Nonsmoothness in Machine Learning: specific structure, proximal identification, and applications

Franck Iutzeler, Jérôme Malick

▶ To cite this version:

Franck Iutzeler, Jérôme Malick. Nonsmoothness in Machine Learning: specific structure, proximal identification, and applications. Set-Valued and Variational Analysis, 2020, 28 (4), pp.661-678. 10.1007/s11228-020-00561-1 . hal-02953985v2

HAL Id: hal-02953985

https://hal.science/hal-02953985v2

Submitted on 3 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Set-Valued and Variational Analysis manuscript No.
(will be inserted by the editor)

Nonsmoothness in Machine Learning: specific structure, proximal identification, and applications

Franck Iutzeler · Jérôme Malick

November 3, 2020

Abstract Nonsmoothness is often a curse for optimization; but it is sometimes a blessing, in particular for applications in machine learning. In this paper, we present the specific structure of nonsmooth optimization problems appearing in machine learning and illustrate how to leverage this structure in practice, for compression, acceleration, or dimension reduction. We pay a special attention to the presentation to make it concise and easily accessible, with both simple examples and general results.

1 Introduction

Optimization is at the core of machine learning and nonsmoothness plays there a special role. Often, nonsmooth functions are introduced in learning problems to enforce low-complexity of the optimal model parameters. This promoted structure (e.g. sparsity, low-rank, or controlled variation) turns out to be progressively identified by proximal algorithms; most interestingly, it can be leveraged numerically to improve optimization methods. This situation contrasts with a large part of the optimization literature where dealing with nonsmooth functions or checking nonsmooth substructure is usually uneasy.

In this paper, we lay out a generic framework of optimization problems for which nonsmoothness can be exploited, covering most machine learning problems with low-complexity priors. We then go over the main technical tools that enable to mathematically and practically handle nonsmooth structures.

F. Iutzeler
Univ. Grenoble Alpes

J. Malick
CNRS, LJK

We pay a special attention to being accessible for a wide audience[1] in optimization, including graduate students in applied maths. No specific knowledge on machine learning or nonsmooth analysis is necessary to follow our developments, but fundamentals in mathematical optimization are required. Let us mention that we will make a constant (but basic) use of the proximal operator which is a key tool to deal with explicit nonsmoothness. For a function $g\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and a parameter $\gamma > 0$, we define $\mathbf{prox}_{\gamma g}(u)$ for any $u \in \mathbb{R}^n$ as the optimal solution of the following problem:

$$\mathbf{prox}_{\gamma g}(u) := \arg\min_{y\in\mathbb{R}^n} \left\{ g(y) + \frac{1}{2\gamma}\|y-u\|_2^2 \right\}. \tag{prox}$$

This properly defines an operator $\mathbf{prox}_{\gamma g}\colon \mathbb{R}^n \to \mathbb{R}^n$ in many cases, in particular when g is convex. We refer to the textbooks [31] and [6] or to the review [43] for more information and an historical perspective.

The rest of the paper is organized as follows. In Section 2, we explain why nonsmooth models are considered in machine learning. In Section 3, we formalize a large class of nonsmooth optimization problems for which structure can be exploited. Then, in Section 4, we discuss the proximal optimization algorithms that can harness this structure. In Section 5, we lay out the mathematical grounds that enable to properly identify structure. Finally, in Section 6, we review recent advances in the use of identified structure to improve performances of some optimization algorithms in machine learning.

2 Nonsmooth problems in Machine Learning

Standard machine learning tasks such as regression, classification, or clustering (see e.g the textbook [48]) lead to optimization problems. Indeed, many objectives in supervised learning can write as minimizing some *risk* function R measuring the dissimilarity between a learning model, obtained from statistical modeling, with parameters $x \in \mathbb{R}^n$, and a set of m examples $\{a_i, b_i\}_{i=1}^m$, in the form of an input (a_i) and target (b_i) pair. This problem, often called Empirical Risk Minimization (ERM), has the following form:

