Universidad Autónoma de Madrid
Escuela Politécnica Superior
Departamento de Ingeniería Informática

Trend Filtering Techniques for Time Series Analysis

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

By
Daniel López Arias

under the direction of
José R. Dorronsoro Ibero

Madrid, June 30, 2016

Contents

Abstract

Time series can be found almost everywhere in our lives and because of this being capable of analysing them is an important task. Most of the time series we can think of are quite noisy, being this one of the main problems to extract information from them. In this work we use Trend Filtering techniques to try to remove this noise from a series and understand the underlying trend of the series, that gives us information about the behaviour of the series aside from the particularities that it can have when we look at it. We can also easily learn by using Trend Filtering the points when the trend of the series changes.

We begin this work introducing the elements of the convex optimization theory, that are key to solve the Trend Filtering problem. Then this problem is tackled as a convex optimization one and we find the solution to a number of different variations on this problem. Some extensions to Trend Filtering are also explored, specially Polished Trend Filtering, which uses the points that Trend Filtering finds to be those where the trend changes to construct a more precise model.

Finally, we apply this technique to analyse and forecast the energy demand on Spain, comparing our results with those of an autoregressive model, finding that, for short-term predictions, Polished Trend Filtering can outperform it.

Acknowledgements

First of all I would like to thank my tutor José Dorronsoro for his guidance throughout this work. His advice has been really useful and has really helped me to find the key to solve the most problematic issues I have faced in this work.

I also want to give thanks to my family, specially my parents for their help and for their support when I need them the most.

And finally I would also like to thank all my friends for their help.

Chapter 1

Introduction

The analysis of time series is a matter of great importance, due to the amount of situations that can be described considering the status of a system or some kind of measure depending on the moment when we look at its variables. When we look at a time series, in most of the cases it is reasonable to think that it is possible to split it into a number of components that, joined together, form the actual series. There will possibly be some variation on the series that is repeated cyclically. An example of this behaviour could be the temperature. Although it is not the same all the days, it does conserves from one day to another a similar structure, at least in terms of the hours at which the temperature is growing or decreasing.

We can also consider time series to have some stochastic or random part. Of course this doesn't mean that this is actually the case, but it is an elegant way to mathematically encode our ignorance about the reasons behind some of the variance on the time series. We can also think of another component, the *trend* of the series. In most of the series it is not difficult to find that besides the random component and the periodicities present in the series there is also some kind of tendency in the sense that, regardless of all the noise present in the series, it is growing, remains approximately at the same level, etc. Since the periodical part of the series is, at least for the short term, well-known and the random component is impossible to know by definition, being able to find the trend that is behind the series is an interesting task and it is really useful to help us understand the behaviour of the series. At this Master thesis we consider a method for achieving this that also helps us to identify the points where important changes on the trend of the series are located.

This method is called L_1 Trend Filtering and was proposed first by Boyd at [1]. The framework in which we will be working to calculate this type of Trend Filtering is that of Convex Optimization. At Chapter 2 the elements of this part of the Optimization theory are introduced, together with those of Subdifferential Calculus, an extension to ordinary calculus that is extensively used in the context of Convex Optimization, being because of this key to understand it. We give a special look at the Lasso problem, one of the most representatives problems in this area.

Then, at Chapter 3 the problem of Trend Filtering (TF) is introduced. Some other types of TF different than L_1 are explored, but the focus is on the L_1 version of it. We solve the TF problem and discuss some variations on it that have proven to be useful.

Once the theory related with TF has been introduced, this technique is applied to the time series of the Energy Demand on the peninsular part of Spain. Being able to analyse and even to try to predict this demand is very useful for energy companies, since depending on the demand that is needed more or less energy will have to be produced. It is necessary to produce more energy than the actual demand to avoid power cuts but if too much energy is produced, that energy will be eventually lost, since we lack a way to keep great amounts

of electric energy for a long time without losing it.

Of course, being this such an important problem, there are already implemented systems to predict the energy demand that have access to a number of relevant variables that we don't have. It is anyway, a good problem to test the capabilities of L_1 TF. First we use only TF to predict the demand but an extension of TF, TF with Polishing is also used to try to improve the performance of the first one.

Chapter 2

Convex Optimization

In this chapter the problem of the optimization of a convex function is considered. Convex functions are particularly interesting when solving optimization problems because they have a number of properties that can be useful when trying to minimize or maximize functions. Because of this we are going to try to set the problems we need to deal with so they fit in this framework and we are able to take advantage of these properties. This chapter is structured as follows:

First, the general Convex Optimization problem is presented and some basic definitions are introduced at section 2.1. In order to solve such problems, ordinary differential calculus is not enough, due to the possibly non-differentiable function that is to be optimized. An extension to differential calculus, the subdifferential calculus, is explored at section 2.2, giving us the mathematical framework we are going to need. At section 2.3, several algorithms that use the concept of the proximal operator are described. This includes the proximal step descent, and ISTA/FISTA algorithms. After this, an example of a problem that we are able to solve using these techniques, the Lasso, is explored at section 2.4. The Standard version of the Lasso is introduced and solved, while for the Generalized version of the Lasso, we introduce it and give a method to solve it in some important cases, by transforming it into a Standard Lasso.

2.1 Introduction

Some definitions to set the framework we are going to work with when solving convex optimization problems are introduced below:

Definition 1. ***Euclidean space:*** *An Euclidean space $\mathbb{E}$ is a finite dimensional vectorial space, with an interior product and a norm defined. This norm is defined by means of the dot product.*

Definition 2. ***Convex set:*** *A set A is said to be convex if $\forall t \in (0,1)$ we have*

$$tx + (1-t)y \in A \ \forall x, y \in A.$$

Loosely speaking, this means that the points on the segment joining x and y are, all of them, inside the set A. See Figure 2.1.1 for an example of a convex set.

Definition 3. ***Convex function:*** *A convex function f is a function, $f : \mathbb{E} \to (-\infty, +\infty]$ whose domain is a convex set for which, with x, y, t as in the previous definition, the following holds:*

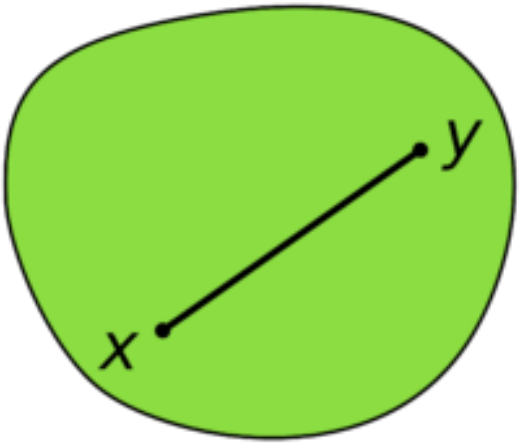


Figure 2.1.1: Example of a convex set [Wikipedia, "Convex set"]

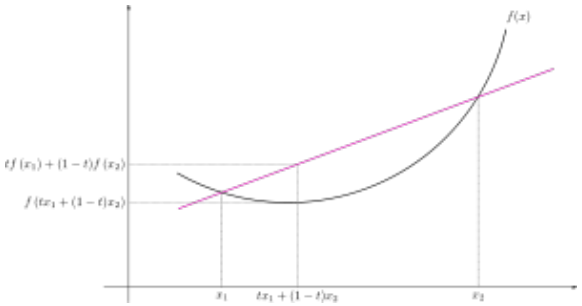


Figure 2.1.2: Example of a convex function [Wikipedia, "Convex function"]

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

An example of such function can be seen at Figure 2.1.2. If the previous inequality is strict, such a function is said to be strictly convex.

Definition 4. ***Effective Domain:*** *The effective domain of a function f is the set*

$$dom(f) = \{x \in \mathbb{E} : f(x) < +\infty\}$$

If $dom(f)$ is not empty, the function f is said to be proper.

Two simple methods to create new convex functions, using functions that we already know that are convex are described next.

Proposition 2.1.1. *If $f_1, ..., f_m : \mathbb{R}^n \to (-\infty, +\infty]$ are convex functions, functions $f(x)$ defined as below will also be convex:*

- *For any $\alpha_i \in \mathbb{R}_+$, $f(x) = \sum_{i=1}^{m} \alpha_i f_i(x)$.*
- $f(x) = \max_{1 \leq i \leq m} f_i(x)$

Proofs: If f_i is convex and $\alpha_i \geq 0 \; \forall i$, we have that

$$f_i(tx + (1-t)y) \leq tf_i(x) + (1-t)f_i(y) \; \forall i$$

and the same will apply for the sum $f = \sum_{i=1}^{m} f_i$ since

$$\forall i \ \alpha_i f_i(tx + (1-t)y) \leq t\alpha_i f_i(x) + (1-t)\alpha_i f_i(y)$$

and this property also applies trivially for the sum:

$$\sum_{i=1}^{n} \alpha_i f_i(tx - (1-t)y) \leq t \sum_{i=1}^{n} \alpha_i f_i(x) + (1-t) \sum_{i=1}^{n} \alpha_i f_i(y)$$

Regarding the second property, for $0 \leq \alpha \leq 1$, we have that for $f(x) = max_i f_i(x)$

$$\begin{aligned} f(\alpha x + (1-\alpha)y) &= \max f_i(\alpha x + (1-\alpha)y) \\ &\leq \alpha \max f_i(x) + (1-\alpha) \max f_i(y) \\ &= \alpha f(x) + (1-\alpha) f(y). \end{aligned}$$

□

Proposition 2.1.2. *Let f be a proper convex function over $\mathbb{E}$. If $x^* \in \mathbb{E}$ is a local minimum, it is necessarily also a global minimum.*

Proof: If x is a local minimum, we know that there exists a neighbourhood A around x, for which for all $y \in A, f(x) \leq f(y)$. We can also therefore find a small $\epsilon > 0$ for which for all $y \in \mathbb{E}$ we have $(1-\epsilon)x + \epsilon y \in A$. But, applying the definition of convex function,

$$f(x) \leq f((1-\epsilon)x + \epsilon y) \leq (1-\epsilon) f(x) + \epsilon f(y)$$

Which leads us to

$$\epsilon f(x) \leq \epsilon f(y)$$

And finally we get

$$f(x) \leq f(y)$$

proving that x is a global minimum. □

2.2 Subdifferential calculus

In this section, subdifferential calculus is introduced. This extension of ordinary differential calculus is very useful when we need to solve a convex optimization problem that is non-differentiable. In such problems, typical methods that use gradients in one way or another cannot be applied unless the concepts of gradient and differential are generalized by those of subgradient and subdifferential respectively.

2.2.1 Basic definitions

First, some essential definitions are introduced, followed by a number of properties of the subdifferential calculus, and some important results.

Definition 5. ***Subgradient:*** *A subgradient of a convex function f at $x \in \mathbb{E}$ is a vector $\xi \in \mathbb{E}$ for which $\forall y \in \mathbb{E}$, the following condition holds:*

$$f(y) - f(x) \geq \langle \xi, y - x \rangle .$$

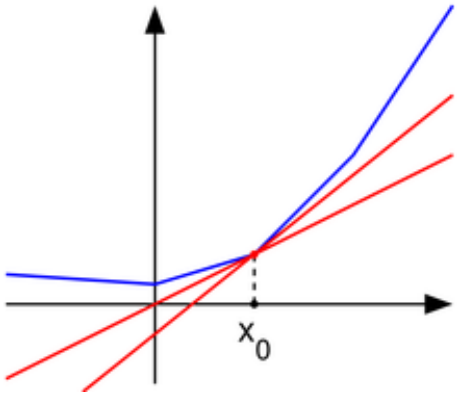


Figure 2.2.1: A convex function and some of its subtangent lines at x_0 in red. [Wikipedia "Subdifferential"]

Definition 6. ***Subdifferential:*** *The subdifferential of a convex function is the set of subgradients* $\partial f : \mathbb{E} \to 2^{\mathbb{E}}$*, defined as*

$$\partial f(x) = \left\{ \xi \in \mathbb{E} : f(x) + \xi^t (y - x) \leq f(y), \forall \in \mathbb{E} \right\}. \tag{2.2.1}$$

Proposition 2.2.1. *For* f *a closed and convex function, differentiable on its domain, we have that*

$$\partial f(x) = \{\nabla f(x)\}$$

for all $x \in$ *int*$(\text{dom}\, f)$.

Proof: To see this, we fix $x \in \text{int}(\text{dom}\, f)$. It is possible to prove that, for any vector $p \in \mathbb{R}^n$ and $g \in \partial f(x)$ we have

$$\langle \nabla f(x), p \rangle = \nabla f(x; p) \geq \langle g, p \rangle .$$

For the complete proof of this point, check Theorem 3.1.14 from [2].

Since we can also choose another vector $p' = -p$ and repeat this argument, we reach to the conclusion that $\langle \nabla f(x), p \rangle = \langle g, p \rangle$ for any $g \in \partial f(x)$. Now we conclude considering the vector e_k formed by zeros in all the components except k, where we have a 1, $p = e_k, k = 1, ..., n$, which leads us to $g = \nabla f(x)$. □

However, in those points where the gradient is not defined, but the subdifferential is, this last one becomes a set of points; therefore, it is not defined by an unique value but it is a set valued operator.

At Figure 2.2.1, a non-differentiable convex function is plotted in blue. At x_0 such function does not have a gradient, but the subdifferential operator can be defined at x_0. Two tangent lines, each of them with a slope fitting in the set of points for the subdifferential at x_0 are marked in red.

2.2.2 Subdifferential calculus

Some of the most relevant properties of the subdifferential operator are introduced next:

- For any $x \in \mathbb{E}, \partial f(x)$ is either a closed convex set or the empty set.
- $\forall x \in \text{int}(dom(f))$, $\partial f(x)$ is not empty and it is bounded.
- $\partial f(x) = \nabla f(x)$ if and only if f is differentiable in $x \in \mathbb{E}$.

- $x \in \mathbb{E}$ is a global minimum of f if and only if $0 \in \partial f(x)$.

For a proof of the first property see proposition 16.3 at [3]. The proofs of second and third properties can be found at [4], lemma 2.16 and proposition 2.6, respectively.
The last property is key in the theory of convex optimization and it is called Fermat's rule. We will see in section 2.3 how this property proves to be extremely useful to solve convex optimization problems. The proof for this last property is given a few lines later, at Theorem 2. But first, an important result related with the subdifferential of two convex functions and its sum is given:

Theorem 1. ***Moreau-Rockafellar:*** *Let f and g be convex functions. Then $\forall x_0 \in \mathbb{R}^n$ the following relation holds:*

$$\partial f(x_0) + \partial g(x_0) \subset \partial(f+g)(x_0). \tag{2.2.2}$$

Besides, if int(dom $f \cap$ dom $g) \neq \emptyset$ *we also have*

$$\partial(f+g)(x_0) \subset \partial f(x_0) + \partial g(x_0). \tag{2.2.3}$$

Proof: 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, $\forall x \in \mathbb{R}^n$,

$$f(x) \geq f(x_0) + \xi_1^t(x - x_0),$$

$$g(x) \geq g(x_0) + \xi_2^t(x - x_0),$$

Summing these inequalities we obtain:

$$f(x) + g(x) \geq f(x_0) + g(x_0) + (\xi_1 + \xi_2)^t(x - x_0).$$

And therefore $(\xi_1 + \xi_2) \in \partial(f+g)(x_0)$, proving this part of the theorem.
The proof of the second inclusion of this theorem can be found in Theorem 2.9 at [4]. □

Theorem 2. ***Fermat's rule:*** *For a proper convex function f we have*

$$\operatorname{argmin} f = \{x \in \mathbb{E}\ 0 \in \partial f(x)\}$$

Proof: $x \in \operatorname{argmin} f$ if there is no other y for which $f(y) \leq f(x)$. Therefore we have

$$(y - x)^t 0 + f(x) \leq f(y)\ \ \forall y$$

and applying the definition of subdifferential, this is equivalent to saying $0 \in \partial f$. □

Subdifferential operators are known to be monotone, being this a really useful property that will be applied several times throughout this text in the resolution of optimization problems. This concept is defined next.

Definition 7. ***Monotone operator:*** *An operator $T : \mathbb{E} \to 2^{\mathbb{E}}$ is said to be monotone if the following condition holds*

$$(\xi_1 - \xi_2)^t(x_1 - x_2) \geq 0, \forall x_1, x_2 \in \mathbb{E},\ \xi_1 \in T(x_1), \xi_2 \in T(x_2).$$

Proposition 2.2.2. *The subdifferential operator is monotone.*

Proof: Using the definition of subdifferential, we know that

$$\forall x_1, x_2 \in \mathbb{E}, \xi_1 \in \partial f(x_1), \xi_2 \in \partial f(x_2)$$

which leads us to

$$(\xi_1 - \xi_2)^t (x_1 - x_2) \geq 0.$$

The proof is quite straightforward, since the definition of subdifferential gives us the two following inequalities:

$$f(x_1) - f(x_2) \leq \xi_1^t (x_1 - x_2),$$
$$f(x_2) - f(x_1) \leq \xi_2^t (x_2 - x_1).$$

Summing these two expressions it is immediate to prove the monotony of the subdifferential:

$$\xi_1^t (x_1 - x_2) + \xi_2^t (x_2 - x_1) \geq 0$$

But we also have

$$\xi_1^t (x_1 - x_2) - \xi_2^t (x_1 - x_2) \geq 0$$

Which leads us to the monotonicity

$$(\xi_1 - \xi_2)^t (x_1 - x_2) \geq 0.$$

□

Monotone operators, and therefore the subdifferential, have a number of interesting properties that are useful when solving convex optimization problems. Some of the most important ones are described next.

