1 Convex Optimization with Sparsity-Inducing Norms

Francis Bach francis.bach@inria.fr
INRIA - Willow Project-Team
23, avenue d'Italie, 75013 PARIS

Rodolphe Jenatton rodolphe.jenatton@inria.fr
INRIA - Willow Project-Team
23, avenue d'Italie, 75013 PARIS

Julien Mairal julien.mairal@inria.fr
INRIA - Willow Project-Team
23, avenue d'Italie, 75013 PARIS

Guillaume Obozinski guillaume.obozinski@inria.fr
INRIA - Willow Project-Team
23, avenue d'Italie, 75013 PARIS

1.1 Introduction

The principle of parsimony is central to many areas of science: the simplest explanation to a given phenomenon should be preferred over more complicated ones. In the context of machine learning, it takes the form of variable or feature selection, and it is commonly used in two situations. First, to make the model or the prediction more interpretable or computationally cheaper to use, i.e., even if the underlying problem is not sparse, one looks for the best sparse approximation. Second, sparsity can also be used given prior knowledge that the model should be sparse.

For variable selection in linear models, parsimony may be directly achieved by penalization of the empirical risk or the log-likelihood by the cardinality of the support of the weight vector. However, this leads to hard combinatorial problems (see, e.g., Tropp, 2004). A traditional convex approximation of the problem is to replace the cardinality of the support by the ℓ_1-norm. Estimators may then be obtained as solutions of convex programs.

Casting sparse estimation as convex optimization problems has two main benefits: First, it leads to efficient estimation algorithms—and this chapter focuses primarily on these. Second, it allows a fruitful theoretical analysis answering fundamental questions related to estimation consistency, prediction efficiency (Bickel et al., 2009; Negahban et al., 2009) or model consistency (Zhao and Yu, 2006; Wainwright, 2009). In particular, when the sparse model is assumed to be well-specified, regularization by the ℓ_1-norm is adapted to high-dimensional problems, where the number of variables to learn from may be exponential in the number of observations.

Reducing parsimony to finding the model of lowest cardinality turns out to be limiting, and *structured parsimony* has emerged as a natural extension, with applications to computer vision (Jenatton et al., 2010b), text processing (Jenatton et al., 2010a) or bioinformatics (Kim and Xing, 2010; Jacob et al., 2009). Structured sparsity may be achieved by regularizing by other norms than the ℓ_1-norm. In this chapter, we focus primarily on norms which can be written as linear combinations of norms on subsets of variables (Section 1.1.1). One main objective of this chapter is to present methods which are adapted to most sparsity-inducing norms with loss functions potentially beyond least-squares.

Finally, similar tools are used in other communities such as signal processing. While the objectives and the problem set-up are different, the resulting convex optimization problems are often very similar, and most of the techniques reviewed in this chapter also apply to sparse estimation problems in signal processing.

This chapter is organized as follows: in Section 1.1.1, we present the optimization problems related to sparse methods, while in Section 1.1.2, we review various optimization tools that will be needed throughout the chapter. We then quickly present in Section 1.2 generic techniques that are not best suited to sparse methods. In subsequent sections, we present methods which are well adapted to regularized problems, namely proximal methods in Section 1.3, block coordinate descent in Section 1.4, reweighted ℓ_2-methods in Section 1.5, and working set methods in Section 1.6. We provide quantitative evaluations of all of these methods in Section 1.7.

1.1.1 Loss functions and Sparsity-Inducing Norms

We consider in this chapter convex optimization problems of the form

$$\min_{\boldsymbol{w}\in\mathbb{R}^p} f(\boldsymbol{w}) + \lambda\Omega(\boldsymbol{w}), \tag{1.1}$$

where $f : \mathbb{R}^p \to \mathbb{R}$ is a convex differentiable function and $\Omega : \mathbb{R}^p \to \mathbb{R}$ is a sparsity-inducing—typically nonsmooth and non-Euclidean—norm.

In supervised learning, we predict outputs y in $\mathcal{Y}$ from observations $\boldsymbol{x}$ in $\mathcal{X}$; these observations are usually represented by p-dimensional vectors, so that $\mathcal{X} = \mathbb{R}^p$. In this supervised setting, f generally corresponds to the empirical risk of a loss function $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$. More precisely, given n pairs of data points $\{(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathbb{R}^p \times \mathcal{Y};\ i = 1, \dots, n\}$, we have for linear models $f(\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^n \ell(y^{(i)}, \boldsymbol{w}^T\boldsymbol{x}^{(i)})$. Typical examples of loss functions are the square loss for least squares regression, i.e., $\ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$ with y in $\mathbb{R}$, and the logistic loss $\ell(y, \hat{y}) = \log(1 + e^{-y\hat{y}})$ for logistic regression, with y in $\{-1, 1\}$. We refer the readers to Shawe-Taylor and Cristianini (2004) for a more complete description of loss functions.

When one knows *a priori* that the solutions $\boldsymbol{w}^\star$ of problem (1.1) only have a few non-zero coefficients, Ω is often chosen to be the ℓ_1-norm, i.e., $\Omega(\boldsymbol{w}) = \sum_{j=1}^p |\boldsymbol{w}_j|$. This leads for instance to the Lasso (Tibshirani, 1996) with the square loss and to the ℓ_1-regularized logistic regression (see, for instance, Shevade and Keerthi, 2003; Koh et al., 2007) with the logistic loss. Regularizing by the ℓ_1-norm is known to induce sparsity in the sense that, a number of coefficients of $\boldsymbol{w}^\star$, depending on the strength of the regularization, will be *exactly* equal to zero.

In some situations, for example when encoding categorical variables by binary dummy variables, the coefficients of $\boldsymbol{w}^\star$ are naturally partitioned in subsets, or *groups*, of variables. It is then natural to select or remove *simultaneously* all the variables forming a group. A regularization norm exploiting explicitly this group structure can be shown to improve the prediction performance and/or interpretability of the learned models (Yuan and Lin, 2006; Roth and Fischer, 2008; Huang and Zhang, 2009; Obozinski et al., 2009; Lounici et al., 2009). Such a norm might for instance take the form

$$\Omega(\boldsymbol{w}) := \sum_{g\in\mathcal{G}} d_g \|\boldsymbol{w}_g\|_2, \tag{1.2}$$

where $\mathcal{G}$ is a partition of $\{1, \dots, p\}$, $(d_g)_{g\in\mathcal{G}}$ are some positive weights, and $\boldsymbol{w}_g$ denotes the vector in $\mathbb{R}^{|g|}$ recording the coefficients of $\boldsymbol{w}$ indexed by g in $\mathcal{G}$. Without loss of generality we may assume all weights $(d_g)_{g\in\mathcal{G}}$ to be equal to

one. As defined in Eq. (1.2), Ω is known as a mixed ℓ_1/ℓ_2-norm. It behaves like an ℓ_1-norm on the vector $(\|\boldsymbol{w}_g\|_2)_{g\in\mathcal{G}}$ in $\mathbb{R}^{|\mathcal{G}|}$, and therefore, Ω induces group sparsity. In other words, each $\|\boldsymbol{w}_g\|_2$, and equivalently each $\boldsymbol{w}_g$, is encouraged to be set to zero. On the other hand, within the groups g in $\mathcal{G}$, the ℓ_2-norm does not promote sparsity. Combined with the square loss, it leads to the group Lasso formulation (Yuan and Lin, 2006). Note that when $\mathcal{G}$ is the set of singletons, we retrieve the ℓ_1-norm. More general mixed ℓ_1/ℓ_q-norms for $q > 1$ are also used in the literature (Zhao et al., 2009):

$$\Omega(\boldsymbol{w}) = \sum_{g\in\mathcal{G}} \|\boldsymbol{w}_g\|_q := \sum_{g\in\mathcal{G}} \Big\{ \sum_{j\in g} |\boldsymbol{w}_j|^q \Big\}^{1/q}.$$

In practice though, the ℓ_1/ℓ_2- and ℓ_1/ℓ_∞-settings remain the most popular ones.

In an attempt to better encode structural links between variables at play (e.g., spatial or hierarchical links related to the physics of the problem at hand), recent research has explored the setting where $\mathcal{G}$ can contain groups of variables that *overlap* (Zhao et al., 2009; Bach, 2008a; Jenatton et al., 2009; Jacob et al., 2009; Kim and Xing, 2010; Schmidt and Murphy, 2010). In this case, Ω is still a norm, and it yields sparsity in the form of specific patterns of variables. More precisely, the solutions $\boldsymbol{w}^\star$ of problem (1.1) can be shown to have a set of zero coefficients, or simply *zero pattern*, that corresponds to a union of some groups g in $\mathcal{G}$ (Jenatton et al., 2009). This property makes it possible to control the sparsity patterns of $\boldsymbol{w}^\star$ by appropriately defining the groups in $\mathcal{G}$. This form of *structured sparsity* has notably proven to be useful in the context of hierarchical variable selection (Zhao et al., 2009; Bach, 2008a; Schmidt and Murphy, 2010), multi-task regression of gene expressions (Kim and Xing, 2010) and also for the design of localized features in face recognition (Jenatton et al., 2010b).

1.1.2 Optimization Tools

The tools used in this book chapter are relatively basic and should be accessible to a broad audience. Most of them can be found in classical books on convex optimization (Boyd and Vandenberghe, 2004; Bertsekas, 1999; Borwein and Lewis, 2006; Nocedal and Wright, 2006), but for self-containedness, we present here a few of them related to non-smooth unconstrained optimization.

Subgradients

Given a convex function $g : \mathbb{R}^p \to \mathbb{R}$ and a vector $\boldsymbol{w}$ in $\mathbb{R}^p$, let us define the

subdifferential of g at $\boldsymbol{w}$ as

$$\partial g(\boldsymbol{w}) := \{\boldsymbol{z} \in \mathbb{R}^p \mid g(\boldsymbol{w}) + \boldsymbol{z}^T(\boldsymbol{w}' - \boldsymbol{w}) \leq g(\boldsymbol{w}') \text{ for all vectors } \boldsymbol{w}' \in \mathbb{R}^p\}.$$

The elements of $\partial g(\boldsymbol{w})$ are called the *subgradients* of g at $\boldsymbol{w}$. This definition admits a clear geometric interpretation: any subgradient $\boldsymbol{z}$ in $\partial g(\boldsymbol{w})$ defines an affine function $\boldsymbol{w}' \mapsto g(\boldsymbol{w}) + \boldsymbol{z}^T(\boldsymbol{w}' - \boldsymbol{w})$ which is tangent to the graph of the function g. Moreover, there is a bijection (one-to-one correspondence) between such "tangent affine functions" and the subgradients. Let us now illustrate how subdifferential can be useful for studying nonsmooth optimization problems with the following proposition:

Proposition 1.1 (Subgradients at Optimality)**.**
For any convex function $g : \mathbb{R}^p \to \mathbb{R}$, a point $\boldsymbol{w}$ in $\mathbb{R}^p$ is a global minimum of g if and only if the condition $0 \in \partial g(\boldsymbol{w})$ holds.