$$\min_{x\in\mathbb{R}^n} \; R\left(x; \{a_i, b_i\}_{i=1}^m\right). \tag{ERM}$$

Often, the risk R is taken as the average of some loss over the training examples: $R\left(x; \{a_i, b_i\}_{i=1}^m\right) = \frac{1}{m}\sum_{i=1}^m \ell(p(x;a_i); b_i)$ where $p(x;a)$ is the model prediction for input a using parameters x, and $\ell(p;b)$ quantifies the error between the predicted point p and the true target b. The simplest case is when the prediction function is linear $(p(x;a) = \langle x; a\rangle)$ and the error is quadratic $(\ell(p;b) = (p-b)^2)$ leading to the well known least-squares regression problem.

[1] Advanced readers can jump directly to Sections 5 (about general identification) and 6 (with entry points to recent research on this topic).

Other popular choices for the model include polynomial and gaussian kernels, or neural networks. The error functions often depend either on (i) the statistical modelling of the problem through the (log) likelihood (squared 2-norm for linear regression of points corrupted by a Gaussian noise, 1-norm for Laplace noise); or (ii) directly from applications (errors in classification).

With the ever-growing collection of data, the size of learning problems has significantly increased, both in terms of number of examples, m, as well as in size of optimized parameter, n. While the increase in number of examples is generally beneficial for the general conditioning of the problem, the increase of the parameter space often makes the problem ill-conditioned and reduces both the interpretability and stability of the model. In order to overcome this issue, a generally admitted solution is to introduce a *prior* on the *structure* of the model x and to *regularize* the (ERM) in order to promote the prior structure. One of the first and most well-known instances of such a prior is the sparsity sought in the parameters of least-squares regression, leading to the well known lasso problem [51]. More generally, regularizing a learning problem to enforce a prior structure conforms to the following steps:

(i) *Define a prior structure.* Let us observe the usual priors in machine learning: sparsity, constant by block, fixed rank, etc. They can be described as subsets of $\mathbb{R}^n$ that are rather easy to describe and project on. As mathematical objects, and to comply with a vast part of the literature, we shall see them as (affine or smooth) manifolds, but most of the arguments in this paper hold for simple closed sets.

(ii) *Find an additive nonsmooth function r enforcing this structure.* Nonsmoothness of functions *traps* optimal solutions in low-dimensional manifolds: small perturbations in the risk around these points would not break down optimal structure as illustrated by Figure 1. Mathematically, this is due to subdifferentials with non-empty relative interiors in relevant directions. For instance, the subdifferential of the ℓ_1-norm has a nonempty relative interior along the axes, promoting sparsity.

At this point, a regularized version of (ERM) can be formulated as

$$\min_{x\in\mathbb{R}^n} \quad \underbrace{R\left(x;\{a_i,b_i\}_{i=1}^m\right)}_{\substack{=:f(x) \\ \text{minimizes the risk}}} \quad + \quad \underbrace{\lambda\, r(x)}_{\substack{=:g(x) \\ \text{enforces structure}}} \quad . \tag{Regularized ERM}$$

To make it more concrete, let us consider the popular example of ℓ_1-regularized least-squares problems (often called lasso [51]): take a linear prediction function, a quadratic loss, and the ℓ_1-norm as a regularizer, then the regularized problem becomes

$$\min_{x\in\mathbb{R}^n} \quad \frac{1}{2}\|A\,x - b\|_2^2 \ + \ \lambda\,\|x\|_1 \ . \tag{lasso}$$

Optimal solutions are then sparser than the one of the original least-squares, and their sparsity pattern is stable under small perturbations; see Figure 1.

In general, the regularized risk minimization has thus a composite form, involving two different parts: f which is linked to the risk itself, and g which only promotes structure, with a hyperparameter $\lambda > 0$ controlling the balance between f and g. This particular formulation enables one to use ad-hoc optimization methods which forms the third part of the learning process:

(iii) *Design an optimization method complying with this composite formulation.* The structure-enforcing part $g = \lambda\, r$ is necessarily nonsmooth but it may be possible to choose it so that its proximity operator (prox) can be computed easily (with an explicit expression or through a computationally cheap procedure). When this is the case, proximal methods (i.e. optimization methods involving at least one proximity operator) are the algorithms of choice to solve (Regularized ERM). Indeed, contrary to subgradient methods, the proximity operator enables taking fixed stepsizes and is thus much faster in both theory and practice. We will come back to them in Section 4.

