Universidad Autónoma de Madrid
Escuela Politécnica Superior
Departamento de Ingeniería Informática

Nesterov Acceleration Schemes for Group Lasso

Master's thesis presented to apply for the
Master's degree of Investigation and Innovation in Information and Communications Technologies

By
Alejandro Catalina Feliú

under the direction of
José R. Dorronsoro Ibero

Madrid, June 22, 2017

Contents

List of Figures

List of Tables

List of Algorithms

Resumen

La teoría de optimización convexa es cuerpo fundamental para la resolución de muchos problemas de aprendizaje automático y problemas del día a día. Estudiar y alcanzar una mejor comprensión de estos métodos es por tanto un aspecto fundamental de cara a construir buenas soluciones y extraer conclusiones adecuadas. Además, centraremos también nuestra atención en el estudio de modelos lineales como Lasso o Group Lasso, que incorporan varias ventajas, siendo las más importantes el bajo coste computacional y la interpretabilidad de los modelos finales.

En esta tesis de máster realizaremos en primer lugar un estudio del campo de la teoría de optimización convexa desde un punto de vista puramente teórico, estudiando conceptos como subdiferencial y minimización de problemas proximales de cara a minimizar funciones compuestas donde alguna de sus componentes es no diferenciable. Este cuerpo teórico es la base para estudiar ISTA, un primer enfoque iterativo para la minimización de tales funciones. Para mejorar la convergencia de ISTA, Nesterov introdujo una optimización para alcanzar una convergencia óptima de $\mathcal{O}(1/k^2)$ que supone una mejora sustancial frente al $\mathcal{O}(1/k)$ de ISTA. Este método optimizado se conoce hoy en día como FISTA. No obstante, a pesar de que esta optimización nos permite alcanzar una convergencia teórica mas rápida, en la práctica puede resultar en una evolución no monótona, que normalmente afecta la convergencia haciéndola más lenta. Para resolver estos problemas se han propuesto ciertas mejoras en la literatura. En este sentido hemos estudiado los esquemas de reinicio propuestos por O'Donogue y Candes y posteriores optimizaciones incluidas por Ito en su método FAPG.

A la hora de mostrar los efectos de estas optimizaciones presentaremos varios experimentos. Un primer experimento consistirá en datos generados de forma sintética con propósitos ilustrativos. Este experimento se centrará en el estudio de los efectos que las distintas optimizaciones tienen sobre los métodos y su ventaja sobre FISTA. En este experimento podemos ver claramente como los métodos optimizados son mejores en términos de iteraciones para converger. No obstante, un ejemplo sintético no nos permite extraer resultados concluyentes. Para ello hemos aplicado estos mismos métodos a un problema de predicción de energía eólica en Sotavento, un parque eólico situado en Galicia. En este experimento perseguimos dos objetivos. En primer lugar, comparar Group Lasso como método competitivo contra otros modelos lineales como pueden ser Lasso y Ridge Regression. Por otro lado, perseguimos replicar los experimentos del ejemplo sintético, estudiando el efecto de las distintas optimizaciones en situaciones complejas de cross validation.

En resumen, hemos observado que las optimizaciones no son solo útiles en un contexto de único experimento, sino que en el peor caso los métodos optimizados muestran un rendimiento similar a los demás, y que por tanto podemos obtener un mayor beneficio en contextos de cross validation, donde probamos muchos modelos y por tanto el coste computacional es mayor. Esto tiene que ver con la forma del problema y del término de regularización. Una penalización pequeña supone un area muy amplio para buscar la solución, y por tanto podemos esperar una optimización mayor. Por otro lado, una penalización muy grande simplifica demasiado el problema y hace que, en esencia, todos los modelos se comporten de forma similar. En términos de competitividad, Group Lasso se comporta de manera muy similar a los demás, teniendo una pequeña ventaja en expresividad cuando hablamos de la interpretabilidad final del modelo.

Abstract

Convex optimization is a fundamental theoretical core for many well known machine learning models used in day-to-day problems. Studying and getting a better understanding of these methods is an important aspect to build good models and extract proper conclusions. Furthermore, we will also focus on the study of linear models such as Lasso and Group Lasso, which incorporates several advantages, being the most important ones the cheap computational cost and the interpretability of the resulting models.

In this Master's Thesis we will first review the field of convex optimization from a pure theoretical point of view, studying concepts such as subdifferential and proximal descent minimization in order to minimize composite functions where one of its components is non differentiable. This theoretical background is the base to study ISTA, a first iterative approach to minimize such functions. To improve the convergence of ISTA, Nesterov first introduced an optimization to reach an optimal convergence ratio of $\mathcal{O}(1/k^2)$, which is a significant improvement over the $\mathcal{O}(1/k)$ convergence of ISTA. This optimized method is known as FISTA. Nonetheless, even if this optimization allows us to reach a faster theoretical convergence, in practice we see that it may not be monotone, which usually affects the convergence, making it slower in real applications. To solve these issues there have been several proposals. We have studied here some restarting schemes proposed by O'Donoghe and Candes and further optimizations introduced by Ito and colleagues that actually make FISTA faster in practice.

To show the effects of these optimizations we have ended this work by presenting some real experiments. A first experiment consists on synthetic data generated from a Gaussian distribution. This experiment focuses on exploring the effects of the different proposed optimizations and its advantage over standard FISTA. In this experiment we clearly see how the optimized methods are clearly better in terms of iterations to convergence than FISTA. Nonetheless, a synthetic experiment is not a conclusive demonstration. For this, we have also applied these methods to a real problem of wind energy prediction in Sotavento, a wind farm located in Galicia, northwest Spain. In this experiment we pursue two goals. First, to test the usefulness of the optimized methods in a complex cross-validation setup where we test many hyperparameters and the effects of such strategies. Second, to test Group Lasso as a competitive model against state of art models such as Lasso and Ridge.

In summary, we have observed that the optimizations are not only useful in a single-run setup, where the worst case shows a performance similar to standard FISTA, but in a cross-validation setup, where the benefit is actually greater. This has to do with the shape of the problem and the penalization term. A small penalization implies a very wide area to find the solution, and then a greater optimization may be expected. On the other hand, a big penalization makes the problem pretty much "straightforward", making all models behave similarly. In terms of competitiveness, we see that Group Lasso performs similarly to other methods such as Lasso, and better than Ridge, with the added advantage of a grouped structure and thus an interesting interpretation of the final model in terms of feature selection.

Acknowledgements

First of all, I would like to thank my advisor, José Ramón Dorronsoro, for giving me the opportunity to work on this and other exciting and innovative projects, and, of course, for all the time invested directing and reviewing this work. His support and guidance have lead me throughout this exciting path of research, of which this is just the beginning.

Moreover, this work has been done with support from the Cátedra UAM–IIC en Ciencia de Datos y Aprendizaje Automático. Thanks also to the Centro de Computación Científica (CCC) at UAM for using its computing facilities and to Red Eléctrica de España and Sotavento (Galicia) for kindly supplying wind energy production data. We would also thank ECMWF for their support and work making available the data we used in this work and for being constantly improving the overall quality of their services and products.

In a special place I would also like to thank my friends, specially Pablo Sánchez (my closest partner this year), Guillermo Sarasa and Sergio Sanz, with whom I've spent uncountable hours. They have been a fundamental support during the longest hours of these years.

Finally, a really special thanks to my family, who has always been an irreplaceable support, making it possible for me to become who I am now.

Chapter 1

Introduction

1.1 Motivation

Convex optimization is an essential theory core needed to solve many machine learning problems. Among these we find well known models in the literature such as Lasso and its *group* extensions, Group Lasso and so on. The solution to these models usually takes an iterative form given their particular analytical conditions, that usually involve non closed expressions. Furthermore, complex problems also involve advance mathematical techniques to solve non differentiable components, which are usually included as regularization terms in these models. As a consequence, a deep study of mathematical branches such as subdifferential calculus and proximal operators is very important to reach a proper understanding of the field.

The basic topics of convex optimization include the very well known gradient descent method, which is the most fundamental procedure to minimize smooth functions that appear in fairly simple models. Nonetheless, we cannot apply gradient descent when trying to work with more complex composite functions of the form $f + g$, where f is convex and differentiable and g is convex but not necessarily differentiable due to, precisely, the non differentiability of one of its components. To overcome this issue we study subdifferential calculus, which allows us to define the proximal operator of such functions. With these tools we can now find an iterative algorithm to solve these problems, known as ISTA.

Nonetheless, ISTA's performance is not good enough for most applications and some accelerations proposals have appeared in the literature, resulting in significant improvements over ISTA's convergence ratio. The main contribution to this was made by Nesterov. Along this line FISTA was later introduced, a fast version of the previous ISTA, as a major improvement on the field. However, for complex problems FISTA does not show in practice its theoretical advantage over ISTA, due to the possible non monotone behavior introduced by Nesterov's modifications.

Thus, the motivation of this work is to study the current state of art and review the mathematical background necessary to reach a good understanding of this field. Apart from this study, we are also motivated to study and experiment with further optimizations that address this non monotonicity issue.

1.2 Objectives

The main objective of this work is to introduce a research line focused around Nesterov's acceleration schemes. These schemes have many applications in well known machine learning models such as the ones we will study here, Lasso and Group Lasso. Nonetheless, they are being currently applied in deep neural networks theory and SVM solvers such as SMO. Based on this general purpose, in this work we pursue the following main objectives:

- To study and understand the mathematical background of smooth optimization.
- To study subdifferential calculus and proximal operators.
- To introduce the convex optimization problem and study ISTA as the first iterative algorithm to find a solution to such problems.
- To study Lasso as a first real model involving these concepts.
- To study general Nesterov's acceleration schemes and FISTA as an improvement over ISTA.
- To study the non monotone behavior of Nesterov's acceleration schemes and some proposals to avoid it such as O'Donoghue's restarting schemes and Ito's FAPG method.
- To study Group Lasso as an extension to Lasso that deals with grouped structured predictors. As far as we know this is the first application of these techniques to Group Lasso, a problem very important in many fields such as genome-based diagnosis and renewable energies.
- To implement these ideas and methods and apply them in both synthetic and real world problems and perform a comparison of all optimizations to extract some conclusions.

1.3 Structure

In this context, this work is structured around three main chapters:

1. **Proximal descent minimization**, where we introduce the original problem and develop the necessary mathematical background. We will work first with smooth functions to later introduce the convex optimization problem we are actually interested in. This chapter also describes in detail the fundamental results of subdifferential calculus and proximal operators needed to introduce ISTA, an iterative algorithm to solve composite optimization problems. At the end of the chapter we introduce Lasso, a first machine learning model that applies these ideas.
2. **Accelerated proximal descent**, where we introduce Nesterov's acceleration schemes and FISTA as a fast version of ISTA applying Nesterov's ideas. This leads us to the rippling non monotone behavior that these accelerated methods may show. To overcome this issue we present some proposals from the literate such as O'Donoghue's and Candes's [7] restarting schemes or the many optimizations included in Ito et al. [6] FAPG method. This chapter ends with the introduction of Group Lasso as a natural extension of Lasso for problems with group structured predictors.
3. **Experiments**, where we will implement and compare these methods. We will work on two different problems; a synthetic one and a real one, a wind forecasting problem, where the predictores are geographically grouped, which makes natural the application of Group Lasso. We will compare the convergence of all optimized methods against standard FISTA.

We also include a final chapter of conclusions and further work, where we hint future lines of work and more advanced applications of these methods.

Chapter 2

Proximal Descent Minimization

In this chapter we will cover a set of mathematical concepts that will be essential to develop the analysis of the models and problems that we will deal with in further sections of this work. This introduction will include a first approach to the general optimization theory and to the more specific convex optimization theory, which involves the functions we are interested in, convex functions. We will also review the field of subdifferential calculus in order to be able to solve this kind of problems, ending up with an explanation of the proximal operator, an essential tool for our analysis and algorithms. After this introduction we will present a very well-known algorithm to solve composite minimization problems, ISTA, giving a glimpse of its convergence ratio. After covering this basic theory we will present the first model, the Lasso problem, which constitutes a first application of the ideas and concepts explained in this Chapter.

The Chapter is divided into four main sections that will cover the following areas

1. We first introduce the context of general optimization theory. In Section 2.1 we cover the most fundamental results in finding the minima of differentiable functions. After this we introduce Lipschitz continuous functions to later introduce the first scheme to find the minima of a general smooth function: the gradient descent method.

2. After covering the minimization of general smooth functions we narrow our study down to convex functions, for whose minimization we need to study subdifferential calculus. This branch of calculus enables us to introduce the proximal operator, an important tool for minimizing non-smooth functions.

3. The proximal operator allows for the definition of an iterative algorithm to find the minima of composite functions, those composed of a smooth and a non-smooth functions. This algorithm is known as ISTA and Section 2.3 covers in detail its description and convergence properties.

4. In Section 2.4 of this Chapter we finally introduce the Lasso problem, a first real application of the ideas detailed in this Chapter. We will cover the solutions of this model from the perspective of the ISTA algorithm.

2.1 Introduction to General Optimization Theory

Before focusing on our specific problem regarding convex optimization theory, we will present a brief introduction to the concepts of the more general nonlinear optimization theory covered in [1]. In this context, the general form of an optimization problem is

$$\arg\min f_0(x) \quad \text{s.t.} \quad f_j(x) \leq 0, \quad j = 1, \dots, m, \quad x \in \mathcal{S}, \tag{2.1.1}$$

where $x = (x^1, \dots, x^n) \in \mathbb{R}^n$, $\mathcal{S} \subset \mathbb{R}^n$, the functions $f_0(x), \dots, f_m(x) \in \mathbb{R}$, where $f_0(x)$ is our *objective* function and the rest are the *functional constraints.* A very well known problem of this form is the machine learning model support vector machines (SVM). We define $\mathcal{S}$ as the *basic feasible* set and set

$$\mathcal{Q} = \{x \in \mathcal{S} \mid f_j \leq 0, j = 1, \dots, m\} \tag{2.1.2}$$

as the *feasible* set of the problem, which is the set of points that verify all the constraints.

Nonetheless, given that many of these problems don't have an analytical closed solution, in this Section we will describe the minimization process for unconstrained functions, given its far simpler derivation. This will serve as an introduction to the problems that we are more interested in, convex optimization problems.

2.1.1 Minima of Differentiable Functions

A great majority of nonlinear problems are usually solved by working on an approximated function of the original one and generating then a decreasing sequence of values on this function. A sequence $\{a_k\}_{k=0}^{\infty}$ is a decreasing sequence if $a_{k+1} \leq a_k$ for all $k \geq 0$. Thus, to solve our original problem, we will try to generate a decreasing sequence of values $\{f(x_{k+1}) \leq f(x_k)\}$ leading to a convergence on a minimum of $f(x)$ if the function is bounded below.

Another important concept is that of the *approximation.* In general, we say that to approximate an object is to replace it with a simpler object that is sufficiently similar to the original one. In nonlinear optimization problems we will generally apply linear and quadratic local approximations based on the derivatives of the original function. From this idea we can derive two essential methods, gradient descent and Newton method, based on the first and second order derivatives, respectively.

The first order approximation to a *continuously differentiable* function f at x can be written as

$$f(y) = f(x) + \langle y - x, \nabla f(x) \rangle + o(\|y - x\|), \tag{2.1.3}$$

where $o(r)$ is a function such that $\lim_{r \downarrow 0} \frac{1}{r} o(r) = 0$.

Consider $\mathcal{L}_f(\alpha) = \{x \in \mathbb{R}^n \mid f(x) = \alpha\}$ the level set of $f(x)$ at α and the set $\mathcal{S}_f(x)$ of tangent directions to $\mathcal{L}_f(\alpha)$ at x such that $f(x) = \alpha$ as

$$S_f(x) = \left\{ s \in \mathbb{R}^n \mid s = \lim_{y_k \to x, f(y_k) = \alpha} \frac{y_k - x}{\|y_k - x\|} \right\}. \tag{2.1.4}$$

Lemma 2.1.1. *If $s \in \mathcal{S}_f(x)$ then $\langle \nabla f(x), s \rangle = 0$.*

Proof. Since $f(y_k) = f(x)$ we have that

$$f(y_k) = f(x) + \langle \nabla f(x), y_k - x \rangle + o(\|y_k + x\|) = f(x).$$

Then we have that $\langle \nabla f(x), y_k - x \rangle + o(\|y_k - x\|) = 0$. If we divide by $\|y_k - x\|$ and take the limit when $y_k \to x$ we finally have

$$\lim_{y_k \to x} \frac{\langle \nabla f(x), y_k - x \rangle}{\|y_k - x\|} = \lim_{y_k \to x} \langle \nabla f(x), s \rangle = 0.$$

□

Consider $s \in \mathbb{R}^n$ such that $\|s\| = 1$. The local decrease in $f(x)$ following the direction s is

$$\Delta(s) = \lim_{\alpha \downarrow 0} \frac{1}{\alpha} [f(x + \alpha s) - f(x)]$$

Since $f(x + \alpha s) - f(x) = \alpha \langle \nabla f(x), s \rangle + o(\alpha)$ we have $\Delta(s) = \langle \nabla f(x), s \rangle$. Using the Cauchy-Shwartz inequality

$$-\|x\| \cdot \|y\| \leq \langle x, y \rangle \leq \|x\| \cdot \|y\|$$

we obtain

$$\begin{aligned} -\|\nabla f(x)\| \|s\| &\leq \langle \nabla f(x), s \rangle \leq \|\nabla f(x)\| \|s\| \\ \implies \Delta(s) &= \langle \nabla f(x), s \rangle \geq -\|\nabla f(x)\| \end{aligned}$$

Let us now take $s = -\nabla f(x)/\|\nabla f(x)\|$. Then

$$\Delta(s) = -\langle \nabla f(x), \nabla f(x) \rangle / \|\nabla f(x)\| = -\|\nabla f(x)\| \tag{2.1.5}$$

Thus, the *antigradient* direction, $-\nabla f(x)$, is the fastest local decreasing direction of f at x, which is an important result because all descent methods are based on reaching the fastest decreasing direction on f.

Theorem 1. *Let x^* be a local minimum of the differentiable function $f(x)$. Then we have $\nabla f(x^*) = 0$.*

Proof. Given that x^* is a local minimum of $f(x)$, then there exists an $r > 0$ such that for all $y \in \mathcal{B}_n(x^*, r)$ we have $f(y) \geq f(x^*)$ where $\mathcal{B}_n(x, r) = \{y \in \mathbb{R}^n \mid \|y - x\| \leq r\}$, that is, the ball with radius r and center x. Since f is differentiable we have

$$f(y) = f(x^*) + \langle \nabla f(x^*), y - x^* \rangle + o(\|y - x^*\|)$$

Then, for all s, $\|s\| = 1$, we have $\langle \nabla f(x^*), s \rangle = 0$. This is clear if we choose some directions s and $-s$, where we have

$$\begin{aligned} -\|s\| \cdot \|\nabla f(x^*)\| &\leq \langle \nabla f(x^*), s \rangle \leq \|s\| \cdot \|\nabla f(x^*)\| \\ -\|-s\| \cdot \|\nabla f(x^*)\| &\leq \langle \nabla f(x^*), -s \rangle \leq \|-s\| \cdot \|\nabla f(x^*)\| \end{aligned}$$

and therefore we must have $\langle \nabla f(x^*), s \rangle = 0 \quad \forall s, \|s\| = 1$.

Finally, choosing $s = e_i$, where e_i is the ith coordinate vector in $\mathbb{R}^n$ we hence have $\nabla f(x^*) = 0$. □

Nonetheless, we have just proved a necessary condition for a local minimum. The points that satisfy this condition are usually known as stationary points. An example of a function where these points are not a minimum is $f(x) = x^3$.

We now introduce the second order approximation to a *continuously twice differentiable* function $f(x)$

$$f(y) = f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} \langle f''(x)(y - x), y - x \rangle + o(\|y - x\|^2) \tag{2.1.6}$$

where $f''(x)$ is the second derivative of f at x, that is, the Hessian.

Theorem 2 (Second order optimality condition)**.** *Let x^* be a local minimum of a twice differentiable function $f(x)$. Then*

$$\nabla f(x^*) = 0, \quad \text{and} \quad f''(x) \geq 0, \tag{2.1.7}$$

that is, the Hessian is semi-positive definite.

Proof. Since x^* is a local minimum of the function $f(x)$ there exists an $r > 0$ such that for all $y \in \mathcal{B}_n(x^*, r)$ and then

$$f(y) \geq f(x^*).$$

We know that $\nabla f(x^*) = 0$, as shown before, and it follows

$$f(y) = f(x^*) + \left\langle f''(x^*)(y - x^*), y - x^* \right\rangle + o(\|y - x\|^2) \geq f(x^*)$$

Then, if we divide by $\|y - x^*\|^2$ we know that $o(\|y - x\|^2)$ tends to 0 on the limit and we are left with the desired result as follows:

$$\lim_{y \to x^*} \frac{f''(x^*)\|y - x^*\|^2}{\|y - x^*\|^2} + \frac{o(\|y - x^*\|^2)}{\|y - x^*\|^2} = f''(x^*) \geq 0$$

□

Nonetheless, this is still a necessary but not a sufficient condition for a minimum in $f(x)$. Let us prove a sufficient condition

Theorem 3. *Let $f(x)$ be a twice differentiable function in $\mathbb{R}^n$ and let x^* be a point satisfying*

$$\nabla f(x^*) = 0, \quad f''(x^*) > 0. \tag{2.1.8}$$

Then x^* is a strict local minimum of $f(x)$.

Proof. If we consider a small neighborhood around x^* in $f(x)$ we can represent $f(y)$ as

$$f(y) = f(x^*) + \frac{1}{2} \left\langle f''(x^*)(y - x^*), y - x^* \right\rangle + o(\|y - x^*\|^2).$$

Since $\frac{1}{r} o(r) \to 0$, there exists a sufficiently small $\overline{r}$ such that for all $r \in [0, \overline{r}]$ we have that

$$|o(r)| \leq \frac{r}{4} \lambda_1 f''(x^*),$$

where λ_1 is the smallest eigenvalue of $f''(x^*)$. Assuming that the Hessian is positive definite at x^* we have then that for all $y \in \mathcal{B}_2(x^*, r)$

$$\begin{aligned} f(y) &\geq f(x^*) + \frac{1}{2}\lambda_1 f''(x^*)\|y - x^*\|^2 + o(\|y - x^*\|^2) \\ &\geq f(x^*) + \frac{1}{4}\lambda_1 f''(x^*)\|y - x^*\|^2 > f(x^*) \end{aligned}$$

□

2.1.2 Lipschitz Continuous Differentiable Functions

From now on we focus on a set of functions with specific properties. This kind of functions allows us to define more interesting properties about the minimization process, that often involve the *Lipschitz condition* for a derivative of certain order of $f(x)$.

Let $\mathcal{Q} \subset \mathbb{R}^n$. We denote $C_L^{k,p}(\mathcal{Q})$ as the class of functions k times differentiable of which the pth derivative is *Lipschitz continuous* with constant L. That means that the pth derivative fulfills that

$$\|f^p(x) - f^p(y)\| \leq L\|x - y\|.$$

where $f^p(x)$ is an abbreviated form of expressing any partial

$$\frac{\partial^{\alpha_1+\ldots+\alpha_d}}{\partial x_1^{\alpha_1}\ldots\partial x_d^{\alpha_d}} f,$$

where $\alpha_1 + \ldots + \alpha_d = p$.