Note that the concept of subdifferential is mainly useful for nonsmooth functions. If g is differentiable at $\boldsymbol{w}$, the set $\partial g(\boldsymbol{w})$ is indeed the singleton $\{\nabla g(\boldsymbol{w})\}$, and the condition $0 \in \partial g(\boldsymbol{w})$ reduces to the classical first-order optimality condition $\nabla g(\boldsymbol{w}) = 0$. As a simple example, let us consider the following optimization problem

$$\min_{w \in \mathbb{R}} \frac{1}{2}(x - w)^2 + \lambda |w|.$$

Applying the previous proposition and noting that the subdifferential $\partial |\cdot|$ is $\{+1\}$ for $w > 0$, $\{-1\}$ for $w < 0$ and $[-1, 1]$ for $w = 0$, it is easy to show that the unique solution admits a closed form called the *soft-thresholding* operator, following a terminology introduced by Donoho and Johnstone (1995); it can be written

$$w^\star = \begin{cases} 0 & \text{if } |x| \leq \lambda \\ (1 - \frac{\lambda}{|x|})x & \text{otherwise.} \end{cases} \tag{1.3}$$

This operator is a core component of many optimization techniques for sparse methods, as we shall see later.

Dual Norm and Optimality Conditions

The next concept we introduce is the dual norm, which is important to study sparsity-inducing regularizations (Jenatton et al., 2009; Bach, 2008a; Negahban et al., 2009). It notably arises in the analysis of estimation bounds (Negahban et al., 2009), and in the design of working-set strategies as will be shown in Section 1.6. The dual norm Ω^* of the norm Ω is defined for any

vector $\boldsymbol{z}$ in $\mathbb{R}^p$ by

$$\Omega^*(\boldsymbol{z}) := \max_{\boldsymbol{w}\in\mathbb{R}^p} \boldsymbol{z}^T\boldsymbol{w} \text{ such that } \Omega(\boldsymbol{w}) \leq 1.$$

Moreover, the dual norm of Ω^* is Ω itself, and as a consequence, the formula above holds also if the roles of Ω and Ω^* are exchanged. It is easy to show that in the case of an ℓ_q-norm, $q \in [1;+\infty]$, the dual norm is the $\ell_{q'}$-norm, with q' in $[1;+\infty]$ such that $\frac{1}{q}+\frac{1}{q'}=1$. In particular, the ℓ_1- and ℓ_∞-norms are dual to each other, and the ℓ_2-norm is self-dual (dual to itself).

The dual norm plays a direct role in computing optimality conditions of sparse regularized problems. By applying Proposition 1.1 to Eq. (1.1), a little calculation shows that a vector $\boldsymbol{w}$ in $\mathbb{R}^p$ is optimal for Eq. (1.1) if and only if $-\frac{1}{\lambda}\nabla f(\boldsymbol{w}) \in \partial\Omega(\boldsymbol{w})$ with

$$\partial\Omega(\boldsymbol{w}) = \begin{cases} \{\boldsymbol{z}\in\mathbb{R}^p;\ \Omega^*(\boldsymbol{z})\leq 1\} \text{ if } \boldsymbol{w}=0, \\ \{\boldsymbol{z}\in\mathbb{R}^p;\ \Omega^*(\boldsymbol{z})\leq 1 \text{ and } \boldsymbol{z}^T\boldsymbol{w}=\Omega(\boldsymbol{w})\} \text{ otherwise.} \end{cases} \tag{1.4}$$

As a consequence, the vector 0 is solution if and only if $\Omega^*\big(\nabla f(0)\big) \leq \lambda$.

These general optimality conditions can be specified to the Lasso problem (Tibshirani, 1996), also known as basis pursuit (Chen et al., 1999):

$$\min_{\boldsymbol{w}\in\mathbb{R}^p} \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1, \tag{1.5}$$

where $\boldsymbol{y}$ is in $\mathbb{R}^n$, and $\boldsymbol{X}$ is a design matrix in $\mathbb{R}^{n\times p}$. From Equation (1.4) and since the ℓ_∞-norm is the dual of the ℓ_1-norm we obtain that necessary and sufficient optimality conditions are

$$\forall j = 1,\dots,p, \quad \begin{cases} |\boldsymbol{X}_j^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})| \leq \lambda & \text{if } \boldsymbol{w}_j = 0 \\ \boldsymbol{X}_j^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}) = \lambda\operatorname{sgn}(\boldsymbol{w}_j) & \text{if } \boldsymbol{w}_j \neq 0, \end{cases} \tag{1.6}$$

where $\boldsymbol{X}_j$ denotes the j-th column of $\boldsymbol{X}$, and $\boldsymbol{w}_j$ the j-th entry of $\boldsymbol{w}$. As we will see in Section 1.6.1, it is possible to derive from these conditions interesting properties of the Lasso, as well as efficient algorithms for solving it. We have presented a useful duality tool for norms. More generally, there exists a related concept for convex functions, which we now introduce.

Fenchel Conjugate and Duality Gaps

Let us denote by f^* the Fenchel conjugate of f (Rockafellar, 1997), defined by

$$f^*(\boldsymbol{z}) := \sup_{\boldsymbol{w}\in\mathbb{R}^p} [\boldsymbol{z}^T\boldsymbol{w} - f(\boldsymbol{w})].$$

The Fenchel conjugate is related to the dual norm. Let us define the indicator function ι_Ω such that $\iota_\Omega(\boldsymbol{w})$ is equal to 0 if $\Omega(\boldsymbol{w}) \leq 1$ and $+\infty$ otherwise.

Then, ι_Ω is a convex function and its conjugate is exactly the dual norm Ω^*.

For many objective functions, the Fenchel conjugate admits closed forms, and can therefore be computed efficiently (Borwein and Lewis, 2006). Then, it is possible to derive a duality gap for problem (1.1) from standard Fenchel duality arguments (see Borwein and Lewis, 2006), as shown below:

Proposition 1.2 (Duality for Problem (1.1))**.**
If f^* and Ω^* are respectively the Fenchel conjugate of a convex and differentiable function f and the dual norm of Ω, then we have

$$\max_{\boldsymbol{z}\in\mathbb{R}^p:\,\Omega^*(\boldsymbol{z})\leq\lambda} -f^*(\boldsymbol{z}) \quad \leq \quad \min_{\boldsymbol{w}\in\mathbb{R}^p} f(\boldsymbol{w}) + \lambda\Omega(\boldsymbol{w}) \tag{1.7}$$

Moreover, equality holds as soon as the domain of f has non-empty interior.

Proof. This result is a specific instance of Theorem 3.3.5 in Borwein and Lewis (2006). In particular, we use the fact that (a) the conjugate of a norm Ω is the indicator function ι_{Ω^*} of the unit ball of the dual norm Ω^*, and (b) the subdifferential of a differentiable function (here, f) reduces to its gradient.

If $\boldsymbol{w}^\star$ is a solution of Eq. (1.1), and $\boldsymbol{w}, \boldsymbol{z}$ in $\mathbb{R}^p$ are such that $\Omega^*(\boldsymbol{z}) \leq \lambda$, this proposition implies that we have

$$f(\boldsymbol{w}) + \lambda\Omega(\boldsymbol{w}) \geq f(\boldsymbol{w}^\star) + \lambda\Omega(\boldsymbol{w}^\star) \geq -f^*(\boldsymbol{z}). \tag{1.8}$$

The difference between the left and right term of Eq. (1.8) is called a duality gap. It represents the difference between the value of the primal objective function $f(\boldsymbol{w}) + \lambda\Omega(\boldsymbol{w})$ and a dual objective function $-f^*(\boldsymbol{z})$, where $\boldsymbol{z}$ is a dual variable. The proposition says that the duality gap for a pair of optima $\boldsymbol{w}^\star$ and $\boldsymbol{z}^\star$ of the primal and dual problem is equal to 0. When the optimal duality gap is zero one says that *strong duality* holds.

Duality gaps are important in convex optimization because they provide an upper bound on the difference between the current value of an objective function and the optimal value which allows to set proper stopping criteria for iterative optimization algorithms. Given a current iterate $\boldsymbol{w}$, computing a duality gap requires choosing a "good" value for $\boldsymbol{z}$ (and in particular a feasible one). Given that at optimality, $\boldsymbol{z}(\boldsymbol{w}^\star) = \nabla f(\boldsymbol{w}^\star)$ is the unique solution to the dual problem, a natural choice of dual variable is $\boldsymbol{z} = \min\big(1, \frac{\lambda}{\Omega^*(\nabla f(\boldsymbol{w}))}\big)\nabla f(\boldsymbol{w})$, which reduces to $\boldsymbol{z}(\boldsymbol{w}^\star)$ at the optimum and therefore yields a zero duality gap at optimality.

Note that in most formulations that we will consider, the function f is of the form $f(\boldsymbol{w}) = \psi(\boldsymbol{X}\boldsymbol{w})$ with $\psi : \mathbb{R}^n \to \mathbb{R}$ and $\boldsymbol{X}$ a design matrix; typically, the Fenchel conjugate of ψ is easy to compute while the design matrix $\boldsymbol{X}$

makes it hard[1] to compute f^*. In that case, (1.1) can be rewritten as

$$\min_{\boldsymbol{w}\in\mathbb{R}^p,\boldsymbol{u}\in\mathbb{R}^n} \quad \psi(\boldsymbol{u}) + \lambda\,\Omega(\boldsymbol{w}) \qquad \text{s.t. } \boldsymbol{u} = \boldsymbol{X}\boldsymbol{w} \tag{1.9}$$

and equivalently as the optimization of the Lagrangian

$$\min_{\boldsymbol{w}\in\mathbb{R}^p,\boldsymbol{u}\in\mathbb{R}^n} \quad \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad \big(\psi(\boldsymbol{u}) - \lambda\boldsymbol{\alpha}^T\boldsymbol{u}\big) + \lambda\,\big(\Omega(\boldsymbol{w}) + \boldsymbol{\alpha}^T\boldsymbol{X}\boldsymbol{w}\big) \tag{1.10}$$

which is obtained by introducing the Lagrange multiplier $\boldsymbol{\alpha}$. The corresponding Fenchel dual[2] is then

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad -\psi^*(\lambda\boldsymbol{\alpha}) \quad \text{such that} \quad \Omega^*(\boldsymbol{X}^T\boldsymbol{\alpha}) \leq \lambda, \tag{1.11}$$

which does not require any inversion of $\boldsymbol{X}$.

1.2 Generic Methods

The problem defined in Eq. (1.1) is convex, as soon as both the loss f and the regularizer Ω are convex functions. In this section, we consider optimization strategies which are essentially blind to problem structure, namely subgradient descent (e.g., see Bertsekas, 1999), which is applicable under weak assumptions, and interior point methods solving reformulations such as linear programs (LP), quadratic programs (QP) or more generally, second-order cone programming (SOCP) or semidefinite programming (SDP) problems (e.g., see Boyd and Vandenberghe, 2004). The latter strategy is usually only possible with the square loss and makes use of general-purpose optimization toolboxes.

Subgradient descent. For all convex unconstrained problems, subgradient descent can be used as soon as one subgradient can be computed efficiently. In our setting, this is possible when a subgradient of the loss f, and a subgradient of the regularizer Ω can be computed. This is true for all classical settings, and leads to the following iterative algorithm

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{\alpha}{t}(\boldsymbol{s} + \lambda\boldsymbol{s}'), \text{ where } \boldsymbol{s} \in \partial f(\boldsymbol{w}_t),\ \boldsymbol{s}' \in \partial\Omega(\boldsymbol{w}_t)$$

with α a positive parameter. These updates are globally convergent. More precisely, we have, from Nesterov (2004), $F(\boldsymbol{w}_t) - \min_{\boldsymbol{w}\in\mathbb{R}^p} F(\boldsymbol{w}) = O(\frac{1}{\sqrt{t}})$.