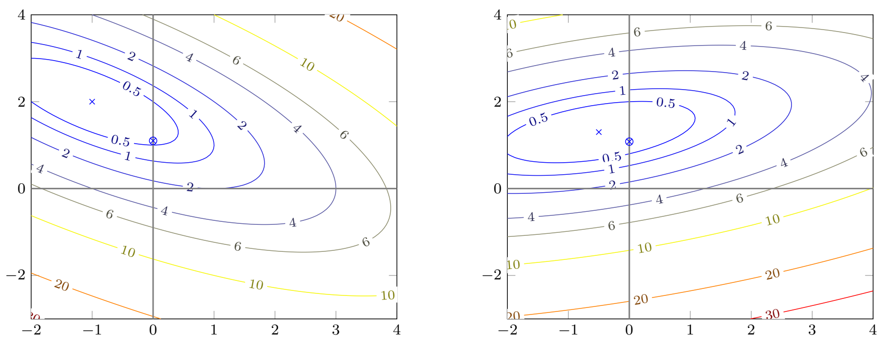


Fig. 1: Illustration of the stability of optimal solutions of lasso in $\mathbb{R}^2$. We plot the level sets of $1/2\|Ax - b\|_2^2$ for two (lasso) problems with different but close design matrices A. We see that while the solutions of the (unregularized) least-squares problem (marked by a x) are different, the solutions of the lasso (marked by a circled cross), although different, lie on the same axis, corresponding to the nonsmoothness loci of the ℓ_1-norm.

To summarize, when solving a machine learning problem, it is common to assume some prior structure and enforce it by adding a regularization term to the (ERM). In the next section, we will see that this prior structure often takes a special and simple form in relation with the proximity operator of g. In this context, we will later explain how proximal methods solving (Regularized ERM) automatically uncover the optimal structure (or at least part of it), and how we can take advantage of this behavior to improve the performance of the algorithms themselves.

3 Noticeable Structure in nonsmooth optimization problems

The type of nonsmoothness encountered in machine learning objectives is often linked to user-defined priors, as in (Regularized ERM). In this case, the nonsmoothness is under control: it is chosen and relatively simple. In this section, we introduce the notion of Noticeable Structure (NS) to describe this favorable class of problems for which structure can be characterized.

Framework NS (Problems with Noticeable Structure)**.** *For a function g and a finite collection*[2] $\mathsf{C} = \{\mathcal{M}_1, \ldots, \mathcal{M}_p\}$ *of closed sets, we say that the couple* (g, C) *has noticeable structure if*

1. *we have a projection mapping* $\mathrm{proj}_{\mathcal{M}_i}$ *onto* $\mathcal{M}_i$ *for all* i *as well as for any intersection* $\mathcal{W}_{\mathcal{I}} = \cap_{i\in\mathcal{I}}\mathcal{M}_i$ *for* $\mathcal{I} \subseteq \{1, .., p\}$*;*
2. g *is non-differentiable at* x *only if* $x \in \mathcal{M}_i$ *for some* i*;*
3. $\mathbf{prox}_{\gamma g}(u)$ *is a singleton and can be computed explicitly for any* u *and* γ*;*
4. *upon computation of* $x = \mathbf{prox}_{\gamma g}(u)$*, we know if* $x \in \mathcal{M}_i$ *or not for all* i*.*

Before providing examples, let us discuss these four properties. Items 1 and 2 are important to harness the structure numerically. The first item simply means that it is computationally possible to project onto any set (or intersection of sets) present in the collection; this is to enable algorithms to harness the iterates structure by updating preferably along the identified manifold (see Sec. 6.3). Item 2 permits localization the non-differentiability and as a consequence better grasp differentiability (outside of the manifolds and along manifolds[3]). Item 3 is typically satisfied for all proper convex lower semi-continuous functions and for some classes of non-convex functions (e.g. for prox-bounded, prox-regular functions [45, Prop. 13.37]). Note though that (generalized) convexity will not be required for most of our developments. Item 4 is essential: to be able to exploit some structural information, the user needs to *know* the current structure. This condition is not so stringent since for many popular structure-enforcing functions, an explicit proximity operator is known, often based on switch-cases that directly determine the output structure.