We are particularly interested in the class of functions with Lipschitz continuous gradient, that is, $C_L^{1,1}$. A particularly important result that we will use in further derivations is

Lemma 2.1.2. *Let $f \in C_L^{1,1}(\mathbb{R}^n)$; then we have that*

$$|f(y) - f(x) - \langle \nabla f(x), y - x\rangle| \leq \frac{L}{2}\|y - x\|^2. \tag{2.1.9}$$

Proof. For all $x, y \in \mathbb{R}^n$ we have

$$\begin{aligned} f(y) &= f(x) + \int_0^1 \langle \nabla f(x + \tau(y - x)), y - x\rangle \, d\tau \\ &= f(x) + \langle \nabla f(x), y - x\rangle + \int_0^1 \langle \nabla f(x + \tau(y - x)) - \nabla f(x), y - x\rangle \, d\tau, \end{aligned}$$

from which follows

$$\begin{aligned} |f(y) - f(x) - \langle \nabla f(x), y - x\rangle\,| &= \left|\int_0^1 \langle \nabla f(x + \tau(y - x)) - \nabla f(x), y - x\rangle \, d\tau\right| \\ &\leq \int_0^1 |\,\langle \nabla f(x + \tau(y - x)) - \nabla f(x), y - x\rangle\,| d\tau \\ &\leq \int_0^1 \|\nabla f(x + \tau(y - x)) - \nabla f(x)\| \cdot \|y - x\| d\tau \\ &\leq \int_0^1 \tau L\|y - x\|^2 d\tau = \frac{L}{2}\|y - x\|^2. \end{aligned}$$

□

2.1.3 Gradient Descent

Now that we have setup the bare minimum set of tools and concepts we need to understand the problem, comes the phase where we actually perform the optimization of the function to solve. In this case we are dealing just with *differentiable* functions, and thus the most reasonable method is the gradient method. As we mentioned before, we can easily derive two methods from the first and second order approximations; the *gradient* method and the *Newton* method, which are based on first and second order derivatives respectively. In this section we will focus on gradient descent, since the Newton method involves the calculation of the Hessian, which is much more expensive from the point of view of the computational cost.

As we just said, the simplest method to find the minimum point of a certain function f is to follow its fastest local decreasing direction, which is, as we saw, the antigradient. This simple scheme would involve two basic steps

1. Choose a starting point $x_0 \in \mathbb{R}^n$.
2. Iterate

$$x_{k+1} = x_k - h_k \nabla f(x_k), \quad k = 0, 1, \dots \tag{2.1.10}$$

   where h_k is a parameter known as the *stepsize*.

For this to be possible we need that the gradient of f is defined and continuous, that is, we assume that $f \in C_L^{1,1}$. The key of this algorithm is how to choose the step size parameter. We have three main strategies

1. Choose the $\{h_k\}_{k=0}^{\infty}$ beforehand.
2. Full relaxation:
$$h_k = \arg\min_{h \geq 0} f(x_k - h\nabla f(x_k)).$$
3. Goldstein-Armijo rule: find the $x_{k+1} = x_k - h\nabla f(x_k)$ such that
$$\begin{aligned} \alpha \langle \nabla f(x_k), x_k - x_{k+1} \rangle &\leq f(x_k) - f(x_{k+1}), \\ \beta \langle \nabla f(x_k), x_k - x_{k+1} \rangle &\geq f(x_k) - f(x_{k+1}), \end{aligned}$$
   where $0 < \alpha < \beta < 1$ are fixed parameters.

Consider now our usual problem

$$\arg\min_{x \in \mathbb{R}^n} f(x). \tag{2.1.11}$$

Take $y = x - h\nabla f(x)$ as the first step of the gradient method. We then have

$$\begin{aligned} f(y) &\leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2 \\ &= f(x) - h\|\nabla f(x)\|^2 + \frac{h^2}{2}L\|\nabla f(x)\|^2 \\ &= f(x) - h(1 - \frac{h}{2}L)\|\nabla f(x)\|^2. \end{aligned}$$

We can obtain the optimal step by minimizing $\Delta(h) = \arg\min_h -h\left(1 - \frac{h}{2}L\right) \implies h^* = \frac{1}{L}$. Hence, a single step can decrease the objective function as much as

$$f(y) \leq f(x) - \frac{1}{2L}\|\nabla f(x)\|^2 \tag{2.1.12}$$

We are now ready to estimate the performance of the gradient method. Let's start by summing the inequalities in (2.1.12) for $k = 0, \dots, N$, reaching

$$\frac{w}{L}\sum_{k=0}^{N} \|\nabla f(x_k)\|^2 \leq f(x_0) - f(x_N) \leq f(x_0) - f^*, \tag{2.1.13}$$

where w is some positive constant and f^* is the optimal value of the problem we want to minimize. As a simple conclusion we see that

$$\|\nabla f(x_k)\| \to 0 \quad \text{as} \quad k \to \infty.$$

Indeed, we can go further and get the convergence rate at which the gradient goes to zero. Let us denote

$$g_N^* = \arg\min_{0 \leq k \leq N} g_k,$$

where $g_k = \|\nabla f(x_k)\|$. In view of (2.1.13), we have

$$g_N^* \leq \frac{1}{\sqrt{N+1}}\left[\frac{1}{w}L(f(x_0) - f^*)\right]^{1/2}, \tag{2.1.14}$$

which describes the *rate of convergence* of the sequence $\{g_N^*\}$ to zero. Note that this still does not say anything about the convergence of the sequences $\{f(x_k)\}$ or $\{x_k\}$.

Recall that, even if our goal in this kind of problems is moderate, namely, we want to find a minimum on the sequence $f(x_k)$, it is generally out of reach for the described gradient method as it is. To show this, consider the following example detailed in [1].

Consider the function of two variables

$$f(x) \equiv f(x^{(1)}, x^{(2)}) = \frac{1}{2}\left(x^{(1)}\right)^2 + \frac{1}{4}\left(x^{(2)}\right)^4 - \frac{1}{2}\left(x^{(2)}\right)^2.$$

The gradient of this function is $\nabla f(x) = \left(x^{(1)}, \left(x^{(2)}\right)^3 - x^{(2)}\right)^T$. Therefore there are only three possible local minima of this function

$$x_1^* = (0,0), \quad x_2^* = (0,-1), \quad x_3^* = (0,1).$$

Computing the Hessian of this function,

$$f''(x) = \begin{pmatrix} 1 & 0 \\ 0 & 3\left(x^{(2)}\right)^2 - 1 \end{pmatrix}, \tag{2.1.15}$$

we conclude that x_2^* and x_3^* are the isolated local minima, while x_1^* is only a *stationary point* of our function. Indeed, $f(x_1^*) = 0$ and $f(x_1^* + \epsilon e_2) = \frac{\epsilon^4}{4} - \frac{\epsilon^2}{2} < 0$ for ϵ small enough.

Consider the trajectory of the gradient method, starting from $x_0 = (1,0)$. We note that the second coordinate is 0 and thus the second coordinate of the gradient is also 0. Consequently, the second coordinate of any x_k is also 0. This means that this sequence can only converge to x_1^*. Thus, without any additional assumptions on the function, it is impossible to guarantee the global convergence of the minimizing sequence to a local minimum, only to a stationary point.

Furthermore, the next theorem states the necessary assumptions for any function f in order to guarantee a global convergence to a local minimum.

Theorem 4. *Let a function $f(x)$ satisfy the following assumptions*

1. *$f \in C_M^{2,2}(\mathbb{R}^n)$.*

2. *There exists a local minimum of function f at which the Hessian is* positive definite.

3. *We know the bounds $0 < \mu \leq L < \infty$ for the Hessian at x^*:*

$$\mu I_n \leq f''(x^*) \leq L I_n. \tag{2.1.16}$$

4. *Our starting point x_0 is close enough to x^*.*

Then we have

$$r_0 = \|x_0 - x^*\| < \bar{r} = \frac{2\mu}{M}, \tag{2.1.17}$$

and then the gradient method with the optimal step size converges with the following rate [1]:

$$\|x_k - x^*\| \leq \frac{\bar{r} r_0}{\bar{r} - r_0} \left(1 - \frac{\mu}{L + \mu}\right)^k \tag{2.1.18}$$

2.2 Convex Optimization

As we have claimed at the beginning of this chapter, in practice we don't have terms that are easily differentiable but, in contrast, we have nonsmooth terms that are not differentiable at some points. For instance, a term we will use here is the well known ℓ_1 norm $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$, which is a convex but not differentiable function. Nonetheless, we will have a natural extension to minimize this kind of functions in the field of subdifferential calculus.

This generalization is based on the theorems of Fermat and Moreau-Rockafellar that we describe later on. We now need a minimization scheme valid for nonsmooth functions, as opposed to the gradient descent method that we described in the Section 2.1.3. Before describing this specific scheme to solve the nonsmooth optimization, let us review first the context of the general subdifferential calculus and a set of results that we will need in further derivations.

2.2.1 Subdifferential Calculus

We now focus on the so-called convex functions, leading thus to the field of convex optimization. Our interest in the convex optimization problems is mainly motivated because of the relevant presence that convex functions have in day-to-day machine learning problems. In this context, the objective of this section is to study subdifferential calculus, which is an extended branch of classical calculus including the notions of subdifferential and subgradient. These two concepts will make possible the analysis of nonsmooth convex functions, that constitute the main problem to solve here. We start with a few important definitions:

Definition 1 (Convex Set). *A set $C \subseteq \mathbb{R}^n$ is a convex set if for all $t \in (0, 1)$ the following condition holds*

$$tx + (1-t)y \in C, \quad \forall x, y \in C. \tag{2.2.1}$$

Intuitively, this condition states that for any two given points x, y in C, any other point in the segment that joins x, y must also be in C. Examples of convex sets are an hyperplane $H = \{x \in \mathbb{R}^n : w \cdot x - \beta = 0\}$ or a ball $B = \{x \in \mathbb{R}^n : |x - x_0| \leq \beta\}$.

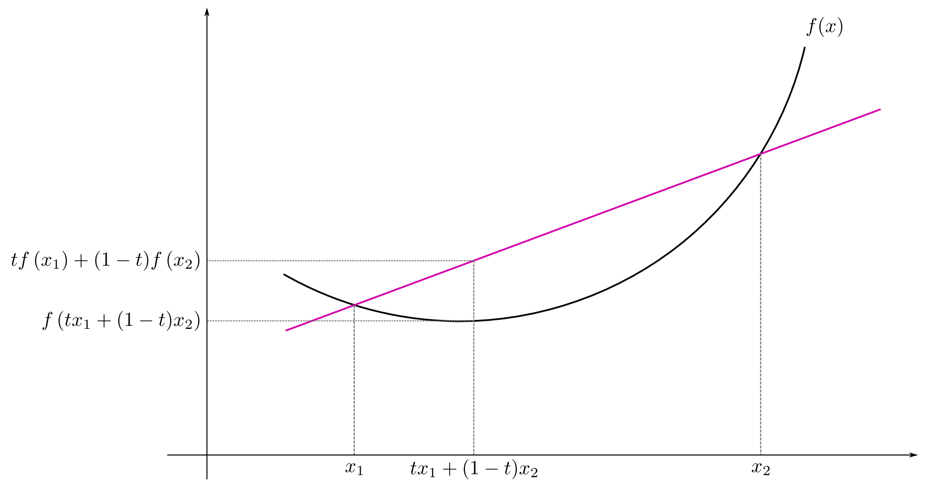


Figure 2.2.1: Convex function definition (extracted from Wikipedia)

Definition 2 (Effective Domain). *The effective domain of f is the set*

$$\mathrm{dom}(f) = \{x \in \mathbb{R}^n : f(x) < +\infty\}. \tag{2.2.2}$$

If $\mathrm{dom}(f)$ *is not empty the function is said to be proper.*

Definition 3 (Convex Function). *A convex function f is a function $f : \mathbb{R}^n \to (-\infty, \infty]$ whose* effective *domain is an* open convex set *and for which $x, y, 0 \le t \le 1$, fulfill that*

$$f(tx + (1-t)y) \le tf(x) + (1-t)f(y). \tag{2.2.3}$$

A graphical example of a convex function is shown in Figure 2.2.1. A fundamental property of convex functions is given in the following theorem

Theorem 5. *Consider a convex function f and a local minimum point $x^* \in \mathbb{R}^n$, then it is necessarily a global minimum.*

Proof. To show this property let us take a point x that is a local minimum. Hence, there must be a ball of points $\mathcal{A}$ around this point x for which we have $\forall y \in \mathcal{A}, f(x) \le f(y)$. We can then find a small $\epsilon > 0$ for which it also holds that $\forall y \in \mathbb{R}^n$ we have $(1-\epsilon)x + \epsilon y \in \mathcal{A}$. Applying the convex function definition that means that

$$f(x) \le f((1-\epsilon)x + \epsilon y) \le (1-\epsilon)f(x) + \epsilon f(y),$$

from which we can easily obtain

$$\epsilon f(x) \le \epsilon f(y) \implies f(x) \le f(y).$$

□

The core concepts of the subdifferential calculus, as we mentioned before, are *subgradient* and *subdifferential.* Their definitions are

Definition 4 (Subgradient). *A subgradient of a convex function f at $x \in \mathbb{R}^n$ is a vector $\xi \in \mathbb{R}^n$ for which $\forall y \in \mathbb{R}^n$ the following condition holds*

$$f(y) \ge f(x) + \xi \cdot (y - x). \tag{2.2.4}$$

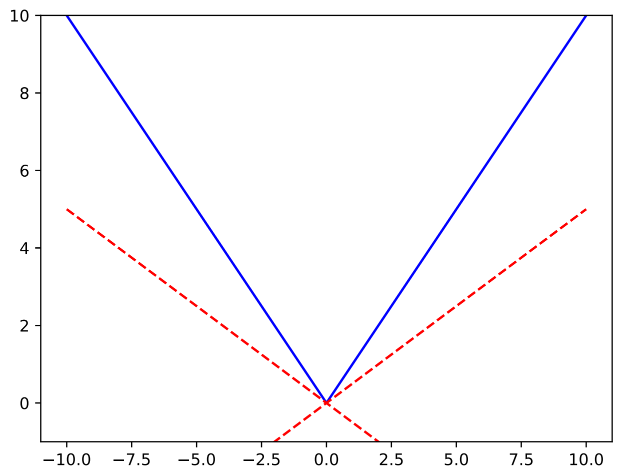


Figure 2.2.2: Graphical representation of some subgradients of $f(x) = |x|$.

That is, the function is left *above* the hyperplane defined by $\xi \cdot (x - y)$.

Definition 5 (Subdifferential)**.** *The subdifferential of a convex function f at x is the set function of subgradients $\partial f : \mathbb{R}^n \to 2^{\mathbb{R}^n}$, defined as*

$$\partial f(x) = \{\xi \in \mathbb{R}^n : f(x) + \xi \cdot (y - x) \leq f(y), \quad \forall y \in \mathbb{R}^n\}\,. \tag{2.2.5}$$

We can easily see that if the function is differentiable then we have $\partial f(x) = \{\nabla f(x)\}$. By definition, we can approximate $f(y)$ as

$$f(y) \geq f(x) + \xi \cdot (y - x),$$

where $\xi = \nabla f(x)$ iff $f(x)$ is differentiable. If we now choose $\xi = 0$ we have proved that $\partial f(x) = \{\nabla f(x)\}$.

A nonsmooth convex function that we will study here is the aforementioned ℓ_1-norm, whose form is $\|x\|_1 = \sum_{i=1}^n |x_i|$, which involves the absolute value function. We can clearly see that this is a convex function that is not differentiable at $x = 0$. Its subdifferential is

$$\partial|x| = \begin{cases} -1 & \text{if} \quad x < 0, \\ [-1, 1] & \text{if} \quad x = 0, \\ 1 & \text{if} \quad x > 0. \end{cases} \tag{2.2.6}$$

The tricky piece of this function is the case where $x = 0$. We can see that any line with slope in $[-1, 1]$ *leaves above* the function, which is, essentially, what the subgradient definition means. In Figure 2.2.2 we can see two subgradients of the absolute value function, with slopes -0.5 and 0.5. Note that we have an *infinity* number of subgradients at $x = 0$.

All the technical definitions presented in this Section are useful to define a way of minimizing a composite function that is typically composed of a smooth part $f(x)$ and a nonsmooth part $g(x)$. This problem is known as composite optimization. Nonetheless, we need yet another theorem that allows us to compute the subgradient of the sum of two given functions. Fortunately, one of the key results of this field includes a theorem that allows us to divide the subdifferential of this function into its two components.

Theorem 6. Moreau-Rockafellar *Let f, g be convex functions defined in $\mathbb{R}^n$. Then, $\forall x_0 \in \mathbb{R}^n$ the following condition holds*

$$\partial f(x_0) + \partial g(x_0) \subset \partial (f+g)(x_0). \tag{2.2.7}$$

If int dom $f \,\cap$ dom $g \neq \emptyset$ *we also have that $\partial(f+g)(x_0) \subset \partial f(x_0) + \partial g(x_0)$.*

Proof. We will divide the proof into two parts, one for proving the each part of the inclusion, as presented in [2].

For the proof of the first inclusion we choose $\xi_1 \in \partial f(x_0)$ and $\xi_2 \in \partial g(x_0)$ for which we have $\forall x \in \mathbb{R}^n$,

$$\begin{aligned} f(x) &\geq f(x_0) + \xi_1 \cdot (x - x_0), \\ g(x) &\geq g(x_0) + \xi_2 \cdot (x - x_0), \end{aligned}$$

By summing these inequalities we end up with the following expression

$$f(x) + g(x) \geq f(x_0) + g(x_0) + (\xi_1 + \xi_2) \cdot (x - x_0),$$

and therefore $(\xi_1 + \xi_2) \subset \partial(f+g)(x_0)$, proving the first part of the theorem.

To prove the second part, let $\xi \in \partial(f+g)(x_0)$. First, we observe that $f(x_0) = +\infty$ implies $(f+g)(x_0) = +\infty$, whence $f + g \equiv +\infty$, which is impossible by $\xi \in \partial(f+g)(x_0)$. Analogously, $g(x_0) = +\infty$ is impossible. Hence, we now that both $f(x_0)$ and $g(x_0)$ belong to $\mathbb{R}$. We form the following two sets in $\mathbb{R}^{n+1}$.

$$\begin{aligned} \Lambda_f &:= \left\{ (x - x_0, y) \in \mathbb{R}^n \times \mathbb{R} : y > f(x) - f(x_0) - \xi \cdot (x - x_0) \right\}, \\ \Lambda_g &:= \left\{ (x - x_0, y) : -y \geq g(x) - g(x_0) \right\}. \end{aligned}$$

Note that both sets are nonempty and convex. From $\xi \in \partial(f+g)(x_0)$ follows that $\Lambda_f \cap \Lambda_g = \emptyset$. Hence, by the set-set-separation Theorem [2], there exists $(\xi_0, \mu), (\xi_0, \mu) \neq (0,0) \in \mathbb{R}^{n+1}$ and $\alpha \in \mathbb{R}$, such that

$$\begin{aligned} \xi_0 \cdot (x - x_0) + \mu y \leq \alpha \quad &\text{for all } (x,y) \text{ with } y > f(x) - f(x_0) - \xi \cdot (x - x_0), \\ \xi_0 \cdot (x - x_0) + \mu y \geq \alpha \quad &\text{for all } (x,y) \text{ with } -y \geq g(x) - g(x_0). \end{aligned}$$

By $(0,0) \in \Lambda_g$ we get $\alpha \leq 0$. But also $(0, \epsilon) \in \Lambda_f$ for every $\epsilon > 0$, and this gives $\mu\epsilon < \alpha$, so $\mu \leq 0$ (take $\epsilon = 1$). In the limit, for $\epsilon \to 0$, we find $\alpha \geq 0$. Hence $\alpha = 0$ and $\mu \leq 0$. We now claim that $\mu = 0$ is impossible. Indeed, if one had $\mu = 0$, then the first of the above two inequalities would give

$$\xi_0 \cdot (x - x_0) \leq 0 \quad \text{for all } (x,y) \text{ with } y > f(x) - f(x_0) - \xi \cdot (x - x_0),$$

which is equivalent to

$$\xi_0 \cdot (x - x_0) \leq 0 \quad \text{for all } x \in \text{dom } f,$$

because we can essentially choose any arbitrarily large y such that $y > f(x) - f(x_0) - \xi \cdot (x - x_0)$. Similarly, the second inequality would give

$$\xi_0 \cdot (x - x_0) \geq 0 \quad \text{for all } x \in \text{dom } g.$$

In particular, for x as above this would imply $\xi_0 \cdot (x - x_0) = 0$. But since x lies in the interior of dom f (so for some $\delta > 0$ the ball $N_\delta(\tilde{x})$ belongs to dom f), the preceding would imply

$$\xi_0 \cdot u = \xi_0 \cdot (\tilde{x} + u - x_0) \leq 0 \quad \text{for all } u \in N_\delta(0).$$

This would give $\xi_0 = 0$ which would be in contradiction to $(\xi_0, \mu) \neq (0, 0)$. Hence, we conclude $\mu < 0$. Dividing the separation inequalities by $-\mu$ and setting $\overline{\xi}_0 := -\xi_0/\mu$, this results in

$$\begin{aligned}
\overline{\xi}_0 \cdot (x - x_0) \leq y \quad & \text{for all } (x, y) \text{ with } y > f(x) - f(x_0) - \xi \cdot (x - x_0), \\
\overline{\xi}_0 \cdot (x - x_0) \geq y \quad & \text{for all } (x, y) \text{ with } -y \geq g(x) - g(x_0).
\end{aligned}$$

The last inequality gives $-\overline{\xi}_0 \in \partial g(x_0)$ and the one but last inequality gives $\xi + \overline{\xi}_0 \in \partial f(x_0)$. Since $\xi = (\xi + \overline{\xi}_0) - \overline{\xi}_0$, this finishes the proof. □

Another fundamental result here is the Fermat rule. The Fermat rule is known for its implications in the field of classical and subdifferential calculus. We present its versions for both the differentiable and non-differentiable cases next.

Theorem 7. Fermat Rule (differentiable case) *For a convex and differentiable function f we have*

$$\arg\min f = \{x \in \mathbb{R}^n \mid \nabla f(x) = 0\}. \tag{2.2.8}$$

This first definition gives us a very explicit way to calculate the minimum point of a given function by calculating its gradient and solving the equation $\nabla f(x) = 0$. The non-differentiable case, on the other hand, will not give us such an explicit tool.

Theorem 8. Fermat Rule (non-differentiable case) *For a convex function f we have*

$$\arg\min f = \{x \in \mathbb{R}^n \mid 0 \in \partial f(x)\}. \tag{2.2.9}$$

Proof. If $0 \in \partial f(x)$ we have that

$$f(y) \geq f(x) + \xi \cdot (y - x),$$

where $\xi = 0$ and then $f(y) \geq f(x) \quad \forall y \in \mathbb{R}^n$. □

To reach a more explicit equation to find this point we will need the proximal operator, that we explain next. The implications of these two theorems (Moreau-Rockafellar and Fermat Rule) are very important to the field of convex optimization because they give a formal condition of the minimum of a composite function f that may not be necessarily smooth keeping the intuitiveness of classical calculus.