1. It would require to compute the pseudo-inverse of $\boldsymbol{X}$.
2. Fenchel conjugacy naturally extends to this case (see for more details Borwein and Lewis, 2006, Theorem 3.3.5)

However, the convergence is in practice slow (i.e., many iterations are needed), and the solutions obtained are usually not sparse. This is to be contrasted with the proximal methods presented in the next section which are less generic but more adapted to sparse problems.

Reformulation as LP, QP, SOCP, SDP. For all the sparsity-inducing norms we consider in this chapter the corresponding regularized least-square problem can be represented by standard mathematical programming problems, all of them being SDPs, and often simpler (e.g., QP). For example, for the ℓ_1-norm regularized least-square regression, we can reformulate $\min_{w\in\mathbb{R}^p} \frac{1}{2n}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\Omega(\boldsymbol{w})$ as

$$\min_{\boldsymbol{w}_+,\boldsymbol{w}_-\in\mathbb{R}_+^p} \frac{1}{2n}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}_+ + \boldsymbol{X}\boldsymbol{w}_-\|_2^2 + \lambda(1^\top \boldsymbol{w}_+ + 1^\top \boldsymbol{w}_-)$$

which is a quadratic program. Other problems can be similarly cast (for the trace norm, see Fazel et al., 2001; Bach, 2008b).

General-purpose toolboxes can then be used, to get solutions with high precision (low duality gap). However, in the context of machine learning, this is inefficient for two reasons: (1) these toolboxes are generic and blind to problem structure and tend to be too slow, or cannot even run because of memory problems, (2) as outlined by Bottou and Bousquet (2008), high precision is not necessary for machine learning problems, and a duality gap of the order of machine precision (which would be a typical result from toolboxes) is not necessary.

1.3 Proximal Methods

1.3.1 Principle of Proximal Methods

Proximal methods are specifically tailored to optimize an objective of the form (1.1), i.e., which can be written as the sum of a generic differentiable function f with Lipschitz gradient, and a non-differentiable function $\lambda\Omega$. They have drawn increasing attention in the machine learning community, especially because of their convergence rates (optimal for the class of first-order techniques) and their ability to deal with large nonsmooth convex problems (e.g., Nesterov 2007; Beck and Teboulle 2009; Wright et al. 2009; Combettes and Pesquet 2010).

Proximal methods can be described as follows: at each iteration the

function f is linearized around the current point and a problem of the form

$$\min_{\boldsymbol{w}\in\mathbb{R}^p} \; f(\boldsymbol{w}^t)+\nabla f(\boldsymbol{w}^t)^T(\boldsymbol{w}-\boldsymbol{w}^t) + \lambda\Omega(\boldsymbol{w}) + \frac{L}{2}\|\boldsymbol{w}-\boldsymbol{w}^t\|_2^2 \tag{1.12}$$

is solved. The quadratic term, called proximal term, keeps the update in a neighborhood of the current iterate $\boldsymbol{w}^t$ where f is close to its linear approximation; $L>0$ is a parameter, which should essentially be an upper bound on the Lipschitz constant of ∇f and is typically set with a linesearch. This problem can be rewritten as

$$\min_{\boldsymbol{w}\in\mathbb{R}^p} \; \frac{1}{2}\|\boldsymbol{w}-\big(\,\boldsymbol{w}^t-\tfrac{1}{L}\nabla f(\boldsymbol{w}^t)\,\big)\|_2^2 \;+\; \tfrac{\lambda}{L}\Omega(\boldsymbol{w}). \tag{1.13}$$

It should be noted that when the nonsmooth term Ω is not present, the solution of the previous proximal problem just yields the standard gradient update rule $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \frac{1}{L}\nabla f(\boldsymbol{w}^t)$. Furthermore, if Ω is the indicator function of a set ι_C, i.e., defined by $\iota_C(x)=0$ for $x\in C$ and $\iota_C(x)=+\infty$ otherwise, then solving (1.13) yields the projected gradient update with projection on the set C. This suggests that the solution of the proximal problem provides an interesting generalization of gradient updates, and motivates the introduction of the notion of a *proximal operator* associated with the regularization term $\lambda\Omega$.

The proximal operator, which we will denote $\mathrm{Prox}_{\mu\Omega}$, was defined by Moreau (1962) as the function that maps a vector $\boldsymbol{u}\in\mathbb{R}^p$ to the unique[3] solution of

$$\min_{\boldsymbol{w}\in\mathbb{R}^p} \; \frac{1}{2}\|\boldsymbol{u}-\boldsymbol{w}\|^2 + \mu\,\Omega(\boldsymbol{w}). \tag{1.14}$$

This operator is clearly central to proximal methods since their main step consists in computing $\mathrm{Prox}_{\frac{\lambda}{L}\Omega}\big(\boldsymbol{w}^t-\frac{1}{L}\nabla f(\boldsymbol{w}^t)\big)$.

In section 1.3.3, we present analytical forms of proximal operators associated with simple norms and algorithms to compute them in some more elaborate cases.

1.3.2 Algorithms

The basic proximal algorithm uses the solution of problem (1.13) as the next update $\boldsymbol{w}^{t+1}$; however fast variants such as the accelerated algorithm presented in Nesterov (2007) or FISTA (Beck and Teboulle, 2009) maintain two variables and use them to combine the solution of (1.13) with information about previous steps. Often, an upper bound on Lipschitz constant of ∇f is

3. Since the objective is strongly convex.

not known, and even if it is, it is often better to obtain a local estimate. A suitable value for L can be obtained by iteratively increasing L by a constant factor until the condition

$$f(\boldsymbol{w}_L^\star) \leq M_f^L(\boldsymbol{w}^t, \boldsymbol{w}_L^\star) := f(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)^T(\boldsymbol{w}_L^\star - \boldsymbol{w}^t) + \tfrac{L}{2}\|\boldsymbol{w}_L^\star - \boldsymbol{w}^t\|_2^2 \quad (1.15)$$

is met, where $\boldsymbol{w}_L^\star$ denotes the solution of (1.13).

For functions f whose gradients are Lipschitz, the basic proximal algorithm has a global convergence rate in $O(\frac{1}{t})$ where t is the number of iterations of the algorithm. Accelerated algorithms like FISTA can be shown to have global convergence rate in $O(\frac{1}{t^2})$. Perhaps more importantly, both basic (ISTA) and accelerated (Nesterov, 2007) proximal methods are adaptive in the sense that if f is strongly convex — and the problem is therefore better conditioned — the convergence is actually linear (i.e., with rates in $O(C^t)$ for some constant $C < 1$; see Nesterov 2007). Finally, it should be noted that accelerated schemes are not necessarily descent algorithms, in the sense that the objective does not necessarily decrease at each iteration in spite of the global convergence properties.

1.3.3 Computing the Proximal Operator

Computing the *proximal operator efficiently* and *exactly* is crucial to enjoy the fast convergence rates of proximal methods. We therefore focus here on properties of this operator and on its computation for several sparsity inducing norms.

Dual proximal operator. In the case where Ω is a norm, by Fenchel duality, the following problem is dual (see Proposition 1.2) to problem (1.13):

$$\max_{\boldsymbol{v} \in \mathbb{R}^p} -\frac{1}{2}\left[\|\boldsymbol{v} - \boldsymbol{u}\|_2^2 - \|\boldsymbol{u}\|^2\right] \quad \text{such that} \quad \Omega^*(\boldsymbol{v}) \leq \mu. \quad (1.16)$$

Lemma 1.3 (Relation to dual proximal operator)**.** *Let $Prox_{\mu\Omega}$ be the proximal operator associated with the regularization $\mu\Omega$, where Ω is a norm, and $Proj_{\{\Omega^*(\cdot)\leq\mu\}}$ be the projector on the ball of radius μ of the dual norm Ω^*. Then $Proj_{\{\Omega^*(\cdot)\leq\mu\}}$ is the proximal operator for the dual problem (1.16) and, denoting the identity I_d, these two operators satisfy the relation*

$$Prox_{\mu\Omega} = I_d - Proj_{\{\Omega^*(\cdot)\leq\mu\}}. \quad (1.17)$$

Proof. By Proposition 1.2, if $\boldsymbol{w}^\star$ is optimal for (1.14) and $\boldsymbol{v}^\star$ is optimal for (1.16), we have[4] $-\boldsymbol{v}^\star = \nabla f(\boldsymbol{w}^\star) = \boldsymbol{w}^\star - \boldsymbol{u}$. Since $\boldsymbol{v}^\star$ is the projection of $\boldsymbol{u}$

4. The dual variable from Fenchel duality is $-\boldsymbol{v}$ in this case.

on the ball of radius μ of the norm Ω^*, the result follows.

This lemma shows that the proximal operator can always be computed as the residual of a projection on a convex set.

ℓ_1-norm regularization. Using optimality conditions for (1.16) and then (1.17) or subgradient condition (1.4) applied to (1.14), it is easy to check that $\mathrm{Proj}_{\{\|\cdot\|_\infty \leq \mu\}}$ and $\mathrm{Prox}_{\mu\|\cdot\|_1}$ respectively satisfy:

$$\big[\mathrm{Proj}_{\{\|\cdot\|_\infty \leq \mu\}}(\boldsymbol{u})\big]_j = \min\big(1, \tfrac{\mu}{|\boldsymbol{u}_j|}\big)\,\boldsymbol{u}_j \quad \text{and} \quad \big[\mathrm{Prox}_{\mu\|\cdot\|_1}(\boldsymbol{u})\big]_j = \Big(1 - \tfrac{\mu}{|\boldsymbol{u}_j|}\Big)_+ \boldsymbol{u}_j,$$

for $j \in \{1, \ldots, p\}$, with $(x)_+ := \max(x, 0)$. Note that $\mathrm{Prox}_{\mu\|\cdot\|_1}$ is componentwise the *soft-thresholding operator* of Donoho and Johnstone (1995) presented in Section 1.1.2.

ℓ_1-norm constraint. Sometimes, the ℓ_1-norm is used as a hard constraint and, in that case, the optimization problem is

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}) \quad \text{such that} \quad \|\boldsymbol{w}\|_1 \leq C$$

This problem can still be viewed as an instance of (1.1), with Ω defined by $\Omega(\boldsymbol{u}) = 0$ if $\|\boldsymbol{u}\|_1 \leq C$ and $\Omega(\boldsymbol{u}) = +\infty$ otherwise. Proximal methods thus apply and the corresponding proximal operator is the projection on the ℓ_1-ball, for which efficient pivot algorithms with linear complexity have been proposed (Brucker, 1984; Maculan and Galdino de Paula, 1989).

ℓ_1/ℓ_q-norm ("group Lasso"). If $\mathcal{G}$ is a partition of $\{1, \ldots, p\}$, the dual norm of the ℓ_1/ℓ_q norm is the $\ell_\infty/\ell_{q'}$ norm, with $\frac{1}{q} + \frac{1}{q'} = 1$. It is easy to show that the orthogonal projection on a unit $\ell_\infty/\ell_{q'}$ ball is obtained by projecting separately each subvector $\boldsymbol{u}_g$ on a unit $\ell_{q'}$-ball in $\mathbb{R}^{|g|}$. For the ℓ_1/ℓ_2-norm $\Omega : \boldsymbol{w} \mapsto \sum_{g \in \mathcal{G}} \|\boldsymbol{w}_g\|_2$ we have

$$[\mathrm{Prox}_{\mu\Omega}(\boldsymbol{u})]_g = \Big(1 - \frac{\lambda}{\|\boldsymbol{u}_g\|_2}\Big)_+ \boldsymbol{u}_g, \quad g \in \mathcal{G}.$$

This is shown easily by considering that the subgradient of the ℓ_2-norm is $\partial\|\boldsymbol{w}\|_2 = \big\{\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}\big\}$ if $\boldsymbol{w} \neq \mathbf{0}$ or $\partial\|\boldsymbol{w}\|_2 = \{\boldsymbol{z} \mid \|\boldsymbol{z}\|_2 \leq 1\}$ if $\boldsymbol{w} = 0$ and by applying the result of (1.4).