Proposition 2.2.3. *If T is a monotone operator, the following conditions hold:*

- *αT is monotone $\forall \alpha \geq 0$.*
- *T monotone $\Rightarrow T^{-1}$ monotone.*
- *The resolvent operator defined by $R_T = (I + T)^{-1}$ is univalued and firmly non-expansive, that is,*

$$\langle R_T(x_1) - R_T(x_2), x_1 - x_2 \rangle \geq ||R_T(x_1) - R_T(x_2)||^2, \forall x_1, x_2 \in \mathbb{E}.$$

Proofs: The first point is immediate from the definition.
In order to check the second one we consider the graph of T, $graph(T) = \{(\xi, x) : \xi \in T(x)\}$. We know that T is monotone, so

$$(\xi_1 - \xi_2)^t (x_1 - x_2) \geq 0, \forall x_1, x_2 \in \mathbb{E}, \xi_1 \in T(x_1), \xi_2 \in T(x_2).$$

Now, the monotonicity of T^{-1} is obvious from the definition.

Regarding the third one, let us check first that R_T is single valued. We consider the graph of R_T. If we suppose that there is z such that $(z, x_1), (z, x_2) \in graph(R_T)$ we would have

$$z \in x_1 + T(x_1), z \in x_2 + T(x_2).$$

Therefore, there would exist $\xi_1 \in T(x_1), \xi_2 \in T(x_2)$ for which $z = x_1 + \xi_1 = x_2 + \xi_2$ and $\xi_1 - \xi_2 = -(x_1 - x_2)$.

Now, applying the fact that T is monotone

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

To check the firmly non-expansiveness, we define $x_i = R_T(z_i)$, and then $z_i \in x_i + T(x_i)$ and $z_i = x_i + \xi_i$ with $\xi_i \in \partial(x_i)$. Finally we have

$$\begin{aligned}(x_1 - x_2)^t(z_1 - z_2) &= (x_1 - x_2)^t(x_1 - x_2) + (x_1 - x_2)^t(\xi_1 - \xi_2) \\ &= ||x_1 - x_2||^2 + (x_1 - x_2)^t(\xi_1 - \xi_2),\end{aligned}$$

and, being T monotone, we have $(x_1 - x_2)^t(\xi_1 - \xi_2) \geq 0$, and firmly non-expansiveness of R_T is immediate. □

Proposition 2.2.4. *For all $\alpha > 0$, the points where zero is in the subgradient of a monotone operator T coincide with the fixed points of the resolvent $R_{\alpha T}$ or, more precisely,*

$$0 \in T(x) \Leftrightarrow x = R_{\alpha T}(x).$$

Proof: In order to prove this, it is just necessary to check that, for $\alpha > 0$ we have

$$0 \in T(x) \Leftrightarrow 0 \in \alpha T(x) \Leftrightarrow x \in x + \alpha T(x)$$

But this expression can be transformed easily into one containing the resolvent, as is shown next

$$x \in (I + \alpha T)(x) \Leftrightarrow x = (I + \alpha T)^{-1}(x) \Leftrightarrow x = R_{\alpha T}(x).$$

□

2.3 Proximal algorithms

In this section, a number of proximal algorithms, that is, algorithms that use the proximal operator to solve an optimization problem, are discussed. A few introductory concepts are introduced first. Then, the problem of the optimization of the sum of two convex functions is introduced. This problem will be a very important one, as we will see later, since many important practical problems fall into this schema. Finally, three proximal algorithms are discussed: the gradient proximal method and the ISTA and FISTA algorithms.

Definition 8. ***Proximal operator:*** *The proximal operator of a function f is defined as the resolvent of the subdifferential operator and it is denoted by $prox_f$. Therefore*

$$prox_f = (I + \partial f)^{-1}. \tag{2.3.1}$$

It is interesting to note that if $x = \text{prox}_f(z)$, this leads us to

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

The previous definition can be, in some cases, difficult to apply in a real problem, since obtaining the inverse of an expression with a subdifferential in it is not, for most of the functions, an easy task. An alternative, more useful in the practical sense definition, is shown next:

Definition 9. ***Proximal operator (alternative version):***

$$prox_f(z) = \underset{y\in\mathbb{R}}{\operatorname{argmin}} \left(f(y) + \frac{1}{2}||z-y||^2 \right).$$

Now we prove the equivalence between these two definitions of the proximal operator. First we show that the alternative definition implies the original one.
We consider $\phi(y) = f(y) + \frac{1}{2}||z-y||^2$ and $x = \operatorname{argmin}\phi(y)$.
Since $\partial\phi(y) = y - z + \partial f(y)$ and x is a minimizer we get back the first definition

$$0 \in x - z + \partial f(x) \Rightarrow x = (I + \gamma\partial f)^{-1}(z)$$

In order to prove the other part of the equivalence we begin by supposing $z - x \in \partial f(x)$, or $x = (I + \gamma\partial f)^{-1}(z)$. If we recall the definition of subdifferential we have

$$f(y) \geq f(x) + (z-x)^t(y-x)$$

Now we apply the alternative version of the definition

$$f(y) + \frac{1}{2}||y-z||^2 \geq f(x) + (z-x)^t(y-x) + \frac{1}{2}||y-z||^2.$$

If $\psi(y) = (z-x)^t(y-x) + \frac{1}{2}||y-z||^2$ and $\nabla\psi(y) = z - x + y - z = y - x$, the minimum of ψ is reached at x. This way we have that $\forall y$

$$f(y) + \frac{1}{2}||y-z||^2 \geq f(x) + \psi(y) \geq f(x) + \frac{1}{2}||x-z||^2$$

We finally get the alternative definition, since $x = \operatorname{argmin}_y \left\{ f(y) + \frac{1}{2}||z-y||^2 \right\}$ □

We will see now some methods that allow us to introduce the restrictions that we might want to consider in a given optimization problem. That is, we want the solution of our problem to be in a determined set $S \subseteq \mathbb{E}$. So we are looking for the solution of the problem

$$\begin{array}{ll} \min_{x\in\mathbb{E}} & f(x) \\ \text{subject to} & x \in S. \end{array}$$

An auxiliary function that will help us tackle this problem is defined next:

Definition 10. ***Indicator function:*** *The indicator function of $S \subseteq \mathbb{E}$ is*

$$I_S(x) = \begin{cases} 0 & x \in S, \\ +\infty & x \notin S \end{cases}$$

If we sum to the function we want to optimize, f, an adequate indicator function, the sum of the two functions will only have finite values in the set where the indicator function is 0, that is, the set matching the constraints of the problem. If f is a convex function, since $g(x) = I_S(x)$ is also convex, the sum $f(x) + g(x)$ is also convex.

This suggests that an interesting approach to solve an optimization problem with restrictions would be to transform such problem into a convex optimization one consisting on the sum of the two former functions f and g. Several ways to proceed when we need to solve these problems are discussed at subsections 2.3.1, 2.3.2.

2.3.1 Optimization of the sum of two convex functions

The problem of the optimization of the sum of two convex functions, one of them being differentiable but not the other, is a very natural problem that we will need to solve frequently, since it fits in the framework where we are trying to minimize a traditional cost function, while using a regularization function (or, in our case a few lines before, the restrictions of the problem).

In such cases, the proximal operator becomes a powerful resource, provided that we are able to calculate it, to solve the optimization problem, combining what we know about the proximal operator with Fermat's rule.

We have talked before about the usual scenario where our optimization problem has a part consisting on the cost function ($f(x)$ in our formulation), and another regularization term (that would be $g(x)$). Most of the times, the cost function will be differentiable and easy to deal with. However, the regularization term is usually not differentiable (an example of this would be the Lasso regularization, at which we will take a look soon).

In the case that the regularization term is not differentiable, we need to use the properties of the subdifferential (Fermat's rule) and Moreau-Rockafellar's to note that the solution will be in the points where $0 \in \partial f(x) + \partial g(x)$. Since $f(x)$ is differentiable this is equivalent to

$$0 \in \nabla f(x) + \partial g(x).$$

Finally, we check that the solution will be $x = prox_{\gamma g}(x - \gamma \nabla f(x))$ taking into account that

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

We introduce a term γ

$$x - \gamma \nabla f(x) \in x + \gamma \partial g(x)$$

Reaching finally an expression for the proximal operator

$$x = \text{prox}_{\gamma g}(x - \gamma \nabla f(x)). \tag{2.3.2}$$

2.3.2 Convex optimization algorithms

Some efficient algorithms for solving convex optimization problems, that make an iterative use of the concept of proximal operator, are explained next. The equation (2.3.2) suggests us that a possible way to obtain the solution to such problems, provided that the convergence can be shown, would be to use the proximal operator to iterate, that is, to proceed as

$$x_{k+1} = prox_{\gamma g}(x_k - \gamma \nabla f(x_k)). \tag{2.3.3}$$

It is remarkable to note that the well-known gradient descent algorithm is just a particular case of this one, when $g = 0$.

The Iterative Shrinkage Thresholding Algorithm, ISTA from now on, can be used to solve optimization problems with a L_1 regularization term. In order to deal with the L_1 norm in optimization problems, it is necessary to be able to calculate the proximal operator of the L_1 norm. This problem is considered next.

Algorithm 1: ISTA algorithm with constant step

Let $L := L(f)$ be the Lipschitz constant of f, and $w_0 \in \mathbb{R}, k = 1.$;
while *Algorithm has not converged* **do**
  $w_k = \text{prox}_{\frac{1}{L}f}(w_{k-1} - \frac{1}{L}\nabla(w_{k-1}))$;
end

Definition 11. ***Soft-Thresholding operator:*** *The soft-thresholding operator $S_\lambda(x)$ corresponds to the proximal operator of the L_1 norm, that is*

$$S_\lambda = (Id + \lambda\partial f(x))^{-1}$$

where $f(x) = |x|$

Proposition 2.3.1. *The soft-thresholding operator can be expressed as*

$$S_\lambda(x) = sgn(x)(|x| - \lambda)_+ \tag{2.3.4}$$

Proof: In order to prove this result we first compute the value of $\partial|x|$:

$$\partial|x| = \begin{cases} 1, & x > 0 \\ [-1, 1], & x = 0 \\ -1, & x < 0 \end{cases} \tag{2.3.5}$$

Now, for $f(x) = |x|$ we calculate the expression $(I + \lambda\partial f)$ as

$$(I + \lambda\partial f)(\hat{x}) = x = \begin{cases} \hat{x} + \lambda & \hat{x} > 0, \\ [-\lambda, \lambda] & \hat{x} = 0, \\ \hat{x} - \lambda & \hat{x} < 0. \end{cases} \tag{2.3.6}$$

now, inverting this is all that remains. We see that there are three different cases:

- If $\hat{x} < 0$ then $x = \hat{x} - \lambda$ and $\hat{x} = x + \lambda$, which is satisfied if $x < -\lambda$. This way:

$$\hat{x} = x + \lambda \text{ if } x < -\lambda$$

- If $\hat{x} = 0$ then $x \in [-\lambda, \lambda]$

$$\hat{x} = 0 \text{ if } x \in [-\lambda, \lambda]$$

- If $\hat{x} > 0$ then similarly we have $\hat{x} = x - \lambda$. Therefore

$$\hat{x} = x - \lambda \text{ if } x > \lambda$$

And we finally obtain (2.3.4) □

It is possible to use this result to the case of a general convex regularization function g, considering a convex function f responsible of measuring the error (like the square error, for instance)

$$\min\{F(\beta) = f(\beta) + g(\beta) : \beta \in \mathbb{R}^n\}.$$

In this context we assume the following statements

Algorithm 2: ISTA algorithm with backtracking

We take $L_0 > 0, \eta > 1, w_0 \in \mathbb{R}$;
while *Algorithm has not converged* **do**
  $i = 0$;
  repeat
    $\tilde{L} \leftarrow \eta_{k-1} L_{k-1}$;
    $w_L^k \leftarrow \text{prox}_{\frac{1}{L}g}\left(w_{k-1} - \frac{1}{\tilde{L}}\nabla f(w_{k-1)}\right)$;
    $i \leftarrow i + 1$;
  until $f(w_{\tilde{L}}^k) + g_{\tilde{L}}^k \leq Q(w_{\tilde{L}}^k, w_{k-1}, \tilde{L})$;
  $L^k \leftarrow \tilde{L}$;
  $w_k \leftarrow w_L^k$;
end

Algorithm 3: FISTA algorithm

For $L := L(f), y_1 = w_0 \in \mathbb{R}^n, t_1 = 1.$;
while *Algorithm has not converged* **do**
  $w_k = \text{prox}_{\frac{1}{L}}(y_k - \frac{1}{L}\nabla f(y_k).$;
  $t_{k+1} = \dfrac{1 + \sqrt{1 + 4t_k^2}}{2},$;
  $w_{k+1} = w_k + \left(\dfrac{t_k - 1}{t_{k+1}}\right)(w_k - w_{k-1}).$;
  Increment k;
end

- $g : \mathbb{R}^n \to \mathbb{R}$ is a convex continuous function.
- $f : \mathbb{R}^n \to \mathbb{R}$ is a continuous, class C^1 convex function for which the Lipschitz condition holds

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

If we consider the quadratic approximation for F in y:

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

whose unique minimizer is, applying the alternative definition of proximal we saw at Definition 9

$$p_L(y) = \text{argmin}\left\{Q_L(\beta, y) : \beta \in \mathbb{R}^n\right\},$$

as it is easy to prove that this expression is nothing more than the proximal of g evaluated at $y - \frac{1}{L}\nabla f(y)$.

$$p_L(y) = prox_{\frac{1}{L}g}\left(y - \frac{1}{L}\nabla f(y)\right)$$

since

$$\begin{aligned} prox_{\frac{1}{L}g}\left(y - \frac{1}{L}\nabla f(y)\right) &= \text{argmin}_\beta \left\{ \frac{1}{L}g(\beta) + \frac{1}{2}|| \left(y - \frac{1}{L}\nabla f(y)\right) - \beta||^2 \right\} \\ &= \text{argmin}_\beta \left\{ g(\beta) + \frac{L}{2}||\beta - \left(y - \frac{1}{L}\nabla f(y)\right)||^2 \right\} = p_L(y). \end{aligned}$$

A limitation of ISTA is that it is necessary to know an appropriate step size so the algorithm can work properly. If the function f is Lipschitz, we can use use its constant L, therefore using a fixed step, as shown in Algorithm 1, to optimize $(f + g)(x)$.

In the case that f is not Lipschitz, or maybe we are not able to determine the value of its Lipschitz constant, we still can use ISTA if we modify slightly the algorithm so that the step size is guaranteed to be a valid one. In order to achieve this we use an alternative version of the algorithm that uses backtracking, at Algorithm 2

The Fast Iterative Shrinkage Thresholding Algorithm, FISTA from now on [5], is very similar to the ISTA algorithm, but in this case the proximal operator is applied to a linear combination of the two previous points, instead of just the very previous point like in ISTA. This small change improves its performance in the worst case, being $O(\frac{1}{k^2})$ instead of the $O(\frac{1}{k})$ in ISTA. The details of the algorithm can be seen at Algorithm 3.

2.4 The Lasso Problems

2.4.1 Standard Lasso Problem

In this section we take a look at the Lasso regression. Lasso regression, which stands for "least absolute shrinkage and selection operator" is a shrinkage method for variable selection that will be used in next chapter to deal with the Trend Filtering problem. It can also be seen as an optimization problem of a function formed by the sum of two convex functions, therefore fitting in the framework we have been discussing at section 2.3.

The idea of the Lasso is to use a regularization method similar to the well-known Tikhonov regularization, or Ridge regression. Ridge regression chooses the optimum values of the variables of a problem by optimizing a combination of the sum of the squared error and a L_2 regularization component that encourages the coefficients not to be too big. The coefficients are calculated as:

$$\hat{\beta} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2. \right\} \tag{2.4.1}$$

This method can have problems when some of the coefficients in the optimal solution are identically zero, since their corresponding coefficients, although will tend to be shrunk to zero, will not be identically zero in most practical situations. This way, those coefficients will need to be considered throughout the entire computation.