2.2.2 The Proximal Operator

Nonetheless, although the general analysis of the gradient method may be useful, we cannot directly apply it to our usual problems due to the non-differentiability of one of its components. Then, we have to define a more powerful tool that makes us able to deal with non-differentiable components in our problem: the *proximal operator*. Actually, this new minimization scheme should behave in a similar fashion as the gradient scheme, and should then perform some sort of descent along a decreasing direction of the objective function. Before actually defining this operator we need a set of technical definitions

Definition 6 (Monotone Operator)**.** *An operator $T : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is monotone if it fulfills the following condition*

$$(\xi_1 - \xi_2) \cdot (x_1 - x_2) \geq 0, \quad \forall x_1, x_2 \in \mathbb{R}^n, \quad \xi_1 \in T(x_1), \xi_2 \in T(x_2). \tag{2.2.10}$$

Lemma 2.2.1. *If T is a monotone operator then T^{-1} is also monotone*

Proof. To prove that the inverse of an operator is also monotone we consider the graph of an operator T

$$G(T) : \{(x, \xi) : \xi \in T(x)\}$$

and define T^{-1} through its graph

$$G(T^{-1}) = \{(\xi, x) : (x, \xi) \in G(T)\} \quad \text{with} \quad \xi \in T(x),$$

which, in other words, means that $(\xi, x) \in G(T^{-1})$ iff $\xi \in T(x)$. Then, we can see that the monotonicity of T^{-1} follows since $x \in T^{-1}(\xi) \iff \xi \in T(x)$. □

Lemma 2.2.2. *If T is a monotone operator then αT is also monotone $\forall \alpha > 0$.*

Proof. Let us have $\eta_1 \in \alpha T(x_1)$ and $\eta_2 \in \alpha T(x_2)$. This means that $\eta_1 = \alpha \xi_1$ for $\xi_1 \in T(x_1)$ and the same holds for η_2. By applying the definition of monotone operator we have that

$$(\eta_1 - \eta_2) \cdot (x_1 - x_2) \geq 0 \implies \alpha(\xi_1 - \xi_2) \cdot (x_1 - x_2) \geq 0.$$

□

Proposition 1. *The subdifferential is a monotone operator.*

Proof. Suppose $\xi_1 \in \partial f(x_1)$ and $\xi_2 \in \partial f(x_2)$, where f is a convex function. Then we have

$$\begin{aligned} f(x_1) &\geq f(x_2) + \xi_1 \cdot (x_1 - x_2) \\ f(x_2) &\geq f(x_1) + \xi_2 \cdot (x_2 - x_1) \end{aligned}$$

If we now sum these inequalities we get

$$\begin{aligned} f(x_1) - f(x_2) \geq f(x_2) - f(x_1) - (\xi_1 + \xi_2) \cdot (x_1 - x_2), \\ \implies (\xi_1 - \xi_2) \cdot (x_1 - x_2) \geq 0 \end{aligned}$$

which fulfills the definition of monotone operator. □

Theorem 9. *The resolvent operator $R_T = (I + T)^{-1}$ is univalued and firmly nonexpansive, that is, it fulfills that*

$$\langle R_T(x_1) - R_T(x_2), x_1 - x_2 \rangle \geq \|R_T(x_1) - R_T(x_2)\|^2, \quad \forall x_1, x_2 \in \mathbb{R}^n. \tag{2.2.11}$$

Proof. To prove that the resolvent operator is single valued, let us have $(Z, x_1), (Z, x_2) \in G(R_T)$. The we have that $Z \in (I + T)(x_1) = x_1 + T(x_1)$, i.e., $Z \in x_2 + T(x_2)$ and there are $\xi_i \in T(x_i)$ such that $Z = x_1 + \xi_1 = x_2 + \xi_2$, i.e., $\xi_1 - \xi_2 = -(x_1 - x_2)$. But since T is monotone it follows that

$$0 \leq (\xi_1 - \xi_2) \cdot (x_1 - x_2) = -\|x_1 - x_2\|^2$$

and therefore we must have $x_1 = x_2$.

To prove the firmly non-expansiveness let us define $x_i = R_T(Z_i)$. Then $Z_i \in x_i + T(x_i)$ and $Z_i = x_i + \xi_i$ with $\xi_i \in T(x_i)$. Now it follows that

$$\begin{aligned}(x_1 - x_2) \cdot (Z_1 - Z_2) &= (x_1 - x_2) \cdot (x_1 - x_2) + (x_1 - x_2) \cdot (\xi_1 - \xi_2) \\ &= \|x_1 - x_2\|^2 + (x_1 - x_2) \cdot (\xi_1 - \xi_2),\end{aligned}$$

and since T is monotone, firmly non-expansiveness of R_T follows.

□

Definition 7 (Proximal Operator)**.** *The proximal operator of a function f is defined as the resolvent of the subdifferential and is denoted by* $\text{prox}_f = (I + \partial f)^{-1}$.

We can see that the previous definition of the Proximal Operator is not particularly useful when it comes to actually solve a problem. For this, we can find a more operative definition of the Proximal Operator.

Definition 8 (Alternative definition of the Proximal Operator)**.**

$$\text{prox}_f(z) = \underset{y \in \mathbb{R}^n}{\arg\min} \left(f(y) + \frac{1}{2}\|z - y\|^2 \right). \tag{2.2.12}$$

We now prove the equality of both definitions

Proposition 2. *Both definitions for the proximal operator are equivalent.*

Proof. Consider $g(y) = f(y) + \frac{1}{2}\|z - y\|^2$ and $x = \arg\min g(y)$. If x is the minimizer of $g(y)$ we have that

$$0 \in \partial g(x) = x - z + \partial f(x).$$

This implies that

$$\begin{aligned} z &\in x + \partial f(x) = (I + \partial f)(x) \\ &\implies x = (I + \partial f)^{-1}(z) \\ &\implies x = \text{prox}_g(z) \end{aligned}$$

by applying the original definition of the proximal operator.

□

As we mentioned before, the purpose of introducing the proximal operator is to be able to solve composite optimization problems, where the final function $F(x)$ is composed by a smooth function $f(x)$ and a nonsmooth function $g(x)$. In this context, and thanks to the theorem of Moreau-Rockafellar, we know that minimizing the sum of two convex functions translates to finding the point that verifies $0 \in \nabla f(x) + \partial g(x)$.

The solution to this problem is thus $x = \text{prox}_{\gamma g}(x - \gamma \nabla f(x))$, taking into account that

$$0 \in \nabla f(x) + \partial g(x) \iff -\nabla f(x) \in \partial g(x) \tag{2.2.13}$$

If we introduce a term γ, since we know that γT, where T is a monotone operator, is still a monotone operator, we get

$$\begin{aligned} x - \gamma \nabla f(x) \in x + \gamma \partial g(x) &= (I + \gamma \partial g)(x) \\ &\implies x = (I + \gamma \partial g)^{-1}(x - \gamma \nabla f(x)) \\ &\implies x = \text{prox}_{\gamma g}(x - \gamma \nabla f(x)) \end{aligned} \tag{2.2.14}$$

and this finally leads us to the next section, where we will introduce an algorithm to compute the solution of the aforementioned composite minimization problem, based on the application of the proximal operator.

2.3 ISTA Algorithm

In this section we present a very well known algorithm, ISTA, for solving proximal problems of the form $F \equiv f + g$. In Section 3.2 we will introduce a fast version of this algorithm. The previous section gave us an interesting tool to solve problems that involve a nonsmooth component, the proximal operator. From equation (2.2.14) we see that we need to find a fixed point of the proximal operator. The simplest idea that comes to our mind to find a fixed point in any sequence is to iterate the expression until we reach such a point. The basic iteration would be then the application of this equation in the following way

$$x_{k+1} = \text{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k)), \tag{2.3.1}$$

which amounts to finding the fixed point of the proximal, leading us directly to the ISTA algorithm. The key of the ISTA algorithm is not only to propose such iterative method but to prove its convergence.

Let us recall the general problem that we are trying to solve.

$$\underset{x \in \mathbb{R}^n}{\arg\min} \left\{ F(x) \equiv f(x) + g(x) \right\}, \tag{2.3.2}$$

where $f(x) \in C_L^{1,1}(\mathbb{R}^n)$ is convex and $g(x)$ is convex but not necessarily differentiable.

Let's start with the quadratic approximation to this function at a point y

$$Q_L(x, y) := f(y) + \langle x - y, \nabla f(y) \rangle + \frac{L}{2} \|x - y\|^2 + g(x), \tag{2.3.3}$$

where we leave the non differentiable term without any modification. The minimizer of this problem is

$$p_L(y) := \underset{x \in \mathbb{R}^n}{\arg\min} \left\{ Q_L(x, y) \right\}. \tag{2.3.4}$$

We can see that by completing squares this can be equally written as

$$\begin{aligned} p_L(y) &= \arg\min_x \left\{ g(x) + \frac{L}{2} \left\| x - \left(y - \frac{1}{L}\nabla f(y) \right) \right\|^2 \right\} \\ &= \arg\min_x \left\{ \frac{1}{L} g(x) + \frac{1}{2} \left\| x - \left(y - \frac{1}{L}\nabla f(y) \right) \right\|^2 \right\}, \end{aligned}$$

which is the proximal operator as we alternatively defined it in equation (2.2.12) evaluated at the point $y - \frac{1}{L}\nabla f(y)$. Before proceeding to the analysis of ISTA we need the following results from [3].

Lemma 2.3.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function with Lipschitz continuous gradient and Lipschitz constant L_f. Then, for any $L \geq L_f$,*

$$f(x) \leq f(y) + \langle x - y, \nabla f(y) \rangle + \frac{L}{2}\|x - y\|^2 \text{ for every } x, y \in \mathbb{R}^n, \tag{2.3.5}$$

and furthermore

$$F(x) \leq Q_L(x, y) \quad \forall y \in \mathbb{R}^n. \tag{2.3.6}$$

Lemma 2.3.2. *For any $y \in \mathbb{R}^n$, one has $z = p_L(y)$ if and only if there exists a $\gamma \in \partial g(z)$, the subdifferential of $g(\cdot)$, such that*

$$\nabla f(y) + L(z - y) + \gamma = 0. \tag{2.3.7}$$

Proof. If $z = p_L(y)$, we have that

$$0 \in \nabla f(y) + L(x - y) + \partial g(x)$$

and then there must exist a $\gamma \in \partial g(x)$ for which

$$0 = \nabla f(y) + L(z - y) + \gamma.$$

□

We can now show that

Lemma 2.3.3. *Let $y \in \mathbb{R}^n$ and $L > 0$ such that*

$$F(p_L(y)) \leq Q_L(p_L(y), y). \tag{2.3.8}$$

Then, for any $x \in \mathbb{R}^n$, it follows that

$$F(x) - F(p_L(y)) \geq \frac{L}{2}\|p_L(y) - y\|^2 + L\langle y - x, p_L(y) - y \rangle \tag{2.3.9}$$

Proof. From (2.3.8) we have that

$$F(x) - F(p_L(y)) \geq F(x) - Q_L(p_L(y), y) \tag{2.3.10}$$

Since f, g are convex we have

$$\begin{aligned} &f(x) \geq f(y) + \langle x - y, \nabla f(y) \rangle \\ &g(x) \geq g(p_L(y)) + \langle x - p_L(y), \gamma \rangle, \end{aligned}$$

Algorithm 1: ISTA algorithm with constant step

```
Input          : $f$ convex with $\nabla f$ Lipschitz constant $L$; $g$ convex
Output         : $x_{k+1} \simeq \arg\min_{x\in\mathbb{R}^{\times}} \{f(\mathbf{x}) + g(\mathbf{x})\}$
Initialization: $x_0 \in \mathbb{R}^n$
for $k = 1, \dots$ do
    $x_{k+1} \leftarrow \operatorname{prox}_{\frac{1}{L}g}\left(\mathbf{x}_{k-1} - \frac{1}{L}\nabla f(\mathbf{x}_{k-1})\right)$     (2.3.13)
end
```

where γ verifies (2.3.7), that is

$$\gamma = -\nabla f(y) + L(y - p_L(y)).$$

Summing these inequalities we obtain

$$F(x) \geq f(y) + \langle x - y, \nabla f(y)\rangle + g(p_L(y)) + \langle x - p_L(y), \gamma\rangle . \tag{2.3.11}$$

On the other hand, by the definition of Q_L at $p_L(y)$ we have

$$Q_L(p_L(y), y) = f(y) + \langle p_L(y) - y, \nabla f(y)\rangle + \frac{L}{2}\|p_L(y) - y\|^2 + g(p_L(y)). \tag{2.3.12}$$

Thus, introducing (2.3.12) and (2.3.11) in (2.3.10) it follows that

$$\begin{aligned}
F(x) - F(p_L(y)) &\geq f(y) + \langle x - y, \nabla f(y)\rangle + g(p_L(y)) + \langle x - p_L(y), \gamma\rangle \\
&\quad - f(y) - \langle p_L(y) - y, \nabla f(y)\rangle - \frac{L}{2}\|p_L(y) - y\|^2 - g(p_L(y)) \\
&\geq -\frac{L}{2}\|p_L(y) - y\|^2 + \langle x - p_L(y), \nabla f(y) + \gamma\rangle \\
&= -\frac{L}{2}\|p_L(y) - y\|^2 + L\,\langle x - p_L(y), y - p_L(y)\rangle \\
&= \frac{L}{2}\|p_L(y) - y\|^2 + L\,\langle y - x, p_L(y) - y\rangle .
\end{aligned}$$

See [3], Lemma 2.3.3, for more details. □

The pseudocode for the standard ISTA algorithm is shown in Algorithm 1. Nonetheless, even if this is the standard case, we note that we may not know the actual Lipschitz constant L_f. In this case we also note from Lemma 2.3.1 that for any $L \geq L_f$ the condition (2.3.8) is always fulfilled for $p_L(y)$. This implies that for any $L \geq L_f$ we are *decreasing* the objective's value each time we iterate the proximal operator, which is the goal in a minimization scheme. Then, in this case, when we don't know the actual Lipschitz constant, we will need to use an estimate. To obtain this approximation we will have to make sure that our estimation L_k for the kth iteration fulfills that $L_k \geq L$ and the condition (2.3.8). This strategy is known as *backtracking.* We obtain the following remark.

Remark 2.3.1. *Since inequality* (2.3.8) *is satisfied for any* $L \geq L_f$*, where* L_f *is the Lipschitz constant of* ∇f*, it follows that for ISTA we can define a backtracking strategy in which we have* $L_k \leq \eta L_f$ *for every* $k \geq 1$*. Overall,*

$$\beta L_0 \leq L_k \leq \eta L_f, \tag{2.3.14}$$

```
Algorithm 2: ISTA algorithm with backtracking
  Input         : $f$ convex; $g$ convex
  Output        : $x_{k+1} \simeq \arg\min_{x\in\mathbb{R}^n} \{f(\mathbf{x}) + g(\mathbf{x})\}$
  Initialization: Take $L_0 > 0$, some $\eta > 1$ and some $x_0 \in \mathbb{R}^n$
  for $k = 1, \dots$ do
      Find smallest nonnegative integers $i_k$ such that with $\bar{L} = \eta^{i_k} L_{k-1}$ we have
          $F(p_{\bar{L}}(x_{k-1})) \leq Q_{\bar{L}}(p_{\bar{L}}(x_{k-1}), x_{k-1})$    (2.3.15)
      Set $L_k = \eta^{i_k} L_{k-1}$
          $x_{k+1} \leftarrow \text{prox}_{\frac{1}{L_k} g}\left(\mathbf{x}_{k-1} - \frac{1}{L}\nabla f(\mathbf{x}_{k-1})\right)$    (2.3.16)
  end
```

Essentially, this amounts to finding a proper η for which our estimation of L_k verifies the condition (2.3.8). The pseudocode of the ISTA algorithm for the backtracking strategy is shown in Algorithm 2.

We can now prove the convergence rate of ISTA, given by the following theorem [3].

Theorem 10. ISTA Convergence Ratio *Let $\{x_k\}$ the sequence generated by either version of the ISTA algorithm. Then, for any $k \geq 1$*

$$F(x_k) - F(x^*) \leq \frac{\alpha L \|x_0 - x^*\|^2}{2k}, \tag{2.3.17}$$

where $\alpha = 1$ for the constant stepsize setting (in the case of known Lipschitz constant) and $\alpha = \eta$ for the backtracking setting.

Proof. We follow the proof in [3], Theorem 3.1. We first invoke Lemma 2.3.3 with $x = x^*, y = x_n$ and $L = L_{n+1}$; then we have that

$$\begin{aligned}\frac{2}{L_{n+1}}(F(x^*) - F(x_{n+1})) &\geq \|x_{n+1} - x_n\|^2 + 2\langle x_n - x^*, x_{n+1} - x_n\rangle \\ &= \|x^* - x_{n+1}\|^2 - \|x^* - x_n\|^2,\end{aligned}$$

which using (2.3.14) and the fact that $F(x^*) - F(x_{n+1}) \leq 0$ gives

$$\frac{2}{\alpha L_f}(F(x^*) - F(x_{n+1})) \geq \|x^* - x_{n+1}\|^2 - \|x^* - x_n\|^2.$$

Summing this inequality over $n = 0, \dots, k$ leads us to

$$\frac{2}{\alpha L_f}\left(kF(x^*) - \sum_{n=0}^{k-1} F(x_{n+1})\right) \geq \|x^* - x_k\|^2 - \|x^* - x_0\|^2. \tag{2.3.18}$$

Again, invoking Lemma 2.3.3 with $x = y = x_n$ and $L = L_{n+1}$ we obtain

$$\frac{2}{L_{n+1}}(F(x_n) - F(x_{n+1})) \geq \|x_n - x_{n+1}\|^2.$$

Since $L_{n+1} \geq \beta L_f$ (because of (2.3.14)) it follows that

$$\frac{2}{\beta L_f}(F(x_n) - F(x_{n+1})) \geq \|x_n - x_{n+1}\|^2 .$$

By multiplying this inequality by n and summing over $n = 0, \dots, k$ we get

$$\begin{aligned} &\frac{2}{\beta L_f}\sum_{n=0}^{k-1} nF(x_n) - \sum_{n=1}^{k} nF(x_n) + \sum_{n=1}^{k} F(x_n) \geq \sum_{n=0}^{k-1} n\|x_n - x_{n+1}\|^2 \\ \implies &\frac{2}{\beta L_f}\sum_{n=0}^{k-1}\Big\{nF(x_n) - (n+1)F(x_{n+1}) + F(x_{n+1})\Big\} \geq \sum_{n=0}^{k-1} n\|x_n - x_{n+1}\|^2. \end{aligned}$$

which can be simplified as

$$\frac{2}{\beta L_f}\left(-kF(x_k) + \sum_{n=0}^{k-1} F(x_{n+1})\right) \geq \sum_{n=0}^{k-1} n\|x_n - x_{n+1}\|^2. \tag{2.3.19}$$

Summing Equations (2.3.19) and (2.3.18) times β/α we finally reach

$$\frac{2k}{\alpha L_f}(F(x^*) - F(x_k)) \geq \|x^* - x_k\|^2 + \frac{\beta}{\alpha}\sum_{n=0}^{k-1} n\|x_n - x_{n+1}\|^2 - \|x^* - x_0\|^2$$

obtaining the final convergence ratio

$$F(x_k) - F(x^*) \leq \frac{\alpha L_f \|x^* - x_0\|^2}{2k}$$

$\square$

This convergence ratio is called *sublinear*. This means that to reach a solution within a tolerance ϵ we need at most $\lceil C/\epsilon \rceil$ iterations, where C is the constant numerator on the convergence ratio.

2.4 The Lasso Problem

We present now the first machine learning model we will work with, the Lasso problem [4, 5]. The Lasso model is a regularized linear regression model that imposes a ℓ_1 norm penalty on the coefficients to avoid overfitting and to improve the interpretability of the model. In contrast to other regularized linear regression models, such as Ridge Regression, the ℓ_1 norm penalty term makes things a bit more difficult due to its non-differentiability. Nonetheless, after the introduction to the concepts of subdifferential calculus and the proximal operator, we are now able to deal with it.

The original formulation of the problem, as presented in [5] is often written as

$$\arg\min_{\beta \in \mathbb{R}^n}\left\{\frac{1}{2N}\|y - X\beta\|^2 + \lambda\|\beta\|_1\right\} \quad \text{s.t.} \quad \lambda \geq 0, \tag{2.4.1}$$

where we have standardized the features and the target to get rid of the β_0 term, known as the *intercept*, to ease the derivations.

We note that the Lasso formulation fits very well the kind of problems we have been dealing with in a more abstract fashion during this Chapter. Specifically, the Lasso problem can be seen as a composite problem, where $f = \frac{1}{2N}\|y - X\beta\|^2$ and $g = \|\beta\|_1$. In this context, we want to solve the following problem

$$\beta^* = \arg\min f(\beta) + g(\beta). \tag{2.4.2}$$

As we have seen in the last Section, to find the minimum of the sum of these two functions is the same as finding the fixed point of the proximal operator applied to the sum, which amounts to iterating the ISTA algorithm

$$x_{k+1} = \operatorname{prox}_{\frac{1}{L}g}\left(x_k - \frac{1}{L}\nabla f(x_k)\right). \tag{2.4.3}$$

In the specific case of Lasso, we can calculate the proximal operator from the subdifferential of the ℓ_1 penalty term, which is

$$\partial|\beta_j| = \begin{cases} -1 & \text{if} \quad \beta_j < 0, \\ [-1,1] & \text{if} \quad \beta_j = 0, \\ 1 & \text{if} \quad \beta_j > 0. \end{cases} \tag{2.4.4}$$

Applying the definition of the proximal operator we find that

$$(I + \lambda\partial f)(\hat\beta) = \begin{cases} \hat\beta + \lambda & \text{if} \quad \hat\beta > 0, \\ [-\lambda,\lambda] & \text{if} \quad \hat\beta = 0, \\ \hat\beta - \lambda & \text{if} \quad \hat\beta < 0, \end{cases}$$

$$\hat\beta = \begin{cases} \hat\beta + \lambda & \text{if} \quad \hat\beta < -\lambda, \\ 0 & \text{if} \quad \hat\beta \in [-\lambda,\lambda], \\ \hat\beta - \lambda & \text{if} \quad \hat\beta > \lambda. \end{cases} \tag{2.4.5}$$

This can be written in a more compact form

$$S_\lambda = \operatorname{sign}(x)(|x| - \lambda)_+ \tag{2.4.6}$$

which is known as the *soft-thresholding* operator.

Thus, we see that solving the Lasso problem by application of the ISTA algorithm translates to iterate the following expression until convergence

$$x_{k+1} = S_\lambda\left(x_k - \frac{1}{L}\nabla f(x_k)\right) \tag{2.4.7}$$

A very interesting advantage of the ISTA algorithm is that it gives a generic formulation that allows us to solve different proximal problems very easily, simply by changing the proximal operator. This advantage often contrasts with the probably better efficiency that specific algorithms for some problems may have. In the case of Lasso there exists an alternative specific algorithm, known as *coordinate descent.* We describe it next.

To solve the Lasso problem by applying the coordinate descent algorithm let us first write the zero-point subgradient equations in the following way

$$\frac{1}{N}\langle x_j, y - x\beta\rangle + \lambda s_j = 0, \quad j = 1, \ldots, p \tag{2.4.8}$$

where p is the number of features and where s_j is an element of the subdifferential of the $|\beta_j|$. Note that we are taking a specific element of the subdifferential $\partial g(\beta)$, s_j.

To derive the full solutions to the Lasso problem we first assume that we have a single predictor z, which is a far simpler case. For this case, we recover the proximal equation in (2.4.5), the soft-thresholding operator.

Going back to the system we can see that the actual solutions are

$$\hat{\beta} = S_\lambda \left(\frac{1}{N} \langle z, y \rangle \right) \tag{2.4.9}$$

by applying the soft-thresholding operator we defined earlier.

In the more general case, of multiple predictors, we can express the solution of $\hat{\beta}_j$ as an iterative process fixing at each iteration one of the predictors. We then get an iterative solution where the $\hat{\beta}_j$ is solved as

$$\hat{\beta}_j = S_\lambda \left(\frac{1}{N} \left\langle x_j, r^{(j)} \right\rangle \right) \tag{2.4.10}$$

where the $r^{(j)} = y_i - \sum_{k \neq j}^{p} x_{ik} \hat{\beta}_k$ are the partial residuals fixing all but the jth predictor.

The Lasso model has been widely studied and has received some recent attention in the literature, mainly because of its cheap computational cost and its good general results for many applications, among which we find many related to energy prediction problems, a field in which we have seen that Lasso is a very competitive model [14, 16]. Furthermore, there are a lot of different models that somehow derive from Lasso, such as Elastic Net, which employs a combination of both the ℓ_1- and ℓ_2-norms in order to avoid the erratic behavior that Lasso may show in case of highly correlated features. Another kind of derived models comes from a generalization of the Lasso, where the penalty term is modified by applying a linear operator to it. In this case we can extract multiple models, and one of the most famous is Fused Lasso, particularly useful in applications of image denoising or signal approximation. In Chapter 3 we will introduce another extension of this model known as Group Lasso, which is useful when our features are somehow structured in a grouped fashion.

2.5 Conclusions

In this Chapter we have reviewed some important and interesting concepts. The main ones are summarized next.