For the ℓ_1/ℓ_∞-norm, whose dual norm is the ℓ_∞/ℓ_1-norm, an efficient algorithm to compute the proximal operator is based on (1.17). Indeed this equation indicates that the proximal operator can be computed on each group g as the residual of a projection on an ℓ_1-norm ball in $\mathbb{R}^{|g|}$; the latter is done efficiently with the previously mentioned linear time algorithms.

In general, the case where groups overlap is more complicated because the

regularization is no longer separable. Nonetheless, in some cases it is still possible to compute efficiently the proximal operator.

Hierarchical ℓ_1/ℓ_q-norms. Hierarchical norms were proposed by Zhao et al. (2009). Following Jenatton et al. (2010a), we focus on the case of a norm $\Omega : \boldsymbol{w} \mapsto \sum_{g \in \mathcal{G}} \|\boldsymbol{w}_g\|_q$, with $q \in \{2, \infty\}$, where the set of groups $\mathcal{G}$ is *tree-structured*, meaning that two groups are either disjoint or one is included in the other. Let $\preceq$ be a total order such that $g_1 \preceq g_2$ if and only if either $g_1 \subset g_2$ or $g_1 \cap g_2 = \emptyset$.[5] Then, if $g_1 \preceq \ldots \preceq g_m$ with $m = |\mathcal{G}|$, and if we define Π_g as (a) the proximal operator $\boldsymbol{w}_g \mapsto \mathrm{Prox}_{\mu\|\cdot\|_q}(\boldsymbol{w}_g)$ on the subspace corresponding to group g and (b) the identity on the orthogonal, it can be shown (Jenatton et al., 2010a) that:

$$\mathrm{Prox}_{\mu\Omega} = \Pi_{g_m} \circ \ldots \circ \Pi_{g_1}. \tag{1.18}$$

In other words, the proximal operator associated with the norm can be obtained as the composition of the proximal operators associated to individual groups provided that the ordering of the groups is well chosen. Note that this result does not hold for $q \notin \{1, 2, \infty\}$.

Combined $\ell_1 + \ell_1/\ell_q$-norm ("sparse group Lasso"). The possibility of combining an ℓ_1/ℓ_q-norm that takes advantage of sparsity at the group level with an ℓ_1-norm that induces sparsity within the groups is quite natural (Friedman et al., 2010; Sprechmann et al., 2010). Such regularizations are in fact a special case of the hierarchical ℓ_1/ℓ_q-norms presented above and the corresponding proximal operator is therefore readily computed by applying first soft-thresholding and then group soft-thresholding.

Overlapping ℓ_1/ℓ_∞-norms. When the groups overlap but do not have a tree structure, computing the proximal operator has proven to be more difficult, but it can still be done efficiently when $q = \infty$. Indeed, as shown by Mairal et al. (2010), there exists a dual relation between such an operator and a quadratic min-cost flow problem on a particular graph, which can be tackled using network flow optimization techniques.

1.4 (Block) Coordinate Descent Algorithms

Coordinate descent algorithms solving ℓ_1-regularized learning problem go back to Fu (1998). They optimize (exactly or approximately) the objective with respect to one variable at a time while all others are kept fixed.

5. For a tree-structured $\mathcal{G}$ such an order exists.

1.4.1 Coordinate descent for ℓ_1-regularization

We consider first the following special case of ℓ_1-regularized problem:

$$\min_{w\in\mathbb{R}} \frac{1}{2}(w-w_0)^2 + \lambda|w| \tag{1.19}$$

As shown in (1.3), $w^\star$ can be obtained by *soft-thresholding*:

$$w^\star = \mathrm{Prox}_{\lambda|\cdot|}(w_0) := \Big(1 - \frac{\lambda}{|w_0|}\Big)_+ w_0 \tag{1.20}$$

Lasso case. In the case of the least-square loss the minimization with respect to a single coordinate can be written as

$$\min_{\boldsymbol{w}_j\in\mathbb{R}} \nabla_j f(\boldsymbol{w}^t)\,(\boldsymbol{w}_j - \boldsymbol{w}_j^t) + \frac{1}{2}\nabla^2_{jj} f(\boldsymbol{w}^t)(\boldsymbol{w}_j - \boldsymbol{w}_j^t)^2 + \lambda|\boldsymbol{w}_j|,$$

with $\nabla_j f(\boldsymbol{w}) = \boldsymbol{X}_j^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$ and $\nabla^2_{jj} f(\boldsymbol{w}) = \boldsymbol{X}_j^T\boldsymbol{X}_j$ independent of $\boldsymbol{w}$. Since the above equation is of the form (1.19), it is solved in closed form:

$$\boldsymbol{w}_j^\star = \mathrm{Prox}_{\lambda|\cdot|}\big(\,\boldsymbol{w}_j^t - \nabla_j f(\boldsymbol{w}_j^t)/\nabla^2_{jj} f\,\big). \tag{1.21}$$

In words, $\boldsymbol{w}_j^\star$ is obtained by solving the unregularized problem with respect to coordinate j and soft-thresholding the solution.

This is the update proposed in the shooting algorithm of Fu (1998), which cycles through all variables in a fixed order.[6]

An efficient implementation is obtained if the quantity $\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}$ or even better $\nabla f(\boldsymbol{w}^t) = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y}$ is kept updated.[7]

Smooth loss. For more general smooth losses, like the logistic loss, the optimization with respect to a single variable cannot be solved in closed form. It is possible to solve it numerically using a sequence of modified Newton steps as proposed by Shevade and Keerthi (2003). We present here a fast algorithm of Tseng and Yun (2009) based on solving just a quadratic approximation of f with an inexact line search at each iteration.

Given $d = \boldsymbol{w}_j^\star - \boldsymbol{w}_j^t$ where $\boldsymbol{w}_j^\star$ is the solution of (1.21), a line search is performed to choose the largest step of the form $\alpha^k d$ with $\alpha \in (0,1)$, $k \in \mathbb{N}$

6. Coordinate descent with a cyclic order is sometimes called Gauss-Seidel procedure.
7. In the former case, at each iteration, $\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}$ can be updated in $\Theta(n)$ operations if $\boldsymbol{w}_j$ changes and $\nabla_{j^{t+1}} f(\boldsymbol{w})$ can always be updated in $\Theta(n)$ operations. The complexity of one cycle is therefore $O(pn)$. However a better complexity is obtained in the latter case, provided the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is precomputed (with complexity $O(p^2n)$). Indeed $\nabla f(\boldsymbol{w}^t)$ is updated in $\Theta(p)$ iterations only if w_j does not stay at 0. Otherwise, if w_j stays at 0 the step costs $O(1)$; the complexity of one cycle is therefore $\Theta(ps)$ where s is the number of non-zero variables at the end of the cycle.

such that the following modified Armijo condition is satisfied:

$$F(\boldsymbol{w}^t + \alpha d \boldsymbol{e}_j) - F(\boldsymbol{w}^t) \leq \sigma\alpha\big(\nabla_j f(\boldsymbol{w}) d + |\boldsymbol{w}_j^t + d| - |\boldsymbol{w}_j^t|\big)$$

where $F(\boldsymbol{w}) := f(\boldsymbol{w}) + \lambda\,\Omega(\boldsymbol{w})$ and $\sigma < 1$. Tseng and Yun (2009) show that for f continuously differentiable and if H^t has uniformly upper and lower bounded spectrum, the sequence generated by the algorithm is decreasing and its cluster points are stationary points of F. It should be noted that the algorithm generalizes to other separable regularizations than the ℓ_1 norm.

Variants of coordinate descent algorithms have also been considered by Genkin et al. (2007), by Krishnapuram et al. (2005), by Wu and Lange (2008). Generalizations based on the Gauss-Southwell rule have been considered by Tseng and Yun (2009).

1.4.2 Block-coordinate descent for ℓ_1/ℓ_2 regularization

When $\Omega(\boldsymbol{w})$ is the ℓ_1/ℓ_2-norm with groups $g \in \mathcal{G}$ forming a partition of $\{1, \ldots, p\}$, the previous methods are generalized by block-coordinate descent (BCD) algorithms, and in particular the algorithm of Tseng and Yun (2009) generalizes easily to that case.

Specifically, the BCD generalization solves at each iteration a problem of the form:

$$\min_{\boldsymbol{w}_g \in \mathbb{R}^{|g|}} \nabla_g f(\boldsymbol{w}^t)^T(\boldsymbol{w}_g - \boldsymbol{w}_g^t) + \frac{1}{2}(\boldsymbol{w}_g - \boldsymbol{w}_g^t)^T H_{gg}(\boldsymbol{w}_g - \boldsymbol{w}_g^t) + \lambda\|\boldsymbol{w}_g\|_2, \quad (1.22)$$

where H_{gg} equals or approximates[8] $\nabla^2_{gg} f(\boldsymbol{w}^t)$. The above problem is solved in closed form if $H_{gg} = h_{gg} I_{|g|}$ in which case the solution $\boldsymbol{w}_g^\star$ is obtained by *group-soft-thresholding* of the Newton step:

$$\boldsymbol{w}_g^\star = \mathrm{Prox}_{\lambda\|\cdot\|_2}\big(\boldsymbol{w}_g^t - h_{gg}^{-1}\nabla_g f(\boldsymbol{w}_g^t)\big) \quad \text{with} \quad \mathrm{Prox}_{\lambda\|\cdot\|_2}(\boldsymbol{w}) = \Big(1 - \frac{\lambda}{\|\boldsymbol{w}\|_2}\Big)_+ \boldsymbol{w}.$$

In univariate learning problems regularized by the ℓ_1/ℓ_2-norm, and for the square loss, it is common to orthonormalize the set of variables belonging to a given group (Yuan and Lin, 2006; Wu and Lange, 2008), in which case it is natural to choose $H_{gg} = \nabla^2_{gg} f(\boldsymbol{w}^t) = I_{|g|}$. If H_{gg} is not a multiple of the identity, the solution of (1.22) can be found by replacing $\lambda\|\boldsymbol{w}_g\|_2$ by $\lambda'\|\boldsymbol{w}_g\|_2^2$ in (1.22), which yields an analytic solution; it is then a standard result in optimization that there exists a value of λ'– which can be found by binary search– such that the obtained solution also solves (1.22). More simply, it is

8. It is however not required to have good approximation properties of H_{gg} to obtain convergence guarantees for the algorithm.

sufficient to choose $H_{gg} = h_{gg} I_{|g|}$ with h_{gg} an approximation of the largest eigenvalue of $\nabla^2_{gg} f(\boldsymbol{w}^t)$[9].

In the case of general smooth losses, the descent direction is given by $\boldsymbol{d} = \boldsymbol{w}^\star_g - \boldsymbol{w}^t_g$ with $\boldsymbol{w}^\star_g$ as above and with a stepsize chosen to satisfy the following modified Armijo rule

$$F(\boldsymbol{w}^t + \alpha \boldsymbol{d}) - F(\boldsymbol{w}^t) \leq \sigma\alpha\big(\nabla_g f(\boldsymbol{w})^T \boldsymbol{d} + \|\boldsymbol{w}^t_g + \boldsymbol{d}\|_2 - \|\boldsymbol{w}^t_g\|\big).$$

1.5 Reweighted-ℓ_2 Algorithms

Approximating a nonsmooth or constrained optimization problem by a series of smooth unconstrained problems is common in optimization (see, e.g., Nesterov, 2005; Boyd and Vandenberghe, 2004; Nocedal and Wright, 2006). In the context of objective functions regularized by sparsity-inducing norms, it is natural to consider variational formulations of these norms in terms of squared ℓ_2-norms, since many efficient methods are available to solve ℓ_2-regularized problems (e.g., linear system solvers for least-squares regression).

