. Mach. Learn. Res.*, 2020.

P. McCullagh and J.A. Nelder. *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. London: Chapman & Hall, 1989.

Kevin Menden, Mohamed Marouf, Sergio Oller, Anupriya Dalmia, Daniel Sumner Magruder, Karin Kloiber, Peter Heutink, and Stefan Bonn. Deep learning–based cell composition analysis from tissue expression profiles. *Science Advances*, 6(30), 2020. doi: 10.1126/sciadv.aba2619.

S. Mohammadi, N. Zuckerman, A. Goldsmith, and A. Grama. A critical survey of deconvolution methods for separating cell types in complex tissues. *Proceedings of the IEEE*, 105(2):340–366, Feb 2017.

Michinari Momma and Kristin P Bennett. A pattern search method for model selection of support vector regression. In *Proceedings of the 2002 SIAM International Conference on Data Mining*, pages 261–274. SIAM, 2002.

Gregory Moore, Charles Bergeron, and Kristin P Bennett. Model selection for primal svm. *Machine learning*, 85(1-2):175–208, 2011.

Brian B Nadel, David Lopez, Dennis J Montoya, Feiyang Ma, Hannah Waddel, Misha M Khan, Serghei Mangul, and Matteo Pellegrini. The gene expression deconvolution interactive tool (gedit): Accurate cell type quantification from gene expression data. *GigaScience*, 10(2):giab002, 2021.

I. Necoara and A. Patrascu. A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints. *Computational Optimization and Applications*, 57(2):307–337, 2014.

I. Necoara, Y. Nesterov, and F. Glineur. Linear convergence of first order methods for non-strongly convex optimization. *Mathematical Programming*, 175(1-2):69–107, 2019.

Y. Nesterov. *Introductory lectures on convex optimization*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, Boston, MA, 2004.

Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.*, 22(2):341–362, 2012.

A. M. Newman, C.L. Liu, M. R. Green, A. J. Gentles, W. Feng, Y. Xu, C. D. Hoang, M. Diehn, and A. A. Alizadeh. Robust enumeration of cell subsets from tissue expression profiles. *Nature methods*, 12(5):453–457, May 2015. ISSN 1548-7105.

Aaron M Newman, Chloé B Steen, Chih Long Liu, Andrew J Gentles, Aadel A Chaudhuri, Florian Scherer, Michael S Khodadoust, Mohammad S Esfahani, Bogdan A Luca, David Steiner, et al. Determining cell type abundance and expression from bulk tissues with digital cytometry. *Nature biotechnology*, 37(7):773–782, 2019.

Terri T Ni, William J Lemon, Yu Shyr, and Tao P Zhong. Use of normalization methods for analysis of microarrays containing a high degree of gene effects. *BMC bioinformatics*, 9 (1):1–11, 2008.

J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

J. Nutini. *Greed is good: greedy optimization methods for large-scale structured problems*. PhD thesis, University of British Columbia, 2018.

J. Nutini, M. W. Schmidt, I. H. Laradji, M. P. Friedlander, and H. A. Koepke. Coordinate descent converges faster with the Gauss-Southwell rule than random selection. In *ICML*, pages 1632–1641, 2015.

J. Nutini, I. Laradji, and M. Schmidt. Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence. *arXiv preprint arXiv:1712.08859*, 2017.

J. Nutini, M. Schmidt, and W. Hare. "active-set complexity" of proximal gradient: How long does it take to find the sparsity pattern? *Optimization Letters*, 13(4):645–655, 2019.

P. Ochs, R. Ranftl, T. Brox, and T. Pock. Bilevel optimization with nonsmooth lower level problems. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 654–665, 2015.

B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. SCS: Splitting conic solver, version 2.1.2, 2019.

Sang-Yun Oh, Bala Rajaratnam, and Joong-Ho Won. Towards a sparse, scalable, and stably positive definite (inverse) covariance estimator, 2015.

Joel S Parker, Michael Mullins, Maggie CU Cheang, Samuel Leung, David Voduc, Tammi Vickery, Sherri Davies, Christiane Fauron, Xiaping He, Zhiyuan Hu, et al. Supervised risk predictor of breast cancer based on intrinsic subtypes. *Journal of clinical oncology*, 27 (8):1160, 2009.

F. Pedregosa. Hyperparameter optimization with approximate gradient. In *ICML*, volume 48, pages 737–746, 2016.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

G. Peyré and J. M. Fadili. Learning analysis sparsity priors. In *Sampta*, 2011.

J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research, 1998.

J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

R. Poliquin and R. Rockafellar. Generalized hessian properties of regularized nonsmooth functions. *SIAM Journal on Optimization*, 6(4):1121–1137, 1996a.

R. Poliquin and R. Rockafellar. Prox-regular functions in variational analysis. *Transactions of the American Mathematical Society*, 348(5):1805–1838, 1996b.

B. T. Polyak. Introduction to optimization. optimization software. *Inc., Publications Division, New York*, 1, 1987.

C. Poon and J. Liang. Trajectory of alternating direction method of multipliers and adaptive acceleration. In *Advances In Neural Information Processing Systems*, pages 7357–7365, 2019.