- We have reviewed the general nonlinear optimization theory, focusing on some interesting properties of the minimum of a general nonlinear function. We explored the gradient method as the most common minimization scheme for smooth setups.

- From this we have come to study the subdifferential calculus as an extension applicable to convex functions, including the notions of subgradient and subdifferential, allowing us to deal with nonsmooth terms, that usually arise in complex machine learning models.

- This has lead us to present the proximal operator as a tool to solve optimization problems that involve these nonsmooth terms. This new scheme generalizes the common gradient descent strategy carried out in schemes of smooth optimization.

- The proximal operator is the main idea behind the ISTA algorithm. We have first presented this algorithm along with the backtracking strategy, which comes in very useful when we do not know the Lipschitz constant of the function. We have then analyzed its convergence ratio, highlighting the importance of this result. We will come back later to this algorithm to introduce an improved version, known as FISTA.

- Finally, we have described the Lasso problem as a first application of the ideas described in this Chapter, ending up with a solution by applying the ISTA algorithm. We have also analyzed and presented an alternative algorithm to solve the problem, the cyclic coordinate descent algorithm.

Chapter 3

Accelerated Proximal Descent

In the previous Chapter we described some very foundational concepts covering the basic theory and tools we need in order to understand more advanced topics regarding convex optimization theory. This Chapter focuses on presenting these topics, that mainly evolve around the ideas introduced by Nesterov to accelerate gradient descent methods and thus reach an optimal convergence ratio in the algorithms that solve these problems. In short, these Nesterov methods are all based upon the idea of *momentum*, in which we will not perform a gradient step directly on the previous point but instead on a smartly chosen combination of previous points known as the momentum point.

This Chapter is divided into the next main sections that will cover the following areas:

1. In Section 3.1 we present a brief introduction to Nesterov acceleration schemes, where we will give a general notion of these methods by describing some important intuitions about the minimization process and the effects that these accelerated methods may have.

2. After this, we will present FISTA, an extension of an already known algorithm, ISTA, in Section 3.2, that reaches an optimal convergence ratio for first order problems, those based on gradient steps.

3. The FISTA algorithm will give us the first application of Nesterov steps to solve our kind of problems. Nonetheless, as we will see in Section 3.1, the acceleration may not be monotone towards convergence, describing thus a somehow erratic behavior. In Section 3.3 we cover some restarting schemes that could help avoiding this kind of behavior.

4. In Section 3.4 we describe a set of further optimizations over FISTA that conform a new algorithm, FAPG, presented by Ito *et al.* in [6]. These optimizations are focused on achieving a faster practical convergence. In Section 3.5 we prove its convergence.

5. In Section 3.6 of this Chapter we describe Group Lasso as an extension of the Lasso problem we previously described and analyze its solution.

3.1 Introduction to Nesterov Acceleration Schemes

Generally, Nesterov accelerated schemes can be thought of as some sort of *momentum–*based methods, where we increase the *impulse* of our descent step in each iteration of the minimization process. Intuitively, we can already see the sense of this idea applied as a way to essentially accelerate the convergence. Nonetheless, and opposed to standard

```
Algorithm 3: Accelerated Scheme 1
  Initialization: $y_0 = x_0 \in \mathbb{R}^n$, $\theta_0 = 1$ and $q \in [0,1]$
  for $k = 0, 1, \dots$ do
      $x_{k+1} = \mathbf{y}_k - t_k \nabla f(\mathbf{y}_k)$
      $\theta_{k+1}$ solves $\theta_{k+1}^2 = (1-\theta_{k+1})\theta_k^2 + q\theta_{k+1}$
      $\beta_{k+1} = \theta_k(1-\theta_k)/(\theta_k^2 + \theta_{k+1})$
      $y_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k)$
  end
```

```
Algorithm 4: Accelerated Scheme 2
  Initialization: $y_0 = x_0 \in \mathbb{R}^n$
  for $k = 0, 1, \dots$ do
      $x_{k+1} = \mathbf{y}_k - t_k \nabla f(\mathbf{y}_k)$
      $y_{k+1} = x_{k+1} + \beta^*(x_{k+1} - x_k)$
  end
```

first–order methods, these accelerated schemes do not guarantee a monotone behavior towards convergence. If we are increasing our *impulse* in each iteration we may end up having *too much* impulse, leading us to take steps in our gradient direction that take us out of the optimal range, resulting in a somehow rippling or bumping behavior. In this section we will show how this erratic performance is characteristic of situations in which we *overestimate* the required *momentum*, getting past the *critical point.* In this context, the restarting techniques that we will later introduce seem a reasonable way to recover the optimal convergence ratio that accelerated schemes promised.

Let us first introduce the set of concepts and properties that will help us go through all the later derivations. Suppose we want to minimize a certain convex function that depends on a variable $x \in \mathbb{R}^n$, where $f : \mathbb{R}^n \to \mathbb{R}$ has *Lipschitz continuous* gradient. We recall that this means that the following conditions holds

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

A function is considered *strongly convex* if there exists a certain μ such that

$$f(x) \geq f^* + \frac{\mu}{2}\|x - x^*\|^2, \quad \forall x \in \mathbb{R}^n.$$

In this context, we define the *condition number* of a strongly convex function as L/μ.

The essential idea underlying these schemes is to add an increasing sequence θ_k that represents the momentum of the algorithm. This parameter is then used to generate a new point y_k from two past points x_k and x_{k-1}, known as the momentum point. An example of accelerated scheme to solve this problem is presented in Algorithm 3. Note that there are many possibilities to choose β_{k+1} and we will see another example when we study the FISTA algorithm in Section 3.2. As we will see next, this algorithm converges for any $t_k \leq 1/L$. For a setting of $t_k = 1/L, q = 1$ this algorithm shows a *linear convergence ratio.* More precisely, choosing $q = 1$ recovers the full gradient scheme that we previously saw in Section 2.1.3. This can be easily derived since we have $\theta_0 = 1$ and then this update means that $\theta_{k+1} = 1$ is constant and thus $\beta_{k+1} = 0$, leading to $y_{k+1} = x_{k+1}$ and we are back to performing the gradient step over the previous x_k point, whose convergence rate is:

$$F(x_k) - F(x^*) \leq \frac{\alpha L_f \|x - x_0\|^2}{2k}$$

For a setting of $t_k = 1/L, q = 0$ we have a guaranteed convergence ratio of

$$f(x^k) - f^* \leq \frac{4L\|x^0 - x^*\|^2}{(k+2)^2},$$

which is significantly better than the linear convergence ratio we obtained earlier and whose proof can be found in [3] (proof of ISTA convergence).

If the function is strongly convex we can go further and choose $q = \mu/L$, for which the convergence shows that [7]

$$f(x^k) - f^* \leq L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k \|x^0 - x^*\|^2,$$

which implies that we can reach an ϵ–accurate solution within

$$\mathcal{O}\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\epsilon}\right)$$

iterations.

In the case of a strongly convex function we can write the algorithm in a simplified way, shown in Algorithm 4. In this scenario we write the β^* parameter in a very specific form

$$\beta^* = \frac{1 - \sqrt{\mu/L}}{1 + \sqrt{\mu/L}},$$

which is the same optimal β_k that we recover in Algorithm 3 if we choose an optimal $q = \mu/L$.

Nonetheless, even if we study here the case of strongly convex functions in detail, we should ask ourselves for the nature of the μ and L parameters. Since the most common case is that of unknown parameters we would have to estimate them. For the case of L we already know that there exists a *backtracking* strategy that we showed in the ISTA algorithm to recover a good estimate of L. Unfortunately, this is not the case of μ, where the estimation is more delicate. Even though Nesterov proposed a *backtracking*-alike method to estimate μ [8], it seems that this method does not recover a very good estimate and thus we will have a slower convergence. In this context, we can already see that a bad estimation of q could lead to very different scenarios. Depending on whether we underestimate or overestimate our q parameter we could see a characteristic *rippling* behavior.

More precisely, in the case of an overestimated q the algorithm would show a *monotone* descent towards convergence, whilst in the underestimated case we will see a rippling behavior of the cost function in which we wouldn't have a monotone convergence. This can be seen by briefly analyzing the form of our β^* parameter, that we recall is the one defining the *momentum.* In the case of an overestimated q we are somehow getting a β generally *smaller* than the optimal one; intuitively we can see that we would have a smaller numerator and a bigger denominator in equation (3.1). This implies that the *momentum* of our minimization process will be smaller, which, even though we will get a worse convergence, is guaranteed to be monotone. On the other hand, if we instead underestimate q we will end up with a *bigger* β than the optimal one, which implies that the momentum in our process may be too big, provoking then a bigger gradient step that could take us out of the optimal direction.

We see then that the optimum momentum is closely related to the condition number of the problem. Specifically, when we have a very big condition number we will need a bigger impulse in our descent direction. We can see that this makes sense if we think of μ and L as the slope of quadratic boundaries to $f(x)$. In this context, a very big condition number means that the inferior bound to our function is somehow very flat, resulting in very small gradient steps that need to be balanced with a bigger momentum. The opposite case is now clear: if we have a very small condition number then both boundaries to our function are very steep and then a big momentum could easily lead us to big jumps resulting then in a non–monotone behavior that we are trying to avoid.

3.2 FISTA Algorithm

In Section 2.3 we introduced ISTA, a first iterative algorithm to solve proximal descent problems. While ISTA is a generic and good algorithm to solve proximal problems, its *sublinear* convergence ratio is far from optimal. Suppose we want to find a solution for an ϵ accuracy of $\epsilon = 10^{-6}$; that means that we may need at least one million iterations, which is generally too many iterations for real problems, where we work with big datasets. In this context, in [3] the authors propose a modification of this algorithm, called FISTA (Fast Iterative Shrinkage–Thresholding Algorithm), including a *Nesterov* step to speed up the iterations, reaching a much better convergence ratio of $\mathcal{O}(1/k^2)$. As we described in Section 3.1, the Nesterov step is actually an increasing sequence t_k with which we generate a linear combination of past points of x_k that we name y_k, the momentum point, over which we later perform the gradient step.

We show the algorithms in Algorithms 5 and 6 and their analysis next. As we see, we have a first version of the algorithm provided that we know the Lipschitz constant of the problem, which is often not the case, and a second one where we make an estimation of this parameter through a *backtracking* strategy. We remember from Section 2.3 that the backtracking strategy tries to find a $L_k \geq L_f$, where L_f is the Lipschitz constant, such that the quadratic approximation to the function is above the function at the current point, that is, find a L_k that fulfills (2.3.8).

As we easily see from the algorithms, the basic idea behind FISTA is the same as ISTA's, with the difference on where we apply the proximal operator. In this case we apply the proximal operator on a specific combination of x_k and x_{k-1} times the Nesterov step $\frac{t_k-1}{t_k}$. The reason for this step is motivated by the following lemma

Lemma 3.2.1. *The sequences $\{x_k, y_k\}$ generated by either version of the FISTA algorithm where $v_k := F(x_k) - F(x^*)$, $u_k := t_k x_k - (t_k - 1)x_{k-1} - x^*$ satisfy for any $k \geq 1$ and $L_k \geq L_f$*

$$\frac{2}{L_k} t_k^2 v_k - \frac{2}{L_{k+1}} t_{k+1}^2 v_{k+1} \geq \|u_{k+1}\|^2 - \|u_k\|^2,$$

whose proof is given in [3]. Before analyzing FISTA's convergence ratio we need the following result.

Lemma 3.2.2. *Let $\{a_k, b_k\}$ be positive sequences of reals satisfying*

$$a_k - a_{k+1} \geq b_{k+1} - b_k \quad \forall k \geq 1,$$

with $a_1 + b_1 \leq c, c > 0$. Then, $a_k \leq c$ for every $k \geq 1$.

```
Algorithm 5: FISTA algorithm with constant step
  Input         : $f$ convex with $\nabla f$ Lipschitz constant $L$; $g$ convex
  Output        : $x_k \simeq \arg\min_{x\in\mathbb{R}^n} \{f(\mathbf{x}) + g(\mathbf{x})\}$
  Initialization: $x_0 \in \mathbb{R}^n$
  $\mathbf{y}_1 = \mathbf{x}_0$;
  $t_1 = 1$;
  for $k = 1, \dots$ do
      $$x_k = \mathrm{prox}_{\frac{1}{L}g}\left(\mathbf{y}_k - \frac{1}{L}\nabla f(\mathbf{y}_k)\right)$$
      $$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$$
      $$\mathbf{y}_k = \mathbf{x}_k + \frac{t_k - 1}{t_{k+1}}(\mathbf{x}_k - \mathbf{x}_{k-1})$$
  end
```

```
Algorithm 6: FISTA algorithm with backtracking
  Input         : $f$ convex; $g$ convex
  Output        : $x_k \simeq \arg\min_{x\in\mathbb{R}^n} \{f(\mathbf{x}) + g(\mathbf{x})\}$
  Initialization: Take $L_0 > 0$, some $\eta > 1$ and some $x_0 \in \mathbb{R}^n$
  $\mathbf{y}_1 = \mathbf{x}_0$;
  $t_1 = 1$;
  for $k = 1, \dots$ do
      Find smallest non-negative integers $i_k$ such that with $\bar{L} = \eta^{i_k} L_{k-1}$ we have
      $$F(p_{\bar{L}}(x_{k-1})) \leq Q_{\bar{L}}(p_{\bar{L}}(x_{k-1}), x_{k-1})$$
      Set $L_k = \eta^{i_k} L_{k-1}$
      $$x_k = \mathrm{prox}_{\frac{1}{L}g}\left(\mathbf{y}_k - \frac{1}{\bar{L}}\nabla f(\mathbf{y}_k)\right)$$
      $$t_k = \frac{1 + \sqrt{1 + 4(t_k)^2}}{2}$$
      $$\mathbf{y}_k = \mathbf{x}_k + \frac{t_k - 1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1})$$
  end
```

Theorem 11. FISTA Convergence Ratio *Let $\{x_k\}$ be the sequence generated by either version of the FISTA algorithm. Then*

$$F(x_k) - F(x^*) \le \frac{2\alpha L_f \|x_0 - x^*\|^2}{(k+1)^2},$$

where $\alpha = 1$ for the constant step-size setting and $\alpha = \eta$ for the backtracking step-size setting.

Proof. Let us define the quantities

$$a_k := \frac{2}{L_k} t_k^2 v_k, \quad b_k := \|u_k\|^2, \quad c := \|y_1 - x^*\|^2 = \|x_0 - x^*\|^2,$$

and recall that $v_k := F(x_k) - F(x^*)$ from the previous Lemma 3.2.1. Then, by Lemma 3.2.1 we have for every $k \ge 1$

$$a_k - a_{k+1} \ge b_{k+1} - b_k,$$

and hence assuming that $a_1 + b_1 \le c$ holds true. By Lemma 3.2.2 we obtain that

$$\frac{2}{L_k} t_k^2 v_k \le \|x_0 - x^*\|^2,$$

which, combined with the fact that $t_k \ge (k+1)/2$, yields

$$v_k \le \frac{2L_k \|x_0 - x^*\|^2}{(k+1)^2}.$$

Therefore, we just have to show that

$$\frac{2}{L_1} t_1^2 v_1 + \|u_1\|^2 \le \|x_0 - x^*\|^2.$$

We know that an upper bound for on L_k is $L_k \le \alpha L_f$ (recall Lemma 2.3.1), and then the desired result follows. Thus, all that is left to prove is the validity of the relation $a_1 + b_1 \le c$. Since $t_1 = 1$, and using the definition of u_k, we have

$$a_1 = \frac{2}{L_1} t_1^2 v_1 = \frac{2}{L_1} v_1, \quad b_1 = \|u_1\|^2 = \|x_1 - x^*\|^2, \quad c = \|y_1 - x^*\|^2 = \|x_0 - x^*\|^2.$$

Applying Lemma 2.3.3 to the points $x = x^*, y = y_1$ with $L = L_1$ we get

$$F(x^*) - F(p(y_1)) \ge \frac{L_1}{2} \|p(y_1) - y_1\|^2 + L_1 \langle y_1 - x^*, p(y_1) - y_1 \rangle.$$

And then,

$$\begin{aligned} F(x^*) - F(x_1) &= F(x^*) - F(p(y_1)) \\ &\ge \frac{L_1}{2} \|p(y_1) - y_1\|^2 + L_1 \langle y_1 - x^*, p(y_1) - y_1 \rangle \\ &= \frac{L_1}{2} \|x_1 - y_1\|^2 + L_1 \langle y_1 - x^*, x_1 - y_1 \rangle \\ &= \frac{L_1}{2} \left\{ \|x_1 - x^*\|^2 - \|y_1 - x^*\|^2 \right\}. \end{aligned}$$

This is equivalent to,

$$-v_1 \geq \frac{L_1}{2}\|x_1 - x^*\|^2 - \frac{L_1}{2}\|y_1 - x^*\|^2 = \frac{L_1}{2}\|u_1\|^2 - \frac{L_1}{2}\|x_0 - x^*\|^2,$$

or, in other words,

$$-\frac{L_1}{2}a_1 \geq \frac{L_1}{2}b_1 - \frac{L_1}{2}c$$

which proves that $a_1 + b_1 \leq c$ holds true. □

And thus the number of iterations of the FISTA algorithm to obtain an ϵ-optimal solution is at most $\lceil C/\sqrt{\epsilon} - 1\rceil$, where $C = 2\alpha L_f\|x_0 - x^*\|^2$ is the numerator in the convergence ratio.

3.3 Nesterov Restarting Schemes

We have just seen the FISTA algorithm as a first real application of Nesterov ideas to accelerate proximal descent problems. This opens up a wide variety of derivations that have as common joint the exploitation of this kind of momentum steps in order to find an optimal and computationally fast algorithm to solve high dimensional problems or, at least, improve on the convergence ratio offered by ISTA.

Within this wide set of methods we find the restarting schemes proposed by O'Donoghue and Candes in [7], that we describe next. Other techniques following this lead are described, for instance, in [9].

These restarting schemes, as we briefly presented in the introduction to this Chapter, have the general objective of avoiding the possible erratic behavior of accelerated methods when showing a non–monotone trajectory towards convergence. These techniques make the algorithm restart the t_k sequence of the FISTA algorithm when they detect an erratic behavior, given certain conditions that we explain next. As we described in Section 3.1, this will be the most common case due to our lack of knowledge about the optimal parameters μ and L of the function. In this context, the first *naive* idea that may come to mind is a fixed scheme that restarts the algorithm every k iterations.

The optimal restarting interval k will be determined by the evolution of the system in a given iteration outer j, which is the number of restarts, just before the restart occurs, at the inner iteration k, which is the iteration counter within each restarting cycle. This evolution is given by

$$f(x^{(j+1,0)}) - f^* = f(x^{(j,k)}) - f^* \leq 4L\|x^{(j,0)} - x^*\|/k^2 \leq (8L/\mu k^2)(f(x^{(j,0)}) - f^*),$$

where the first inequality comes from the convergence ratio of Algorithm 3 and the second from the strong convexity condition we defined earlier. After jk steps we get

$$f(x^{(j,0)}) - f^* \leq \left(8L/\mu k^2\right)^j (f(x^{(0,0)}) - f^*).$$

By minimizing the term $\left(8L/\mu k^2\right)^j$ over both j, k jointly we have that

$$k^* = \sqrt{8L/\mu},$$

which implies that we will reach an ϵ-optimal solution after approximately $\mathcal{O}(\sqrt{L/\mu}\log(1/\epsilon))$ iterations.

Again, this method will only show a good performance if we know the aforementioned parameters μ and L, which, as we said, is often not the case.

The previous analysis suggests that an adaptive restarting scheme may be useful in cases when we don't know μ and L. In the original paper the authors propose the two following options

- Function scheme: we restart the algorithm whenever
$$f(x^k) > f(x^{k-1}).$$
- Gradient scheme: we restart the algorithm whenever
$$\nabla f(y^{k-1}) \cdot (x^k - x^{k-1}) > 0.$$

In Section 3.3.3 we report some experiments applying these optimizations for the Lasso problem. Nonetheless, we would like to remark some interesting facts about both schemes. On one hand, we can intuitively think that the first scheme tells us to restart the algorithm whenever we see an increase in the cost function, which is a very reasonable criteria. Nonetheless, that forces us to compute the cost function at two points, x^k and x^{k-1}, which may end up being quite expensive computationally. On the other hand, the second scheme tells us to restart whenever the momentum seems to be taking us in a bad direction, that is, an ascent direction. Furthermore, this scheme expresses its criteria in terms of the gradient, which we already had computed (remember that we are working with gradient accelerated methods, so calculating the gradient is a must), and then this scheme does not imply any additional computational costs, which makes this a very interesting option.

3.3.1 Analysis

In this section we focus on analyzing the simplest optimization case, that of a pure quadratic form. The objective of this analysis is to observe the behavior of the methods from the dynamical systems point of view. Having this context in mind, let us define the general framework of the problem as the minimization of the following quadratic

$$\arg\min_x f(x),$$

where $f(x) = \frac{1}{2}xAx$, and $A \in \mathbb{R}^{n\times n}$.

In this simple case we have the optimal point $x^* = 0$ and optimal cost $f^* = 0$. Furthermore, we know the parameters of the function, $\mu = \lambda_{\min} > 0$ y $L = \lambda_{\max}$. Now let us assume a constant step $t_k = 1/L$ for simplicity and we have that, since $y_0 = x_0$ the Algorithm 3 evolves at iteration k as

$$\begin{aligned} x^{k+1} &= y^k - (1/L)Ay^k, \\ y^{k+1} &= x^{k+1} + \beta_k(x^{k+1} - x^k). \end{aligned}$$

Let as now consider the eigen–decomposition of $A = V\Lambda V^T$, where V is the eigenvector matrix and Λ is the eigenvalues diagonal matrix. Let us denote $w^k = V^T x^k$ y $v^k = V^T y^k$ and write the system in terms of the previous eigendecomposition of A and its eigenvalues as

$$
\begin{aligned}
w^{k+1} &= v^k - (1/L)\Lambda v^k, \\
v^{k+1} &= w^{k+1} + \beta(w^{k+1} - w^k),
\end{aligned}
$$

by means of simple algebraic operations. We can see that these equations now describe n dynamical systems where the ith component evolves as

$$
\begin{aligned}
w_i^{k+1} &= v_i^k - (1/L)\lambda_i v_i^k, \\
v_i^{k+1} &= w_i^{k+1} + \beta(w_i^{k+1} - w_i^k).
\end{aligned}
$$

We can now write each equation in terms of only one variable by substituting the other and then we get the following recurrences

$$
\begin{aligned}
w_i^{k+2} &= (1+\beta)(1-\lambda_i/L)w_i^{k+1} - \beta(1-\lambda_i/L)w_i^k, \\
v_i^{k+2} &= (1+\beta)(1-\lambda_i/L)v_i^{k+1} - \beta(1-\lambda_i/L)v_i^k,
\end{aligned}
$$

where w_i^0 is known and $w_i^1 = w_i^0(1-\lambda_i/L)$, $v_i^0 = w_i^0$ and $v_i^1 = ((1+\beta)(1-\lambda_i/L)-\beta)v_i^0$.

3.3.2 Convergence Properties

The behavior of this system is determined by the characteristic polynomial of the recurrent relation, that can be written in the following form

$$
r^2 - (1+\beta)(1-\lambda_i/L)r + \beta(1-\lambda_i/L).
$$

This system has a critical point for β for which the recurrence has repeated roots. This β^* can be found by solving the quadratic equation and then solving with respect to β for the case of equal roots. This β is

$$
\beta_i^* := \frac{1-\sqrt{\lambda_i/L}}{1+\sqrt{\lambda_i/L}}.
$$

This value of β divides the topological solutions of the dynamical system into the following three regions

- $\beta_i < \beta_i^*$, where the system is over-damped.
- $\beta_i = \beta_i^*$, where the system is critically damped.
- $\beta_i > \beta_i^*$, where the system is under-damped.