In the remainder of this section, we detail some popular examples of structures and regularizers[4] falling into the framework (NS) often considered as priors in machine learning [4].

[2] We mention that the sets in the collection $\mathsf{C} = \{\mathcal{M}_1, \ldots, \mathcal{M}_p\}$ are denoted by $\mathcal{M}_i$ since in most of the literature these sets are described as manifolds. The manifold structure will only be used when needed. However, the closedness of the sets is primordial. Its finiteness is less essential but is used to describe properly the structure of a neighborhood points (see the proof of Theorem 1).

[3] In many examples, the nonsmooth function g turns out to be smooth with respect to smooth (sub)manifolds of the collection. We will come back to this relative smoothness of the problem in Section 6.3.1.

[4] In the following examples, we take $\lambda = 1$ out of simplicity since in that case $g \equiv r$. This does not change the discussion on proximity operators since there are explicit formulas for scaling, translation, etc. [8, Th. 6.1].

3.1 Sparse structure with ℓ_1-norm

The natural manner to look at the sparsity pattern of a vector is to look at each coordinate and see if it is null or not. In terms of collection of manifolds, we consider

$$\mathsf{C} = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\} \quad \text{with} \quad \mathcal{M}_i = \{x \in \mathbb{R}^n : x_i = 0\}.$$

First, the projections on the $\mathcal{M}_i$'s are direct (NS-1). Taking $g(x) = \|x\|_1$, its non-differentiability points exactly match the described manifolds (NS-2).

The proximity operator of the ℓ_1-norm is the well-known soft-thresholding operator which can be described coordinate-wise as

$$[x]_i = \Big[\mathbf{prox}_{\gamma\|\cdot\|_1}(u)\Big]_i = \begin{cases} 0 & \text{if } [u]_i \in [-\gamma, \gamma] \\ [u]_i - \gamma & \text{if } [u]_i > \gamma \\ [u]_i + \gamma & \text{if } [u]_i < -\gamma \end{cases}$$

with $[u]_i \in \mathbb{R}$ denoting the i-th coordinate of vector $u \in \mathbb{R}^n$. This proximity operator is thus explicitly computable for any u (NS-3). Finally, after completing the tests on the right hand side above, the structure of the output x is perfectly known (NS-4).

For other types of sparsity-inducing functions and structure (e.g. ℓ_1/ℓ_2 or group sparsity), see [3]. Note that these functions do not need to be convex as discussed in the following remark.

Remark 1 (Sparser structure with non-convex functions) The ℓ_1-norm is sometimes seen as a convex approximation of the ℓ_0-"norm". Although $g(x) = \|x\|_0 = \operatorname{Card}\{i : [x]_i \neq 0\}$ is not a norm and is non-convex, it is quite easy to see that it fits the framework (NS), with the same collection and the proximal operator

$$[x]_i = \Big[\mathbf{prox}_{\gamma\|\cdot\|_0}(u)\Big]_i = \begin{cases} 0 & \text{if } [u]_i \in [-\gamma, \gamma] \\ [u]_i & \text{if } |[u]_i| > \gamma \end{cases},$$

sometimes called the hard-thresholding operator.