Lasso regression is a variation on Ridge Regression that avoids the previous problem while yielding a similar performance to that of the Ridge method. The only difference between these problems is that Lasso uses a L_1 norm in the regularization term instead of the L_2 norm we saw at Ridge regression. Therefore the coefficients are obtained as:

$$\hat{\beta} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|. \right\} \tag{2.4.2}$$

At Figure 2.4.1 we show a picture showing an intuitive explanation of the reason why the Ridge Regression only manages to shrink the coefficients towards zero without actually

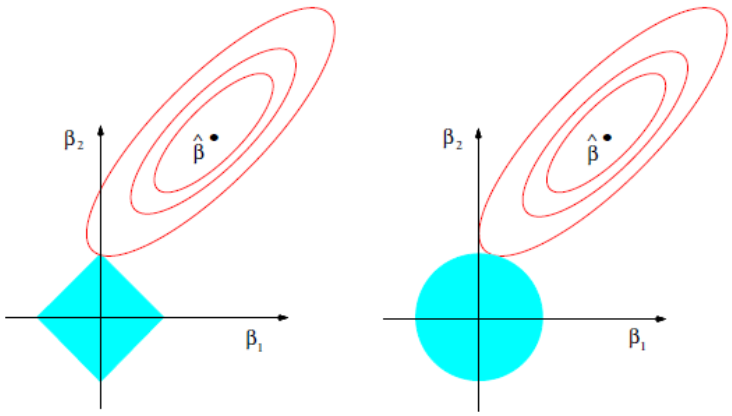


Figure 2.4.1: Lasso and Ridge estimates. The Elements of Statistical Learning

reaching zero. The Lasso restricted area, being a square, has edges on points where the coefficients are zero, and encourages the less important coefficient to be identically zero while Ridge does not, since now the intersection of the restricted area with the possible values of $\hat{\beta}$ will be most likely to be reached on such points, while in the restricted area of the Ridge there are no privileged points in this sense.

The use of the L_1 norm transforms what was a differentiable and easy to solve problem into a non-differentiable one, but as we have seen, we have several algorithms that allow us to solve the Lasso. These algorithms, however, solve the Lasso for a given value of λ. If we want to know all the set of solutions for the different values of λ, it is necessary to solve the problem for the different values of λ.

This way we can compute the whole solution path of the problem. By solution path we mean the different set of coefficients that are obtained by changing the value of the regularization term λ. As λ increases more and more estimators will become zero. The exact values of λ where the number of non-zero estimators changes define this concept of solution path of the problem.

In order to solve the Lasso, we can use the ISTA and FISTA algorithms that have been explained at Algoritms 1 and 3 at section 2.3

2.4.2 Generalized Lasso Problem

Now we consider a natural extension of the Lasso problem, the so called Generalized Lasso Problem [6]. In this generalized problem, a general penalization matrix D is introduced in the part of the function responsible of the regularization. This way, the coefficients are not necessarily shrunk like in Lasso, but they are adjusted to a specific restriction encoded in the matrix D. This allows us to use our previous knowledge about the structure of the coefficients that will solve the problem (such as possible relationships between coefficients, or similarities between them). This generalized Lasso problem is:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}||y - X\beta||_2^2 + \lambda||D\beta||_1 \tag{2.4.3}$$

Depending on the election of the matrix D a specific structure in the coefficients of the solution will be encouraged. For instance, a typical election for the penalization matrix, similar to the extra penalization matrix used at the Fused Lasso problem [7], similar to the Total Variation, is shown below:

$$D_{1d} = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ & & & \cdots & & \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{pmatrix} \tag{2.4.4}$$

By introducing such a $(n-1) \times n$ matrix in the equation we are transforming the original penalization, that only shrinks coefficients towards 0, into a penalization on the L_1 norm of the differences between two consecutive coordinates (or coefficients). This is easy to see since each row, when multiplied by the vector β, is transformed into $|\beta_{i+1} - \beta_i|$. This kind of adjustment can be very useful in problems where each coordinate is supposed to be highly correlated to their neighbours.

This idea can naturally be extended to a general adjacency graph. This way, we would work with a $m \times n$ matrix, being n the number of nodes of the graph, and m the number of edges. Therefore, in each row of the matrix, ones and minus ones will be placed depending on the pairs of nodes jointed by an edge.

The have already faced the problem of solving the Lasso, in its Standard version. Because of this, and also because it is reasonable to think that the original Lasso problem will be easier to solve than its Generalized version, it is interesting to try to find the conditions that are necessary for a generalized Lasso problem to be possible to be transformed into a normal Lasso problem. Although it is possible to find the solution for the Generalized Lasso, in this work we are going to work only with Generalized Lasso that can be transformed into Standard Lassos. For a discussion on the solution of the Generalized Lasso, check [6].

In order to perform such transformation, we first note that, in the case of D being invertible, an immediate method is available, since the generalized Lasso problem can be transformed into the normal Lasso by using the variable change $\theta = D\beta$

$$\min_{\theta \in \mathbb{R}^p} \frac{1}{2}||y - XD^{-1}\theta||_2^2 + \lambda||\theta||_1, \tag{2.4.5}$$

If the $m \times p$ D matrix is not invertible, it is still possible to transform the problem into a standard Lasso, provided that $rank(D) = m$ for a $D \in \mathbb{R}^{m \text{ x } p}$ matrix.

To do so, first, we construct a $\mathbb{R}^{p \text{ x } p}$ matrix $\hat{D} = \begin{bmatrix} D \\ A \end{bmatrix}$, where A is chosen so that $rank(\hat{D}) = p$, finding rows that are orthogonal to those of D.

This way, using a variable change we get $\theta = (\theta_1, \theta_2)^T = \hat{D}\beta$, the original problem is almost transformed into a standard Lasso, being the only difference between this problem and the Lasso that now the penalization term only covers the first part of the coefficients vector, (θ_1):

$$\min_{\theta_1 \in \mathbb{R}^m} \frac{1}{2}||(I - P)y - (I - P)X_1\theta_1||_2^2 + \lambda||\theta_1||_1.$$

where $P = X_2^T(X_2^T X_2)^{-1}X_2^T$, that is, the projection over the X_2 column space.

So we solve the problem formed by the θ_1 part of the problem, plus the corresponding part of the θ_2 part that affects the problem in the coefficients of θ_1. It is easy to check that this works properly since

$$-X_2\theta_2 = -P(y - X_1\theta_1)$$

By using any of the algorithms for the Lasso that we have seen, like ISTA or FISTA, we are able to find the solution of the problem. This can be done for the complete solution path, or just for some specific values of λ. Once we have the solutions of the Lasso, we will be able to obtain the solutions of the generalized problem (2.4.3) as $\hat{\beta} = \tilde{D}^{-1}\hat{\theta}$.

As it has been stated, in order to apply this method, it is necessary that $rank(D) = m$. If this condition does not hold, it is necessary to find alternative methods to solve the problem (2.4.3). We recall that a full discussion on the solution of the Generalized Lasso can be found at [6].

Chapter 3

Trend filtering

In this chapter, a number of methods for estimating the underlying trend of a time series are introduced. For a given time series or sequence of data points y_t, it is reasonable to believe that it could be split into two different components, one of them corresponding to the trend of the series and another one representing the noise of the sample. We will call x_t the trend part of the series and z_t the stochastic part of it or the noise of the sample. So z_t can be interpreted as the time series that is the result of substracting the trend to the original series $z_t = y_t - x_t$, since the original series is the addition $y_t = x_t + z_t$. We will consider in this chapter the Trend Filtering problem, that is, to estimate the value of x_t, (obtaining at the same time an estimation of z_t).

In section 3.1 we take a look at one of the most extensively used methods, the Hodrick–Prescott (H–P) filter or L_2 Trend Filtering. We draw our attention towards L_1 TF in section 3.2. The Lagrange theory, which will be used to prove some useful properties and also for the resolution of TF. At section 3.4 a couple of efficient algorithms to solve TF are introduced. Finally a couple of extensions on the Trend Filtering problem, Polishing and Level Shifts, are discussed in 3.5 together with an idea for calculating the Polishing extension as a Standard Lasso.

3.1 H-P Trend Filtering

The nowadays called H-P filter was first proposed by Hodrick and Prescott [8]. This Trend Filtering (TF) method is probably the one that has been used the most so far, being especially relevant in the study of processes related to economy [9], partially due to its simplicity and robustness. To calculate the trend x_t using this method, an optimization problem consisting in the minimization of the following sum is to be solved:

$$\frac{1}{2}\sum_{t=1}^{n}(y_t - x_t)^2 + \lambda \sum_{t=2}^{n-1}(x_{t-1} - 2x_t + x_{t+1})^2 \tag{3.1.1}$$

The left–hand size measures the square error of the fitting, while minimizing the second one contributes to the linearity of the trend x_t, since $\lambda \geq 0$ is a regularization parameter that penalizes the non-smoothness of x_t. The reason for this is that $x_{t-1} - 2x_t + x_{t+1}$, that is the second difference of the time series, would be equal to 0 if and only if the 3 points $\{x_{t-1}, x_t, x_{t+1}\}$ are in a straight line. This way, the bigger λ is, the more likely the solution will be a straight line. We will refer to this kind of regularization from now on to be of order $k = 1$, or also 2^{nd} order, since it uses the second discrete differences.

The previous problem can be written in a more compact way by using the L_2 norm

$$\frac{1}{2}||y-x||_2^2 + \lambda||Dx||_2^2$$

where D corresponds to the $(n-k-1)\times n$ matrix at (3.1.2). We recall that k is the order of the difference matrix that is being used.

$$D = \begin{pmatrix} 1 & -2 & 1 & 0 & & & \\ 0 & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \end{pmatrix} \tag{3.1.2}$$

Using matricial notation it is immediate to obtain the solution to this problem: we just need to compute the gradient with respect to x and equal the derivative to 0 to obtain that the H-P estimation is

$$x = (I + 2\lambda D^T D)^{-1} y$$

Therefore, the solution to this problem is analytic and doesn't require any type of iterative algorithm for its resolution, being this one of its main advantages. Some of the main properties of H–P filtering are introduced next:

- **Computational complexity:** Being D a band, tridiagonal matrix, the inverse that needs to be computed can be solved in only $O(n)$ operations.

- **Convergence to original data when $\lambda \to 0$:** First we check that the spectral norm of D is upper bounded by 4. If we consider $x_{i:j}$ to be the vector beginning with the i-th element of x and ending with the j–th one, we have

$$||Dx||_2 \leq ||x_{1:n-2}||_2 + 2||x_{2:n-1}||_2 + ||x_{3:n}||_2 \leq 4||x||_2$$

  Therefore, the eigenvalues of $D^T D$ will be between 0 and 16, and those of $2\lambda D^T D(I + 2\lambda D^T D)^{-1}$, between 0 and $\frac{32\lambda}{1+32\lambda}$. This implies

$$||y-x||_2 \leq \frac{32\lambda}{1+32\lambda}||y||_2$$

  And now the property is trivial since for $\lambda \to 0$, the error vanishes $||y-x||_2 \to 0$

- **Convergence to best affine fit when $\lambda \to \infty$:** Loosely speaking, when the penalty imposed for those points that are not in a straight lines with their neighbours is high enough, the result trend will be a straight line that matches that of the linear regression, that is, a line with parameters

$$\alpha = \frac{\sum_{t=1}^n t^2 \sum_{t=1}^n y_t - \sum_{t=1}^n t \sum_{t=1}^n t y_t}{n\sum_{t=1}^n t^2 - (\sum_{t=1}^n t)^2}$$
$$\beta = \frac{n\sum_{t=1}^n t y_t - \sum_{t=1}^n t \sum_{t=1}^n y_t}{n\sum_{t=1}^n t^2 - (\sum_{t=1}^n t)^2}$$

3.2 L_1 Trend Filtering

In this section we take a look at L_1 Trend Filtering (L_1TF). This type of TF consists on the resolution of a problem similar to H-P TF, with the difference that now the norm to be used is the L_1 one instead of the L_2 norm that was used before.

First, we are going to consider the L_1 Trend Filtering problem (or TF L_1) using the second differences like we did in the previous section (that is, 2^{nd} order or of order $k = 1$), and after that we will widen this point of view for the case of TF with order $k \neq 1$.

3.2.1 2^{nd} order Trend Filtering

First we consider the case with parameter $k = 1$, that is, the 2^{nd} order TF. In this case, the function that is to be minimized is analogous to (3.1.1), with the only difference that the L_2 norm will now be changed to L_1:

$$\frac{1}{2}\sum_{i=1}^{n}(y_t - x_t)^2 + \lambda\sum_{i=2}^{n-1}|x_{t-1} - 2x_t + x_{t+1}|, \tag{3.2.1}$$

or, using matricial notation with a D matrix, like in (3.1.2),

$$\frac{1}{2}||y - x||_2^2 + \lambda||Dx||_1, \tag{3.2.2}$$

The most important reason why the L_1 norm could be preferable over the L_2 norm is that the trend estimation given by the resolution of the L_1 problem is piecewise linear in t. To understand the meaning of this more precisely we are going to define a concept that will be extensively used from now on.

Definition 12. Kink point*: Given a time series for which the solution for (3.2.2) has been found, that is, we know x_t, the kink points are those input points $1 = t_1 < t_2 < ... < t_p = n$ for which there is a change in the slope of x_t.*

This way, x_t is an affine function of t for every interval $[t_i, t_{i+1}]$. Besides, if we call α_i and β_i the parameters that define the lines that are the solution calculated for each interval, it can also be proved that those parameters are, as expected, consistent in the kink points, that is,

$$\alpha_k + \beta_k t_{k+1} = \alpha_{k+1} + \beta_{k+1} t_{k+1},\ k = 1, ..., p-1$$

Being the kink points those points where there is a change in the slope of the calculated trend, it is natural to think that they correspond with important changes in the underlying trend of the time series, hence the reason why kink points can give us a precise idea of the time where the most relevant changes in the time series happen. This is specially true when we have a reason to believe that such a trend would be linear or approximately linear.

At picture 3.2.1 we show the trend obtained by a 2^{nd} order Trend Filtering L_1 from a sinusoidal signal $f(x) = \sin\frac{x}{5}$, where the value of λ is $\lambda = 0.425$.

Some basic properties of L_1 TF are mentioned next:

- **Computational complexity:** Although there is no analytic formula for solving this problem, its complexity can be shown to be linear, as we will see at section 3.4.

- **Nonlinearity:** The trend estimate is not necessarily a linear function of the data y

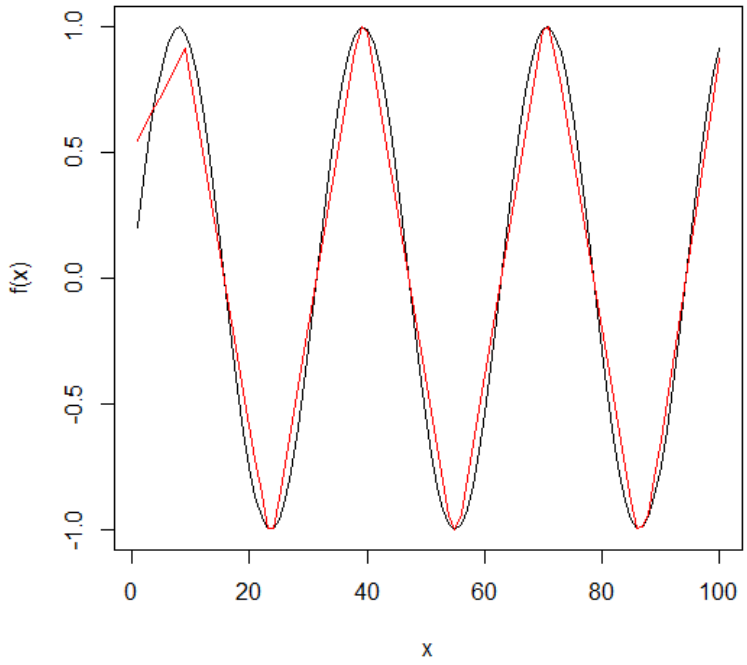


Figure 3.2.1: A sinusoidal time series coloured in black is estimated by the red piecewise linear trend obtained using 2^{nd} order Trend Filtering L_1 with penalty $\lambda = 0.425$.

- **Convergence to original data as $\lambda \to 0$:** This property is immediate if we consider that the error satisfies
$$||y - \hat{x}||_\infty \leq 4\lambda$$
We prove this relation at (3.3.8).

- **Finite convergence to best affine fit when $\lambda \to \infty$:** Not only we get the best affine fit when λ grows, but also we get this convergence at a finite value. The convergence of L_1 TF is obtained at a fixed value of $\lambda = \lambda_{\max}$
$$\lambda_{\max} = ||(DD^T)^{-1}Dy||_\infty$$
This means that, for $\lambda \geq \lambda_{\max}$ the solution will always be the best affine one. The proof for this is given later, at (3.3.9).

- **Piecewise-linearity:** The obtained trend is, for any value of λ, piecewise-linear, in the intervals defined between the kink points of the trend.

- **Linear extension property:** Given a L_1 TF fit $(x_1, ...x_n)$, an interval $[l, u]$ can be found for which, if a new point in the series is observed, satisfying $y_{n+1} \in [u, l]$, the L_1 trend including y_{n+1} will be $x_1, ..., x_n, 2x_n - x_{n-1}$. It is interesting to note that the part of the penalization from the D matrix due to this extension is zero. This property means, intuitively, that if the new observation is close enough to y_n, the trend calculated for the series including it will just extend the last segment linearly.