In the first region, where $\beta_i < \beta_i^*$ the polynomial has two real roots, r_1 and r_2, and the system evolves according to

$$
w_i^k = c_1 r_1^k + c_2 r_2^k,
$$

where for each i the c_1, c_2 are the constants $(1+\beta_i)(1-\lambda_i/L)$ and $\beta_i(1-\lambda_i/L)$. In this region we say that the system is in the low momentum regime and then the system is over-damped, that is, the system will show a slow but monotone trajectory to convergence. When we have $\beta = \beta^*$ then the system behaves according to the optimal convergence ratio $\propto (1-\sqrt{\lambda_i/L})^k$

and the roots are equal at the point $r^* = (1+\beta_i^*)(1-\lambda_i/L)/2 = (1-\sqrt{\lambda_i/L})$, which corresponds to critical dumping. For the case when $\beta_i > \beta_i^*$ the roots of the polynomial are complex and we say that we are in the high momentum regime and the system is under-damped. In this case the system shows a periodic rippling whose periodicity depends on the condition number, as we show next.

In this last case the characteristic solution is given by

$$w_i^k = c_i(\beta_i(1-\lambda_i/L))^{k/2}\cos(k\psi_i - t_i),$$

where c_i and t_i depend on the initial conditions and ψ_i is defined as

$$\psi_i = \cos^{-1}\left((1-\lambda_i/L)(1+\beta_i)/2\sqrt{\beta_i(1-\lambda_i/L)}\right).$$

Similarly, we have

$$v_i^k = \hat{c}_i(\beta_i(1-\lambda_i/L))^{k/2}\left(\cos(k\psi_i - t_i)\right).$$

Since for a small angle θ we know that $\cos^{-1}(\sqrt{1-\theta}) \approx \sqrt{\theta}$ and since $\lambda_i \ll L$ we can simplify ψ_i as

$$\psi_i \approx \sqrt{\lambda_i/L},$$

where ψ_i symbolizes the oscillation frequency for the mode corresponding to the smallest eigenvalue, which is approximately given by $\sqrt{\mu/L}$.

We show next the analysis of the convergence ratio for both adaptive restarting schemes, beginning with the function scheme. To derive the convergence we start from

$$f(w^k) = \sum_{i=1}^{n}(w_i^k)^2\lambda_i, \tag{3.3.1}$$

which is the value of the function at the kth iteration. Since the sum will be dominated by the smallest eigenvalue we can assume that

$$f(w^k) \approx (w_k^0)^2\mu\beta(1-\mu/L)^k\cos^2(k\sqrt{\mu/L}), \tag{3.3.2}$$

which is interesting because it clearly shows the existence of the rippling behavior and its generation. We can see that this function is formed by two components, $(1-\mu/L)^k$ on one hand, which is decreasing towards 0, and $\cos^2\left(k\sqrt{\mu/L}\right)$, which incorporates the rippling structure.

For the case of the gradient scheme we have a similar expression

$$\begin{aligned}\nabla f(y^k)\cdot(x^{k+1}-x^k) &\approx \mu v_\mu^k(w_\mu^{k+1}-w_\mu^k)\\ &\propto \beta^k(1-\mu/L)^k\sin\left(2k\sqrt{\mu/L}\right),\end{aligned} \tag{3.3.3}$$

which also explains the rippling behavior.

In addition, these two equations show us that the expected number of iterations after which we should apply the restarting condition is $(\pi/2)\sqrt{L/\mu}$. We can easily see that this value nullifies the sin and cos functions. This guarantees us an ϵ–tolerance solution after $\mathcal{O}\left(\sqrt{L/\mu}\log(1/\epsilon)\right)$ iterations, which means that we have effectively recovered the optimum convergence ratio of the Nesterov Accelerated Gradient.

These schemes do not have application only on the strict cases of a explicit quadratic problem, but also apply to all problems that may be approximated by a quadratic form inside some regions. As we can expect, the behavior outside these regions is not clear. Nonetheless, we must note that within these regions we may have a particular estimation of $\mu_k > \mu$ that leads to an even faster convergence ratio. This is due to μ being the dominant eigenvalue (the smallest) in the sum of the forms (3.3.2) or (3.3.3), and thus, being possibly bigger in these regions, the convergence should be faster. Note that this is a very interesting result, leading to a more general discussion on the estimation of the Lipschitz constant L. Similarly, we may have specific regions where the estimated L (by means of backtracking) is smaller than the global parameter of the function, leading to a bigger step in that region and then to a possibly faster convergence. In Section 3.4 we explore this and other issues related to further optimize the convergence of the methods.

3.3.3 Application for Lasso

In this section we present the adaptation of the previous results to the Lasso problem. We must note that the results and definitions were given supposing a smooth function f. Since Lasso has a non-smooth regularization component it does not directly apply to the generality of these schemes. Nonetheless, we want a sparse solution and we note that once the non-zero basis of the solution has been identified we are loosely speaking minimizing a quadratic form. Given this interpretation, we could expect that the conclusions at which we previously arrived also apply to this kind of problems.

In the case of convex optimization problems we will not use the generic and first accelerated schemes introduced by Nesterov but the FISTA algorithm we already explained in detail, which is, essentially, a variation of these methods. For the case of Lasso, as we explained in Chapter 2, the proximal operator is the known soft-thresholding operator

$$T_\alpha(x) = \operatorname{sign}(x)\max(|x| - \alpha, 0),$$

where all operations are applied element-wise.

It is easy to show that for the function restarting criteria we do not need an extra application of the X matrix since we are already computing it in the gradient. On the other hand, the gradient condition is expressed under the assumption of the existence of such gradient, which is not the case in Lasso. Nonetheless, we can make use of the subdifferential as a sort of generalized gradient to write our criteria as

$$x_{k+1} = T_{\lambda t}(y_k - tA \cdot (A \cdot y_k - b)) = y_k - t\partial F(y_k),$$

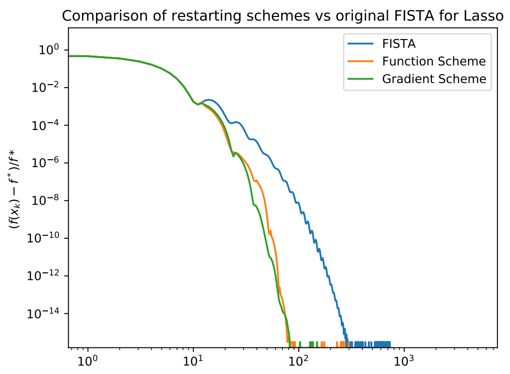
where $F = f + g$ with f being convex and differentiable and g convex.

Under this setting the gradient restarting scheme amounts to restarting whenever

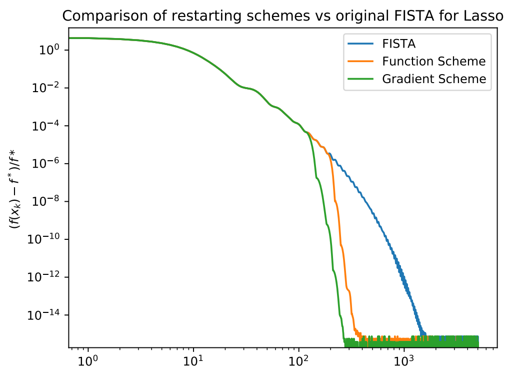
$$\partial F(y_k) \cdot (x_{k+1} - x_k) > 0 \equiv (y_k - x_{k+1}) \cdot (x_{k+1} - x_k) > 0,$$

which greatly simplifies the calculations, avoiding extra computational costs.

To experiment with these criteria we have generated synthetic data in the following way. We first have generated a matrix X from a standard normal distribution and dimension $n \times m$, where n indicates the number of patterns and m indicates the number of predictors. Then, we generated a random sparse vector w with m entries. We now set $y = X \cdot w + b$, where b is a white noise vector. We fix a step-size $t = 1/\lambda_{\max}(X^T X)$ for which we know we should have an optimal convergence ratio of $\mathcal{O}(1/k^2)$ (since the function is convex) and run FISTA for two simple experiments, one with $n = 2000$ and $m = 500$ and the other $n = 2000$ and $m = 100$. This experiment is extracted from [7].



(a) Comparison of FISTA algorithm convergence with adaptive restarting for $m = 100$ predictors.



(b) Comparison of FISTA algorithm convergence with adaptive restarting for $m = 500$ predictors.

In Figures 3.3.1a and 3.3.1b we can see the dramatic boost of the performance obtained after running the algorithm for 5000 iterations (to show the full path towards convergence). The y axis shows the difference between the cost function at iteration k and the optimal value of f^* normalized by the optimal f^* itself.

Even if it is not clear in the figures, we can guess the rippling behavior on the FISTA trajectory given its thicker line towards convergence in both experiments. We see that the adaptive restarting schemes do not get any advantage during the first iterations of the algorithm, and the first scheme to notice the bad dynamic is the gradient restarting one, thanks to its more sophisticated criterion. While FISTA converges around iteration 1500, both the gradient and function restarting schemes converge around iteration 500 with a slight edge in favor of the gradient scheme, which is able to see the ascending trend earlier than the function scheme condition.

Note that this is a synthetic experiment sampled from Gaussian data, so it is probably not a good example of a real application and only serves theoretical purposes. In general, both restarting schemes should show very similar performance. The condition we use will depend on the specific problem we want to tackle, although, as explained earlier, the gradient scheme seems to be not only better in terms of performance but also *cheaper* to compute. In Chapter 4 we will see more realistic examples.

3.4 Further Accelerations

Following the lead set by O'Donoghue and Candes in [7] there has been recent attention in the literature looking for approaches to further improve the performance boost achieved by the restarting schemes described in Section 3.3. In this context, Ito *et al.* have proposed a new algorithm named Fast Accelerated Proximal Gradient in [6], adding further optimizations to the ones already described.

These strategies are focused on making the Accelerated Proximal Gradient a practically efficient method. Some of these strategies have been already described in earlier sections from another point of view.

3.4.1 Bactracking Strategy

We have already seen that in many cases we do not know the exact value of the Lipschitz constant of a problem and thus we are forced to estimate one. From that point of view, the backtracking strategy is depicted as a trade-off on the optimization process. Nonetheless, in Section 3.1 we already hinted that this backtracking estimation may actually be an advantage. Since the global Lipschitz constant L is generally too conservative, we may have a better estimate for some regions of the function, where the backtracking L_k may actually perform better.

Thus, we adopt the backtracking strategy described in the FISTA algorithm (and ISTA) in Section 3.2 given some constants $\eta_u > 1$, $L_0 > 0$, that is

'bt': While

$$F(x^k) > Q_{L_k}(x^k; y^k)$$

update $L_{k+1} \leftarrow \eta_u L_k$ and $x^k \leftarrow T_{L_k}(y^k)$.

This condition ensures that the convergence result for the convergence of the algorithm is still $\mathcal{O}(1/k^2)$.

3.4.2 Decreasing Strategy for L_k

The previous backtracking strategy ensures that the sequence L_k is non-decreasing. Actually, L_k must be non-decreasing to fulfill the conditions required for the complexity analysis in [3]. However, it would be advantageous to decrease the value of L_k to allow for greater gradient steps whenever possible.

In [10] the authors propose a modification in the original APG method to allow L_k to decrease so that t_k satisfies $t_k/L_k \geq t_{k+1}(t_{k+1}-1)/L_{k+1}$ ($\forall k \geq 1$). To achieve this we have to recompute y_k and t_k in each step of the backtracking. Moreover, the new computation of t_k is done by

$$t_{k+1} = \frac{1 + \sqrt{1 + 4(L_{k+1}/L_k)t_k^2}}{2} \tag{3.4.1}$$

The decreasing strategy in the algorithm is

'dec': Set $L_{k+1} \leftarrow L_k/\eta_d$ for some $\eta_d > 1$.

Note that this is a somehow contrary strategy to the backtracking one we detailed in the previous subsection. In this sense, this strategy forces L_k to be as small as possible to allow for a greater step. In case the decreased L_k is too small, the backtracking strategy will control for a sufficiently large estimation.

3.4.3 Restarting Strategy

We have extensively covered in Section 3.3 the restarting strategies proposed in [7] and thus we will only summarize here the restarting criteria for the algorithm, supposing any convex function $F \equiv f + g$.

're': If $\nabla f(y^k) \cdot (x^k - x^{k-1}) + g(x^k) - g(x^{k-1}) > 0$ then update $t_{k+1} \leftarrow 1, t_k \leftarrow 0$, $y^{k+1} \leftarrow x^{k-1}$ and $x^{k+1} \leftarrow x^{k-1}$.

Note that this criteria slightly differs from the pure strategies described earlier. This is because in this case we are considering any convex function F instead of only differentiable functions. However, the convergence for a general convex function is unknown. In this context, as we mentioned before, there are other restarting techniques as such of [9], where the authors model the APG method as a second-order ordinary differential equation.

3.4.4 Top-Speed Strategy

As we have explained, what the restarting strategy precisely does is to cancel high-momentum and prevent overshooting on the performed step, which avoids taking big jumps that may direct us into the wrong direction. Nonetheless, in neighborhoods near the optimum a high-momentum may be more effective rather than continuously restarting the algorithm because large overshooting may not occur. For this, a heuristic optimization is to put a prohibition period of restart for the K_i iteration after the ith restart occurs, where K_i increases as $K_{i+1} = 2K_i$ for some $K_1 \geq 2$.

'mt': If $k \leq kre + K_i$ then skip **'re'**. If restart occurs, update $kre \leftarrow k$, $K_{i+1} \leftarrow 2K_i$.

Later on we will show that this modification ensures a convergence rate of $\mathcal{O}((\log k/k)^2)$ under some assumptions.

3.4.5 Stability Strategy

This strategy is more of a practical addition and it does not affect the convergence analysis that we show in Subsection 3.5. This optimization tries to correct the trade-off generated between the decreasing strategy and the restarting one. We can see that both strategies speed-up the practical convergence of the APG method, especially in early iterations and near the optimum.

Nonetheless, they have opposing effects. Decreasing L_k enlarges the step $1/L_k$, which extends $x^k - x^{k-1}$, which inherently induces high momentum. On the other hand, the restarting strategy cancels high momentum. As a consequence, combining these two strategies triggers the restarting criteria frequently and makes the APG method unstable. In order to avoid this instability we decrease the η_d so that in the limit we have that $\eta_d = 1$ and then near the optimum we don't decrease L_k and then the same convergence analysis that perform [3] is equally valid.

'st': Update $\eta_d \leftarrow \delta \cdot \eta_d + (1 - \delta)$ when the restart occurs with $\delta < 1$.

3.4.6 Skipping Extra Computations

We show the full modified algorithm in Algorithm 7. Nonetheless, for practical efficiency we can add the following algorithmic optimizations

On one hand, instead of computing t_k and y^k in the backtracking section, we can fix them and store the value for $f(y^k)$ and $\nabla f(y^k)$, which reduces the computation cost, as pointed out in [3]. On the other hand, we see that the termination criteria involves the extra computation of $\text{prox}_{\frac{1}{L_k}g}(x^k - \frac{1}{L_k}\nabla f(x^k))$. This termination criteria checks whether the current solution is a fixed point of the proximal operator. Instead of checking this condition we can check if $L_k\|x^k - y^k\| < \epsilon$ at each iteration. As analyzed in [6], it follows that if $x^k = y^k$ then x^k is an optimal solution. We can see that checking this is similar to the actual termination condition because intuitively

Algorithm 7: Modified APG Algorithm

```
Input          : $f$ convex and differentiable; $g$ convex
Output         : $x_k \simeq \arg\min_{x \in \mathbb{R}^n} \{f(\mathbf{x}) + g(\mathbf{x})\}$
Initialization: Take $L_0 > 0$, $\eta_u > 1$, $\eta_d > 1$, $K_1 \geq 2$, $x_0 \in \mathbb{R}^n$, $\delta \in (0,1)$
$\mathbf{y}_1 = \mathbf{x}_0$
$t_1 = 1$
kre $= 0$
for $k = 1, \dots$ do
    $x^k = \mathrm{prox}_{\frac{1}{L_k} g}\left(y^k - \frac{1}{L_k}\nabla f(y^k)\right)$
    // backtracking
    while $F(x^k) > Q_{L_k}(x^k; y^k)$ do
        $L_k = \eta_u L_k$
        $t_k = \frac{1 + \sqrt{1 + 4(L_k/L_{k-1})t_{k-1}^2}}{2}$
        $y^k = x^{k-1} + \frac{t_{k-1} - 1}{t_k}\left(x^{k-1} - x^{k-2}\right)$
        $x^k = \mathrm{prox}_{\frac{1}{L_k} g}\left(y^k - \frac{1}{L_k}\nabla f(y^k)\right)$
    end
    if $\left\| L_k \left(\mathrm{prox}_{\frac{1}{L_k} g}\left(x^k - \frac{1}{L_k}\nabla f(x^k)\right) - x^k\right)\right\| < \epsilon$ then
        break
    end
    $L_{k+1} = L_k/\eta_d$
    $t_{k+1} = \frac{1 + \sqrt{1 + 4(L_{k+1}/L_k)t_k^2}}{2}$
    $y^{k+1} = x^k + \frac{t_k - 1}{t_{k+1}}\left(x^k - x^{k-1}\right)$
    if $k >$ kre *and* $\nabla f(y^k) \cdot (x^k - x^{k-1}) + g(x^k) - g(x^{k-1}) > 0$ then
        kre $= k$; $K_{i+1} = 2K_i$
        $\eta_d = \delta \cdot \eta_d + (1 - \delta)$
        $t_{k+1} = 1$; $y^{k+1} = x^{k+1} = x^{k-1}$
    end
end
```

Algorithm 8: Final Modified APG Algorithm

```
Input          : $f$ convex; $g$ convex
Output         : $x_k \simeq \arg\min_{x\in\mathbb{R}^n} \{f(\mathbf{x}) + g(\mathbf{x})\}$
Initialization: Take $L_0 > 0$, $\eta_u > 1$, $\eta_d > 1$, $K_1 \geq 2$, $x_0 \in \mathbb{R}^n$
$\mathbf{y}_1 = \mathbf{x}_0$
$t_1 = 1$
kre $= 0$
for $k = 1, \dots$ do
    $x^k = \operatorname{prox}_{1/L_k g}\left(y^k - \frac{1}{L_k}\nabla f(y^k)\right)$
    while $F(x^k) > Q_{L_k}(x^k; y^k)$ do
        $L_k = \eta_u L_k$
        $x^k = \operatorname{prox}_{\frac{1}{L_k}g}\left(y^k - \frac{1}{L_k}\nabla f(y^k)\right)$
    end
    if $L_k\|x^k - y^k\| < \epsilon$ then
        break
    end
    $L_{k+1} = L_k/\eta_d$
    $t_{k+1} = \frac{1+\sqrt{1+4(L_{k+1}/L_k)t_k^2}}{2}$
    $y^{k+1} = x^k + \frac{t_k - 1}{t_{k+1}}\left(x^k - x^{k-1}\right)$
    if $k >$ kre and $\nabla f(y^k)\cdot(x^k - x^{k-1}) + g(x^k) - g(x^{k-1})$ then
        kre $= k$; $K_{i+1} = 2K_i$
        $\eta_d = \delta\cdot\eta_d + (1-\delta)$
        $t_{k+1} = 1$; $y^{k+1} = x^{k+1} = x^{k-1}$
    end
end
```

$$x^k = \operatorname{prox}_{\frac{1}{L_k}g}\left(y^k - \frac{1}{L_k}\nabla f(y^k)\right) = y^k - \frac{1}{L_k}G(y^k),$$

where G is the generalized gradient. This equation holds only when we are in a region of the proximal when the solution doesn't change. In this case it follows that

$$\|G(y^k)\| = L_k\|y^k - x^k\|, \tag{3.4.2}$$

so we are checking whether this *residual* is small enough.

The final and more efficient version of the modified APG method is shown in Algorithm 8.

3.5 Convergence Analysis

We now prove the convergence ratio of FAPG method taking into account all the accelerations we just described. Let us use the following notation: assume that x^k is generated as step j within a restarting iteration i. That is,

$$x^k = x^{(i,j)}, \quad \text{for some } j = 0, \dots, \overline{J}_i$$

where $\overline{J}_i$ is the last iteration within the ith restarting cycle. Then we have the following Lemma, which can be found in [6], Lemma 5.

Lemma 3.5.1. *We have* $F(x^{(i,j)}) \le F(x^{(i,0)}), \quad j = 0, \dots, \overline{J}_i$ *and*

$$F(x^{(i+1,0)}) = F(x^{(i,\overline{J}_i-1)}) \le F(x^{(i,0)}) \le F(x^{(i-1,0)}) \dots \le F(x^0) \tag{3.5.1}$$

As a consequence, for any $x^k = x^{(i,j)}$*, we have that*

$$F(x^k) \le F(x^0), \quad \forall k \ge 1. \tag{3.5.2}$$

Proof. Assume that the restart does not occur until the kth iteration. From Lemma 2.3.3 we have

$$F(x^1) - F(x^0) \le -\frac{L_1}{2}\|x^1 - x^0\|^2 \le 0. \tag{3.5.3}$$

For all $n = 1, 2, \dots,$ we also have

$$\begin{aligned} F(x^{n+1}) - F(x^n) &\le \frac{L_{n+1}}{2}\left\{\|y^{n+1} - x^n\|^2 - \|x^{n+1} - x^n\|^2\right\} \\ &= \frac{L_{n+1}}{2}\left\{\left(\frac{t_n - 1}{t_{n+1}}\right)\|x^n - x^{n-1}\|^2 - \|x^{n+1} - x^n\|^2\right\}. \end{aligned}$$

Summing over $n = 1, 2, \dots, k-1$ we have

$$\begin{aligned} F(x^k) - F(x^1) \le & \frac{1}{2}\left\{L_2\left(\frac{t_1 - 1}{t_2}\right)^2\|x^1 - x^0\|^2\right. \\ &\left. + \sum_{n=2}^{k-1}\left(L_{n+1}\left(\frac{t_n - 1}{t_{n+1}}\right)^2 - L_n\right)\|x^n - x^{n-1}\|^2 - L_k\|x^k - x^{k-1}\|^2\right\}. \end{aligned}$$

Note that $t_1 - 1 = 0$ and $L_{n+1}\left(\frac{t_n - 1}{t_{n+1}}\right)^2 - L_n \le 0$ for all $n \ge 1$ because of

$$\frac{t_n - 1}{t_{n+1}} = \frac{2(t_n - 1)}{1 + \sqrt{1 + 4(L_{n+1}/L_n)t_n^2}} \le \frac{2(t_n - 1)}{\sqrt{4(L_{n+1}/L_n)t_n^2}} = \sqrt{\frac{L_n}{L_{n+1}}}\frac{t_n - 1}{t_n}.$$

Thus we have that

$$F(x^k) - F(x^1) \le 0.$$

Similarly, since the restart does not occur from $x^{(i,0)}$ to $x^{(i,\overline{J}_i)}$ we have

$$F(x^{(i,j)}) \le F(x^{(i,0)}), \quad \forall j \in \{0, 1, \dots, \overline{J}_i\}$$

because since we only restart whenever the function is not monotone, we have a monotone behavior within each restarting cycle, and hence by definition

$$F(x^{(i+1,0)}) = F(x^{(i,\overline{J}_i)}) \le F(x^{(i,0)}).$$

□

This Lemma essentially states that the initial value $F(x^{(i,0)})$ of each restarting cycle gives an upper bound of the subsequent function values and hence the sequence $\left\{F(x^{(i,0)})\right\}_{i=1}^{\infty}$ is non-increasing. For more details about its proof see [6], Lemma 5.

We now prove the convergence rate for the modified algorithm (see Algorithm 8) as exposed in [6], Theorem 6.