In this section, we show on our motivating example of sums of ℓ_2-norms of subsets how such formulations arise (see, e.g. Pontil et al., 2007; Rakotomamonjy et al., 2008; Jenatton et al., 2010b; Daubechies et al., 2010).

Variational formulation for sums of ℓ_2-norms. A simple application of Cauchy-Schwarz inequality and the inequality $\sqrt{ab} \leq \frac{1}{2}(a+b)$ leads to

$$\begin{aligned}
\Omega(\boldsymbol{w}) &= \sum_{g \in \mathcal{G}} \|\boldsymbol{w}_g\|_2 = \frac{1}{2} \min_{\forall g \in \mathcal{G},\ \boldsymbol{\eta}_g \geqslant 0} \sum_{g \in \mathcal{G}} \Big\{ \frac{\|\boldsymbol{w}_g\|_2^2}{\boldsymbol{\eta}_g} + \boldsymbol{\eta}_g \Big\} \\
&= \frac{1}{2} \min_{\forall g \in \mathcal{G},\ \boldsymbol{\eta}_g \geqslant 0} \Big\{ \sum_{j=1}^{p} \big(\sum_{g \in \mathcal{G}, j \in g} \boldsymbol{\eta}_g^{-1} \big) \boldsymbol{w}_j^2 + \sum_{g \in \mathcal{G}} \boldsymbol{\eta}_g \Big\},
\end{aligned}$$

with equality if and only if $\forall g \in \mathcal{G},\ \boldsymbol{\eta}_g = \|\boldsymbol{w}_g\|_2$ (Pontil et al., 2007; Rakotomamonjy et al., 2008; Jenatton et al., 2010b). In the case of the ℓ_1-norm, it simplifies to $\sum_{j=1}^p |\boldsymbol{w}_j| = \frac{1}{2} \min_{\boldsymbol{\eta} \geqslant 0} \sum_{j=1}^p \big\{ \frac{\boldsymbol{w}_j^2}{\boldsymbol{\eta}_j} + \boldsymbol{\eta}_j \big\}$.

The variational formulation we have presented in the previous proposition

9. This can be done easily for joint feature selection in multi-task learning, since in that case the Hessian $\nabla^2_{gg} f(\boldsymbol{w}^t)$ is diagonal (Obozinski et al., 2009).

allows to consider the following function $H(\boldsymbol{w}, \boldsymbol{\eta})$ defined as

$$H(\boldsymbol{w}, \boldsymbol{\eta}) = f(\boldsymbol{w}) + \frac{\lambda}{2} \sum_{j=1}^{p} \Big\{ \sum_{g \in \mathcal{G}, j \in g} \boldsymbol{\eta}_g^{-1} \Big\} \boldsymbol{w}_j^2 + \frac{\lambda}{2} \sum_{g \in \mathcal{G}} \boldsymbol{\eta}_g.$$

It is jointly convex in $(\boldsymbol{w}, \boldsymbol{\eta})$; the minimization with respect to $\boldsymbol{\eta}$ can be done in closed form, and the optimium is equal to $F(\boldsymbol{w}) = f(\boldsymbol{w}) + \lambda \Omega(\boldsymbol{w})$; as for the minimization with respect to $\boldsymbol{w}$, it is a ℓ_2-regularized problem.

Unfortunately, the alternating minimization algorithm that is immediately suggested is not convergent in general, because the function H is not continuous (in particular around $\boldsymbol{\eta}$ which has zero coordinates). In order to make the algorithm convergent, two strategies are usually used:

- **Smoothing**: we can add a term of the form $\frac{\varepsilon}{2} \sum_{g \in \mathcal{G}} \boldsymbol{\eta}_g^{-1}$, which yields a joint cost function with compact level sets on the set of positive numbers. Alternating minimization algorithms are then convergent (as a consequence of general results on block coordinate descent), and have two different iterations: (1) minimization with respect to $\boldsymbol{\eta}$ in closed form, through $\boldsymbol{\eta}_g = (\|\boldsymbol{w}_g\|_2 + \varepsilon)$, and (2) minimization with respect to $\boldsymbol{w}$, which is an ℓ_2-regularized problem, which can be for example solved in closed form for the square loss. Note however, that the second problem needs not be optimized exactly at all iterations.
- **First order method in $\boldsymbol{\eta}$**: While the joint cost function $H(\boldsymbol{\eta}, \boldsymbol{w})$ is not continuous, the function $I(\boldsymbol{\eta}) = \min_{\boldsymbol{w} \in \mathbb{R}^p} H(\boldsymbol{w}, \boldsymbol{\eta})$ is continuous, and under general assumptions, continuously differentiable, and is thus amenable to first-order methods (e.g., proximal methods, gradient descent). When the groups in $\mathcal{G}$ do not overlap, one sufficient condition is that the function $f(\boldsymbol{w})$ is of the form $f(\boldsymbol{w}) = \psi(\boldsymbol{X}\boldsymbol{w})$ for $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ any matrix (typically the design matrix) and ψ a strongly convex function on $\mathbb{R}^n$. This strategy is particularly interesting when evaluating $I(\boldsymbol{\eta})$ is computationally cheap.

1.6 Working-Set Methods

Working-set algorithms address optimization problems by solving a increasing sequence of small subproblems of (1.1). The working set, that we will denote J, refers to the subset of variables involved in the optimization of these subproblems.

Working-set algorithms proceed as follows: after computing a solution to the problem restricted to the variables in J, global optimality is checked to determine whether the algorithm has to continue. If this is the case,

new variables enter the working set J according to a strategy that has to be defined. Note that we only consider *forward* algorithms, i.e., where the working set grows monotonically. In other words, there are no *backward* steps where variables would be allowed to leave the set J. Provided this assumption is met, it is easy to see that these procedures stop in a finite number of iterations.

This class of algorithms takes advantage of sparsity from a computational point of view (Lee et al., 2007; Szafranski et al., 2007; Bach, 2008a; Roth and Fischer, 2008; Obozinski et al., 2009; Jenatton et al., 2009; Schmidt and Murphy, 2010), since the subproblems that need to be solved are typically much smaller than the original one.

Working-set algorithms require three ingredients:

- **Inner-loop solver**: At each iteration of the working-set algorithm, problem (1.1) has to be solved on J, i.e., subject to the additional equality constraint that $\boldsymbol{w}_j = 0$ for all j in J^c:

$$\min_{\boldsymbol{w} \in \mathbb{R}^p} f(\boldsymbol{w}) + \lambda \Omega(\boldsymbol{w}), \text{ such that } \boldsymbol{w}_{J^c} = 0. \tag{1.23}$$

The computation can be performed by any of the methods presented in this chapter. Working-set algorithms should therefore be viewed as "meta-algorithms". Since solutions for successive working sets are typically close to each other the approach is efficient if the method chosen can use *warm-restarts*.

- **Computing the optimality conditions**: Given a solution $\boldsymbol{w}^\star$ of problem (1.23), it is then necessary to check whether $\boldsymbol{w}^\star$ is also a solution for the original problem (1.1). This test relies on the duality gaps of problems (1.23) and (1.1). In particular, if $\boldsymbol{w}^\star$ is a solution of problem (1.23), it follows from Proposition 1.2 in Section 1.1.2 that

$$f(\boldsymbol{w}^\star) + \lambda \Omega(\boldsymbol{w}^\star) + f^*(\nabla f(\boldsymbol{w}^\star)) = 0.$$

In fact, the Lagrangian parameter associated with the equality constraint ensures the feasibility of the dual variable formed from the gradient of f at $\boldsymbol{w}^\star$. In turn, this guarantees that the duality gap of problem (1.23) vanishes. The candidate $\boldsymbol{w}^\star$ is now a solution of the full problem (1.1), i.e., without the equality constraint, if and only if

$$\Omega^*(\nabla f(\boldsymbol{w}^\star)) \leq \lambda. \tag{1.24}$$

Condition (1.24) points out that the dual norm Ω^* is a key quantity to monitor the progress of the working-set algorithm (Jenatton et al., 2009). In simple settings, for instance when Ω is the ℓ_1-norm, checking

condition (1.24) can be easily computed since Ω^* is just the ℓ_∞-norm. In this case, condition (1.24) becomes

$$|[\nabla f(\boldsymbol{w}^\star)]_j| \leq \lambda, \text{ for all } j \text{ in } \{1, \ldots, p\}.$$

Note that by using the optimality of problem (1.23), the components of the gradient of f indexed by J are already guaranteed to be no greater than λ. For more general sparsity-inducing norms with overlapping groups of variables (see Section 1.1.1), the dual norm Ω^* cannot be computed easily anymore, prompting the need for approximations and upper-bounds of Ω^* (Bach, 2008a; Jenatton et al., 2009; Schmidt and Murphy, 2010).

- **Strategy for the growth of the working set**: If condition (1.24) is not satisfied for the current working set J, some inactive variables in J^c have to become active. This point raises the questions of *how many* and *how* these variables should be chosen.
First, depending on the structure of Ω, a *single* or a *group* of inactive variables have to be considered to enter the working set. Furthermore, one natural way to proceed is to look at the variables that violate condition (1.24) most. In the example of ℓ_1-regularized least squares regression with normalized predictors, this strategy amounts to selecting the inactive variable that has the highest correlation with the current residual.

The working-set algorithms we have described so far aim at solving problem (1.1) for a fixed value of the regularization parameter λ. However, for specific types of loss and regularization functions, the set of solutions of problem (1.1) can be obtained efficiently for all possible values of λ, which is the topic of the next section.

1.6.1 LARS - Homotopy

We present in this section an active-set method for solving the Lasso problem (Tibshirani, 1996) of Eq. (1.5). Active-set and working-set methods are very similar; their differ in that active-set methods allow variables returning to zero to exit the set. The problem of the Lasso is again

$$\min_{\boldsymbol{w} \in \mathbb{R}^p} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1, \tag{1.25}$$

where $\boldsymbol{y}$ is in $\mathbb{R}^n$, and $\boldsymbol{X}$ is a design matrix in $\mathbb{R}^{n \times p}$. Even though generic working-set methods introduced above could be used to solve this formulation, a specific property of the ℓ_1-norm associated with a quadratic loss makes it possible to address it more efficiently.

Under mild assumptions (which we will detail later), the solution of

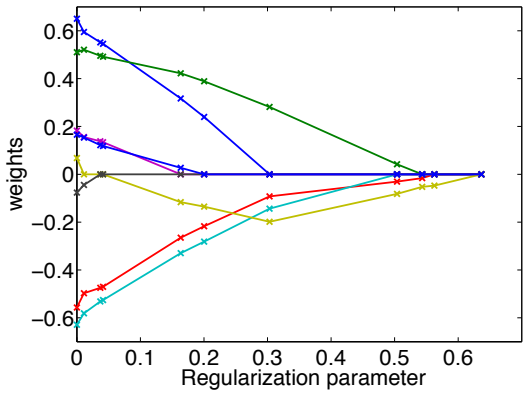


Figure 1.1: The weights $\boldsymbol{w}^\star(\lambda)$ are represented as functions of the regularization parameter λ. When λ increases, more and more coefficients are set to zero. These functions are all piecewise linear.