C. Poon, J. Liang, and C.-B. Schönlieb. Local convergence properties of SAGA/Prox-SVRG and acceleration. In *ICML*, volume 90, pages 4121–4129, 2018.

Sidharth V Puram, Itay Tirosh, Anuraag S Parikh, Anoop P Patel, Keren Yizhak, Shawn Gillespie, Christopher Rodman, Christina L Luo, Edmund A Mroz, Kevin S Emerick, et al. Single-cell transcriptomic analysis of primary and metastatic tumor ecosystems in head and neck cancer. *Cell*, 171(7):1611–1624, 2017.

E. Purdom and S. P. Holmes. Error distribution for gene expression data. *Statistical applications in genetics and molecular biology*, 4(1), 2005.

W. Qiao, G. Quon, A. Csaszar, M. Yu, Q. Morris, and P. W. Zandstra. Pert: A method for expression deconvolution of human blood samples from varied microenvironmental and developmental conditions. *PLOS Computational Biology*, 8(12):1–14, 12 2012.

Z. Qu and P. Richtárik. Coordinate descent with arbitrary sampling i: Algorithms and complexity. *Optimization Methods and Software*, 31(5):829–857, 2016a.

Z. Qu and P. Richtárik. Coordinate descent with arbitrary sampling ii: Expected separable overapproximation. *Optimization Methods and Software*, 31(5):858–884, 2016b.

A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. Meta-learning with implicit gradients. In *Advances in neural information processing systems*, pages 113–124, 2019.

Youlan Rao, Yoonkyung Lee, David Jarjoura, Amy S Ruppert, Chang-gong Liu, Jason C Hsu, and John P Hagan. A comparison of normalization techniques for microrna microarray data. *Statistical applications in genetics and molecular biology*, 7(1), 2008.

L. A. Rastrigin. The convergence of the random search method in the extremal control of a many parameter system. *Automaton & Remote Control*, 24:1337–1342, 1963.

M. Razaviyayn, M. Hong, and Z.-Q. Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM J. Optim.*, 23(2):1126–1153, 2013.

M. Reck, D. Rodríguez-Abreu, A. G. Robinson, R. Hui, T. Csőszi, A. Fülöp, M. Gottfried, N. Peled, A. Tafreshi, S. Cuffe, et al. Pembrolizumab versus chemotherapy for pd-l1–positive non–small-cell lung cancer. *N engl J med*, 375:1823–1833, 2016.

S. Reddi, A. Hefny, C. Downey, A. Dubey, and S. Sra. Large-scale randomized-coordinate descent methods with non-separable linear constraints. *arXiv preprint arXiv:1409.2617*, 2014.

C. Reichling, J. Taieb, V. Derangere, Q. Klopfenstein, K. Le Malicot, J.-M. Gornet, H. Becheur, F. Fein, O. Cojocarasu, M. C. Kaminsky, et al. Artificial intelligence-guided tissue analysis combined with immune infiltrate assessment predicts stage iii colon cancer outcomes in petacc08 study. *Gut*, 69(4):681–690, 2020.

P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

R. T. Rockafellar. *Convex analysis*. Princeton Landmarks in Mathematics. Princeton University Press, Princeton, NJ, 1997.

R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1998.

S. Rosset and J. Zhu. Piecewise linear regularized solution paths. *Ann. Statist.*, 35(3): 1012–1030, 2007.

A. Saha and A. Tewari. On the nonasymptotic convergence of cyclic coordinate descent methods. *SIAM J. Optim.*, 23(1):576–601, 2013.

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

B. Schölkopf, P. Bartlett, A. Smola, and R. Williamson. Shrinking the tube: A new support vector regression algorithm. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, pages 330–336, Cambridge, MA, USA, 1999. MIT Press. ISBN 0-262-11245-0.

G. Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6(2):461–464, 1978.

M. W. Seeger. Cross-validation optimization for large scale structured classification kernel methods. *J. Mach. Learn. Res.*, 9:1147–1178, 2008.

S. Shalev-Shwartz and A. Tewari. Stochastic methods for l1-regularized loss minimization. *Journal of Machine Learning Research*, 12:1865–1892, 2011.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

J. She and M. Schmidt. Linear convergence and support vector identification of sequential minimal optimization. In *10th NIPS Workshop on Optimization for Machine Learning*, volume 5, 2017.

H.-J. M. Shi, S. Tu, Y. Xu, and W. Yin. A primer on coordinate descent algorithms. *ArXiv e-prints*, 2016.

Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, August 2004. ISSN 0960-3174.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

S. E. Stanton, S. Adams, and M. L. Disis. Variation in the incidence and magnitude of tumor-infiltrating lymphocytes in breast cancer subtypes: a systematic review. *JAMA oncology*, 2(10):1354–1360, 2016.