For completeness, notice that all the functions $g(x) = \|x\|_p^p$ for $p \in [0, 1]$ satisfy (NS)-1 & 2 and are thus sparsity-inducing. However, the only explicitly computable proximity operators are for $p = 0, 0.5, 2/3, 1$ [12], they are thus the only ones matching (NS)-3 & 4. Finally, only $p = 1$ (presented above) leads to a convex function. $\square$

3.2 Flat structure with total variation

The denoising of piecewise-constant one-dimensional[5] signals has attracted a lot of attention, notably thanks to the introduction of the total variation

[5] The 2D problem of matrix regression with flat regions is much harder to define and to solve, see e.g. [15] and references therein.

function $g(x) = \sum_{i=2}^{n} |[x]_i - [x]_{i-1}|$ [47]. The structure of piecewise constant signals can be described using the following collection:

$$\mathsf{C} = \{\mathcal{M}_1, \ldots, \mathcal{M}_{n-1}\} \quad \text{with} \quad \mathcal{M}_i = \{x \in \mathbb{R}^n : x_i = x_{i-1}\},$$

and once again (NS)-1 & 2 are easily satisfied. The computation of the proximity operator is less direct than before and does not benefit from a closed form solution, however, there are efficient dynamic programming algorithms computing it exactly and enabling to know exactly the structure of the output, see [14] for details. Hence, (NS)-3 & 4 are also satisfied.

Remark 2 (More structure with non-convex functions) In the same vein as before, the total variation can be seen as a convex relaxation of $g(x) = \text{Card}(\{i : [x]_i \neq [x]_{i-1}\})$, sometimes called the Potts problem [54]. (NS)-1 & 2 are then easily satisfied, while 3 & 4 are less direct but the proximity operator can also be obtained by dynamic programming [24]. □

3.3 Low rank with nuclear norm

One of the most prominent types of structure sought in matrix-values problems is low-rank. Indeed, it is widely used (notably stemming from PCA and spike models) in order to summarize the information in the matrix to its most informative subspaces. It also pops out in matrix factorization and matrix completion problems, with applications in recommender systems. Naturally, the collection writes

$$\mathsf{C} = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\} \quad \text{with} \quad \mathcal{M}_i = \{\text{rank}(X) = i\}$$

and projection onto the manifolds can be obtained numerically by truncating the singular value decomposition (for NS 1).

To express the rank of a matrix $X \in \mathbb{R}^{r \times c}$, it is natural to examine its singular values $(\sigma_1, \ldots, \sigma_n)$ (where $n = \min\{r, c\}$). A direct way to promote low-rank matrices is through nuclear norm regularization, which is the ℓ_1-norm of the singular values vector: $g(X) = \|X\|_* = \sum_{i=1}^{n} \sigma_i$. One can show that the proximal operator $X = \mathbf{prox}_{\gamma\|\cdot\|_*}(U)$ of a matrix $U = V\text{diag}(\sigma_i)W^*$ can be obtained by

$$X = V\text{diag}(\nu_i)W^* \text{ with } \nu_i = \mathbf{prox}_{\gamma|\cdot|}(\sigma_i) = \begin{cases} 0 & \text{if } \sigma_i \in [-\gamma, \gamma] \\ \sigma_i - \gamma & \text{if } \sigma_i > \gamma \\ \sigma_i + \gamma & \text{if } \sigma_i < -\gamma \end{cases}$$

which matches NS 3 & 4.

A remarkable point is that even though two close matrices may have completely different singular vectors, their singular values will be close (this is sometimes called Weyl's lemma, see [55,49]). Hence, while a new singular value decomposition has to be computed at each application of the proximity operator of the nuclear norm, the rank of the output will be somewhat stable to small perturbations, which is of utmost importance when using sparsity.

Thus, even though the problem of checking the rank of a matrix can be problematic numerically, the simplicity of the proximity operator (modulo the computation of the SVD) enables it to fall into our framework (NS); especially, the rank of the output of the proximity operator is known by construction.

Finally, note that as in the previous case, the "ℓ_0-equivalent" of the nuclear norm is simply the rank whose proximity operator involves a hard thresholding of the singular values. Finally, for completeness, several types of matrix structures falling into our framework can be found in [9].

4 Proximal algorithms

From an optimization point of view, regularized empirical risk minimization (see (Regularized ERM) in Section 2) is often seen as a composite problem:

$$\min_{x\in\mathbb{R}^n} \; f(x) + g(x) \qquad (\mathcal{P})$$

where g induces a Noticeable Structure (NS) as per the framework introduced in the previous section.