Eq. (1.25) is unique, and we denote it by $\boldsymbol{w}^\star(\lambda)$. We call *regularization path* the function $\lambda \mapsto \boldsymbol{w}^\star(\lambda)$ that associates to a regularization parameter λ the corresponding solution. We will show that this function is piecewise linear, a behavior illustrated in Figure 1.1, where the entries of $\boldsymbol{w}^\star(\lambda)$ for a particular instance of the Lasso are represented as functions of λ.

An efficient algorithm can thus be constructed by chosing a particular value of λ, for which finding this solution is trivial, and by following the piecewise linear path, computing the directions of the current linear parts, and the points where the direction changes (a.k.a. kinks). This piecewise linearity was first discovered and exploited by Markowitz (1952) in the context of portfolio selection, revisited by Osborne et al. (2000) describing an *homotopy* algorithm, and popularized by Efron et al. (2004) with the LARS algorithm.

Let us show how to construct the path. From the optimality conditions we have presented in Eq. (1.6), denoting by $J := \{j; |\boldsymbol{X}_j^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}^\star)| = \lambda\}$ the set of active variables, and defining the vector $\boldsymbol{\epsilon}$ in $\{-1; 0; 1\}^p$ as $\boldsymbol{\epsilon} := \operatorname{sgn}\big(\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}^\star)\big)$, we have the following closed form

$$\begin{cases} \boldsymbol{w}_J^\star(\lambda) & = (\boldsymbol{X}_J^T\boldsymbol{X}_J)^{-1}(\boldsymbol{X}_J^T\boldsymbol{y} - \lambda\boldsymbol{\epsilon}_J) \\ \boldsymbol{w}_{J^c}^\star(\lambda) & = 0, \end{cases}$$

where we have assumed the matrix $\boldsymbol{X}_J^T\boldsymbol{X}_J$ to be invertible (which is a sufficient condition to guarantee the uniqueness of $\boldsymbol{w}^\star$). This is an important point: if one knows in advance the set J and the signs $\boldsymbol{\epsilon}_J$, then $\boldsymbol{w}^\star(\lambda)$ admits a simple closed-form. Moreover, when J and $\boldsymbol{\epsilon}_J$ are fixed, the function

$\lambda \mapsto (\boldsymbol{X}_J^T \boldsymbol{X}_J)^{-1}(\boldsymbol{X}_J^T \boldsymbol{y} - \lambda \boldsymbol{\epsilon}_J)$ is affine in λ. With this observation in hand, we can now present the main steps of the path-following algorithm. It basically starts from a trivial solution of the regularization path, follows the path by exploiting this formula, updating J and $\boldsymbol{\epsilon}_J$ whenever needed so that optimality conditions (1.6) remain satisfied. This procedure requires some assumptions—namely that (a) the matrix $\boldsymbol{X}_J^T \boldsymbol{X}_J$ is always invertible, and (b) that updating J along the path consists of adding or removing from this set a single variable at the same time. Concretely, we proceed as follows

1. Set λ to $\|\boldsymbol{X}^T \boldsymbol{y}\|_\infty$ for which it is easy to show from Eq. (1.6) that $\boldsymbol{w}^\star(\lambda) = 0$ (trivial solution on the regularization path).

2. Set $J := \{j; |\boldsymbol{X}_j^T \boldsymbol{y}| = \lambda\}$.

3. Follow the regularization path by decreasing the value of λ, with the formula $\boldsymbol{w}_J^\star(\lambda) = (\boldsymbol{X}_J^T \boldsymbol{X}_J)^{-1}(\boldsymbol{X}_J^T \boldsymbol{y} - \lambda \boldsymbol{\epsilon}_J)$ keeping $\boldsymbol{w}_{J^c}^\star = 0$, until one of the following events occur

   - There exists j in J^c such that $|\boldsymbol{X}_j^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}^\star)| = \lambda$. Then, add j to the set J.
   - There exists j in J such that a non-zero coefficient $\boldsymbol{w}_j^\star$ hits zero. Then, remove j from J.

We suppose that only one of such events can occur at the same time (b). It is also easy to show that the value of λ corresponding to the next event can be obtained in closed form.

4. Go back to 3.

Let us now briefly discuss assumptions (a) and (b). When the matrix $\boldsymbol{X}_J^T \boldsymbol{X}_J$ is not invertible, the regularization path is non-unique, and the algorithm fails. This can easily be fixed by addressing instead a slightly modified formulation. It is possible to consider instead the elastic-net formulation of Zou and Hastie (2005) that uses $\Omega(\boldsymbol{w}) = \lambda\|\boldsymbol{w}\|_1 + \frac{\gamma}{2}\|\boldsymbol{w}\|_2^2$. Indeed, it amounts to replacing the matrix $\boldsymbol{X}_J^T \boldsymbol{X}_J$ by $\boldsymbol{X}_J^T \boldsymbol{X}_J + \gamma \boldsymbol{I}$, which is positive definite and therefore always invertible, even with a small value for γ, and to apply the same algorithm. in practice. The second assumption (b) can be unsatisfied in practice because of the precision machine. To the best of our knowledge, the algorithm will fail in such cases, but we consider this scenario unlikely with real data.

The complexity of the above procedure depends on the number of kinks of the regularization path (which is also the number of iterations of the algorithm). Even though it is possible to build examples where this number is large, we often observe in practice that the event where one variable gets out of the active set is rare. The complexity also depends on the implemen-

tation. By maintaining the computations of $\boldsymbol{X}_j^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}^\star)$ and a Cholesky decomposition of $(\boldsymbol{X}_J^T\boldsymbol{X}_J)^{-1}$, it is possible to obtain an implementation in $O(psn + ps^2 + s^3)$ operations, where s is the sparsity of the solution when the algorithm is stopped (which we approximately consider as equal to the number of iterations). The product psn corresponds to the computation of the matrices $\boldsymbol{X}_J^T\boldsymbol{X}_J$, ps^2 to the updates of the correlations $\boldsymbol{X}_j^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}^\star)$ along the path, and s^3 to the Cholesky decomposition.

1.7 Quantitative Evaluation

To illustrate and compare the methods presented in this chapter, we consider in this section three benchmarks. These benchmarks are chosen to be representative of problems regularised with sparsity-inducing norms, involving different norms and different loss functions. To make comparisons that are as fair as possible, each algorithm is implemented in `C/C++`, using efficient BLAS and LAPACK libraries for basic linear algebra operations. All subsequent simulations are run on a single core of a 3.07Ghz CPU, with 8GB of memory. In addition, we take into account several criteria which strongly influence the convergence speed of the algorithms. In particular, we consider (a) different problem scales, (b) different levels of correlations, (c) different strengths of regularization. We also show the influence of the required precision by monitoring the time of computation as a function of the objective function.

For the convenience of the reader, we list here the algorithms compared and the acronyms we use to refer to them throughout this section: the LARS algorithm (LARS), coordinate-descent (CD), reweighted-ℓ_2 schemes (Re-ℓ_2), simple proximal method (ISTA) and its accelerated version (FISTA); we will also include in the comparisons generic algorithms such as a subgradient descent algorithm (SG), and a commercial software (Mosek) for cone (CP), quadratic (QP) and second-order cone programming (SOCP) problems.

1.7.1 Speed Benchmarks

We first present a large benchmark evaluating the performance of various optimization methods for solving the Lasso.

We perform small-scale ($n = 200, p = 200$) and medium-scale ($n = 2000, p = 10000$) experiments. We generate design matrices as follows. For the scenario with low correlation, all entries of $\boldsymbol{X}$ are independently drawn from a Gaussian distribution $\mathcal{N}(0, 1/n)$, which is a setting often used to evaluate optimization algorithms in the literature. For the scenario with

large correlation, we draw the rows of the matrix $\boldsymbol{X}$ from a multivariate Gaussian distribution for which the *average absolute value* of the correlation between two different columns is eight times the one of the scenario with low correlation. Test data vectors $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{n}$ where $\boldsymbol{w}$ are randomly generated, with two levels of sparsity to be used with the two different levels of regularization. $\boldsymbol{n}$ is a noise vector whose entries are i.i.d. samples from a Gaussian distribution $\mathcal{N}(0, 0.01\|\boldsymbol{X}\boldsymbol{w}\|_2^2/n)$. In the low regularization setting the sparsity of the vectors $\boldsymbol{w}$ is $s = 0.5\min(n, p)$, and in the high regularization one $s = 0.01\min(n, p)$, corresponding to fairly sparse vectors. For SG, we take the step size to be equal to $a/(k+b)$, where k is the iteration number, and (a, b) are the best[10] parameters selected on a logarithmic grid $(a, b) \in \{10^3, \dots, 10\} \times \{10^2, 10^3, 10^4\}$; we proceeded this way not to disadvantage SG by an arbitrary choice of stepsize.

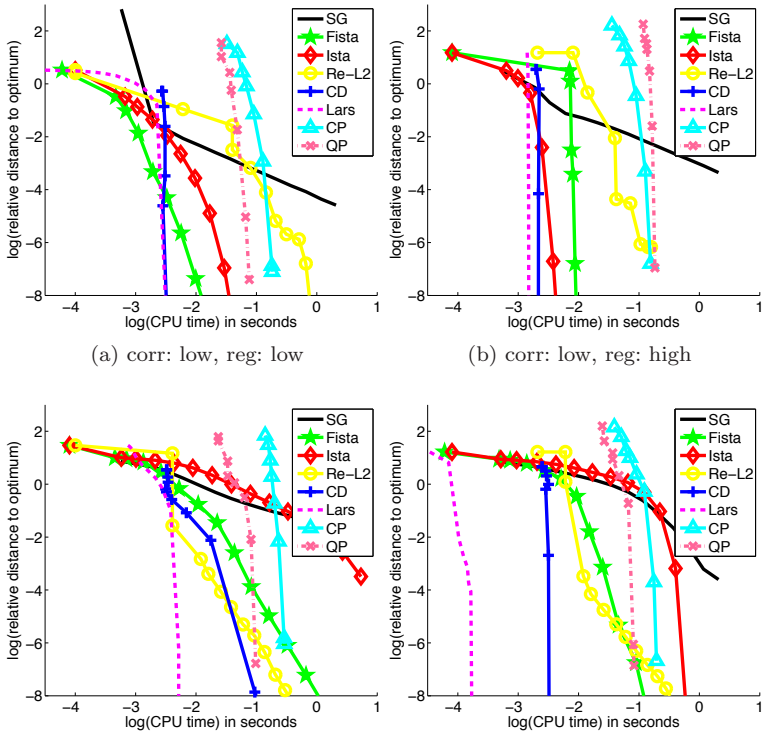
To sum up, we make a comparison for 8 different conditions (2 scales $\times$ 2 levels of correlation $\times$ 2 levels of regularization). All results are reported on figures 1.2, 1.3, by averaging 5 runs for each experiment. Interestingly, we observe that the relative performance of the different methods change significantly with the scenario.

Our conclusions for the different methods are as follows:

- **LARS**: For the small-scale problem, LARS outperforms all other methods for almost every scenario and precision regime. It is therefore *definitely the right choice for the small-scale setting.*
Unlike first-order methods, its performance does not depend on the correlation of the design matrix $\boldsymbol{X}$, but rather on the sparsity s of the solution. In our larger scale setting, it has been competitive either when the solution is *very sparse* (high regularization), or when there is *high correlation* in $\boldsymbol{X}$ (in that case, other methods do not perform as well). More importantly, LARS gives an exact solution and computes the regularization path.