In the next section of this chapter we are going to explore some of the many methods that are available for the resolution of the Trend Filtering L_1 problem, and explain the advantages and weaknesses of them, to finally draw our attention to some of the natural extensions of this problem. But before that, some of the consequences of using a D matrix different from the one corresponding to second differences are going to be analysed at the next section.

3.2.2 Higher Order Trend Filtering

At this section we change slightly our problem. Our new approach is motivated by the thought that using a different D matrix than (3.1.2), that only encouraged the solution to be locally linear, can also be interesting. A number of analogous matrices can be defined, depending on the order of the discrete differences that are penalized.

We begin by defining the matrix related to order $k = 0$ TF, which corresponds to matrix (2.4.4), which penalizes the consecutive points of the series for not being similar. This matrix that we call $D^{(1)}$ is the same as the extra penalization of the fused lasso penalty with respect to the standard Lasso [7], [13].

$$D^{(1)} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & & & & & \\ 0 & & & & -1 & 1 \end{bmatrix} \tag{3.2.3}$$

For $k \geq 1$ the difference matrices $D^{(k+1)} \in \mathbb{R}^{(n-k) \text{ x } n}$ are defined recursively as in the following formula if we consider $D^{(1)}$ to be the $(n-k-1)$ x $(n-k)$ version of (3.2.3).

$$D^{(k+1)} = D^{(1)} D^{(k)}$$

One can easily check that such a matrix with $k = 1$, $D^{(2)}$, is just the same matrix that was defined in the previous section, proving that this matrix is, as was stated before, the second difference matrix.

Using this new definition of the penalization matrix D, we have a new family of problems, analogous to (3.2.2), depending on the order of the matrix. These kind of problems would be defined as

$$\min_{\beta \in \mathbb{R}} \frac{1}{2} ||y - \beta||_2^2 + \lambda ||D^{(k+1)} \beta||_1, \tag{3.2.4}$$

It is easy to check that the solution of this problem is unique, since the target function on (3.2.4) is formed by the addition of two strictly convex functions and therefore it is strictly convex.

In order to solve this problem, we recall from the previous chapter that it is possible to write it like a generalized Lasso (2.4.3) in which the predictor matrix is the identity, $X = I$ and it is only necessary to choose the order k of the penalization matrix to obtain the D matrix of (2.4.3), that would correspond to $D^{(k+1)}$ in our current discussion.

Since it is possible to transform the problem (3.2.4) into a generalized Lasso, it is possible to apply some interesting findings from [6] about its properties. For instance, we can obtain the degrees of freedom of the problem (3.2.4) with the following formula, obtained from Theorem 2 at [6]:

$$df(\hat{\beta}) = \mathbb{E}[\#\text{ knots in } \hat{\beta}] + k + 1, \tag{3.2.5}$$

Here the expectation has been taken over y and # knots means the number of non-zero entries in $D^{(k+1)}\hat{\beta}$, since the entries equal to zero will be those matching the condition on $D^{(k+1)}$. That is, for $k = 1$, an entry will be zero if and only if the 3 points that are considered are in a straight line, that is, none of them is a kink point. For $k = 2$, this means that for every 4 consecutive points, they are placed in such a place that there exists some quadratic function that fits perfectly them all. Higher values of k correspond to analogous versions of this schema.

3.3 Lagrange theory

In this section, Lagrange theory for optimization problems is introduced. This formulation gives us a way to proceed when we face optimization problems that have some constraints in their formulation, not only defined by equalities but also by inequalities. As we will see later at section 3.4 this theory will prove to be very useful for solving the TF problem.

The general formulation of these problems can be summarized as follows: For functions f, $g_i, i = 1, ..., k$ and $h_i, i = 1, ..., m$, defined on $\Omega \in \mathbb{R}^n$ the following optimization problem is to be solved:

$$\begin{array}{lll} \min & f(w) & w \in \Omega, \\ \text{subject to} & g_i(w) \leq 0 & i = 1, ..., k, \\ & h_i(w) = 0 & i = 1, ..., m. \end{array} \tag{3.3.1}$$

The inequality-based constraints are said to be active if we have for the solution w^* that $\forall i$, $g_i(w^*) = 0$, or inactive otherwise. It is important to note that even for inactive constraints, they can be transformed into active constraints by introducing what we call slack variables, that is, variables that, when summed to the inactive constraints, are actually active. For instance, if $g_i(w^*) \leq \xi$, for some ξ we could say that $g(w^*) - \xi = 0$ and is therefore active. In this situation, all the theory that is presented next would apply with the only change of the introduction of this new variable.

The well-known concept of Lagrange multipliers, see chapter 5 of [14] for further reference, gives us a method to solve optimization problems subject to equalities restrictions. We will discuss this method as a previous step to the Kuhn-Tucker theory, that includes inequality restrictions in its formulation. First we define the concept of Lagrangian function, key to the Lagrangian theory we are discussing at this chapter.

Definition 13. ***Lagrangian function.*** *Given a target differentiable function $f(w)$ with equality constraints defined as before ($h_i(w) = 0$) the Lagrangian function associated to the problem is*

$$L(w, \beta) = f(w) + \sum_{i=1}^{m} \beta_i h_i(w).$$

where the β_i are called the Lagrange multipliers of the problem.

It is important to note that in a problem with constraints, it is possible to have a local minimum w^* for which the condition $\frac{\partial f(w^*)}{\partial w} \neq 0$, but in this case we could try to move our solution towards the directions that would allow us to reduce the value of $f(w)$ even more. In order to achieve this, respecting the constraints h, we could move in a perpendicular direction to $\frac{\partial h_i(w^*)}{\partial w}$, for each of these constraints. If this is not possible without breaking the constraints of the problem, since such direction at the point where we are at is perpendicular to the subspace defined by the constraints we have reached the solution. This situation is presented more precisely at Lagrange Theorem.

Theorem 3. ***Lagrange.*** *If $L(w,\beta)$ is the Lagrangian defined at Definition 13, the necessary conditions for w^* to be a global minimum of $f(w)$, sufficient if $L(w,\beta)$ is convex, are:*

$$\frac{\partial L(w^*,\beta^*)}{\partial w} = 0, \tag{3.3.2}$$

$$\frac{\partial L(w^*,\beta^*)}{\partial \beta} = 0. \tag{3.3.3}$$

Now we consider the more general problem when we do not only have equality constraints, but also inequality constraints, $g_i(w)$. The Lagrangian associated to the problem will be

$$L(w,\alpha,\beta)_{\alpha_i\geq 0} = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{m} \beta_i h_i(w) = f(w) + \alpha g(w) + \beta h(w). \tag{3.3.4}$$

We will work from now on with function θ, defined as

$$\theta(\alpha,\beta)_{\alpha\geq 0} = \inf_{w} L(w,\alpha,\beta).$$

The primal problem is, most of the times, difficult to solve. In order to find the solution, we are going to use the dual form of the primal problem, for which the constraints are no longer a problem.

Definition 14. ***Lagrangian Dual Problem.*** *For every Primal $L(w,\alpha,\beta)$ problem* (3.3.4)*, a dual problem can be defined as follows*

$$\max \theta(\alpha,\beta), \text{ for } \alpha \geq 0.$$

Theorem 4. *For a given $w \in \Omega$ being a feasible point w of the primal problem, and (α,β) one of the dual one, the following inequality holds $f(w) \geq \theta(\alpha,\beta)$.*

Proof: Taking into account the already known relations between the primal and dual problems we have

$$\theta(\alpha,\beta) = \inf_{u\in\Omega} L(u,\alpha,\beta) \leq L(w,\alpha,\beta) = f(w) + \alpha g(w) + \beta h(w) \leq f(w). \square$$

Here we have used that, for any feasible w (matching the constraints) $h(w) = 0$ and $g(w) \leq 0$.

As a direct corollary of this theorem, it is easy to see that, since the solution of the dual problem is upper bounded by that of the primal, the following relation holds

$$\sup_{\alpha\geq 0} \theta(\alpha,\beta) \leq \inf_{g(w)\leq 0, h(w)=0} f(w)$$

Considering Theorem 4, it is evident that, the closer we are to the solution of both problems, the smaller the difference between them will be. If such difference, called Dual Gap, is zero, we can be sure that we have found the optimal solution. However, there is nothing that ensures that both solutions are actually the same. Anyway, for the kind of problems we are studying, that is, convex ones, this is actually true, due to next theorem.

Theorem 5. ***Strong duality theorem.*** *For a given optimization problem in a given convex domain $\Omega \subseteq \mathbb{R}^n$:*

$$\begin{array}{lll} \min & f(w) & w \in \Omega, \\ \textit{subject to} & g_i(w) \leq 0 & i = 1, ..., k, \\ & h_i(w) = 0 & i = 1, ..., m. \end{array}$$

where all the functions g_i and h_i are affine and f is convex, the difference between the solutions of the primal and dual problems is 0.

This last result suggests that, for solving a convex problem, there is no problem at all in considering the dual, since its solution will coincide with that of the primal. This leads us to consider the Karush-Kuhn-Tucker conditions, which give the necessary and sufficient conditions for a primal-dual problem to have an optimum, where w^* is the optimum of the primal, and α^*, β^*, the optimum values of the dual variables. If $f \in C^1$ is convex and g_i, h_i affine, these conditions are:

$$\begin{array}{l} \nabla_w L(w^*, \alpha^*, \beta^*) = 0, \\ \nabla_\beta L(w^*, \alpha^*, \beta^*) = 0, \\ \alpha_i^* g_i(w^*) = 0, i = 1, ..., k, \\ g_i(w^*) \leq 0, i = 1, ..., k, \\ \alpha_i^* \geq 0, i = 1, ..., k. \end{array} \tag{3.3.5}$$

Most of the times the constrains of the primal are more complicated than those of the dual, hence the reason why we would prefer to solve the dual version of the problem.

Before giving an example of this situation, we need to define a new concept:

Definition 15. ***Positive-definite matrix:*** *A $n \times n$ matrix A is said to be positive definite if, for all non-zero column vector $z \in \mathbb{R}^n$ the following condition holds*

$$z^t A z > 0$$

A good example of a typical situation follows, where the primal-dual method is applied to the case of a quadratic target function, where Q is a $n \times n$ definite-positive matrix and k and c vectors. Considering the method explained before, the primal problem can be rewritten in the following way

$$\begin{array}{ll} \min & \frac{1}{2} w^t Q w - k^t w \\ \text{subject to} & X w \leq c \end{array}$$

Now we write this problem in its dual version as

$$\max_{\alpha \geq 0} \left(\min_w \left(\frac{1}{2} w^T Q w - k^T w + \alpha^T (X w - c) \right) \right)$$

The inner minimum does not have any kind of constraint, and is obtained when $w = Q^{-1}(k - X^T \alpha)$. Substituting this into the original problem, we transform the original primal problem into its dual

$$\max_{\alpha > 0} \quad -\tfrac{1}{2} \alpha^T X Q^{-1} X^T \alpha - \alpha^T (c - X Q^{-1} k) - \tfrac{1}{2} k^T Q k,$$

We see the dual version of a quadratic problem is also quadratic, but the difference is that the constraints cause no more trouble, making the task of solving it much easier.

Now the theory we have seen is applied to the L_1 TF problem. Since it is not differentiable, it is necessary to use the subdifferential calculus theory introduced in the previous chapter, in order to handle properly those points where there is not a standard derivative.

Recalling Theorem 2, Fermat, we calculate the subdifferential and equal to zero to see that the necessary and sufficient condition for x to minimize (3.2.2) is that $\exists \nu \in \mathbb{R}^n$ such that (3.3.6) holds. At this point we are using the rule chain adapted to subdifferential calculus, that is, if f is a convex function, the subdifferential of $h(x) = f(Ax + b)$ is

$$\partial h(x) = A^T \partial f(Ax + b).$$

$$y - x = D^T \nu, \nu_t \in \begin{cases} \lambda, & (Dx)_t > 0, \\ -\lambda, & (Dx)_t < 0, \\ [-\lambda, \lambda], & (Dx)_t = 0, \end{cases} \quad t = 1, ..., n-2 \tag{3.3.6}$$

It is possible to eliminate ν from (3.3.6) if we take into account that DD^T is invertible and therefore (3.3.6) can be written, multiplying by $(DD^T)^{-1}$, as

$$((DD^T)^{-1}D(y - x))_t \in \begin{cases} \{\lambda\}, & (Dx)_t > 0, \\ \{-\lambda\}, & (Dx)_t < 0, \\ [-\lambda, \lambda], & (Dx)_t = 0, \end{cases} \quad t = 1, ..., n-2. \tag{3.3.7}$$

These equations allow us to prove some of the properties about L_1 TF that were already discussed at 3.2. It is possible to obtain bounds for the maximum fitting error. We can see that the maximum error of (3.2.2) would be

$$||y - x||_\infty \leq 4\lambda, \tag{3.3.8}$$

This result follows from the optimality condition (3.3.6). In fact, if $\nu \in \mathbb{R}^{n-2}$ with $\nu_t \in [-\lambda, \lambda]$, we have

$$-4\lambda \leq (D^T \nu)_t \leq 4\lambda, \quad t = 1, ..., n.$$

Moreover, the central part of the equation above can be transformed into $x_t - y_t$ and therefore we have our bound

$$-4\lambda \leq x_t - y_t \leq 4\lambda, \quad t = 1, ..., n.$$

Note that this implies that, for $\lambda \to 0$, the error is 0. However, this is a situation we must avoid, since the calculated trend would be the original time series.

Now we able to obtain the expression for $\lambda_{\max}$, or the maximum value of λ for which x is affine. For x affine, we have $Dx = 0$, and therefore we must have that $((DD^T)^{-1}D(y-x)) \in [-\lambda, \lambda]$ or

$$||(DD^T)^{-1}D(y - x)||_\infty = ||(DD^T)^{-1}Dy||_\infty \leq \lambda \tag{3.3.9}$$

3.4 ADMM algorithms

In this section a type of really effective algorithms that allow us to find the Trend Filtering solution are introduced, the Augmented Lagrangian Methods [15] and specifically one of them, the Alternating Direction Method of Multipliers (ADMM). First we introduce the concept of Augmented Lagrangian, which is needed to perform the ADMM Algorithm. Then the ADMM is introduced and finally a specialized, more efficient version for TF.

The general version of the problem we want to optimize can be written the following way. Here we want to find the optimum value of x, subject to a number of constraints represented by the equation $Ax = b$. Therefore we solve

$$\begin{array}{ll} \min & f(x) \\ \text{subject to} & Ax = b \end{array} \tag{3.4.1}$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $f : \mathbb{R}^n \to \mathbb{R}$ convex.

We consider the Augmented Lagrangian approach which consists on solving, instead of the original primal problem, a slightly different one, formed by the sum of the original primal plus a new term. In order to do this, first we need to define a new concept.

Definition 16. ***Augmented Lagrangian:*** *The Augmented Lagrangian for the problem* (3.4.1) *is*

$$L_\rho(x, y) = f(x) + y^T(Ax - b) + \frac{\rho}{2}||Ax - b||_2^2 \tag{3.4.2}$$

Here $\rho > 0$ can be seen as a penalty parameter, for which $\rho = 0$ gives the standard Lagrangian, but ρ will also be, as will be clear soon, the step size for the dual update.

It is interesting to see that the augmented Lagrangian would be the standard Lagrangian of the problem

$$\begin{array}{ll} \min & f(x) + \frac{\rho}{2}||Ax - b||_2^2 \\ \text{subject to} & Ax = b \end{array} \tag{3.4.3}$$

for which its solution is the same than that of (3.4.1), since for all feasible x, the term added to $f(x)$ will be zero.

A general method to solve this problem, the Method of Multipliers is presented now. It has the advantage that it is not necessary that f is strictly convex to ensure the convergence of the method and can also be used in cases when f takes the value ∞. [11]

This method works by minimizing over x and then updating the dual variable in an iterative way, in a similar way than gradient descent does, but on the dual variable. First we optimize the primal

$$x^{k+1} = \underset{x}{\operatorname{argmin}}\, L_\rho(x, y^k)$$

The dual variable is then updated

$$y^{k+1} = y^k + \rho(Ax^{k+1} - b)$$

Now we consider the optimality conditions of problem (3.4.2), proceeding as shown in Theorem 3. These conditions are, for an optimal x^*:

$$\begin{gathered} Ax^* - b = 0, \\ \nabla f(x^*) + A^T y^* = 0. \end{gathered}$$

To check that this procedure works properly and that it converges we see that, considering that x_{k+1} minimizes $L_\rho(x, y_k)$, the following holds

$$\begin{aligned} 0 &= \nabla_x L_\rho(x^{k+1}, y^k) \\ &= \nabla_x f(x^{k+1}) + A^T(y^k + \rho(Ax^{k+1} - b)) \\ &= \nabla_x f(x^{k+1}) + A^T y^{k+1} \end{aligned}$$

Comparing this result with the optimality conditions, it is clear that if both x^k and y^k converge, the do so to the solution of the problem. For a deeper discussion on the convergence of this method, check section 2.3 at [11].