The study of optimization algorithms associated with proximity operators, notably in relation with monotone operators [46] and splitting methods [20], has attracted a considerable lot of attention in the optimization community since the 70s; see e.g. [13,43] for reviews of the essential points for signal processing and machine learning applications. These proximal algorithms are written quite generally as follows with iterations consisting of an Update step, that defines the algorithm, followed by a proximity operation:

$$\begin{cases} u_{k+1} = \mathsf{Update} \\ x_{k+1} = \mathbf{prox}_{\gamma g}(u_{k+1}) \end{cases} \qquad (\mathbf{prox} - \mathsf{Update})$$

The main purpose of the Update step is to handle f which is the part of the problem not managed by $\mathbf{prox}_{\gamma g}$. To do so, it can use the properties of f as well as previous iterates.

We will see in the next section that such proximal methods are the most generic algorithms that can harness nonsmoothness in the framework (NS). The only prerequisite is that these methods converge (or converge almost surely, when randomness is involved), which we formalize in the next assumption.

Assumption 1. *The method* ($\mathbf{prox} - \mathsf{Update}$) *is such that*

1. (u_k) *converges (almost surely) to a point* $u^\star$,
2. (x_k) *converges (almost surely) to a point* $x^\star$,
3. $x^\star = \mathbf{prox}_{\gamma g}(u^\star)$ *is a minimizer of* $(\mathcal{P})$.

Let us list some popular proximal algorithms solving ($\mathcal{P}$) with g convex[6] lower semi-continuous, that satisfy Assumption 1:

– *Proximal gradient*: for f convex with an L-Lipchitz gradient

$$\begin{cases} u_{k+1} = x_k - \gamma \nabla f(x_k) \\ x_{k+1} = \mathbf{prox}_{\gamma g}(u_{k+1}) \end{cases} \tag{PG}$$

satisfies Assumption 1 for $\gamma \in (0, 2/L)$ [8, Th. 10.24].

– *Accelerated Proximal Gradient*: for f convex with an L-Lipchitz gradient,

$$\begin{cases} u_{k+1} = x_k + \alpha_k(x_k - x_{k-1}) - \gamma \nabla f(x_k + \alpha_k(x_k - x_{k-1})) \\ x_{k+1} = \mathbf{prox}_{\gamma g}(u_{k+1}) \end{cases} \tag{APG}$$

with $\alpha_k = (k-1)/(k+3)$ satisfies Assumption 1 for $\gamma \in (0, 1/L]$ [11, Th. 3].

– *Douglas-Rachford*: for f convex lower semi-continuous but not necessarily differentiable, the proximity operator of f can be used with a splitting[7] algorithm such as Douglas-Rachford

$$\begin{cases} u_{k+1} = \mathbf{prox}_{\gamma f}(2x_k - u_k) + u_k - x_k \\ x_{k+1} = \mathbf{prox}_{\gamma g}(u_{k+1}) \end{cases} \tag{DR}$$

which satisfies Assumption 1 for any $\gamma > 0$ [6, Cor. 27.4].

– *Variance-Reduced Incremental Methods*: for $f(x) = \frac{1}{m}\sum_{j=1}^{m} f^j(x)$ with each (f^j) strongly convex with an L-Lipchitz gradient,

$$\begin{cases} u_{k+1} = x_k - \gamma\Big(\nabla f^{i_k}(x_k) - \nabla f^{i_k}(x_{k-d_k^{i_k}}) + \displaystyle\sum_{j=1}^{m} \nabla f^j(x_{k-d_k^j})\Big) \\ x_{k+1} = \mathbf{prox}_{\gamma g}(u_{k+1}) \end{cases} \tag{SAGA}$$

where i_k is drawn uniformly in $\{1, \ldots, m\}$ and $k - d_k^j$ is the last time before k when ∇f^j was computed (i.e. for which $i_{k-d_k^j} = j$). SAGA satisfies Assumption 1 for $\gamma = 1/(3L)$ [17, Cor. 1].