- **Proximal methods (ISTA, FISTA)**: FISTA outperforms ISTA in all scenarios but one. Both methods are close for high regularization or low correlation, but FISTA is significantly better for high correlation or/and low regularization. These methods are almost always outperformed by LARS in the small-scale setting, except for *low precision and low correlation.*
Both methods *suffer from correlated features*, which is consistent with the fact that their convergence rate is proportional to the Lipschitz constant of the gradient of f, which grows with the amount of correlation. They are *well adapted to large-scale settings, with low or medium correlation.*

10. "The best step size" is understood here as being the step size leading to the smallest objective function after 500 iterations.



(a) corr: low, reg: low (b) corr: low, reg: high

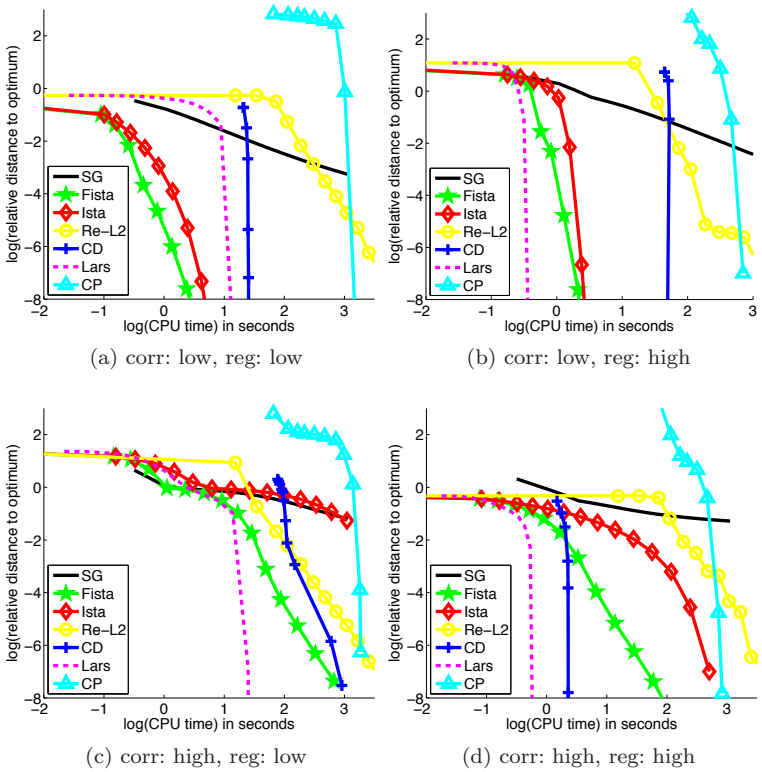
(c) corr: high, reg: low (d) corr: high, reg: high

Figure 1.2: Benchmark for solving the Lasso for the small-scale experiment ($n = 200$, $p = 200$), for the two levels of correlation and two levels of regularization, and 8 optimization methods (see main text for details). The curves represent the relative value of the objective function as a function of the computational time in second on a $\log_{10} / \log_{10}$ scale.

- **Coordinate descent (CD)**: To the best of our knowledge, no theoretical convergence rate is available for this method. Empirically, we have observed that the behavior of CD often translates into a first "warm-up" stage followed by a fast convergence phase.
Its performance in the *small-scale setting is competitive* (even though always behind LARS), but *less efficient in the large-scale one*. For a reason we can not explain, *it suffers less than proximal methods from correlated features.*

- **Reweighted-ℓ_2**: This method was outperformed in all our experiments by other dedicated methods.[11] Note that we considered only the smoothed

11. Note that the reweighted-ℓ_2 scheme requires solving iteratively large-scale linear system that are badly conditioned. Our implementation uses LAPACK Cholesky decompositions, but a better performance might be obtained using a pre-conditioned conjugate



(a) corr: low, reg: low (b) corr: low, reg: high

(c) corr: high, reg: low (d) corr: high, reg: high

Figure 1.3: Benchmark for solving the Lasso for the medium-scale experiment $n = 2000,\ p = 10000$, for the two levels of correlation and two levels of regularization, and 8 optimization methods (see main text for details). The curves represent the relative value of the objective function as a function of the computational time in second on a $\log_{10} / \log_{10}$ scale.

alternating scheme of Section 1.5 and not first order methods in $\boldsymbol{\eta}$ such as that of Rakotomamonjy et al. (2008). A more exhaustive comparison should include these as well.

- **Generic Methods (SG, QP, CP)**: As expected, generic methods are not adapted for solving the Lasso and are always outperformed by dedicated ones such as LARS.

Among the methods that we have presented, some require an overhead computation of the Gram matrix $\boldsymbol{X}^T\boldsymbol{X}$: this is the case for coordinate descent and reweighted-ℓ_2 methods. We took into account this overhead time in all figures, which explains the behavior of the corresponding convergence

gradient, especially in the very large scale setting.

curves. Like the LARS, these methods could also benefit from an offline pre-computation of $\boldsymbol{X}^T\boldsymbol{X}$ and would therefore be more competitive if the solutions corresponding to several values of the regularization parameter have to computed.

We have considered in the above experiments the case of the square loss. Obviously, some of the conclusions drawn above would not be valid for other smooth losses. On the one hand, the LARS does no longer apply; on the other hand, proximal methods are clearly still available and coordinate descent schemes, which were dominated by the LARS in our experiments, would most likely turn out to be very good contenders in that setting.

1.7.2 Structured Sparsity

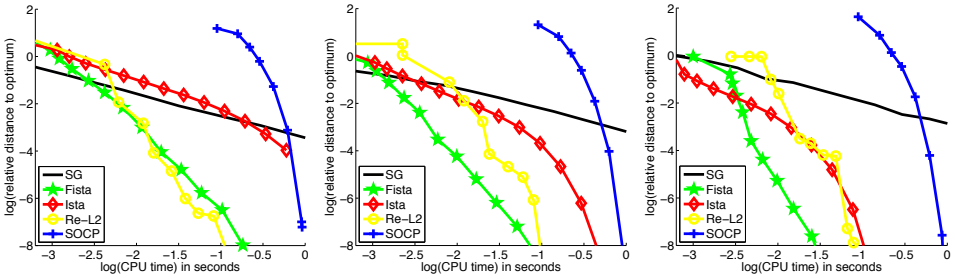
In this second series of experiments, the optimization techniques of the previous sections are further evaluated when applied to other types of loss and sparsity-inducing functions. Instead of the ℓ_1-norm previously studied, we focus on the particular *hierarchical* ℓ_1/ℓ_2-norm Ω introduced in Section 1.3. From an optimization standpoint, although Ω shares some similarities with the ℓ_1-norm (e.g., the convexity and the non-smoothness), it differs in that it cannot be decomposed into independent parts (because of the overlapping structure of $\mathcal{G}$). CD schemes hinge on this property and as a result, cannot be straightforwardly applied in this case.

1.7.2.1 Denoising of natural image patches

In this first benchmark, we consider a least-squares regression problem regularized by Ω that arises in the context of the denoising of natural image patches (Jenatton et al., 2010a). In particular, based on a hierarchical set of features that accounts for different types of edge orientations and frequencies in natural images, we seek to reconstruct noisy 16×16-patches. Although the problem involves a small number of variables (namely $p = 151$), it has to be solved repeatedly for thousands of patches, at moderate precision. It is therefore crucial to be able to solve this problem efficiently.

The algorithms that take part in the comparisons are ISTA, FISTA, Re-ℓ_2, SG, and SOCP. All results are reported in Figure 1.4, by averaging 5 runs.

We can draw several conclusions from the simulations. First, we observe that across all levels of sparsity, the accelerated proximal scheme performs better, or similarly, than the other approaches. In addition, as opposed to FISTA, ISTA seems to suffer in non-sparse scenarios. In the least sparse setting, the reweighted-ℓ_2 scheme matches the performance of FISTA. However this scheme does not yield truly sparse solutions, and would therefore



(a) scale: small, regul: low (b) scale: small, regul: medium (c) scale: small, regul: high

Figure 1.4: Benchmark for solving a least-squares regression problem regularized by the hierarchical norm Ω. The experiment is small scale, $n = 256, p = 151$, and shows the performances of five optimization methods (see main text for details) for three levels of regularization. The curves represent the relative value of the objective function as a function of the computational time in second on a $\log_{10} / \log_{10}$ scale.

require a subsequent thresholding operation, which can be difficult to motivate in a principled way. As expected, the generic techniques such as SG and SOCP do not compete with the dedicated algorithms.

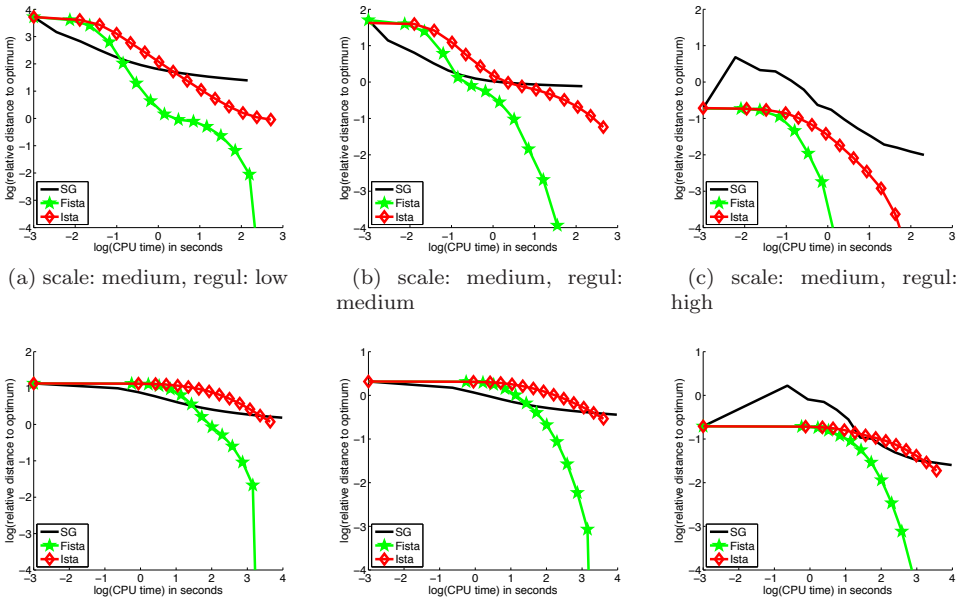
1.7.2.2 Multi-class classification of cancer diagnosis.

The second benchmark involves two datasets[12] of gene expressions in the context of cancer diagnosis. More precisely, we focus on two multi-class classification problems in the "small n, large p" setting. The medium-scale dataset contains $n = 83$ observations, $p = 4615$ variables and 4 classes, while the large-scale one contains $n = 308$ samples, $p = 30017$ variables and 26 classes. In addition, both datasets exhibit highly-correlated features. Inspired by Kim and Xing (2010), we build a tree-structured set of groups $\mathcal{G}$ by applying Ward's hierarchical clustering (Johnson, 1967) on the gene expressions. The norm Ω built that way aims at capturing the hierarchical structure of gene expression networks (Kim and Xing, 2010).