Now the Alternating Direction Method of Multipliers, ADMM algorithm is described. This algorithm solves problems like the one below, where the target function has been split into two different functions, f and g

$$\begin{array}{ll} \min & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

with $x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$, $A \in \mathbb{R}^{p \text{ x } n}$, $B \in \mathbb{R}^{p \text{ x } m}$ and $c \in \mathbb{R}^p$. The functions f and g are also assumed to be convex.

The first step is to construct the augmented Lagrangian as we did at (3.4.2), obtaining

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}||Ax + Bz - c||_2^2.$$

The ADMM algorithm is defined by the following updates, where the step $\rho > 0$:

$$\begin{aligned} x_{k+1} &= \text{argmin}_x \, L_\rho(x, z_k, y_k) \\ z_{k+1} &= \text{argmin}_z \, L_\rho(x_{k+1}, z, y_k) \\ y_{k+1} &= y_k + \rho(Ax_{k+1} + Bz_{k+1} - c), \end{aligned}$$

If we compare this approach with the method of multipliers we have just discussed, we see the reason for the name of alternating directions for the method. In ADMM the x and z updates are performed sequentially, instead of calculating both x_{k+1} and z_{k+1} at the same time as the method of multipliers would do if applied to this problem. This two-step minimization can be done because we have been able to separate the target function in f and g.

Now we apply the ADMM Algorithm to solve the k-th order TF problem.

$$\min_{\beta \in \mathbb{R}^n, \alpha \in \mathbb{R}^{n-k-1}} \frac{1}{2}||y - \beta||_2^2 + \lambda||\alpha||_1 \text{ with } \alpha = D^{(k+1)}\beta. \tag{3.4.4}$$

The (Augmented) Lagrangian associated to this problem can be written as:

$$L(\beta, \alpha, u) = \frac{1}{2}||y - \beta||_2^2 + \lambda||\alpha||_1 + \frac{\rho}{2}||\alpha - D^{(k+1)}\beta + u||_2^2 - \frac{\rho}{2}||u||_2^2$$

where a new term u has been introduced when we transform it into its Augmented version. It is easy to check that this expression is equivalent to the actual Augmented Lagrangian seeing that

$$||\alpha - D^{(k+1)}\beta + u||_2^2 = ||\alpha - D^{(k+1)}\beta||_2^2 + u(\alpha - D^{(k+1)}\beta) + ||u||_2^2$$

Taking derivatives with respect to each of the variables and using the proximal operator when needed, we obtain the updates of the parameters that we need to solve this problem, that are:

$$\beta \leftarrow (I + \rho(D^{(k+1)})^T D^{(k+1)})^{-1}(y + \rho(D^{(k+1)})^T(\alpha + u)), \tag{3.4.5}$$

$$\alpha \leftarrow S_{\lambda/\rho}(D^{(k+1)}\beta - u), \tag{3.4.6}$$

$$u \leftarrow u + \alpha - D^{(k+1)}\beta, \tag{3.4.7}$$

where $S_{\lambda/\rho}$ here means to perform coordinate-wise the Soft-Thresholding operator (2.3.4) at level λ/ρ. It is important to note that the update on α does not actually optimize α on a single step. Several α updates could be performed to try to obtain better results, but we are going to see that we can do better than this slightly modifying the way we proceed.

At [16], an improvement for this algorithm is described. This method consists on rewritting (3.4.4) in a different way:

$$\underset{\beta\in\mathbb{R}^n,\alpha\in\mathbb{R}^{n-k}}{\operatorname{argmin}} \frac{1}{2}||y-\beta||_2^2 + \lambda||D^{(1)}\alpha||_1 \text{ with } \alpha = D^{(k)}\beta, \tag{3.4.8}$$

Now the Lagrangian is

$$L(\beta,\alpha,u) = \frac{1}{2}||y-\beta||_2^2 + \lambda||D^{(1)}\alpha||_1 + \frac{\rho}{2}||\alpha - D^{(k)}\beta + u||_2^2 - \frac{\rho}{2}||u||_2^2, \tag{3.4.9}$$

and therefore the new algorithm have these updates:

$$\beta \leftarrow (I + \rho(D^{(k)})^T D^{(k)})^{-1}(y + \rho(D^{(k)})^T(\alpha + u)), \tag{3.4.10}$$

$$\alpha \leftarrow \underset{\alpha\in\mathbb{R}^{n-k}}{\operatorname{argmin}} \frac{1}{2}||D^{(k)}\beta - u - \alpha||_2^2 + \frac{\lambda}{\rho}||D^{(1)}\alpha||_1 \tag{3.4.11}$$

$$u \leftarrow u + \alpha - D^{(k)}\beta. \tag{3.4.12}$$

It is important to take into account that, although the α update is a new minimization problem in itself, it is indeed the 1d fused lasso, for which several extremely efficient resolution methods exist, like the ones developed in [17] and [10].

Regarding the performance of the methods, it is important to note that, although the theoretical complexity of both the standard ADMM and its specialized version is linear, the empirical performance of them is not the same at all, being the specialized version clearly superior than the standard ADMM [16]. First, we check that the complexity of both versions of ADMM is linear per iteration. In order to do so, we note that the β and u and the same in both algorithms, expect for the order k. The complexity of the β update is $O(n(k+2)^2)$, since the associated system that is to be solved has a banded matrix structure with width $k+2$, while the update for u has also a linear complexity, exactly $O(n(k+2))$, as it can be reduced to a banded matrices multiplication in both algorithms. Regarding the α update, in the original algorithm we need to perform a soft-thresholding operation in every iteration, therefore requiring $O(n-k-1)$ operations. However, for the specialized version (3.4.8), if we use some efficient dynamic programming method like the one at [10], only $O(n-k)$ operations are needed at this step. Besides, since at the specialized version we are actually optimizing the α update it is to be expected, and is what actually happens [16], that the specialized version works faster, since it requires less iterations to achieve convergence.

3.5 Extensions on the Trend Filtering problem

In this section, we are going to consider some natural extensions of the Trend Filtering problem that has been discussed all over this chapter. These extensions try to improve in a way or another the results that can be obtained by using only the typical Trend Filtering, while keeping its essence.

3.5.1 Polishing

When a L_1 TF problem is solved, we have to handle the two different terms that are added to obtain the total penalization: on the one hand, the square loss and, on the other, the penalization related to the D matrix. Therefore, although the found solution minimize the sum of both penalties, this doesn't mean that the result obtained by TF is the one that minimizes the square loss while respecting the constraints of the problem.

Taking this into account, it is a natural question to ask whether the solution would be better if we split the problem into two steps. First TF is applied to learn which are the kink points of the series. After that, the square loss is minimized, under the constraints defined by the kink points, that is, the solution must be piecewise-linear in the intervals defined by them. The solution will be similar than that of the original TF problem, the points where the trend changes will be the same, but the square loss will be better than that of the Standard TF. Formally, if the kink points are $\{T_1, ..., T_p\}$ this new problem is:

$$\begin{array}{cc} \min & \sum_{k=1}^{p-1} \sum_{T_k \leq t \leq T_{k+1}} ||y - \alpha_k - \beta_k t||_2^2 \\ \text{subject to} & \alpha_k + \beta_k T_{k+1} = \alpha_{k+1} + \beta_{k+1} T_{k+1},\ k = 1, ..., p. \end{array} \tag{3.5.1}$$

It is interesting to note that it is possible to solve this problem by encoding the new restrictions of the solution in a generalized Lasso problem, one that can be translated into a Standard Lasso. Since every line of the solution for each interval is defined by the α_i, β_i of (3.5.1), we can write the coefficient vector as

$$\hat{\beta} = (\alpha_1, \beta_1, ..., \alpha_{p-1}, \beta_{p-1}).$$

Now we only need to define the matrix D so that it captures the restrictions on (3.5.1) and the X matrix to indicate, for every value in y_t, which is the pair of (α_i, β_i) that is active for its interval. More precisely, each row of X will have, at time t, values $(1, t)$ at the columns corresponding with the (α_i, β_i) of the interval where falls t and zeroes in the columns corresponding to the rest of the coefficients of the intervals. This way, each row of X will be transformed into a x_t of our trend after the multiplication with the coefficients column vector is performed since $(1, t)(\alpha_i, \beta_i)^\intercal = \alpha_i + \beta_i t = x_t$

We denote by $T_1, ..., T_{p-1}$ the kink points detected in the resolution of the Trend Filtering problem. Regarding the inputs we consider $t_{i,j}$ to be the time when the original inputs are given, being the first sub-index the number of the interval where the time falls, and the second one the order of the input that we are considering into the interval where it is located, the D matrix that encodes the constraints (3.5.1) is

$$D = \begin{pmatrix} 1 & T_2 & -1 & -T_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T_3 & -1 & -T_3 & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & T_{p-1} & -1 & -T_{p-1} \end{pmatrix} \tag{3.5.2}$$

All that remains is to define, for every t_i, which coefficients will be the ones that would be used for every input (that is, those that correspond to its interval). In each row the valid

coefficients are marked. The α_i will be valid and set to 1, the coefficient for β_i would be, for the i-th point, t_i. The rest of the entries will be set to 0. This way the matrix X can be written as

$$X = \begin{pmatrix} 1 & t_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & & \\ 1 & T_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & t_{2,1} & 0 & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & 1 & t_{2,2} & 0 & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & t_{p-1,m-1} \\ \vdots & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & T_n = t_{p-1,m} \end{pmatrix} \tag{3.5.3}$$

This way, if we choose the solution corresponding to the highest λ of the solution path, $\lambda_{\max}$ we are strongly penalizing that the solution does not match the condition in (3.5.1) and our generalized Lasso formulation will also solve the Lagrangian of (3.5.1).

Putting all these pieces together, all that remains is to solve the following generalized Lasso

$$\underset{\beta}{\operatorname{argmin}} (y - X\beta)^2 + \lambda_{\max} ||D\beta||_1$$

It is important to note that, since $\operatorname{rank}(D) = p-1$, it is possible to transform this problem into a Standard Lasso, for which efficient algorithms can be applied. There could be another issues applying this transformation if $\operatorname{rank}(X)$ is not maximum since as we saw at section 2.4.2 we need to calculate $P = X_2^T (X_2^T X_2)^{-1} X_2^T$, where X_2 corresponds to some of the last rows of X. $(X_2^T X_2)$ can only be singular if there are two consecutive kink points but in this case we could either delete one of the kink points without losing much information. The *genlasso* package that has been used in the experiments solves this problem by adding some ϵ to the matrix so the inverse can be calculated, which is also a valid way to deal with this problem.

3.5.2 Level Shifts

It is not unusual to find some time series that have abrupt level shifts. Such level changes can be modelled as a piecewise constant component, so the 1d fused lasso penalization matrix can be used to extract the part of the trend due to these level shifts. The new minimization problem becomes

$$\min_{x,w} \frac{1}{2} ||y - x - w||_2^2 + \lambda ||Dx||_1 + \rho \sum_{t=2}^{n} |w_t - w_{t-1}|, \tag{3.5.4}$$

Now the parameter $\lambda \geq 0$ controls the smoothness of the trend x, while $\rho \geq 0$ controls the number of level shifts that will be found in w

A way to solve this minimization would be to apply the ADMM algorithm, in its variant for Mixed Trend Filtering, that is, to obtain

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} ||y - \beta||_2^2 + \sum_{i=1}^{n} \lambda_i ||D^{(k_i+1)} \beta||_1, \tag{3.5.5}$$

where $k_i \geq 0, i = 1, ..., n$ are the orders that are going to be used in the Mixed Trend Filtering.

For this particular problem, we use the specialized ADMM algorithm for equation (3.5.5) with $k_1 = 1, k_2 = 0$, obtaining the following updates, as shown at [16]:

$$\begin{gathered} \beta \leftarrow \left(I + \sum_{i=1}^{2} \rho_i (D^{(k_i)})^T D^{(k_i)}\right)^{-1} \left(y + \sum_{i=1}^{2} \rho_i (D^{(k_i)})^T (\alpha_i + u_i)\right), \\ \alpha_i \leftarrow DP_{\lambda_i/\rho_i}(D^{(k_i)}\beta - u_i), \ i = 1, 2, \\ u_i \leftarrow u_i + \alpha_i - D^{(k_i)}, \ i = 1, 2. \end{gathered} \tag{3.5.6}$$

By DP at the α update we mean the use of some dynamic programming algorithm to solve the optimization problem of computing the proximal operator at level λ_i/ρ_i.

Calculating these updates we will get a compromise between the solution given by the Polishing problem and the extra constraints from $D^{(0)}$ that helps us identify the abrupt changes in the trend and the result obtained will be more accurate when analysing time series with level shift in it.

Chapter 4

Experiments

In this chapter, the experimental part of this work is presented. The structure of this chapter is as follows. At section 4.1 the different libraries and technologies that have been used in the experimental part of the work are introduced, in order to set the framework we have worked with, when solving TF problems, or applying them to find the solution to other problems. At section 4.2 we reproduce some of the well-known experiments related to Trend Filtering that the authors that started this field of research used to show their results. Finally these techniques are applied to analyse the Energy Demand over Spain at section 4.3.

4.1 Description of the used software

The main programming languages that have been used in this work have been R and Python. Our first idea was to do all the programming in the same language, which would have been Python due to its advantages over R for the pre-processing of the data.
However, the amount of quality functions related with the TF problem written in this language was a clear disadvantage compared to the software available in R.

Therefore R was also chosen as a complementary language, mainly because of the following reasons:

- Some of the authors of the Trend Filtering theory, have developed some extremely efficient implementations of the algorithms they develop in their papers. These implementations are available only in the form of R packages. Since one of the things we want to achieve now is to reproduce some of those experiments, or perform similar ones, it is interesting to have the same software these authors have develop.

- Although implementations for solving the Lasso can also be found as part of some Python libraries, an efficient implementation for solving both the Standard and the Generalized Lasso problem, yielding the complete path of solutions, could not be found, apart from the R packages mentioned before. The complete path of solutions is needed in our experiments, since we don't have a straightforward way to know what is the best choice of λ for our problem.

- TF is a quite specific and specialized method for which there are little reliable packages, aside from the R packages from the authors of this theory, is available. This makes the utilization of these packages the best choice to check the results of the authors.

There are a number of R packages providing methods for solving problems related with TF or the Generalized Lasso Problem. The most relevant ones are the ones developed by Ryan J. Tibshirani and Taylor B. Arnold, that are listed below:

- Taylor B. Arnold and Ryan J. Tibshirani (2014). genlasso: Path algorithm for generalized lasso problems.
  R package version 1.3. https://CRAN.R-project.org/package=genlasso

- Taylor Arnold, Veeranjaneyulu Sadhanala and Ryan Tibshirani (2014). glmgen: Fast algorithms for generalized lasso problems. R package version 0.0.3.

These packages are available on CRAN under the names *glmgen* and *genlasso*.

The *glmgen* package provides us with some specialized functions able to compute the path of solutions for some of the most important generalized Lasso problems. It comes with a specific function to compute TF of any order, being this the main reason why we tried this package.

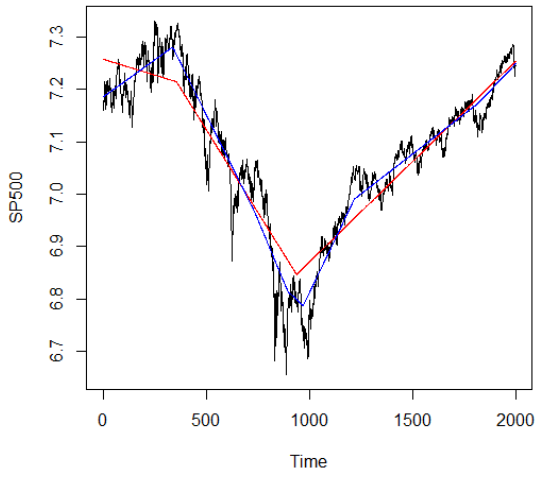
The *genlasso* package offers a specialized, very efficient function called *trendfilter* that is able to calculate the TF of a time series, giving the complete path of solutions for the different values of λ at which the TF changes. This allows us a complete set of valid values of λ for which the solution given for the algorithm changes. All that remains is to choose an appropriate λ that produces the trade-off we are looking for correctness of fit and good generalization properties. This last package has one advantage over *glmgen*, since it is able to solve easily any Generalized Lasso with it (that is, not just the most important problem like TF, but also any other Generalized Lasso). Its function *genlasso* provides us an easy way to obtain the complete path of solutions for this kind of problem, and is able to solve a Generalized Lasso for two specific X and D matrices, in which we can encode the particularities of our problem easily.