[6] In the case where g is convex and lower semi-continuous, points *1* and *3* imply point *2* and are actually sufficient for convergence. Observe indeed that $\|x_k - x^\star\| = \|\mathbf{prox}_{\gamma g}(x_k) - \mathbf{prox}_{\gamma g}(u^\star)\| \le \|u_k - u^\star\|$; see more in [6, Chap. 23]. In non-convex cases, the situation is more complex, all three points are necessary and can be investigated by looking at the prox-regularity and prox-boundedness of g at a local minimizer $x^\star$; see [30] and references therein.

[7] A splitting method is necessary here to separate the two proximity operators, since they cannot be directly composed [57] or averaged [7].

- *Asynchronous Proximal Gradient*: for $f(x) = \frac{1}{m}\sum_{j=1}^{m} f^j(x)$ with each (f^j) μ-strongly convex with an L-Lipchitz gradient,

$$\begin{cases} u_{k+1} = \dfrac{1}{m}\displaystyle\sum_{j=1}^{m}\Big(x_{k-d_k^j} - \gamma\nabla f^j(x_{k-d_k^j})\Big) \\ x_{k+1} = \mathbf{prox}_{\gamma g}(u_{k+1}) \end{cases} \tag{DAve-PG}$$

  where an iteration is performed as soon as an oracle (say i) finishes computing a gradient $(\nabla f^i(x_{k-d_k^i}))$; then, u_{k+1}, x_{k+1} are updated. The time $k - d_k^j$ is then the iteration of the last point for which j has sent a response. Provided that no oracle completely stops working, DAve-PG satisfies Assumption 1 for $\gamma \in (0, 2/(\mu+L)]$ [38, Th. 3.2].

This inventory, far from complete, gives an idea of the diversity of favorable proximal methods. We end this section by mentioning methods that *do not satisfy* Assumption 1; they mostly fall into two categories:

- *Stochastic versions of proximal gradient.* There are situations where the gradient is obtained via a stochastic oracle g_k satisfying $\mathbb{E}[g_k] = \nabla f(x_k)$ and $\mathbb{E}[\|g_k - \nabla f(x_k)\|^2] \leq \sigma^2$. The proximal stochastic gradient, obtained by replacing the gradient in (PG) by such stochastic one, may not recover the structure of the problem; see [44, Sec. 1.3]. This case indeed fails to satisfy Assumption 1: there is convergence only in probability and not almost surely (or any other types of weaker convergence giving localization guarantees).

- *Inexact proximal methods.* When the proximal operator is not cheap, the step $x_{k+1} = \mathbf{prox}_{\gamma g}(u_{k+1})$ may be computed only approximately (see e.g. [46]). The scheme ($\mathbf{prox} - \mathsf{Update}$) is not exactly satisfied and thus fall out of the scope of the present paper. In addition, inexact methods may not recover the structure of the optimization problem.

5 Proximal Identification

Proximal methods are the natural algorithms for solving regularized learning problems. The convergence analysis of these algorithms is well-known and studied in the references given in the previous section. Less-known and studied is the qualitative behavior (beyond convergence) of these methods. In this section, we question where the iterates of proximal methods are located and show that they reach some near-optimal structure after some finite time: this is the *identification* property of proximal methods.

5.1 A generic identification result

The observation that the iterates produced by optimization algorithms reach some structure dates can be traced back to e.g. [56] for constrained problems and [29] for nonsmooth minimization. Our presentation here is original regarding its simplicity and generality. For additional details on identification, including early references, we point to [21] and references therein.

In order to properly study structure membership, let us define the *sparsity vector* that describes the structure of a point $x \in \mathbb{R}^n$ relatively to the collection $\mathsf{C} = \{\mathcal{M}_1, \ldots, \mathcal{M}_p\}$ introduced in the framework (NS), as the vector $\mathsf{S}(x) \in \{0,1\}^p$ defined as

$$[\mathsf{S}(x)]_i = 0 \quad \text{if } x \in \mathcal{M}_i \text{ and } 1 \text{ elsewhere.}$$

An identification theorem is a result describing the eventual structure of the iterates of an algorithm, i.e. the value of $\mathsf{S}(x_k)$ for large enough k, with respect to the optimal point $x^\star$. We provide below an elementary but powerful enough identification theorem.