C. M. Stein. Estimation of the mean of a multivariate normal distribution. *Ann. Statist.*, 9 (6):1135–1151, 1981.

L. R. A. Stone and J.C. Ramer. Estimating WAIS IQ from Shipley Scale scores: Another cross-validation. *Journal of clinical psychology*, 21(3):297–297, 1965.

R. Sun and M. Hong. Improved iteration complexity bounds of cyclic block coordinate descent for convex problems. In *Advances in Neural Information Processing Systems*, pages 1306–1314, 2015.

Y. Sun, H. Jeong, J. Nutini, and M. Schmidt. Are we there yet? manifold identification of gradient-related proximal methods. In *AISTATS*, volume 89, pages 1110–1119, 2019.

H. Sung, J. Ferlay, R. L. Siegel, M. Laversanne, I. Soerjomataram, A. Jemal, and F. Bray. Global cancer statistics 2020: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: A Cancer Journal for Clinicians*, n/a(n/a), 2021.

J. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300, 06 1999.

S. Tao, D. Boley, and S. Zhang. Local linear convergence of ISTA and FISTA on the LASSO problem. *SIAM J. Optim.*, 26(1):313–336, 2016.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 58(1):267–288, 1996.

R. Tibshirani, J. Bien, J. Friedman, T. J. Hastie, N. Simon, J. Taylor, and R. J. Tibshirani. Strong rules for discarding predictors in lasso-type problems. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 74(2):245–266, 2012.

R. J. Tibshirani. The lasso problem and uniqueness. *Electron. J. Stat.*, 7:1456–1490, 2013.

A. N. Tikhonov. On the stability of inverse problems. *Dokl. Akad. Nauk SSSR*, 39:176–179, 1943.

Itay Tirosh, Benjamin Izar, Sanjay M Prakadan, Marc H Wadsworth, Daniel Treacy, John J Trombetta, Asaf Rotem, Christopher Rodman, Christine Lian, George Murphy, et al. Dissecting the multicellular ecosystem of metastatic melanoma by single-cell rna-seq. *Science*, 352(6282):189–196, 2016.

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109(3):475–494, 2001.

P. Tseng and S. Yun. Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *J. Optim. Theory Appl.*, 140(3):513, 2009.

S. Vaiter, M. Golbabaee, J. Fadili, and G. Peyré. Model selection with low complexity priors. *Information and Inference: A Journal of the IMA*, 4(3):230–287, 2015.

S. Vaiter, G. Peyré, and J. Fadili. Model consistency of partly smooth regularizers. *IEEE Transactions on Information Theory*, 64(3):1725–1737, 2018.

T. Van Gestel, J. A. K. Suykens, D. . Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, and J. Vandewalle. Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on Neural Networks*, 12(4):809–821, July 2001. ISSN 1045-9227.

C. L. Ventola. Cancer immunotherapy, part 3: challenges and future trends. *Pharmacy and Therapeutics*, 42(8):514, 2017.

B. E. Wahlin, B. Sander, B. Christensson, and E. Kimby. Cd8+ t-cell content in diagnostic lymph nodes measured by flow cytometry is a predictor of survival in follicular lymphoma. *Clinical Cancer Research*, 13(2):388–397, 2007.

Xuran Wang, Jihwan Park, Katalin Susztak, Nancy R Zhang, and Mingyao Li. Bulk tissue cell type deconvolution with multi-subject single-cell expression reference. *Nature communications*, 10(1):1–9, 2019.

Zhong Wang, Mark Gerstein, and Michael Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 10(1):57–63, 2009.

R. E. Wengert. A simple automatic derivative evaluation program. *Commun. ACM*, 7(8): 463–464, August 1964. ISSN 0001-0782.

E. Winston and Z. Kolter. Neural monotone operator equilibrium networks. *Advances in neural information processing systems*, 2020.

S. J. Wright. Identifiable surfaces in constrained optimization. *SIAM Journal on Control and Optimization*, 31(4):1063–1079, 1993.

S. J. Wright. Accelerated block-coordinate relaxation for regularized optimization. *SIAM J. Optim.*, 22(1):159–186, 2012.

S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Y. Xu and W. Yin. A globally convergent algorithm for nonconvex optimization based on block coordinate update. *Journal of Scientific Computing*, 72(2):700–734, 2017.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 68(1):49–67, 2006.

L. Zhang, M. Mahdavi, and R. Jin. Linear convergence with condition number independent access of full gradients. *Advances in Neural Information Processing Systems*, 26:980–988, 2013.

T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116, 2004.

Y. Zheng. Chapter 1 - introduction to non-coding rnas and high throughput sequencing. In Yun Zheng, editor, *Computational Non-coding RNA Biology*, pages 3–31. Academic Press, 2019. ISBN 978-0-12-814365-0.

H. Zou. The adaptive lasso and its oracle properties. *J. Amer. Statist. Assoc.*, 101(476): 1418–1429, 2006.

H. Zou and T. J. Hastie. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67(2):301–320, 2005.

H. Zou, T. J. Hastie, and R. Tibshirani. On the "degrees of freedom" of the lasso. *Ann. Statist.*, 35(5):2173–2192, 2007.