Theorem 12. *Let the sequence $\left\{x^k\right\}$ be the sequence generated by the modified APG method. Let S^* be the set of optimal solutions and $B := \left\{x \mid F(x) \leq F(x^0)\right\}$ be the level set. Assume that there exists a finite R such that*

$$R \geq \sup_{x^* \in S^*} \sup_{x \in B} \|x - x^*\|. \tag{3.5.4}$$

Then we have

$$F(x^k) - F(x^*) \leq 2\eta_u L_f R^2 \left(\frac{\log_2(k+2)}{k - \log_2(k+2)}\right)^2, \quad \forall k \geq 3. \tag{3.5.5}$$

Proof. From Lemma 3.5.1, we have $x^k \in B$ for all $k \geq 1$. Let $x^k = x^{(i,j)}$. We assume $j \geq 1$ without loss of generality. From Theorem 11 we have

$$F(x^{(i,\overline{J}_i)}) - F(x^*) \leq \frac{2\eta_u L_f \|x^{(i,0)} - x^*\|^2}{\overline{J}_i^2}.$$

Furthermore, for all $i \geq 1$, Lemma 3.5.1 leads to

$$\begin{aligned} F(x^{(i,\overline{J}_i)}) - F(x^*) &\leq F(x^{(i,0)}) - F(x^*) \\ &= F(x^{(i-1,\overline{J}_{i-1}-1)} - F(x^*)) \\ &\leq \frac{2\eta_u L_f \|x^{(i-1,0)} - x^*\|^2}{(\overline{J}_{i-1} - 1)^2} \\ &\leq \frac{2\eta_u L_f R^2}{(\overline{J}_{i-1} - 1)^2}. \end{aligned}$$

Note that $\overline{k}_i \geq K_i \geq 2$. Thus we obtain

$$\begin{aligned} F(x^{(i,j)}) - F(x^*) &\leq \frac{2\eta_u L_f R^2}{\max\left\{\overline{J}_\ell - 1, \quad 1 \leq \ell \leq i\right\}^2} \\ F(x^{(i,j)}) - F(x^*) &\leq \frac{2\eta_u L_f R^2}{(\max\left\{\overline{J}_\ell, \quad 1 \leq \ell \leq i\right\} - 1)^2} \end{aligned}$$

Assume we are at iteration $k = (i,j), 0 \leq j \leq \overline{J}_i$. From

$$k \geq \sum_{\ell=1}^{i} K_\ell = \frac{K_1(2^i - 1)}{2 - 1} \geq 2^{i+1} - 2,$$

the number of restarts i is at most $\log_2(k+2) - 1$. Hence, since the maximum is bigger than the mean, we must have

$$\max\left\{\overline{J}_\ell, \quad 1 \le \ell \le i+1\right\} \ge \frac{k}{i+1} \ge \frac{k}{\log_2(k+2)},$$

which leads to

$$\begin{aligned} F(x^{(i,J_i)}) - F(x^*) &\le 2\eta_u L_f R^2 \frac{1}{\left(\dfrac{k}{\log_2(k+2)} - 1\right)^2} \\ &\le 2\eta_u L_f R^2 \left(\frac{\log_2(k+2)}{k - \log_2(k+2)}\right)^2, \quad \forall k \ge 3. \end{aligned}$$

□

3.6 The Group Lasso Extension

Group Lasso is a natural extension of the Lasso problem where we consider a grouped structure nature on the predictors. This may not only help with getting a better estimation and thus a better model but also to improve the interpretability of the model, which is often a very important feature. Group Lasso first appears in [11] as a new way to model and select grouped variables for regression problems. Nonetheless, this model has received some recent attention in the literature and new models have been proposed for other tasks such as classification by modifying the cost function. This group modeling appears as a nice feature in cases where we want to select or discard a whole group of variables that usually has some higher meaning than a single variable. This is achieved by applying the ℓ_{21} norm known as the regularization term, which translates to performing the ℓ_1 norm to the ℓ_2 norm of each group.

For the purposes of the theoretical description, consider a linear regression model involving J groups of predictors, where for $j = 1, \dots, J$, the vector $Z_j \in \mathbb{R}^{p_j}$ represents the group of predictors in group j with dimension p_j. Our goal is then to predict a real-valued target $Y \in \mathbb{R}$ based on the collection of predictors. A linear model for this regression function takes the form $w_0 + \sum_{j=1}^J Z_j^T w_j$, where $w_j \in \mathbb{R}^{p_j}$ represents a group of p_j regression coefficients.

Given a collection of N samples $\{(y_i, z_{i1}, z_{i2}, \dots, z_{iJ})\}_{i=1}^N$ the Group Lasso model solves the convex problem

$$\underset{w_0 \in \mathbb{R}, w \in \mathbb{R}^{p_j}}{\arg\min} \left\{ \frac{1}{2} \sum_{i=1}^N \left(y_i - w_0 - \sum_{j=1}^J z_{ij} \cdot w_j \right)^2 + \lambda \sum_{j=1}^J \|w_j\|_2 \right\},$$

where $\|w_j\|_2$ is the well-known Euclidean norm of the group j.

Since this model is a group generalization of the Lasso, some interesting properties remain

- Depending on the parameter $\lambda \ge 0$, either the entire vector w_j will be zero or all its elements will be nonzero.
- When $p_j = 1$ then we have that $\|w_j\|_2 = |w_j|$, which means that if all groups contain a single element then this optimization problem reduces to the ordinary Lasso.

Since each of the groups may have a different length, the authors of the original paper recommend weighting the penalty term by a factor $\sqrt{p_j}$. For simplicity, we will consider here the case where all the groups have the same number of elements.

Similarly as we did in the Lasso problem, we calculate the zero-subgradient equations, which take the form

$$-Z_j \cdot \left(y - \sum_{\ell}^{J} Z_\ell \hat{w}_\ell \right) + \lambda \hat{s}_j = 0, \quad \forall j = 1, \dots, J,$$

where $\hat{s}_j \in \mathbb{R}^{p_j}$ is an element of the subdifferential of the norm $\|\cdot\|_2$ at $\hat{w}_j$.

We can easily calculate the subdifferential of the ℓ_2 norm as follows

$$\partial \ell_2 = \begin{cases} \hat{w}_j / \|\hat{w}_j\|_2 & \text{if} \quad \hat{w}_j \neq 0 \\ s \in \mathbb{R}^{p_j} \quad \text{such that} \quad \|\hat{s}_j\|_2 \leq 1 & \text{if} \quad \hat{w}_j = 0 \end{cases}.$$

One method for solving these equations is by holding fixed all block vectors $\left\{ \hat{\theta}_k, k \neq j \right\}$, and then solving for $\hat{\theta}_j$. Doing so amounts to performnig block coordinate descent on the Group Lasso objective function. Furthermore, since the problem is convex and the penalty is block separable, that is, we can write the penalty as a sum of its individual components, then the problem is guaranteed to converge to an optimal solution [5]. With $\left\{ \theta_k, \hat{k} \neq j \right\}$ fixed, we write

$$-Z_j \cdot (r_j - Z_j \hat{\theta}_j) + \lambda \hat{s}_j$$

where $r_j = y - \sum_{k \neq j} Z_k \hat{\theta}_k$ is the jth partial residual.

The subdifferential of this penalty term looks very similar to the subdifferential of the ℓ_1 norm, specially in the second case, when $\hat{w}_j = 0$, in the sense that any vector with norm less than 1 is a valid subgradient, as happened in the ℓ_1 with any slope in the interval $[-1, 1]$. From this we must have $\hat{\theta}_j = 0$ if $\|Z_j \cdot r_j\|_2 \leq \lambda$, and otherwise

$$\hat{\theta}_j = \left(Z_j \cdot Z_j + \frac{\lambda}{\|\hat{\theta}_j\|_2} I \right)^{-1} Z_j \cdot r_j$$

As we can see, this form depends on $\hat{\theta}_j$ itself and so we need iterative methods. We can find the form of these iterates by applying the proximal operator to the Group Lasso problem. This leads to a proximal descent algorithm. To do this, we remember that the proximal operator may be defined by

$$\mathrm{prox}_{\nu g}(\theta) = \underset{\theta \in \mathbb{R}^p}{\arg\min} \left\{ \frac{1}{2} \|\theta - w\|_2^2 + \lambda \nu g(w) \right\}.$$

In our case, we have $g = \|\cdot\|_2$ and $\theta = w_j - \nu Z_j \cdot (r_j - Z_j)$, where r_j is the partial residual.

Applying the definition and differentiating with respect to w_j in the previous equation we can find that the iterates are

$$\hat{w}_j = \left(1 - \frac{\lambda \nu}{\|\theta\|_2} \right)_+ \theta,$$

where ν is the step-size parameter calculated as $\nu = 1/L$, $L = \lambda_{\max}(X^T X)$.

We note that this algorithm is actually performing a proximal descent and we can then plug this equation into the FISTA algorithm we described earlier to end up with an accelerated version of these iterates, taking the form

$$\begin{aligned}
x_{k+1} &= \operatorname{prox}_{\frac{1}{L}g}\left(y_k - \frac{1}{L}\nabla f(y_k)\right),\\
t_{k+1} &= \frac{1+\sqrt{1+4t_k^2}}{2},\\
y_{k+1} &= x_{k+1} + \frac{t_k - 1}{t_{k+1}}(x_{k+1} - x_k).
\end{aligned}$$

As we saw in Section 3.2, this sequence $F(x_k)$ is guaranteed to converge for any $t_k \leq 1/L$ and for a suitable sequence of the t_k.

3.7 Conclusions

In this Chapter we have introduced the advanced topics of this work, focused around the ideas introduced by Nesterov on accelerating gradient methods in order to recover an optimal convergence ratio. These ideas have received some recent attention in the literature in multiple areas of the convex optimization theory, ranging from introducing this Nesterov step in the SMO algorithm [12] to even deep learning applications [13].

We summarize next the main concepts we have presented in this Chapter

- We have first introduced the general framework of Nesterov accelerated gradient methods, where we have seen that even if the performance boost these methods may provide is large, there is no guarantee of monotone convergence, which, sometimes, may lead to a non optimal performance.
- The first and probably most important accelerated algorithm in our field is FISTA, a fast version of the ISTA algorithm we presented in Section 2.3. FISTA is a widely used algorithm given its generality and performance for many problems.
- Nonetheless, our main interest in this chapter is related to how to avoid the non monotone convergence of the FISTA algorithm. To address this issue we have presented the ideas of O'Donoghue and Candes in [7], introducing the restarting schemes built upon the same ideas of Nesterov. We have shown that these restarting schemes may have a dramatic impact in the performance of the algorithm, as we showed in the case of Lasso.
- As further optimizations to the restarting techniques we have also studied the work by Ito *et al.* in [6], the FAPG algorithm, a FISTA algorithm combined with several optimizations. In Chapter 4 we will show an extensive experimental setup in order to compare the performance of these optimized methods against standard FISTA.
- After this discussion we came to the Group Lasso model, a structured extension of the Lasso, where the predictors are grouped. This is a very interesting feature for many problems, such as the ones we are interested in, regarding renewable energy prediction.

Chapter 4

Experiments

After covering the concepts and ideas of Nesterov Accelerated Proximal Gradient and some of its optimizations from a pure theoretical point of view, we will explore now its applicability to real-world problems in day-to-day machine learning settings. We will study the following applications

1. First, we will measure the convergence of standard FISTA against accelerated methods in terms of iterations in a synthetic setup, being a natural extension of the experiments we run for the Lasso problem in Section 3.3.3. In this case, though, we will use the Group Lasso model we recently described and further complete the experiments of Lasso.

2. Second, we will explore the application and usefulness of these methods in a wind energy prediction problem. This is an interesting case since the data is naturally grouped in terms of geographical grid points, where we have a set of variables defined for each point. In this setting we will explore both convergence results and the models' results themselves for Group Lasso, comparing them to other known linear models such as Lasso and Ridge.

4.1 Synthetic Experiments

These first experiments are an enriched version of the ones we performed for the Lasso problem in Section 3.3.3. In this case we will apply the same setting for the Group Lasso model that we introduced in Section 3.6. Along with the analysis of Group Lasso we will continue the previous experiment of Lasso by adding experiments running FAPG.

We first show the results for the Lasso experiments adding an extensive analysis of FAPG's performance on the following problem. We first have generated a matrix X from a standard normal distribution with dimension $n \times m$, where n are the sample size and m the number of predictors. Then, we generated a random vector w with m entries for which we defined ten groups of predictors, each with a different mean (ranging from -5 to 5) and all the same standard deviation $\sigma = 0.25$, in order to obtain a grouped structure. We now set $y = X \cdot w + b$, where b is a white noise vector. We fix a step-size $t = 1/\lambda_{\max}(X^T X)$ for which we know we should have an optimal convergence ratio of $\mathcal{O}(1/k^2)$ for the case of Lipschitz convex functions and run the algorithm for a simple experiment with $n = 2000$ and $m = 500$.

For the convergence benchmark we will test FISTA, FISTA with the function restarting scheme, FISTA with the gradient restarting scheme and the FAPG method proposed by Ito *et al.* and adapted by us to our needs.

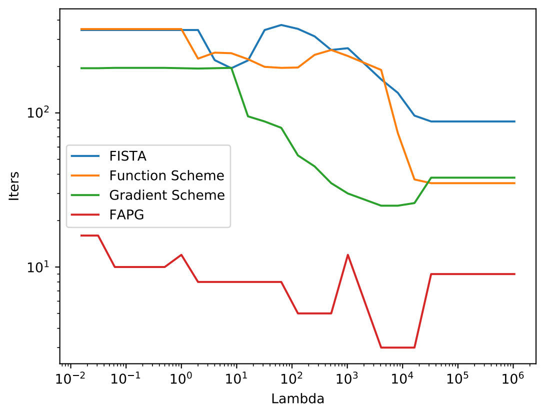


Figure 4.1.1: Iterations taken to converge depending on the λ value for all methods for Lasso.

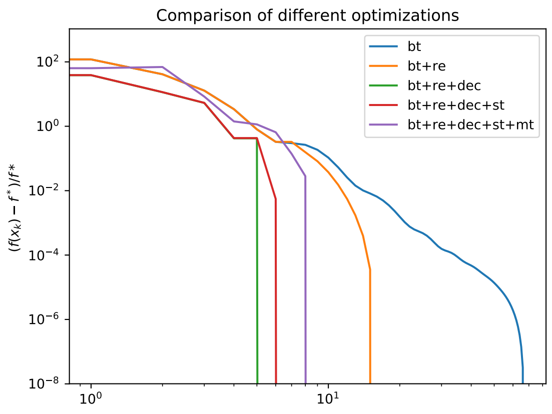


Figure 4.1.2: Detailed contribution of each of the optimizations on the FAPG method for Lasso for optimal $\lambda = 1$, where *bt* refers to the backtracking strategy, *dec* to the decreasing one, *re* to the restarting scheme, *st* to the stability strategy and *mt* to the maintaining top-speed one.

4.1.1 Lasso Performance Benchmarking

Given the relevance of the chosen λ in the Lasso solution, that greatly impacts the performance of the method, we have also run a benchmark comparing the performance of all methods depending on the λ parameter. In Figure 4.1.1 we show a relation between the iterations needed to converge for each method against the λ parameter for Lasso. This is essentially a cross-validation setting, where we run a model for many possible parameters and try to obtain the best ones. It would be very advantageous to optimize this process since it is usually quite computationally expensive for complex problems. These λ are sampled from a logarithmic interval $\left[2^{-6}, 2^{20}\right]$, from which we take 27 values.

Here, in Figure 4.1.1, we see a great benefit from the optimization, particularly from FAPG against standard FISTA. It is also interesting to note the similar performance of the function restarting scheme and the gradient restarting scheme for λ at the end of the interval. Overall, though, the gradient scheme performs much better. Interestingly, we see that there is a certain value of λ around 1 (which is precisely the optimal) for which the

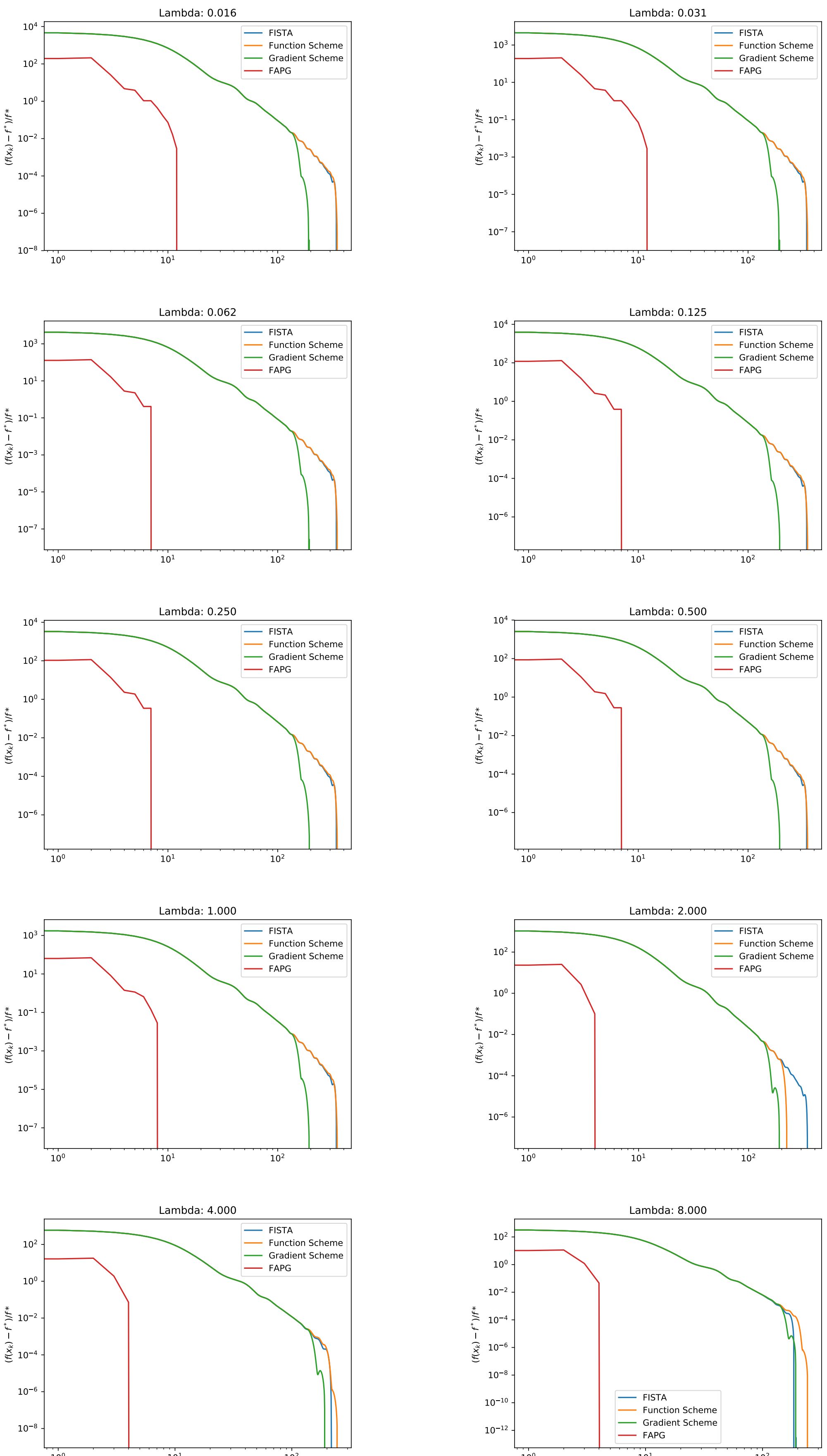


Figure 4.1.3: Performance plots for several λ for Lasso

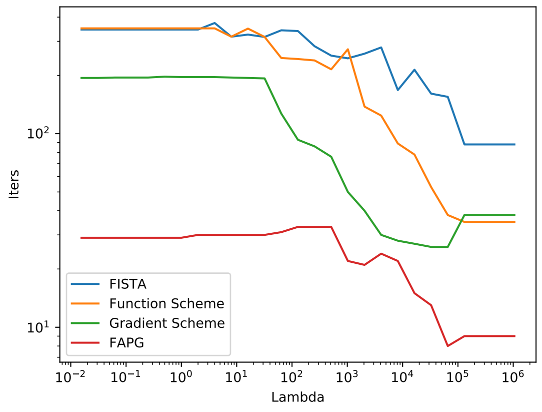


Figure 4.1.4: Iterations taken to converge depending on the λ for all methods for Group Lasso.

performance of the methods improves, having for the last λ a lower difference in terms of iterations across all methods. As we can see, these optimizations result particularly useful in cross validation setups where we want to try tens, or even hundreds, of models.

In Figure 4.1.3 we show the evolution of the methods for 10 particular λ for Lasso. In these figures we see a very similar shape for small λ, whereas for larger ones we see a bigger difference between FAPG and the rest. Interestingly, the larger the λ the more ripples we find, which may be explained because it is easier to exceed the momentum required in a much simpler problem, and thus overestimate the step in each iteration.

For the optimal value of λ we have also run an experiment to analyze the exact contribution of each of the different strategies of the FAPG method. We show this plot in Figure 4.1.2. For this particular case we can appreciate that *dec* is the most helpful strategy. The full FAPG method still shows a very good performance, but we think that it is not the best due to the small number of iterations the method needs to converge, given that some strategies such as *mt* or *st* show its potential in longer runs. Along this line, we also appreciate that the *re* strategy is not particularly good on its own.

4.1.2 Group Lasso Performance Benchmarking

After the analysis of Lasso we show the same results for Group Lasso. Nonetheless, let us briefly recall the Group Lasso model, that, for J groups, solves a problem of the form

$$\operatorname*{arg\,min}_{w_0\in\mathbb{R}, w\in\mathbb{R}^{p_j}} \left\{ \frac{1}{2}\sum_{i=1}^{N}\left(y_i - w_0 - \sum_{j=1}^{J} z_{ij}\cdot w_j\right)^2 + \lambda\sum_{j=1}^{J}\|w_j\|_2 \right\},$$

where $\|w_j\|_2$ is the well-known Euclidean norm of the group j.

In the case of our current setting we could set the number of groups to any arbitrary quantity and, as we mentioned before, we chose $J = 10$, which means that we will have ten groups of fifty predictors. We next test the same methods as we did for Lasso: FISTA, FISTA with function restarting scheme, FISTA with gradient restarting scheme and FAPG.

As we mentioned before, the specific performance of these methods is related to the chosen λ. We show Group Lasso's results for a benchmark comparing the performance varying λ in Figure 4.1.4. The 27 values of λ have been selected as in the case of Lasso, from an evenly spaced logarithmic interval $[2^{-6}, 2^{20}]$ to explore a sufficiently wide range.