For this reason we decided to use the *genlasso* package. All the experiments in this text dealing with TF or with Generalized or Standard Lasso Problems have been processed using this software. At this point we considered using the *glmnet* package, also by Tibshirani, to deal with the Standard Lasso, but finally we decided to use only *genlasso*, since this last package is also able to solve Standard Lassos efficiently, while being capable of translating Generalized Lassos to Standard ones to solve them quickly when possible.

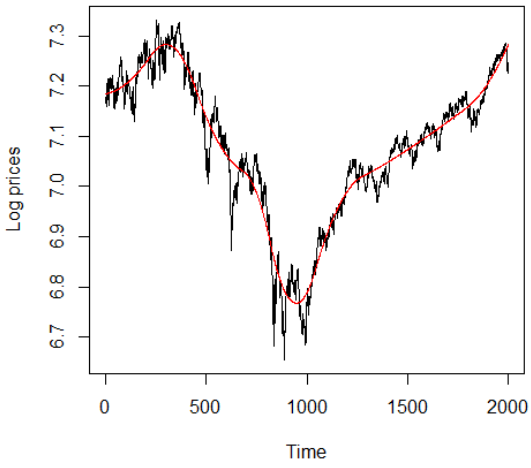
Finally, our results have also been compared with those of an AR(7) model. In order to fit this model to our data, function *arima* from the R package *stats*.

4.2 Basic experiments

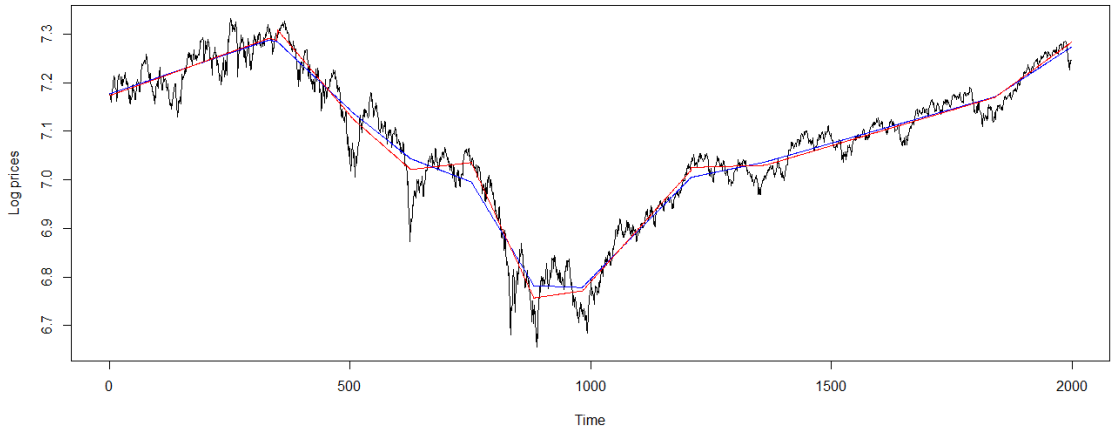
In this section we are going to perform some representative experiments about the type of problems we have been discussing regarding TF. We will use the software that has been explained at the previous section to check that we get the same results predicted by the already presented literature, and also to apply Polishing over the original TF series.



(a) Two TF solutions for the series of the S & P 500 index with $\lambda = 1776$ (red) and $\lambda = 240$ (blue)



(b) 2nd order L_1 TF using $\lambda = 3720$



(c) TF with $\lambda = 77$ (blue) and Polished TF performed over the obtained kink points (red)

Figure 4.2.1: Results of the application of TF over the S&P500 series between 25/3/1999 and 9/3/2007

First of all we apply TF over the classical S&P 500 time series between March 25, 1999, to March 9, 2007. This time series was used at [1] to show that their approach actually works for the problem of filtering the noise that is known to be present in every financial time series.

In the following figures, results from applying L_1 TF are shown. The results for two different values of the parameter λ are shown at Figure 4.2.1a. Here, the L_1 norm is used for the regularization term, therefore giving as a result two piecewise-linear solutions. The values of λ that have been used are $\lambda = 1776$ for the red line and $\lambda = 240$ for the blue one.

As we can see, for a higher λ the result is a more general, similar to a linear regression over the entire series one, while the lower λ is, the more the solution of TF will adapt to the original value of the time series, as the theory seen in the previous chapter predicted. The square error of the obtained trend also decreases with λ, falling from 5.89 to 3.01,

respectively.

These results can be softened, if the penalization matrix D is changed. For instance, instead of using the second differences matrix as we have just done, we can use the $D^{(2)}$ penalty matrix. This way, the results are visually smoother.

At Figure 4.2.1b we show an example showing the result of applying this to the same series of the S&P 500.

Now one of the variants for TF we have discussed before at section 3.5.1, Polished TF, is applied. At figure 4.2.1c we can see the result of applying this strategy. Here we have proceed as follows:

- First we consider the classical L_1 TF, the one we have discussed at the beginning of this section. After finding an appropriate choice of λ, we can obtain the kink points, which we recall that are those points where the slope of the solution changes. These points, provided that the algorithm is working properly and we have been able to find a correct value for λ, will coincide with the changes in the direction of the trend of the original time series.

- Now that we have the kink points, we can use them to perform Polished TF, just by solving (3.5.1). That is, we minimize the square error, forcing the intervals at the left and right side of each kink point to be consistent with the other side.

At Figure 4.2.1c we compare the results of both approaches. Standard L_1 TF, with $\lambda = 77$, is shown in blue, while the Polished version is in red. The two trends, visually, are equally acceptable, but the Polished's square error along the series is better regarding that of the standard TF.

4.3 Energy Demand prediction

Now we aim to use the Trend Filtering techniques we have discussed to analyse the problem of the Energy Demand over the Peninsular part of Spain. As a first step we will use TF to learn more about the structure of this data, but our main objective is to try to use TF to predict the Energy Demand for the next day, given the data of the demand of the previous days. A variation on TF, Polished TF is also applied with this purpose.

First, at section 4.3.1 we analyse the problem and give the first ideas that have been followed. Then at section 4.3.2 we take a look at the aspect of the data looking for missing values or outliers and trying to spot the peculiarities of the time series. Then At section 4.3.3 we develop the model and finally its results are discussed at section 4.3.4.

4.3.1 Introduction

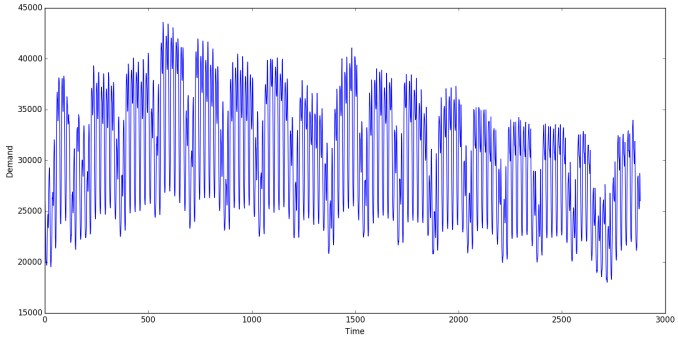
First, we take a look at the whole time series. A plot of the time series showing the energy demand between 2011 and 2013 is shown at Figure 4.3.1a

It is obvious that this series is really noisy to just apply TF and use it directly. A number of variables can modify dramatically the value of the demand. A non-exhaustive list of such variables could be

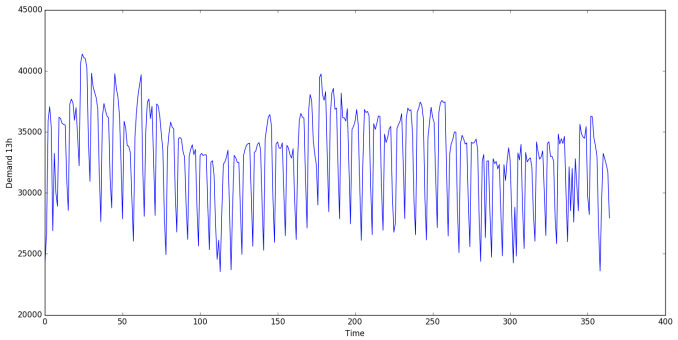
- The day of the year.

- The day of the week.

- The hour of the day.



(a) Series of the Demand for the first 3 months of 2011



(b) Series of the Demand at hour 13 on 2011

Figure 4.3.1: Original series of the energy demand over 2011. At 4.3.1a we show the demand on January, February and March 2011. At 4.3.1b, the demand over the whole year 2011, at hour 13, is shown.

- The temperature.
- How many people are working this day.
- The proximity of a holiday.

It is clear, however, that regardless of all those variables, every day will follow a, at least visually, similar pattern. At Figure 4.3.2 we can see the typical curves of the demand for each of the days of the week. These curves represent the demand of all days between 7/1/11 and 14/1/11. In all cases, the energy demand has a minimum at around 4:00 a.m. Then the demand rises until a maximum at approximately 10 a.m is reached. After that, the demand falls until 16:00, to rise again reaching the global maximum at 20:00. There is therefore, a more or less clear pattern of periodicity on the demand.

Because of this, our first idea was to create some sort of "Standard Day" and normalize it accordingly depending on a number of variables, such as the ones we have already listed and maybe some others, to try to extract the periodical part of the series that is repeated every day, eliminating in the process most of the variance that is present is the series.

However, this approach is not a good one, since there are too little "similar days". If we consider the list of variables above, and try to consider similar days in our analysis we will find that there are almost no days from the same day of the week, with a comparable number of people on holiday that are also close enough in terms of the month of the year.

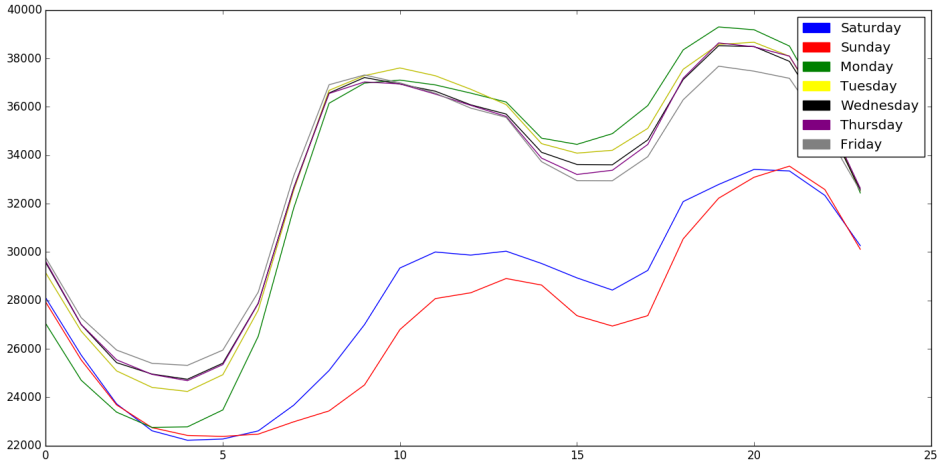


Figure 4.3.2: Demand per day on the 7/1/11 to 14/1/11 week

So eliminating the component of the series due to the periodicity of each day is not an easy task. In order to eliminate some of this noise, from now on we consider the problem not as just one time series, but as 24 time series, one per hour of the day. This is an easy and effective way to partially solve this problem, that is the one responsible for a big part of the variance in the series. Of course this does not include the periodicities we have due to each week and year, but those are much easier to consider.

To see how this simplifies the series we can check at Figure 4.3.1b the series on year 2011 for hour 13 and compare it with the original.

The resulting time series are still really noisy. Although TF is designed to filter the noise, there are still a good number of factors that strongly influence the demand that introduce a good degree of noise that are predictable. At the next section we take a look at those factors and create a normalization algorithm for the data, with the aim of applying TF to the normalized series, so we can denormalize the results later to obtain the correspondence in the original time series. Of course, such a denormalized TF results will lose some of its properties like the piecewise-linearity, but those properties will hold for the normalized time series, where we can make our analysis. The de-normalization of the results will only be done at the end of the process, to check the results we have obtained.

4.3.2 Preprocessing of the data

In this section we take a look at the data in order to check its quality, looking for missing data, or clearly incorrect entries. After some checking we realize that there are no missing data at all and there seems to be almost no outliers that could give us an indication of incorrect values. The only exception in this sense are a number of days, for which we have extremely low (zero) or high entries. A quick look at them is enough to see what is happening. At the hours when the time changes from Summer to Winter time or vice versa we have the following problem: If the hour changes from two to three, the demand at hour two will be zero, while for days with one hour less, we will have more or less double the expected demand at two. In order to delete this noise, linear interpolation has been used to fill the gaps or to reduce the extra demand, respectively.

Now that the data have been cleaned, our objective is to find a way to homogenize them, so we can minimize the oscillations in the demand due to predictable variables. We

recall that we have 24 time series, and want to find the underlying trend for each of them, but this is more difficult than should be, since the demand changes dramatically depending on, for instance, the day of the week. Our intention is to, after some normalization process, modify each of those series into another one, the one that we would expect the series to have been, provided that all the variables we are considering followed some previously defined definition of "Standard Day".

The following variables have been identified as relevant regarding the normalization process:

- Day of the week.
- Working day or holiday.

It has been decided that our Standard Day will be a full working day, that is a day when all Spain is working. This way, days not matching this criteria will be normalized to transform its actual time series into the one we would expect it to be considering our prior knowledge.

In order to decide whether some days apart from Sundays and holidays require an special treatment, we take a look at a week without any holidays in it like the one that goes from 7/1/11 to 14/1/11. Looking again at Figure 4.3.2 we can see that most of the days of the week are similar, with the clear exception of the weekend. Because of this, we will normalize these days in a different, more precise day. We considered to normalize each of the day of the week separately but this would lead to having less data to consider in order to compute the normalization, which would probably have incremented the errors.

For each of these special days (Saturday and Sunday) and each hour of the day we calculate the mean of the demand considering the first two years of our data, that is the series over 2011 and 2012. We call these $\bar{D}_{\text{Sat},h}$ and $\bar{D}_{\text{Sun},h}$, respectively, where h is the hour of the day we are considering and the first index corresponds to the day of the week.

In order to create some kind of standard about working days and holidays we are going to consider from now on full working days as those days for which no province or capital of province is on holiday and full holiday days Sundays or National holidays. Considering this, we can calculate the means of the Energy Demand for both the typical full holiday day, $\bar{D}_{H,h}$ from now on, and that of a standard full working day or $\bar{D}_{W,h}$. Taking these two measures as a reference, we transform the original time series into its normalized version.

It is important to remind that in this process of normalization we work splitting the original series into the 24 series corresponding to each of the hours of the day. This is important since the impact of the normalization is not at all the same for all the hours of the day. For instance, at night the variation between the demand on a working day and a holiday day is lesser, since almost everybody is sleeping no matter the day of the week.

For each day of the year, the fraction of the population in Spain that is on holiday ρ_H is considered. In order to calculate ρ_H the holiday calendar and the population of each of the provinces in Spain, taken from Wikipedia, have been used. This fraction will be used to normalize those days where there are some people working in Spain, while others are on holidays. The fraction of the population working is then $\rho_W = 1 - \rho_H$. In these cases, we consider the normalized demand D to be a linear combination of a normalized working day component $D_{W,h}$ and a normalized holiday component $D_{H,h}$. Therefore we have

$$D_h = D_{W,h}\rho_W + D_{H,h}\rho_H$$

In this schema, we could extract the working-day component, the one we will use as the

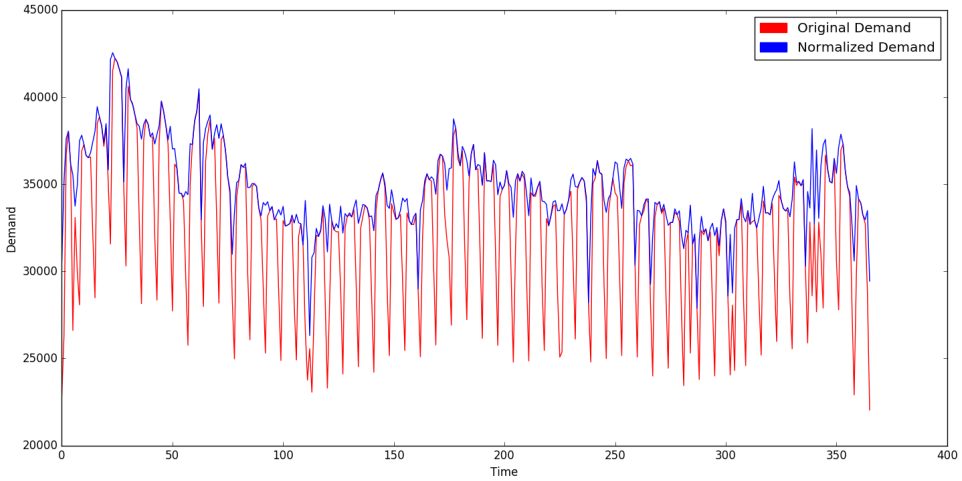


Figure 4.3.3: Comparison between the original (red) and normalized (blue) energy demand on 2011.

normalized demand D_h^n as

$$D_h^n = \frac{D_h - \bar{D}_H \rho_H}{\rho_W}$$

And our predicted demand D_h^n can be transformed into a non-normalized version D_h just as

$$D_h = \rho_W D_h^n + \rho_H \bar{D}_H$$

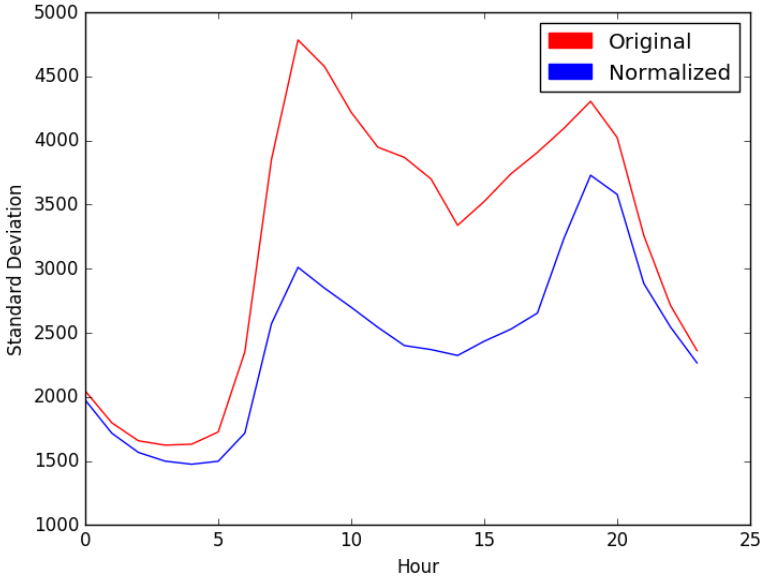


Figure 4.3.4: Standard Deviation per hour before and after the normalization

We tried this method, but there is a problem with it, since when $\rho_W \approx 0$ it is numerically unstable. This could be solved by using in these cases an analogous formula having ρ_H at the denominator, but then we could have problems transforming our predictions into a non-normalized version of them, since the formula for this transformation would consist on a subtraction that could be, if the prediction is not accurate, close to zero or even negative.