Instead of the square loss function, we consider the multinomial logistic loss function, which is better suited for multi-class classification problems. As a direct consequence, the algorithms whose applicability crucially depends on the choice of the loss function are removed from the benchmark. This is for instance the case for reweighted-ℓ_2 schemes that have closed-form updates available only with the square loss (see Section 1.5). Importantly,

12. The two datasets we use are *SRBCT* and *14_Tumors*, which are freely available at `http://www.gems-system.org/`.

the choice of the multinomial logistic loss function requires to optimize over a matrix with dimensions p times the number of classes (i.e., a total of $4615 \times 4 \approx 18\,000$ and $30017 \times 26 \approx 780\,000$ variables). Also, for lack of scalability, generic interior point solvers could not be considered here. To summarize, the following comparisons involve ISTA, FISTA, and SG.



(a) scale: medium, regul: low (b) scale: medium, regul: medium (c) scale: medium, regul: high

(d) scale: large, regul: low (e) scale: large, regul: medium (f) scale: large, regul: high

Figure 1.5: Medium- and large-scale multi-class classification problems for three optimization methods (see details about the datasets and the methods in the main text). Three levels of regularization are considered. The curves represent the relative value of the objective function as a function of the computation time in second on a $\log_{10} / \log_{10}$ scale. In the highly regularized setting, the tuning of the step-size for the subgradient turned out to be difficult, which explains the behavior of SG in the first iterations.

All the results are reported in Figure 1.5. The benchmark especially points out that the accelerated proximal scheme performs overall better that the two other methods. Again, it is important to note that both proximal algorithms yield sparse solutions, which is not the case for SG. More generally, this experiment illustrates the flexibility of proximal algorithms with respect to the choice of the loss function.

We conclude this section by a couple of general remarks on the experiments that we presented. First, the use of proximal methods is often advocated because of their optimal worst case complexities in $O(\frac{1}{k^2})$. In practice,

in our experiments, these and several other methods exhibit empirically convergence rates that are at least linear, if not better, which suggests that the adaptivity of the method (e.g., its ability to take advantage of local curvature) might be more crucial to its practical success. Second, our experiments concentrated on regimes that are of interest for sparse methods in machine learning where typically p is larger than n and where it possible to find good sparse solutions. The setting where n is much larger than p was out of scope here, but would be worth a separate study, and should involve methods from stochastic optimization. Also, even though it might make sense from an optimization viewpoint, we did not consider problems with low levels of sparsity, that is with more dense solution vectors, since it would be a more difficult regime for many of the algorithms that we presented (namely LARS, CD or proximal methods).

1.8 Extensions

We obviously could not cover exhaustively the literature on algorithms for sparse methods in this chapter.

Surveys and comparisons of algorithms for sparse methods have been proposed by Schmidt et al. (2007) and Yuan et al. (2010). These papers present quite a few algorithms, but focus essentially on ℓ_1-regularization and unfortunately do not consider proximal methods. Also, it is not clear that the metrics used to compare the performance of various algorithms is the most relevant to machine learning; in particular, we present the full convergence curves that we believe are more informative than the ordering of algorithms at fixed precision.

Beyond the material presented here, there a few topics that we did not develop and that are worth mentioning.

In terms of norms, we did not consider regularization by the nuclear norm, also known as the trace norm, which seeks low-rank matrix solutions (Fazel et al., 2001; Srebro et al., 2005; Recht et al., 2007; Bach, 2008b). Most of the optimization techniques that we presented do however apply to this norm (with the exception of coordinate descent).

In terms of algorithms, it is possible to relax the smoothness assumptions that we made on the loss. For instance, some proximal methods are applicable with weaker smoothness assumptions on the function f, such as the Douglas-Rachford algorithm (see details in Combettes and Pesquet, 2010). The related augmented Lagrangian techniques (Glowinski and Le Tallec, 1989; Combettes and Pesquet, 2010, and numerous references therein), also known as alternating-direction methods of multipliers are also relevant in

that setting. These methods are in particular applicable to cases where several regularizations are mixed.

In the context of proximal methods, the metric used to define the proximal operator can be (1) modified by judicious rescaling operations, in order to fit better the geometry of the data (Duchi et al., 2010), or even (2) substituted with norms associated with functional spaces, in the context of kernel methods (Rosasco et al., 2009).

Finally, from a broader outlook, our —*a priori* deterministic— optimization problem (1.1) may also be tackled with stochastic optimization approaches, which has been the focus of much research (Bottou, 1998; Bottou and LeCun, 2004; Shapiro et al., 2009).

1.9 Conclusion

We presented and compared four families of algorithms for sparse methods: proximal methods, block-coordinate descent algorithms, reweighted-ℓ_2 algorithms and the LARS that are representative of the state-of-the-art. We did not aim at being exhaustive. The properties of these methods can be summarized as follows:

- Proximal methods provide efficient and scalable algorithms that are applicable to a wide family of loss functions, that are simple to implement, compatible with many sparsity-inducing norms and often competitive with the other methods considered.
- For the square loss, the LARS remains the fastest algorithm for (a) small and medium scale problems, since its complexity depends essentially on the size of the active sets, (b) cases with very correlated designs. It computes the whole path up to a certain sparsity level.
- For smooth losses, block-coordinate descent provides one of the fastest algorithms but it is limited to separable regularizers.
- For the square-loss and possibly sophisticated sparsity inducing regularizers, ℓ_2-reweighted algorithms provides generic algorithms, that are still pretty competitive compared to subgradient and interior point methods. For general losses, these methods currently require to solve iteratively ℓ_2-regularized problems and it would be desirable to relax this constraint.

References

F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Neural Information Processing Systems*, volume 21, 2008a.

F. Bach. Consistency of trace norm minimization. *Journal of Machine Learning Research*, 9:1019–1048, 2008b.

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

D. P. Bertsekas. *Nonlinear programming*. Athena Scientific Belmont, 1999.

P. Bickel, Y. Ritov, and A. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Annals of Statistics*, 37(4):1705–1732, 2009.

J. M. Borwein and A. S. Lewis. *Convex analysis and nonlinear optimization: Theory and Examples*. Springer-Verlag, 2006.

L. Bottou. Online algorithms and stochastic approximations. *Online Learning and Neural Networks*, 5, 1998.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Neural Information Processing Systems*, volume 20, pages 161–168, 2008.

L. Bottou and Y. LeCun. Large scale online learning. In *Advances in Neural Information Processing Systems*, volume 16, pages 217–224, 2004.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. ISBN 0521833787.

P. Brucker. An $O(n)$ algorithm for quadratic knapsack problems. *Operations Research Letters*, 3(3):163–166, 1984.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1999.

P.L. Combettes and J.C. Pesquet. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, chapter Proximal Splitting Methods in Signal Processing. New York: Springer-Verlag, 2010.

I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, 2010.

D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90 (432):1200–1224, 1995.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. Technical report, 2010.

B. Efron, T. Hastie, and R. Johnstone, I.and Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings American Control Conference*, volume 6, pages 4734–4739, 2001.

J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *preprint*, 2010.

W. J Fu. Penalized regressions: the bridge versus the lasso. *Journal of computational and graphical statistics*, 7(3):397–416, 1998.

A. Genkin, D. D Lewis, and D. Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

R. Glowinski and P. Le Tallec. *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*. Society for Industrial Mathematics, 1989.

J. Huang and T. Zhang. The benefit of group sparsity. Technical report, 2009. Preprint arXiv:0901.2962.

L. Jacob, G. Obozinski, and J.-P. Vert. Group Lasso with overlaps and graph Lasso. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

R. Jenatton, J-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. Technical report, 2009. Preprint arXiv:0904.3523v1.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010a.

R. Jenatton, G. Obozinski, and F. Bach. Structured sparse principal component analysis. In *Proceedings of International Workshop on Artificial Intelligence and Statistics*, 2010b.

S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *Proc. Intl. Conf. Machine Learning*, 2010.

K. Koh, S. J. Kim, and S. Boyd. An Interior-Point Method for Large-Scale l 1-Regularized Logistic Regression. *Journal of Machine Learning Research*, 8:1555, 2007.

B. Krishnapuram, L. Carin, et al. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 957–968, 2005.

H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms.

In *Neural Information Processing Systems*, volume 20, 2007.

K. Lounici, M. Pontil, A. B. Tsybakov, and S. van de Geer. Taking advantage of sparsity in multi-task learning. Technical report, Preprint arXiv:0903.1468, 2009.

N. Maculan and G. Galdino de Paula. A linear-time median-finding algorithm for projecting a vector on the simplex of $\mathbb{R}^n$. *Operations research letters*, 8(4):219–222, 1989.

J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. In *Neural Information Processing Systems*, 2010.

H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

J.J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *CR Acad. Sci. Paris Sér. A Math*, 255:2897–2899, 1962.

S. Negahban, P. Ravikumar, M. J. Wainwright, and B. Yu. A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers. In *Neural Information Processing Systems*, 2009.

Y. Nesterov. *Introductory lectures on convex optimization: a basic course*. Kluwer Academic Publishers, 2004.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, Tech. Rep, 2007.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Verlag, 2006. second edition.

G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):231–252, 2009.

M. R. Osborne, B. Presnell, and B. A. Turlach. On the Lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–37, 2000.

M. Pontil, A. Argyriou, and T. Evgeniou. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, 2007.

A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization. Technical report, Preprint arXiv:0706.4138, 2007.

R. T. Rockafellar. *Convex analysis*. Princeton University Press, 1997.

L. Rosasco, A. Verri, M. Santoro, S. Mosci, and S. Villa. Iterative Projection Methods for Structured Sparsity Regularization. Technical report, CBCL-282, 2009.

V. Roth and B. Fischer. The Group-Lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *Proc. Intl. Conf. Machine Learning*, 2008.

M. Schmidt and K. Murphy. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proceedings of International Workshop on Artificial Intelligence and Statistics*, 2010.

M. Schmidt, G. Fung, and R. Rosales. Fast optimization methods for L1 regularization: A comparative study and two new approaches. *Machine Learning: ECML 2007*, pages 286–297, 2007.

A. Shapiro, D. Dentcheva, A. Ruszczyński, and A. P. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. Society for Industrial Mathematics, 2009.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

S. K. Shevade and S. S. Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246, 2003.

P. Sprechmann, I. Ramirez, G. Sapiro, and Y.C. Eldar. Collaborative Hierarchical Sparse Modeling. *Arxiv preprint arXiv:1003.0400*, 2010.

N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Neural Information Processing Systems*, volume 17, pages 1329–1336, 2005.

M. Szafranski, Y. Grandvalet, and P. Morizet-Mahoudeaux. Hierarchical penalization. In *Neural Information Processing Systems*, volume 20, 2007.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *J. Roy. Stat. Soc. B*, 58(1):267–288, 1996.

J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Signal Processing*, 50(10):2231–2242, October 2004.

P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1):387–423, 2009.

M. J. Wainwright. Sharp thresholds for noisy and high-dimensional recovery of sparsity using ℓ_1- constrained quadratic programming. *IEEE transactions on information theory*, 55(5):2183, 2009.

S.J. Wright, R.D. Nowak, and M.A.T. Figueiredo. Sparse reconstruction by

separable approximation. *IEEE Transactions on Signal Processing*, 57(7): 2479–2493, 2009.

T. T Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *Annals*, 2(1):224–244, 2008.

G.X. Yuan, K.W. Chang, C.J. Hsieh, and C.J. Lin. A comparison of optimization methods for large-scale l1-regularized linear classification. Technical report, Department of Computer Science, National University of Taiwan, 2010.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Roy. Stat. Soc. B*, 68:49–67, 2006.

P. Zhao and B. Yu. On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Annals of Statistics*, 37(6A): 3468–3497, 2009.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B*, 67(2):301–320, 2005.