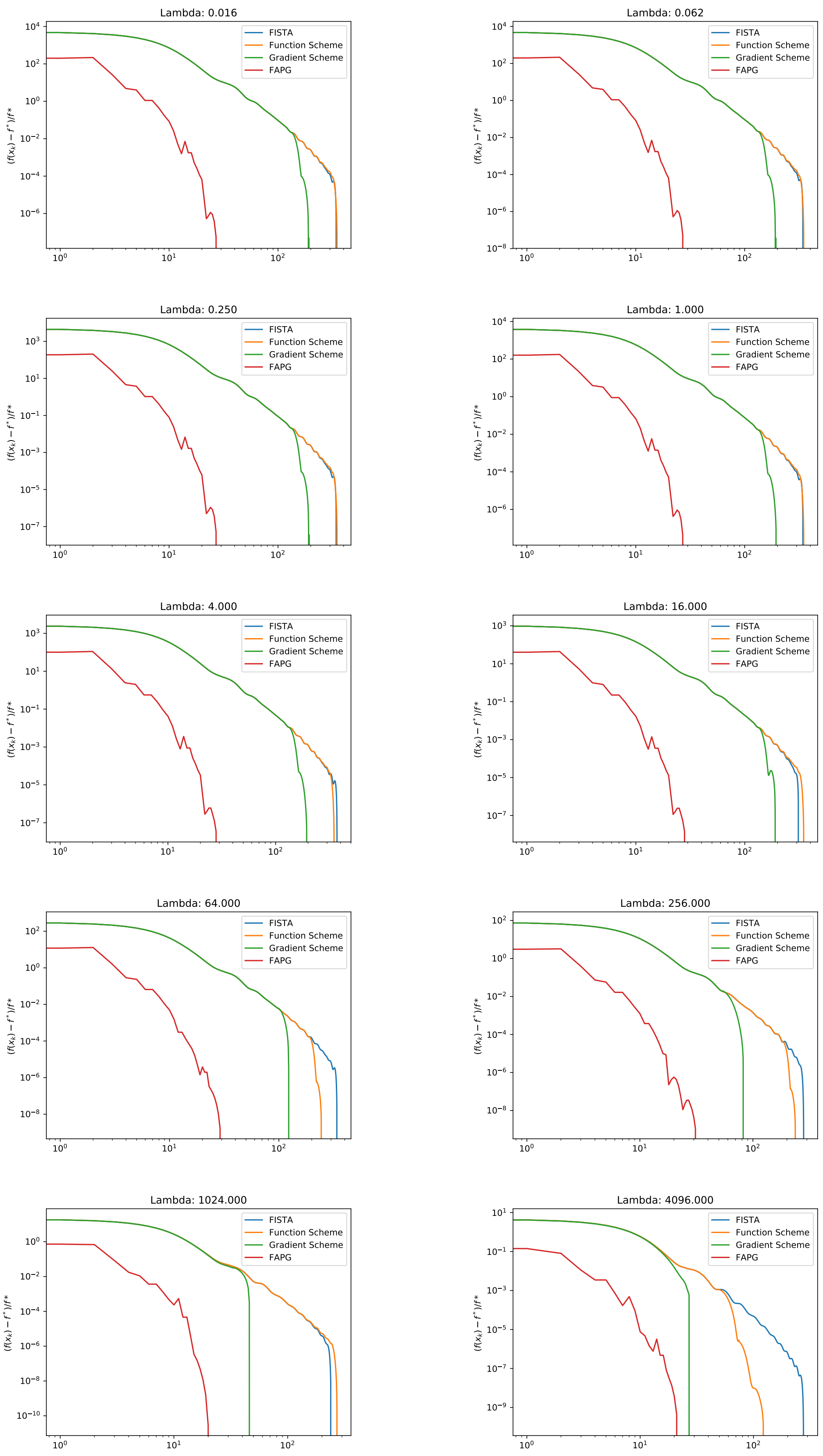


Figure 4.1.5: Performance plots for several λ for Group Lasso

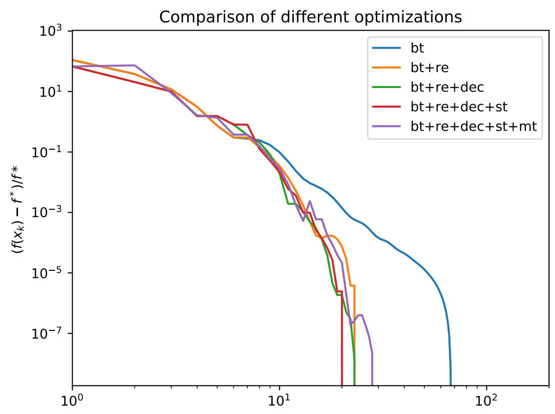


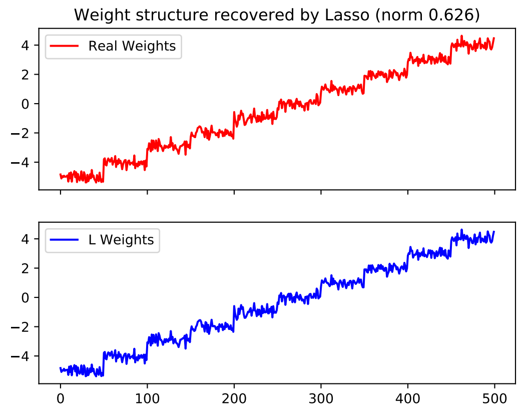
Figure 4.1.6: Detailed contribution of each of the optimizations on the FAPG method for Group Lasso for $\lambda = 8.0$, where *bt* refers to the backtracking strategy, *dec* to the decreasing one, *re* to the restarting scheme, *st* to the stability strategy and *mt* to the maintaining top-speed one.

It is interesting to see how FAPG is outperforming the other methods for every λ we tried. Nonetheless, we also note that there is a certain λ for which FISTA starts performing better. This critical point seems to be around the optimal $\lambda = 8$. This suggests that, even if for any λ FAPG outperforms FISTA in number of iterations, we could get an even higher boost in performance for complex cross validation settings, which is a very interesting conclusion. More interestingly, we see that the optimization of FAPG with respect to standard FISTA is greater for small values of λ and that the difference between methods decreases as we increase λ.

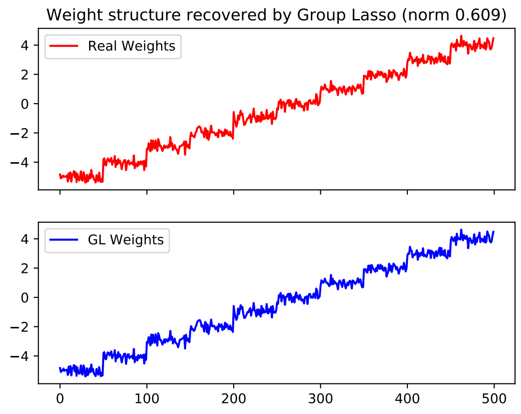
We can see that this intuition makes sense when we think of what each λ value means. A small value would impose a very small penalty and then the region we have to explore to find a solution for the problem is very wide. In this context the method has to work a lot more and then it is reasonable to expect a higher benefit from the optimization. On the other hand, a very large λ value would make the model very simple, taking only into account the regularization term and forgetting about a good error. This makes the model pretty much straightforward, resulting in a very small region to find a solution for our, and then it is possible to not get such a good performance boost.

Given the relevance of λ for the model, in Figure 4.1.5 we show the detailed performance path of all methods for 10 particular λ. We can see how FAPG is the best overall, as we saw in previous figures. From these figures we can see several things. In the first place, we note that the performance of the function restarting scheme is actually closer to FISTA's than to the gradient restarting scheme. Finally, we remark the very good performance of the gradient restarting scheme, very close to FAPG for many λ.

Following the analysis of Lasso, we also analyze the exact contribution that each of the explained strategies (see Section 3.4) have on the FAPG method for the optimal λ for Group Lasso. In Figure 4.1.6 we show these results. We see that the method combining all the strategies is not the best in terms of convergence but is very close to other combined strategies such as $bt + re + dec$. Another interesting insight we extract from this figure is that the key strategy that seems to accelerate the most is the decreasing of the L_k estimation. We also highlight the very bad behavior of the *bt* strategy when it is run alone. In the second place, for this particular case, and probably caused by the low number of iterations we need to converge, we also note that *mt* strategy does not seem to be very



(a) Lasso model's weight structure



(b) Group Lasso model's weight structure

Figure 4.1.7: Recovered weight structure for both Lasso and Group Lasso models.

important for the convergence. As we pointed out before, the method may benefit from it in longer runs. Finally, we observe that the *st* strategy shows the best performance along with $bt + re + dec$, interestingly. For this particular problem it seems that we can take a significant advantage from decreasing L_k in a progressive way.

4.1.3 Recovered Weight Structure

A final aspect we will analyze of these models is the recovered structure of the coefficients. We recall that for this synthetic example we introduced an artificial grouped structure in the original weights by sampling each group from a Gaussian distribution with different mean (from -5 to 5) and $\sigma = 0.25$. In Figure 4.1.7b we show the original against the recovered weights by Group Lasso, where we can clearly see this very same structure. In the title of the figures we also indicate the norm of the difference of the original weights against the recovered weights. Also as a result of the simplicity of the problem we note that the weight structure recovered by Lasso is actually very similar to that of Group Lasso, showed in Figure 4.1.7a.

As a summary, we see that both models behave very similarly and both recover the expected group structure. Nonetheless, we may conclude that Group Lasso actually recovers a better fitted weights, as we can see by comparing the norm of the difference of the weights of both models. Nonetheless, we must take into account that these results correspond to the best parameter of both models, and thus its goal is not to recover the best structure but to obtain the best error. In this sense, it is possible that with another regularization (most likely with a higher one) we could recover a better fitted version of the weights. Regarding the comparison between Lasso and Group Lasso, we recall that this is a synthetic problem and thus in real problems the performance of both models would not be so similar.

4.2 Wind Energy Forecasting

In this Section we will perform a similar analysis as the one we described for the previous synthetic experiment but with a more realistic and interesting problem, wind energy forecasting. The interest in renewable energies is increasingly growing due to, on one side, the state of foil energy sources, whose availability is rapidly decreasing, such as petrol or gas, and, on the other, the need for a stable electricity grid management. In this section we will cover, essentially, the latter, which is directly related to the forecasting problem. The forecasting problem tries to approach the energy prediction for medium to large horizons, namely, one to up to several days in advance.

In this case we will work with a single wind farm in Peninsular Spain, Sotavento, located in Galicia, Northwest Spain. The Numerical Weather Prediction data for our problem are organized in a geographical grid where for each point we have defined several variables. Our data source for this problem is the European Centre for Medium-Range Weather Forecasts (ECMWF). Our dataset consists on the following variables that we directly download from the ECMWF MARS data archive at an hourly frequency:

- U and V wind speed components at 10 and 100 meters.
- Pressure at surface.
- Temperature at 2 meters.

From these variables we will calculate the following ones

- Wind speed module at 10 and 100 meters.

These make a total of 8 variables for each of the grid points in the selected area. For Sotavento we take a geographical grid of 120 points distributed evenly at a resolution of $0.125°$ in a 8×15 rectangular grid whose corners are $(-9.5, 44.0)$ and $(-6.0, 42.25)$ longitude, latitude. This amounts to 960 predictors with 8760 hours yearly. To build our models we employ three years' data: 2013 for training, 2014 for validation of the hyper parameters of the model and 2015 for test. To split such datasets we also apply a zero mean unit variance normalization.

In this context we are interested in two fundamental aspects:

1. First, as we did for the synthetic example, we will study the performance of the different optimizations in a cross validation setup where we build models for several values of the hyper parameters in order to select the best overall. We analyze these results for both Lasso and Group Lasso models.
2. Second, we will compare the results of a Group Lasso model for wind energy forecasting with the related models of Lasso and Ridge Regression.

4.2.1 Performance Benchmarking

For the first of our purposes, that is, the analysis of the performance of the different optimizations, we have essentially run the same cross validation setup focusing our attention in convergence. In this aspect we will test both Group Lasso and Lasso.

4.2.1.1 Lasso Analysis

We first run the Lasso model and we show its results in two figures, similar to those we showed for the synthetic example, in Figures 4.2.1 and 4.2.3.

In the first of them, in Figure 4.2.1, we show the iterations each method took to converge while varying the λ parameter of the model. As we saw in the synthetic example, FAPG is the clear winner for small values of λ, after which all methods seem to show a similar

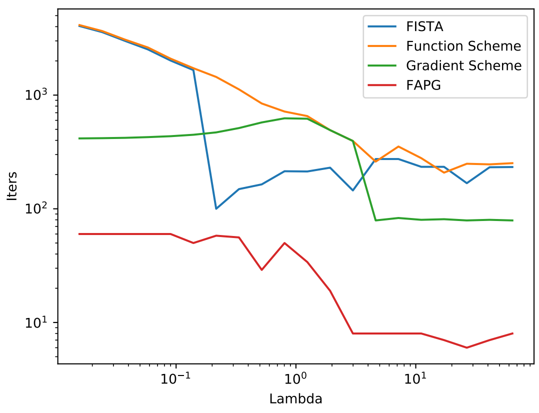


Figure 4.2.1: Iterations taken to converge depending on the λ value for all methods for Lasso

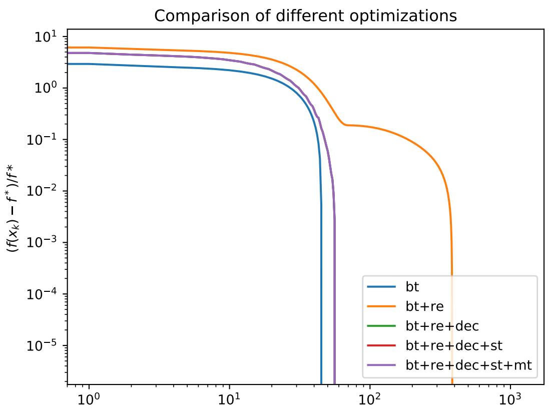


Figure 4.2.2: Detailed contribution of each of the optimizations on the FAPG method for Lasso for $\lambda = 0.090$, where *bt* refers to the backtracking strategy, *dec* to the decreasing one, *re* to the restarting scheme, *st* to the stability strategy and *mt* to the maintaining top-speed one.

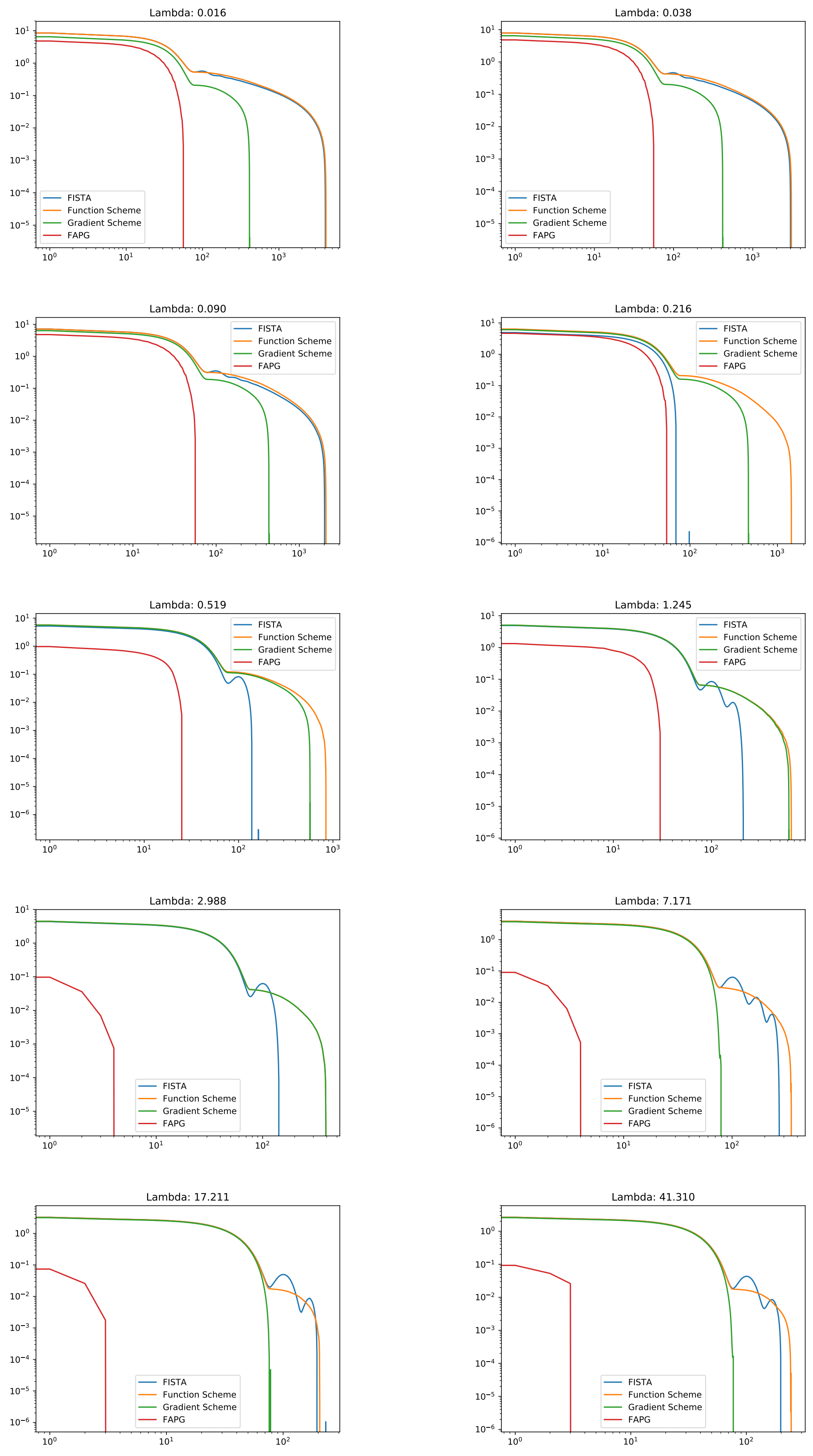


Figure 4.2.3: Performance plots for several λ for Lasso

performance, or at least they show a smaller difference (note the logarithmic scale). As we described for the synthetic example, this is because small values of λ impose a very low penalty and then FISTA has to explore a very wide area, where the optimizations accelerate significantly the convergence of the method. For very large values of λ we have to explore a smaller space, and thus the performance across the methods is similar. This confirms our idea that FAPG would be very valuable in complex cross validation settings, saving a lot of time in general and, overall, for small values of λ, where we often find the best model.

In this sense, it is remarkable to see how FISTA starts showing a closer performance to FAPG for the *optimum* λ around 0.1 and larger values (note the logarithmic scale). As we hinted before, this may be caused because the optimum value of λ balances the model and then FISTA finds a good path towards the solution more easily. We can also see an exceptional performance on the side of the gradient restarting scheme, being the closest method to FAPG for many λ, and much better than the function restarting scheme. Finally, it is also remarkable how the function restarting scheme does not seem to work well at all for this particular problem. We think that this is related to the very low non monotonicity of FISTA in this case, which isn't particularly strong and thus the function restarting scheme would not be triggered.

On a side note, we see that the shape of Figure 4.2.1 is different from that of the synthetic example (Figure 4.1.1), which is natural since for this case a small difference in the value of λ affects more the shape of the problem and the path to the solution, resulting in errors and solutions very different, where we can appreciate better the effects of the optimization strategies. Take into account that we are only exploring here $\lambda \in \left[2^{-6}, 2^{6}\right]$ in logarithmic scale.

In Figure 4.2.3 we show 10 particular cases where we have detailed the evolution of all methods towards convergence for a fixed λ. An interesting case is shown for $\lambda = 0.216$, which is very close to the optimum (the optimum is $\lambda = 0.0903$). In this case we see that FISTA converges very close to FAPG. We note that the optimum $\lambda = 0.09$ for FAPG is not exactly the best for FISTA, but close enough. These results further confirm our intuition: a good value of λ helps the model to find a good direction, even showing a non monotone trajectory.

Finally, we also analyze the contribution of the optimizations method applied to FAPG on Lasso in terms of convergence. In Figure 4.2.2 we show these results. We first note that the restarting strategy alone does not work for this example and needs to be balanced with more powerful accelerations. Interestingly, we also see that for this particular example the full FAPG method does not show the best convergence for the optimal λ. This may be explained because in such few number of iterations we can not get the best out of the *mt* and *st* strategies, that tend to smooth the effects of all strategies in longer experiments. For this particular example we see that the *bt* strategy performs very well, being the best strategy. As we saw in previous examples, the *dec* strategy accelerates the most the method and shows the best performance overall, tied with the *bt* strategy for this particular example.

4.2.1.2 Group Lasso Analysis

We now show the same results for Group Lasso. In Figure 4.2.4 we show the iterations each method took to converge while varying the λ parameter of the model. As we saw in the Lasso example for this same problem, FAPG is the winner for many λ, showing pretty much the same performance as others or little difference for other λ. For this case we see that the gradient restarting scheme also shows an outstanding performance, highly

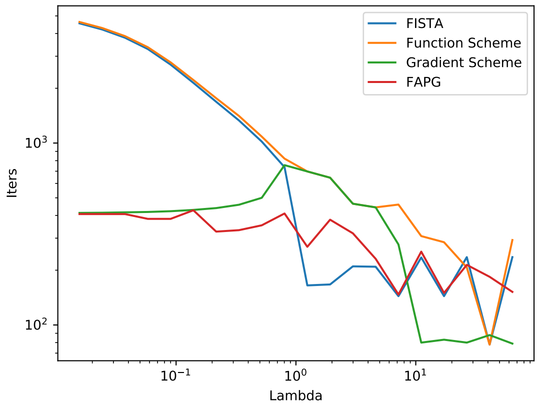


Figure 4.2.4: Iterations taken to converge depending on the λ value for all methods for Group Lasso.

competitive with FAPG for several λ. On the other hand, the function restarting scheme is not just worse than the gradient restarting scheme, but also slower than FISTA for many λ. From these results we see that the performance of the function restarting scheme is highly dependent on the problem, while the other methods seem to be more consistent.

In Figure 4.2.5 we show the detailed evolution of all methods towards convergence for 10 particular λ. An interesting case is shown for $\lambda = 1.245$ and $\lambda = 2.988$, which are close to the optimum $\lambda = 1.928$. In these cases we see that FISTA converges earlier than FAPG (although the difference is small) even being non monotone. As we remarked before, these results confirm our previous intuitions about the effects that λ may have on the performance of the method. It is clear that a good or large λ simplifies the problem and we usually see a faster convergence. Interestingly, these intuitions are even more clear in real and more complex problems than it were in the synthetic case.

To end our analysis of Group Lasso and its performance we also show the contribution of each of the optimizations included in the FAPG method for an optimal λ. We see these results in Figure 4.2.6. A first important difference is that Group Lasso seems to take more iterations to converge than Lasso, obtaining a similar solution. This longer path to convergence has several consequences. In the first place, we recall from Section 3.4.2 that the key to the decreasing strategy is that in the limit we are actually not doing anything, smoothing the effects of the decreasing. In this case, since the method converges around iteration 200, the decreasing constant is already $\eta_d = 1$ and then we are not decreasing any more. In the second place, given the longer run of the experiment, we see that the methods show a closer performance. Particularly, it seems that the mt strategy is not slowing the convergence as much as it did in previous cases, such as in Lasso. Still, the best performance is shown by bt and dec strategies, although bt shows some ripples. We also note that $bt + re$ is the worst strategy, probably caused by restarting too frequently, as we previously saw for Lasso, given that the st strategy seems to accelerate the method. Even if the full FAPG method seems to not have the best performance, a final insight we would like to extract from these analysis is the consistently good convergence of the $bt + re + dec$ strategy, very often better than the full FAPG itself across several of the experiments we have run in this chapter.