Because of these problems, we decided to use a more stable approach, which uses the same ideas, but applies an unique multiplication to complete the transformation. This way, transforming the predictions back to the original scale cannot increase the errors made in such predictions, since the relative errors on both scales will be exactly the same.

In our new approach, we transform the original series into a normalized one following these steps:

- For each day ρ_W is computed. For all days except Saturdays, we proceed as:

$$D_h^n = D_h \left(\frac{\bar{D}_{W,h}}{\bar{D}_{H,h}} + \rho_W \left(1 - \frac{\bar{D}_{W,h}}{\bar{D}_{H,h}} \right) \right)$$

  For a full working day we have $\rho_W = 1$ and we keep the demand without any transformation. Otherwise, the expected demand is transformed depending on the number of people that is expected to be on holiday.

- Saturdays are treated in a similar way, but now we use the combination between the expected demand for a Saturday and that of a holiday. The formula that integrates this is

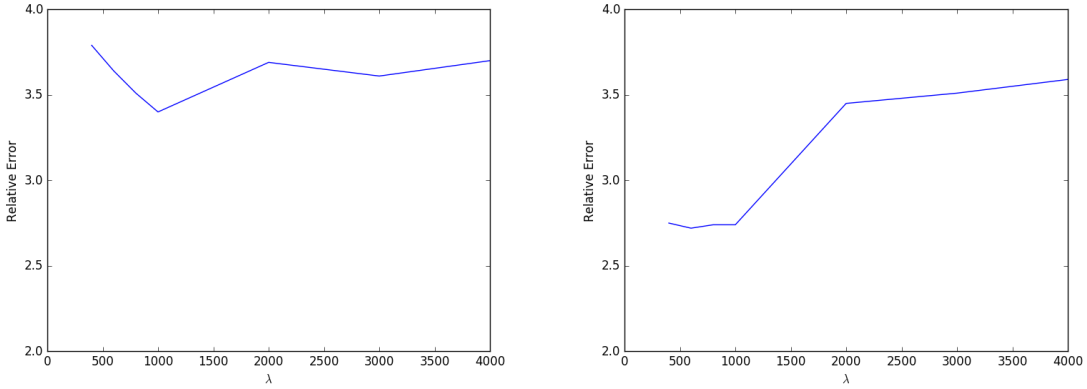
$$\tilde{D}_h = D_h \left(\frac{\bar{D}_{W,h}}{\bar{D}_{\text{Sat,h}}} \rho_W + (1 - \rho_W) \frac{\bar{D}_{W,h}}{\bar{D}_{H,h}} \right)$$

After these normalizations, the series of the demand has been transformed into a less complicated one. At Figure 4.3.3 we show a comparison between the original series, at hour 13, and the series after the normalization. The variance on these series is important and can cause us trouble.

At Figure 4.3.4 we show the standard deviation of both series, separated in the 24 hours. It is easy to see that the normalization has reduced the variance consistently for every hour of the day, which was one of our objectives when trying this normalization.

We can see that, for most of the series this normalization has achieved its objective, translating the series into another one that for most of the days is precisely the original series, and for those days for which the demand has been changed, their demand has been transform using values that allow us to think of a stationary time series when we look at the normalized series. We see however, that there are some days when this process has not worked quite well. An example of this is the day around 110, which corresponds to 23/04/2011. Although the normalized demand is higher than the original one, it is easy to see that it should be even higher to be at more or less the same level that its neighbours. When we look closer at what is happening there we see that it is a day for which a number of unusual factors coincide. This day is Easter Saturday, which is a holiday for most of the cities at Spain, and being a Saturday for which the previous Friday and the next Sunday are also holidays it could probably be considered like a Sunday without any problem. However, the algorithm for normalization that we have described is not taking this into account.

Regardless of this day, and a couple more, the normalization has worked properly and we can work with it from now on in order to apply TF in a more precise way. We remember that TF is designed to extract the trend of the series, not the periodical elements of it, and because of this it is necessary to normalize them before applying TF so it can give us the best possible results.



(a) Relative Error of TF predictions depending on λ

(b) Relative Error of Polished TF predictions depending on λ

Figure 4.3.5: Results of the Validation for TF and Polished TF for next-day predictions. We show the relative errors on the validation year, 2012 depending on the chosen λ

4.3.3 Election of the λ parameter

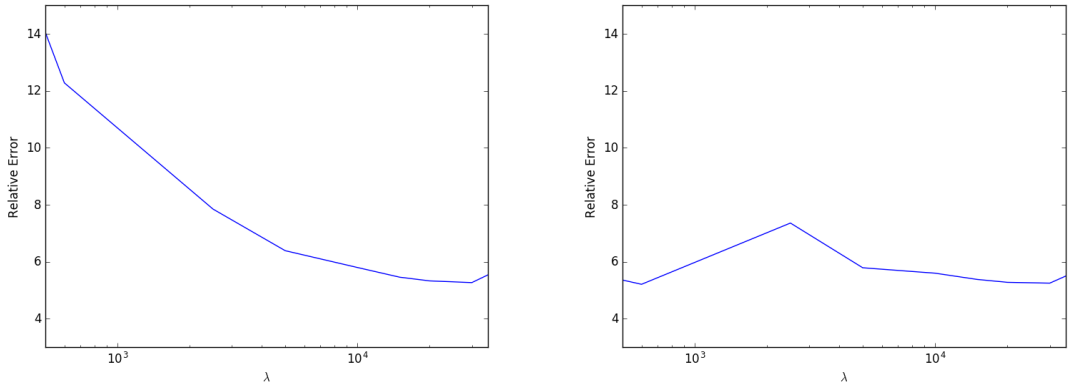
Now that we have 24 time series from which we have deleted all the variance we can using our prior knowledge, we are able to use what we have learnt about TF to analyse the data and try to predict the Energy Demand for the next day. The election of an adequate value for the λ parameter is critical, since it is the way we have to obtain a good compromise between the fitness of the TF series to the original one and its generalization capabilities.

Our first approximation regarding this problem is to use the one standard error rule, that is, to choose the largest value of λ such that the validation error is not greater than one standard deviation greater than the minimal error. The *genlasso* package provides a function capable of calculating this. However, analysing the results obtained by this method, we realised soon that these were not good enough, and that the results could be improved greatly considering the full path of solutions of the Lasso.

Since we have worked with 3 years of data, the method that has been used to adjust the λ parameter has been testing different values of λ to predict the second year of the demand, using the λ that yields the best performance to predict the demand of the third year to test the results. At this point it is important to recall that we have a closed formula (3.3.9) to obtain $\lambda_{\max}$, so we only need to calculate the errors for $\lambda < \lambda_{\max}$. Besides, the length of the time series that is used in each case can affect the optimum λ. Because of this, we always apply TF to time series of exactly one year. This way, when we want to predict the first day of 2012 we use the whole 2011, but for the first day of 2013 we only use the information of 2012, and not that of 2011.

In order to find the optimum λ we test the obtained values of λ against the second year of the Demand. First we try to predict the demand for the next day. At Figure 4.3.5a we show the relative error obtained for a number of values of λ, that we use to finally justify our final election. It is important to note that we are using the same λ for all the hours of the day. One could argue that this is not optimum and that changing this approach could improve the performance of the model. This will definitely be something interesting to test in the future, but as we will see at Figure 4.3.10, since the error on every hour of the day is similar, the improvement that we can expect by this change is not that important, while it is 24 times more computationally expensive than the original approach.

We are not going to use only Standard TF, but we are also going to try the Polished



(a) Relative Error of TF predictions depending on λ

(b) Relative Error of Polished TF predictions depending on λ

Figure 4.3.6: Results of the Validation for TF and Polished TF for 7-days ahead predictions. We show the relative errors on the validation year, 2012 depending on the chosen λ

version of TF. Our intuition is that the optimum values for the λ parameter for the Standard TF will also give us the best results when Polishing is applied, but we will see this is not the case. At Figure 4.3.5b the errors on the validation year for Polished TF are shown.

It is remarkable that the optimum values of λ for the two different methods are not the same. If we believe that TF is giving us the most representative kink points when the relative error at the TF curve is lesser (around $\lambda = 1000$), we would expect the optimum λ for Polished TF to be around the same value. There is no reason of course to argue that both optimum λ need to be the same but intuitively, it is reasonable to think that they should be similar.

The truth is that the optimum value of λ is reached at a point where Standard TF is not providing the best results it can give (although the errors at this point are quite similar to the best ones). The reason behind this behaviour still remains unclear, but the conclusion is that, after performing Polishing, we are able to take advantage of more degrees of freedom before overfitting becomes a problem.

Anyway, after the validation over the second year of the series, the values chosen to perform the definitive experiments have been $\lambda = 1000$ for Standard TF and $\lambda = 600$ for obtaining the kink points to apply Polishing. At Table 4.3.1 we show the details of the validation results for 1-day predictions.

Now we present the results of the validation for predictions 1-week ahead. The procedure that has been followed has been exactly the same and to calculate predictions for TF and Polished TF we have extended linearly the trend 7 days instead of only 1. Although we don't have any mathematical conviction that the trend can reasonably be extended for such a long time like we had for the case of next day prediction, there are intuitive reasons to believe this approach is worth a try.

We can see at Figure 4.3.6 the results of the validation and the details on Table 4.3.7. In this case the values of λ that have been found to work best have been $\lambda = 30000$ for TF and $\lambda = 600$ for Polished TF.

Other window time predictions have been tried (that is, we have tried to perform n-days ahead predictions for $1 \leq n \leq 7$) and have seen that, for Polishing, $\lambda \approx 600$ behaves quite well, regardless of the number of days we are trying to forecast. At Tables 4.3.1 to

Table 4.3.1: Relative Validation Errors for TF and Polish TF in 1 day ahead predictions

λ	TF Error	Polished TF Error
400	3.79	2.75
600	3.64	2.72
800	3.51	2.74
1000	3.40	2.74
2000	3.69	3.45
3000	3.61	3.51
4000	3.70	3.59
5000	3.71	3.58
10000	3.73	3.62
15000	3.75	3.63
20000	3.75	3.65
25000	3.79	3.68
30000	3.85	3.72
40000	3.91	3.72

Table 4.3.2: Relative Validation Errors for TF and Polish TF in 2 days ahead predictions

λ	TF Error	Polished TF Error
400	5.72	3.34
600	5.36	3.39
800	5.10	3.43
1000	4.90	3.47
2000	4.37	3.62
3000	4.14	3.68
4000	4.02	3.71
5000	3.97	3.74
10000	3.92	3.85
15000	3.90	3.87
20000	3.70	3.88
25000	3.91	3.89
30000	3.92	3.90
40000	3.93	3.92

Table 4.3.3: Relative Validation Errors for TF and Polish TF in 3 days ahead predictions

λ	TF Error	Polished TF Error
400	7.65	3.83
600	7.06	3.89
800	6.63	3.95
1000	6.30	4.00
2000	5.35	4.16
3000	4.94	4.23
4000	4.71	4.23
5000	4.58	4.24
10000	4.39	4.29
15000	4.34	4.30
20000	4.31	4.29
25000	4.30	4.27
30000	4.36	4.27
40000	4.32	4.28

Table 4.3.4: Relative Validation Errors for TF and Polish TF in 4 days ahead predictions

λ	TF Error	Polished TF Error
400	9.49	4.23
600	8.63	4.31
800	8.01	4.38
1000	7.54	4.45
2000	6.19	4.62
3000	5.61	4.68
4000	5.50	4.68
5000	5.22	4.68
10000	4.38	4.74
15000	4.85	4.73
20000	4.72	4.68
25000	4.65	4.62
30000	4.60	4.59
40000	4.55	4.65
50000	4.81	

Table 4.3.5: Relative Validation Errors for TF and Polish TF in 5 days ahead predictions

λ	TF Error	Polished TF Error
400	11.41	4.61
600	10.27	4.71
800	9.42	4.79
1000	8.76	4.86
2000	6.99	5.05
3000	6.22	5.13
4000	5.84	5.14
5000	5.63	5.14
10000	5.20	5.14
15000	5.20	5.13
20000	5.09	5.04
25000	4.99	4.96
30000	4.91	5.01
40000	4.83	5.00
50000	4.93	

Table 4.3.6: Relative Validation Errors for TF and Polish TF in 6 days ahead predictions

λ	TF Error	Polished TF Error
400	13.05	4.99
600	11.65	4.90
800	10.62	5.09
1000	9.81	5.18
2000	7.71	5.46
3000	6.84	5.58
4000	6.39	5.57
5000	6.13	5.56
10000	5.72	5.55
15000	5.57	5.48
20000	5.42	5.35
25000	5.30	5.25
30000	5.19	5.16
40000	5.07	5.17
50000	5.08	

Table 4.3.7: Relative Validation Errors for TF and Polish TF in 7 days ahead predictions

λ	TF Error	Polished TF Error
400	14.67	5.25
600	12.28	5.21
800	11.41	5.54
1000	10.53	5.64
2000	8.11	5.95
3000	7.67	6.06
4000	6.82	6.03
5000	6.39	6.39
10000	5.80	5.80
15000	5.37	5.46
20000	5.33	5.33
25000	5.31	5.31
30000	5.27	5.27
40000	5.56	5.48

4.3.7 we can see that this value yields the best performance in both cases (next-day and 7-days ahead predictions). We can therefore think that this is a good value for calculating predictions using Polishing.

Validation errors over 2012 have been calculated for predictions on $n = 2, ..., 6$ days-ahead, so we can test this intuition. We see that this is the case for most of the days that have been tested, although in some cases $\lambda = 400$ is better for applying Polishing. Besides we see that Polished TF is very robust against the election of λ, giving in most of the cases similar results regardless of the λ that has been used.

Regarding the validation errors for TF, we see that the optimum λ tends to be higher when we try to find longer term predictions. Some of the validation tests we have done, like the one for 6-days ahead predictions, it looks like the optimum λ is $\lambda \geq 40000$, although for most of the cases the best results are obtained for lower values of λ.

As a conclusion, we see that, although there is a clear tendency of the optimum λ to be higher when we try to predict more days ahead, they is no a clear good value, like $\lambda = 600$ in the case of Polished TF that works fine for all the cases. Therefore, choosing the best λ is more difficult when we are using TF, although we can see that, with the exception of next-day predictions, choosing $\lambda \approx 10^4$ works fine in most of the cases.

Regarding the election of the λ to use on the test year, for the cases in which a minimum is found in the validation year, this minimum value is the one that has been used in the test year. However, for those cases where the error using TF decreases until $\lambda = 40000$, one more test using $\lambda = 50000$ has been performed to confirm that using a higher λ we are not able to obtain better results.

4.3.4 Results

At this section, the results obtained by the use of TF to forecast the Energy Demand for the test year 2013 are presented. These results are also compared with some other models in order to get an idea of the capabilities of TF and Polish TF over them. The results given in this section have, all of them, being obtained using the optimum λ, as obtained at the previous section.

First we take a look at the performance of TF for predictions for the next day. At Figure 4.3.7 the relative errors over the whole year 2013, the one that has been used for testing,

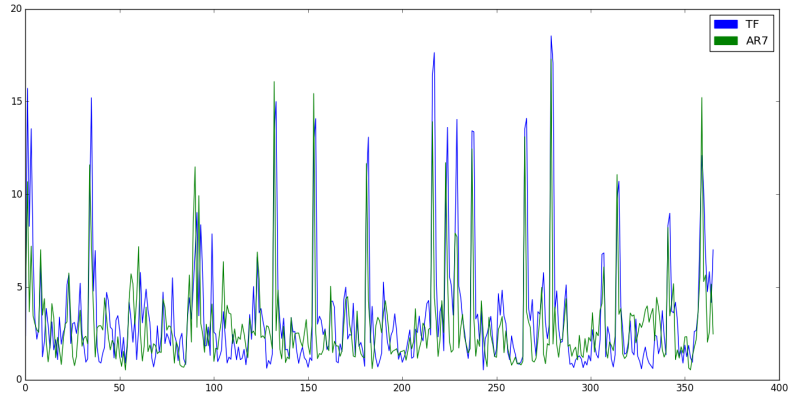


Figure 4.3.7: Relative Error for predictions for the next day on 2013. In blue we show the error of TF and in green that of the AR(7) model

are shown. In the following Figures the relative errors that are reported corresponds to the mean relative error of each day, or more precisely, the mean of the relative errors for all the 24 predictions of each of the hours of the day.

TF is compared against an AR(7) model. The election of $n = 7$ for the AR(n) model is due, evidently, to the fact that it is known that the model can benefit from the information from the same day of the week from the previous week. This is specially true when we try to predict the demand on Saturdays or Sundays. Regarding this last model, two different approaches have been tested, yielding virtually the same results. AR(7) was first applied directly to the normalized series but also to the series of the 7-days differences of the original (not-normalized) demand. After the de-normalization process the results were almost the same, but doing this exercise was necessary to check that the *arima* function was able to perform well with the normalized series since this model requires the series to be stationary to work properly. The *arima* function that we used is theoretically robust enough to extract the non-stationary part of the series, but we wanted to test this feature. The first approach, using function *arima* on the normalized series, has been the one for which the results on this text are shown.

Now, Polishing is applied over TF. A comparison between the errors of this method and AR(7) are shown at Figure 4.3.8. We recall that the λ we use for Polishing is not the same we used to predict when we used Standard TF, but another one in the path of solutions for which, although the forecast applying TF is worse, minimizes the error obtained after performing Polishing. At Table 4.3.8 we show the mean relative error of the methods applied for both next-day and next-week predictions. We see that the Polished version of TF is able to win an AR(7) model for next-day predictions, while AR(7) is the winner for longer-term predictions. We will see later which is the number of days for which Polishing has a better performance than AR(7).

We see that after doing Polishing the errors are lowered, and this method obtains the best performance of all the models that have been tested so far, for next-day predictions. Now we analyse the errors in more detail, to try to learn which are the cases where we are having more trouble to predict the demand.

First we analyse the errors depending on the day of the week that is being predicted. We see at Figure 4.3.9 that these models have problems specially on Fridays. The reason is probably due to the fact that they are close to weekends, while not being part of them.

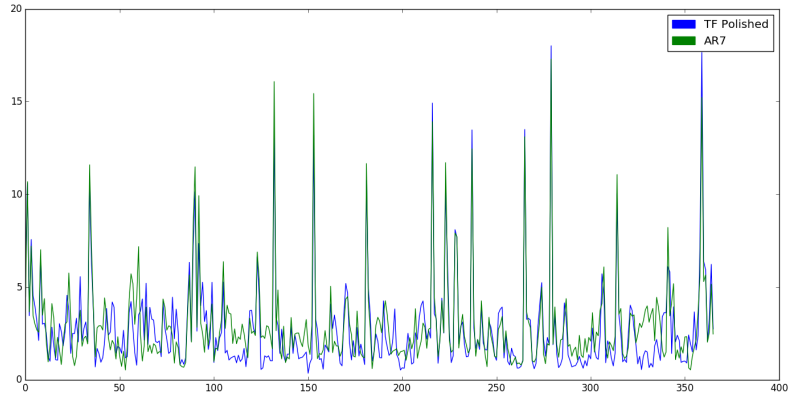


Figure 4.3.8: Relative Error for predictions for the next day on 2013. In blue we show the error of Polished TF and in green that of the AR(7) model

Table 4.3.8: Relative Errors of the models when tested on year 2013

Method	Relative MAE (next-day)	Relative MAE (next-week)
Persistence model	3.50	5.35
TF	3.40	5.27
Polished TF	2.72	5.19
AR(7)	2.93	4.05

This makes the demand of these days very dependent on long weekends close to them. It is interesting to note, however, that although TF has more trouble predicting the demand on weekends, applying Polishing reduces this problem. Although all days of the week have lower errors after applying Polishing, this is specially true when we look at the weekends.

It is also interesting to compare the relative error depending on the hour of the day that we are trying to predict. A full comparison on this, for all the methods that have been tested is shown at Figure 4.3.10. AR(7) and the two types of TF that we have tried have a similar behaviour in this aspect. The easiest hours to predict are the first hours of the day, while the most difficult ones are around 8:00 and 18:00. These hours correspond to those where the slope of the demand is higher, which makes it more complicated to fit a good model for these hours. It is interesting to note that, although this also happens at around 23:00 until 4:00, predicting the demand for these hours is not so difficult. This is due to the fact that since most of the people are sleeping at these hours, the demand on all the days of the week is quite similar and therefore easier to predict.

We show at Figure 4.3.11 the fitting of the prediction made by Polished TF (next-day prediction) for 14-03-2013. This day has been chosen for having, for the Polished model, a mean relative error of 2.769 %, being therefore representative of the mean error that we are to expect when making predictions for the demand of the next day. We can see that, although the prediction has underestimated the demand, the prediction and the actual demand are quite close.

Finally we analyze the results obtained for predictions between the next-day and next-week ones. At Table 4.3.9 Polished TF is compared against Persistence and AR7 models. As we can see, Polished TF is also competitive when compared to AR(7) for predictions for the next two or three days. However, for longer-term predictions, AR(7) clearly outperforms this model.

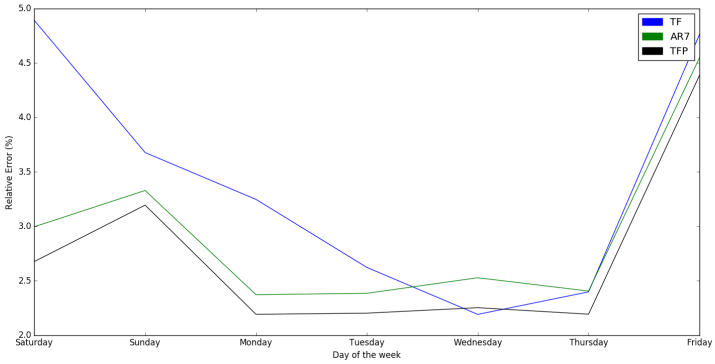


Figure 4.3.9: Relative Errors per day of the week

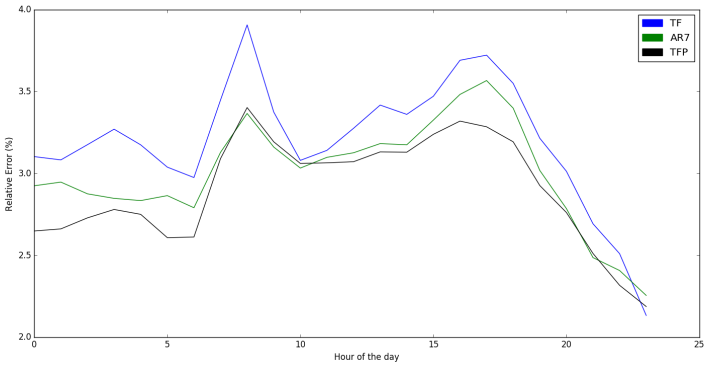


Figure 4.3.10: Relative Error per hour of the day

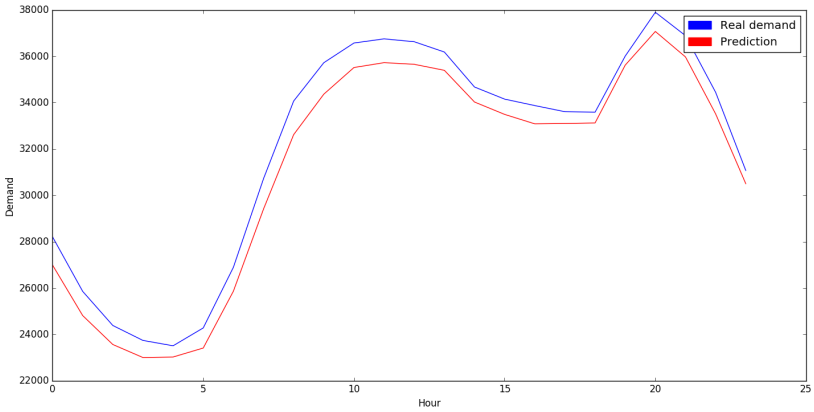


Figure 4.3.11: Original Demand (Blue) vs Prediction (Red) for the Polished TF model on 14-03-2013

Therefore, Polished TF is the most competitive of the tested methods for next-day forecasting and can also challenge, and sometimes win, the AR(7) model. However, this last method is a clear winner when we want to forecast the behaviour of the energy demand on the long term.

Table 4.3.9: Relative Errors of the models when tested on 2013 for predictions more than one day ahead

Number of days	TF Error	Polished TF Error	AR(7)Error	Persistence Error
2	3.61	3.21	3.11	3.71
3	4.22	3.64	3.67	4.28
4	4.32	4.11	3.67	4.44
5	4.65	4.46	3.80	4.91
6	5.01	4.77	3.80	5.22
7	5.27	5.19	4.05	5.35

Chapter 5

Conclusions and future work

5.1 Conclusions

In this Master thesis, a powerful resource to analyse the trend of a time series, that is, L_1 TF has been discussed. The mathematical tools that are required are introduced first and after setting the theoretical framework where it lies, it is applied to a real-life problem: the energy demand.

After calculating the trend of the energy demand by using TF we are able to characterize how the demand in the peninsular part of Spain is behaving in a quite accurate and robust way. We are also able to learn when this trend is changing its behaviour.

Checking that this method is able to provide us some descriptive responses in this sense, we face a more challenging task: to try to forecast the energy demand in Spain using TF. After some experiments we realize that, although the predictions are reasonably accurate, some well-known methods like autoregressive models slightly outperform our first naive approximation of directly using TF.

Therefore we decide to apply a modification on Standard TF, Polished TF, that is based on applying TF over the time series to obtain the points at which the trend changes and then use that information to construct our prediction in a way that yields similar but more adjusted to the actual series results. Using this variation on TF we are able to get better results, slightly better ones than those obtained by AR(7) on the short-term.

It is important to take into account that, since the main objective of this work is to analyse the capabilities of TF we have used no more information than the demand series to construct our model, aside from the knowledge of the calendar and the holidays in Spain and the population of each region in Spain, that have help us adapt our predictions depending on the number of people that is working on a specific day in Spain. We expect therefore that these results should be improvable using more information. Although the mere use of TF and specially its variation with Polishing has allowed us to find a reasonably good model for this problem, we believe the strength of this method lies in the identification of the kink points, or points when the trend changes, and possibly use them to construct more advanced models.

5.2 Future work

The energy demand is heavily influenced by the weather and there is one variable that has an important impact on the demand, the temperature. We have reasons to believe that, if this variable is introduced in the normalization process, the results will be better. We don't expect miracles after this change, since we should be already using indirectly some

of this information in the sense that the temperatures are also influencing the demand of the days we are calculating TF over, and also because this supposes to use the weather forecast in our own forecasting, but we think checking this would be interesting.

Analysis of the kink points that have been obtained as a result of applying TF before predicting might be useful to have an idea of the quality that we can expect from the prediction. For instance, we could analyse the mean of the sizes of the intervals of the trend to have an estimation of how often there is a kink point in the series. This way we could know if the last interval of the trend, the one that is going to be used to perform the prediction is so long that we can think another kink point is about to arise in the series. Or maybe that it is too small to rely on it.

Another possibility would be to calculate several trends depending on λ and choose, for the prediction of each hour, the one that maximizes the interval for which the linear extension property holds. An approach like this will increase the computational cost of the method, but we expect that it would also improve the precision of the predictions.

Finally, the information given by the kink points of the time series could be useful as input for other type of models. Therefore, we could use all we have learnt to pre-process time series when using machine learning models, and hopefully obtain better results after this pre-processing than those that we achieve applying those models to the original time series.

Bibliography

[1] Kim, S.-J., Koh, K., Boyd, S. & Gorinevsky, D. (2009), 'l_1 trend filtering', *SIAM Review* **51**(2), 339-360.

[2] Yu. Nesterov *Introductory Lectures on Convex Optimization Basic Course*

[3] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces.* Springer, 2011.

[4] Erik J. Balder. On subdifferential calculus. Available as `http://staff.science.uu.nl/~balde101/cao10/cursus10_1.pdf`, September 2008.

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

[6] Tibshirani, R.J. The solution path of the generalized Lasso. *The Annals of Statistics* 2011, Vol 39, No 3, 1335-1371

[7] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, *Sparsity and smoothness via the fused Lasso,* J. R. Stat. Soc. Ser. B Stat. Methodol., 67 (2005), pp. 91-108.

[8] Hodrick, R., Prescott, E. (1997). 'Postwar U.S. Business Cycles: An Empirical Investigation'. *Journal of Money, Credit, and Banking* **29** (1): 1-16.

[9] Mills, Terence C. (2003). 'Filtering Economic Time Series'. Modelling Trends and Cycles in Economic Time Series. New York: Palgrave McMillan. pp.75-102.

[10] Johnson, N. (2013). 'A dynamic programming algorithm for the fused lasso and L_0-segmentation', *Journal of Computational and Graphical Statistics* **22**(2), 246-260.

[11] Boyd, S., Parikh, N., Chu, E., Peleato, B. & Eckstein, J. (2011), 'Distributed optimization and statistical learning via the alternative direction method of multipliers', *Foundations and Trends in Machine Learning* **3**(1), 1-122.

[12] Tibshirani, R.J. (2014), Adaptive piecewise polynomial estimation via trend filtering, *Annals of Statistics* **42**(1), 285-323

[13] Rudin,L. I., Osher, S. & Faterni, E. (1992). 'Nonlinear total variation based noise removal algorithms', *Physica D: Nonlinear Phenomena **60**, 259-268.*

[14] Nello Christianini. John Shawe-Taylor. Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Chapter 5. Cambridge University Press, 2000

[15] M. Fortin and R. Glowinski, Eds., *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems.* Amsterdam: North-Holland, 1983.

[16] Ramdas, A., Tibshirani, R.J., *Fast and Flexible ADMM Algorithms for Trend Filtering*

[17] Davies, P.L. & Kovac, A. (2001), Local extremes, runs, string and multiresolution, *Annals of Statistics* **29**(1), 1-65.

[18] Rockafellar, R. T. (1996). *Convex Analysis (Princeton Landmarkds in Mathematics and Physics).* Princeton University Press, Princeton, NJ, USA.

[19] M.R. Hestenes, Multiplier and gradient methods, *Journal of Optimization Theory and Applications,* 4, 1969, 303-320

[20] Fiacco, A. V. and McCormick, G. P. 1968, *Non-linear Programming: Sequential Unconstrained Minimization Techniques,* John Wiley and Sons, New York.

[21] Borwein, J., Lewis, A. *Convex Analysis and NonLinear Optimization,* Springer-Verlag, New York, 2000.

[22] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming,* Stud. Appl. Math. 13, SIAM, Philadelphia, 1994.

[23] Tibshirani, R. J. (2014). Supplement to Adaptive piecewise polynomial estimation via trend filtering. DOI:10.1214/13-AOS1189SUPP.

[24] Patrick L. Combettes, *Fellow, IEEE,* and Jean-Christophe Pesquet, *Senior Member, IEEE.* Proximal Splitting Methods in Signal Processing.

[25] Wang, Y., Smola, A. & Tibshirani, R. J. (2014). 'The falling factorial basis and its statistical properties', *International Conference on Machine* **31**.

[26] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least angle regression. *Annals of Statistics.* **32** 407-499

[27] S. Wright, *Primal-Dual Interior-Point Methods,* SIAM, Philadelphia, 1997.

[28] Mammen, E. and Van De Geer, S. (1997). Localy adaptive regression splines. *Ann. Statist.* **25** 387-413

[29] Donoho, D. L. and Johnstone, I. M. (1998). Minimax estimation via wavelet shrinkage. *Ann. Statist.* **26** 879-921.

[30] Nussbaum, M. (1985). Spline smoothing in regression models and asymptotic efficiency in L_2. *Ann. Statist.* **13** 984-997.

[31] Potra, Florian A., Stephen J. Wright. Interior-Point Methods. *Journal of Computational and Applied Mathematics* **124** (1-2): 281-302.