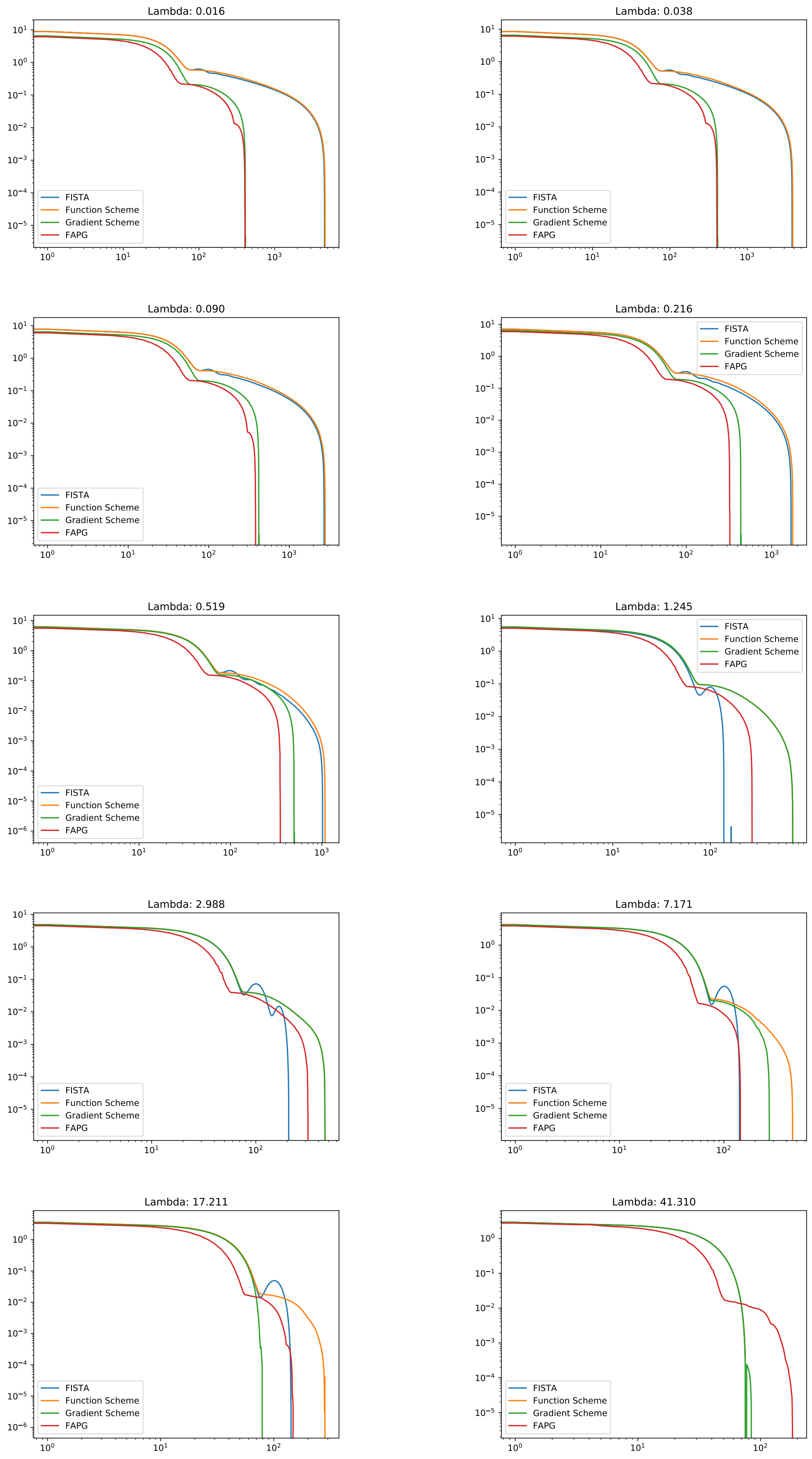


Figure 4.2.5: Performance plots for several λ for Group Lasso

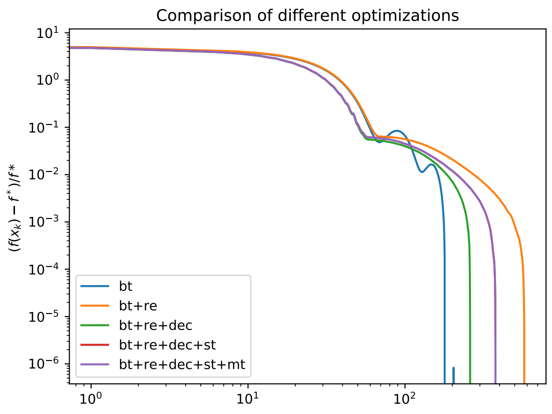


Figure 4.2.6: Detailed contribution of each of the optimizations on the FAPG method for Group Lasso for optimal $\lambda = 1.928$, where *bt* refers to the backtracking strategy, *dec* to the decreasing one, *re* to the restarting scheme, *st* to the stability strategy and *mt* to the maintaining top-speed one.

	FISTA		Function Scheme		Gradient Scheme		FAPG	
	L	GL	L	GL	L	GL	L	GL
MAE Test	**0.082**	0.084	**0.082**	0.0823	**0.082**	0.0823	0.084	**0.0824**
Iters	214	**166**	717	**685**	**625**	685	**60**	178

Table 4.2.1: Results of Group Lasso and Lasso models for Sotavento where we indicate the best model for each method.

4.2.2 Best Model Comparison

For the second of our purposes, that is, the comparison of Group Lasso with other models, we have run a cross validation setup taking as fixed folds 2013's dataset for training, 2014's dataset for validation and 2015's dataset for test. The best parameter is then used to build the final model we use to calculate the test MAE error, that is, the Mean Absolute Error calculated as $\frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$, by training with both 2013 and 2014 datasets and predicting in the 2015 dataset. The results for the best models are show in Table 4.2.1. In the table we represent the MAE for the four methods we have tested: standard FISTA, FISTA with the function restarting scheme, FISTA with the gradient restarting scheme and FAPG for Group Lasso and Lasso. On the other hand, `sklearn's` Ridge model obtains a MAE error of 0.0824, very similar to the errors obtained for Lasso and Group Lasso. We note, again, that for the optimum value of λ FISTA actually reaches convergence faster than FAPG for Group Lasso and close in the case of Lasso, which is something we already hinted before.

We also note that in terms of MAE, even if the convergence we reach is not the same for all methods, the difference is essentially negligible, given that the problem is convex and the solution is unique. It is also interesting to see how both function and gradient restarting schemes reach the same solution in the same number of iterations for Group Lasso, which is often not the case. Another interesting remark is that FAPG reaches a much faster solution for Lasso than Group Lasso, achieving a similar MAE. This may be caused by the shape of the regularization term, that somehow makes the problem a bit more complex. For the rest of the methods we see essentially the same number of iterations in both Lasso and Group Lasso.

In summary, we see that Group Lasso is highly competitive with other linear methods such as Lasso and Ridge Regression, with a slight advantage in terms of interpretability, which makes this a very interesting model. Nonetheless, if we want the best possible model despite of computational cost or interpretability we may find it in a Gaussian SVR model, whose non linear kernel allows for further flexibility. In a similar context, for a PV energy prediction problem, we found ourselves that Gaussian SVR resulted in the best model in [14] or even in [16].

In terms of interpretability we compare now the weights structure of both Lasso and Group Lasso models. For this comparison we will plot the associated weights to each of the variables of the model and grid points and, only for Group Lasso, a final plot considering the norm of each group. This is particularly interesting to select full groups with a clearer meaning than single variables. We recall that Sotavento is a local wind farm located at $(-7.75, 43.25)$ longitude, latitude and is marked with a blue dot in the maps.

This is different in the Lasso weights' structure, as we can see in Figure 4.2.7, where Lasso does not consider groups. At first sight stands out the generally homogeneous weights of the Lasso model. We also see some sparsity geographically, and interestingly we see some weights being zero at points where for other variables we have a nonzero weight. This is because Lasso does not take into account any information at group level. It actually does not even know about the existence of such groups and applies sparsity globally. As a consequence of this, it is harder for Lasso to select points near Sotavento, and instead selects those points that seem to result in a better model, regardless of their interpretability. Also as a consequence of this we see that Lasso results in a sparser model, since it has no other constraint.

In contrast, in Figure 4.2.8 we show the final weights for each of the grid points and variables assigned by the best Group Lasso model for the Sotavento problem. In this sense it is reasonable to expect higher weights around or near its location, which is the case of the most important variables: `v100, v10`. We also note that variables such as temperature

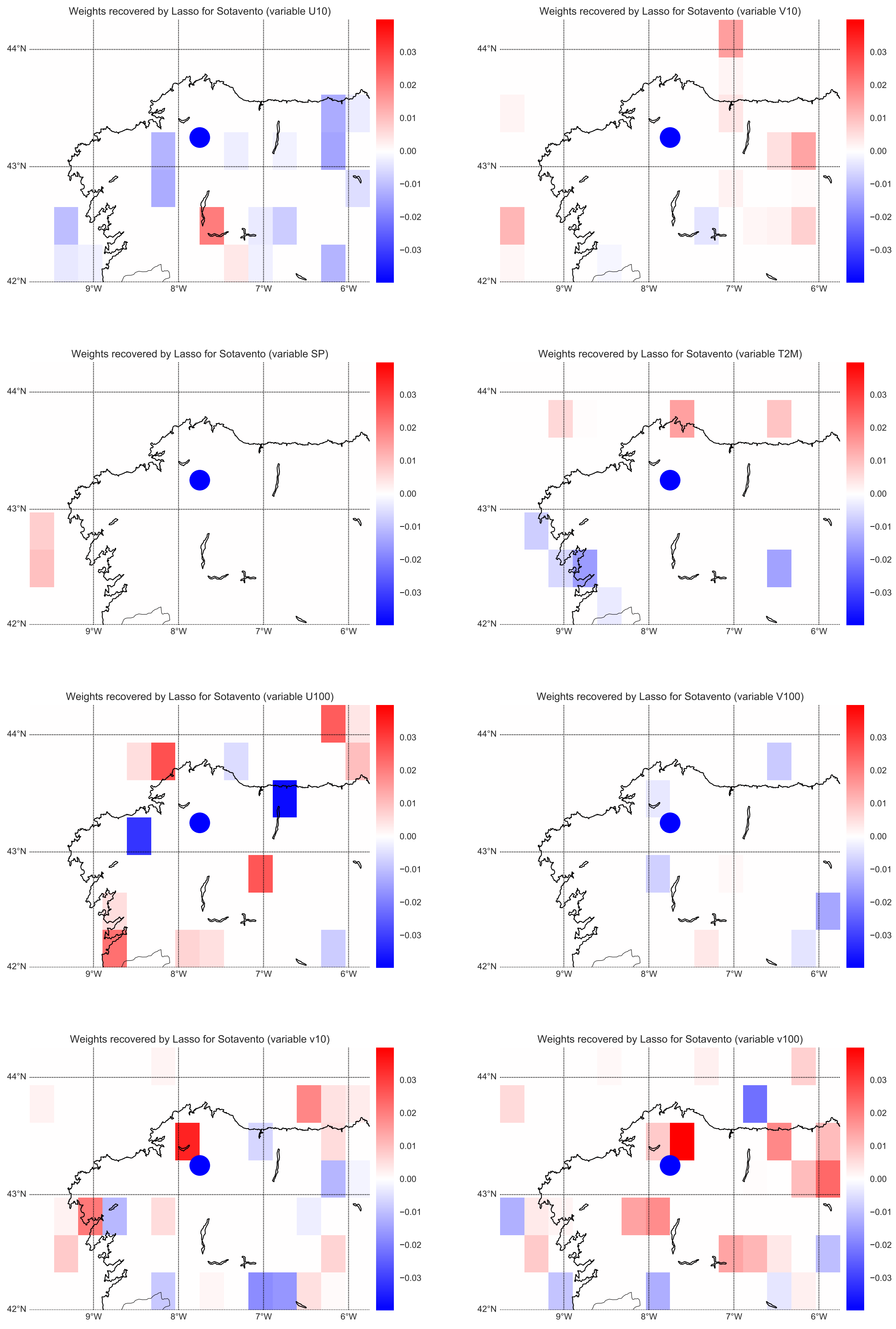

Weights recovered by Lasso for Sotavento (variable U10)
Weights recovered by Lasso for Sotavento (variable V10)
Weights recovered by Lasso for Sotavento (variable SP)
Weights recovered by Lasso for Sotavento (variable T2M)
Weights recovered by Lasso for Sotavento (variable U100)
Weights recovered by Lasso for Sotavento (variable V100)
Weights recovered by Lasso for Sotavento (variable v10)
Weights recovered by Lasso for Sotavento (variable v100)
44°N
43°N
42°N
9°W
8°W
7°W
6°W
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03

Figure 4.2.7: Lasso best model's weight structure

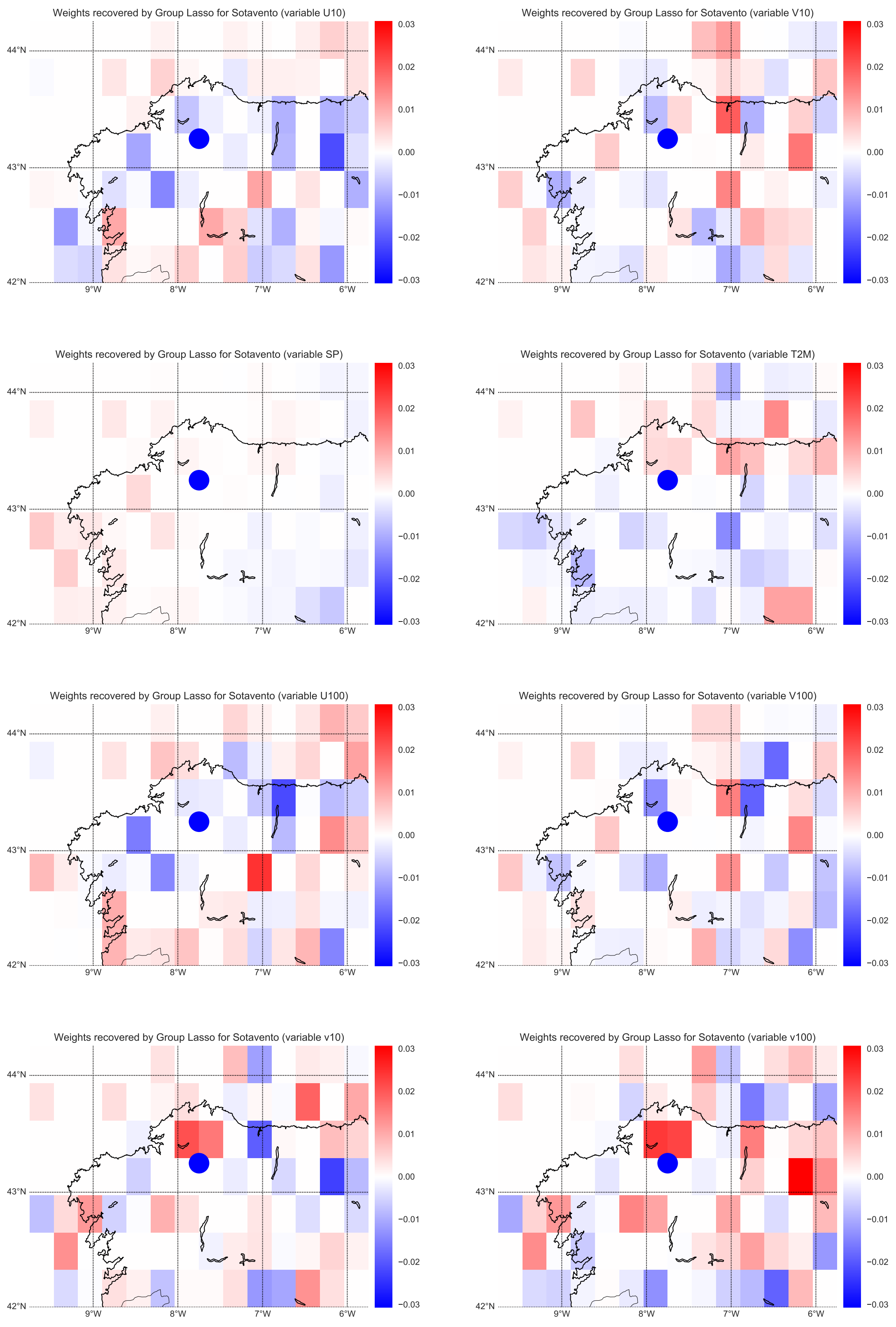

Weights recovered by Group Lasso for Sotavento (variable U10)
Weights recovered by Group Lasso for Sotavento (variable V10)
Weights recovered by Group Lasso for Sotavento (variable SP)
Weights recovered by Group Lasso for Sotavento (variable T2M)
Weights recovered by Group Lasso for Sotavento (variable U100)
Weights recovered by Group Lasso for Sotavento (variable V100)
Weights recovered by Group Lasso for Sotavento (variable v10)
Weights recovered by Group Lasso for Sotavento (variable v100)
44°N
43°N
42°N
9°W
8°W
7°W
6°W
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03

Figure 4.2.8: Group Lasso best model's weight structure

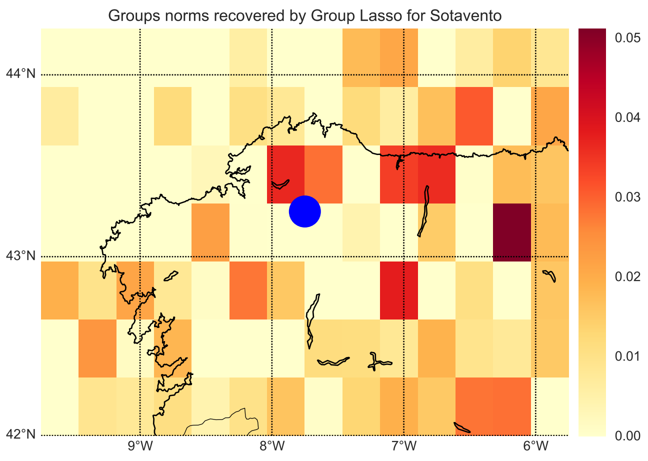


Figure 4.2.9: Group Lasso model groups' norms

or surface pressure do not seem to have the same effect in the wind energy output, as we could expect. It is also interesting to see several points of no importance at all, since Group Lasso selects only those points where all variables' weights are greater than zero (in absolute value). As a consequence of the grouped structure Group Lasso is somehow forced to select some points that may not be that important, as is the case of the point in the far right side of the map that is not particularly close to Sotavento as to be the most important point.

A final interesting interpretation can be extracted from Figure 4.2.9, where we have plotted a single map where each grid point represents the norm of each group according to the Group Lasso weights. We see many white points, indicating points that are not important as the weights are 0. On the other hand, we see about 5 very interesting points located near the real location of Sotavento. Even though we can try to extract similar interpretations from the Lasso model, it is easier and more natural to do so given the already grouped structure of the Group Lasso weights. We can then conclude that for problems with this kind of natural structure Group Lasso is an interesting model.

As we mentioned before, when Group Lasso selects a point it sets all its variables to some nonzero value, even if some variable cancels any other in the group. This is an artificial constraint that may actually result in a worse model, both in terms of accuracy and interpretability. To overcome this issue there are several alternatives, and one that we like is proposed by Friedman et al. in [15]. In this paper the authors propose to add a ℓ_1 norm besides of the $\ell_{2,1}$ norm of standard Group Lasso as the regularization term. This added ℓ_1 norm would introduce sparsity globally to remove some non important points that may be selected by standard Group Lasso. To balance the load of both penalization terms we add an α parameter, in a somehow similar way to Elastic Net. We will approach this and other alternatives in further work.

Chapter 5

Discussion and Further Work

The theory of convex optimization is core to the solution of many machine learning problems. Among them we find interesting models as those studied here, Lasso and Group Lasso. Given their conditions and usually their non differentiable penalty terms we often have to use iterative algorithms to find their solutions, and some of the most common are ISTA and its fast version, FISTA.

In this sense, ISTA provides a generic solution to any composite convex problem by easily changing the proximal operator of the corresponding penalty term. This is an interesting solution when testing several convex models, even if it may not be the most effective method for a particular model. In this context, Nesterov [1] proposed a modified algorithm that accelerates the basic gradient descent method, on which ISTA is based, reaching an optimal convergence ratio of $\mathcal{O}(1/k^2)$ for Lipschitz convex functions that significantly improves the ISTA convergence of $\mathcal{O}(1/k)$.

Nonetheless, even though FISTA reaches a faster theoretical convergence, in practical problems it may show a non monotone behavior, resulting in a worse practical performance. This has motivated many papers in the recent literature trying to recover the optimal convergence ratio of FISTA through the application of many optimizations, some of which have been studied in this work and applied to Lasso and, particularly, Group Lasso. We summarize them next:

- Backtracking: the backtracking strategy allows to find a better suited Lipschitz constant L for specific regions of the function to minimize, which may help to get a better step.

- Restarting strategies: O'Donoghue and Candes [7] proposed two interesting options to restart the FISTA's Nesterov t_k sequence whenever the algorithm overestimates the momentum; that is, the behavior is non monotone. To detect this behavior the authors proposed to restart if the value of the objective function increases or if the direction of the gradient is ascending.

- Decreasing L_k: given that L_k defines the size of the step we take in each iteration, it would be interesting to have the lowest possible L_k at each step, allowing to take greater steps.

- Stability: the aforementioned decreasing strategy may show some instability issues because it somehow cancels the backtracking strategy (ones tries to increase L_k and the other to decrease it). In this sense, this stability strategy gradually stops the decreasing of L_k, so that, in the limit, there is no decreasing.

- Maintain top-speed: restarting is very interesting, overall, in the first iterations of the algorithm, whereas it may cancel a good momentum near the optimum, where far higher momentum is required to make the algorithm advance on the right direction. This strategy, then, avoids restarting the algorithm near the optimum.

After the theoretical explanation we have performed some experiments focused on studying the effect of these optimizations in the actual performance of FISTA. To do so we have run the algorithms in two different problems: a synthetic and a real world one. We summarize our conclusions next:

1. In general, we have observed that FAPG shows better performance than FISTA.
2. Among all the optimizations, the one that more greatly impacts the performance is the decreasing strategy.
3. FAPG not only performs better in a single run, but its boost is even greater in cross-validation setups. In these situations we have found that the benefit from the optimizations is greater under low penalty conditions, where the algorithms need to work more.
4. In the worst case the performance of FAPG is similar to FISTA.
5. With regard to Group Lasso, we have observed that it is competitive with linear models such as Lasso and Ridge. Apart from this, it incorporates an advantage in terms of interpretability, which makes it an interesting option for naturally grouped problems.

This line of research leaves many possible directions to improve in further work, some of which we are currently undertaking. In particular, we are currently pursuing the following main ideas

1. When adding optimizations in order to accelerate the methods it is also very interesting and important to perform a detailed study of the final complexity and computational costs (time, iterations, etc). We are currently working on these issues for Lasso and specially Group Lasso.
2. Along this line, we will also explore the Sparse Group Lasso [15] as an sparse alternative to standard Group Lasso. This may balance the final model and avoid selecting some non important points and produce a more meaningful solution. Furthermore, its proximal operator is a bit more complex than that of standard Group Lasso and, as a consequence, including the optimizations studied here seems interesting and promising.
3. As a natural follow up we are exploring conjugate gradient methods, also based on momentum. The main difference between these and Nesterov accelerated gradient is that in Nesterov we apply the gradient step over the momentum point, whereas in conjugate gradient methods we perform the gradient over the previous x point and then perform the momentum step.
4. Another natural extension of this work is to introduce Nesterov accelerations in other problems, such as Generalized Lasso. Other more complex ideas include adding these optimizations to the SMO algorithm to solve the SVM model or even to deep learning.

5. On the applications side, we will continue to work on a follow up of [14] and [16], on the field of renewable energies prediction, introducing Group Lasso as another model or as a previous feature selection algorithm after adding more Numerical Weather Prediction variables, which would increase the dimensionality of the problem.

Since some of the mentioned problems are also convex (SVM, Deep Learning) we can think of adding these optimizations as a natural idea. Nonetheless, it also raises some issues. In the case of the backtracking, it is usually intractable to compute every iteration the objective function for big problems such as SVM or deep learning. In these cases we think that strategies such as restarting are more likely to work and provide interesting results.

Bibliography

[1] Yurii Nesterov. Introductory lectures on convex optimization. *Applied Optimization*, 2004.

[2] Erik J Balder. On subdifferential calculus. *Lecture notes, Universiteit Utrecht.[51, 52]*, 2010.

[3] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, Jan 2009.

[4] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer, 2009.

[5] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, 2015.

[6] Naoki Ito, Akiko Takeda, and Kim-Chuan Toh. A unified formulation and fast accelerated proximal gradient method for classification. *Journal of Machine Learning Research*, 18(16):1–49, 2017.

[7] Brendan O'Donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, Jul 2013.

[8] Yurii Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, Dec 2012.

[9] Weijie Su, Stephen Boyd, and Emmanuel Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, 2014.

[10] Katya Scheinberg, Donald Goldfarb, and Xi Bai. Fast first-order methods for composite convex optimization with backtracking. *Found. Comput. Math.*, 14(3):389–417, June 2014.

[11] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, Feb 2006.

[12] Alberto Torres-Barrán and José R. Dorronsoro. Nesterov acceleration for the smo algorithm. *Lecture Notes in Computer Science*, pages 243–250, 2016.

[13] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. *Journal of Machine Learning Research*, 2013.

[14] Alejandro Catalina, Alberto Torres-Barrán, and José R. Dorronsoro. Satellite based nowcasting of pv energy over peninsular spain. *Lecture Notes in Computer Science*, pages 685–697, 2017.

[15] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231-245, Apr 2013.

[16] Alejandro Catalina, Alberto Torres-Barrán, and José R. Dorronsoro. Machine learning prediction of photovoltaic energy from satellite sources. *Lecture Notes in Computer Science*, pages 31–42, 2017.