Theorem 1 (Enlarged identification) *Let Assumption 1 hold. Then, the iterates (x_k) produced by (**prox** − Update) identify some manifolds in the collection of the framework (NS)*[8]*. More precisely, for each $\varepsilon > 0$ there is a K such that for all $k > K$ we have, with probability one, for all i*

$$[\mathsf{S}(x^\star)]_i \;\leq\; [\mathsf{S}(x_k)]_i \;\leq\; \max\left\{[\mathsf{S}\big(\mathbf{prox}_{\gamma g}(u)\big)]_i : \|u - u^\star\| \leq \varepsilon\right\}$$

where $x^\star$ is the minimizer of $(\mathcal{P})$ to which (x_k) converges and $u^\star$ is defined as in Assumption 1.

Before going to the proof, we note that a slightly more precise result is shown there: the right part $[\mathsf{S}(x_k)]_i \leq \max\left\{[\mathsf{S}\big(\mathbf{prox}_{\gamma g}(u)\big)]_i : \|u - u^\star\| \leq \varepsilon\right\}$ actually holds as soon as $\|u_k - u^\star\| \leq \varepsilon$.

Proof. For the right part of the result, since $u_k \to u^\star$ almost surely from Assumption 1, we have that for any $\varepsilon > 0$, $\|u_k - u^\star\| \leq \varepsilon$ for k large enough with probability one. This directly gives the right hand side of the inequality. Then, $\mathsf{S}\big(\mathbf{prox}_{\gamma g}(u_k)\big) \leq \max\left\{\mathsf{S}\big(\mathbf{prox}_{\gamma g}(u)\big) : \|u - u^\star\| \leq \varepsilon\right\}$.

The left part of the inequality is more interesting. Consider the collection of sets to which $x^\star$ belongs $\mathsf{C}^\star = \{\mathcal{M}_i \in \mathsf{C} : x^\star \in \mathcal{M}_i\}$ and its complementary collection $\overline{\mathsf{C}^\star} = \mathsf{C} \setminus \mathsf{C}^\star$. Since $\mathcal{M}^\circ := \bigcup\{\mathcal{M} \in \overline{\mathsf{C}^\star}\}$ is a closed set (given that it is a finite[9] union of closed sets), its complementary set $\mathbb{R}^n \setminus \mathcal{M}^\circ$ is open.

[8] For simplicity, we place ourselves within the framework (NS), because it is needed to exploit identification in practice. However, the result uses only existence and uniqueness of the proximal mapping (first part of item 3 in the definition of framework (NS)).

[9] This argument is the main reason why finiteness is assumed in the framework (NS). For countably many sets in the collection, $\mathcal{M}^\circ$ may not be closed (it is then a F_σ set).

Since $x^\star \in \mathbb{R}^n \setminus \mathcal{M}^\circ$ by definition of $\mathsf{C}^\star$, there exists a ball of radius $\varepsilon' > 0$ around $x^\star$ that is included in $\mathbb{R}^n \setminus \mathcal{M}^\circ$. Hence, as $x^k \to x^\star$ almost surely, it will reach this ball in finite time with probability one and thus belong to fewer subspaces than $x^\star$. No point x in that ball belongs to $\mathcal{M}^\circ$ which means that $\mathsf{S}(x^\star) \leq \mathsf{S}(x)$ coordinate-wise. □

This rather generic result demonstrates that the iterates of proximal methods partially identify some structure as soon as u_k is close enough to $u^\star$. After a finite but unknown number of iterations, we have:

– *less structure than* $x^\star$. The left-hand inequality guarantees that all the identified manifolds contain the optimal manifolds, but possibly others (which means that less structure is identified). Note that this does not depend on the algorithm used.
– *some structure.* The identified structure is not random: it is controlled by the right-hand-side term, which encompasses how the structure changes around the pair $(x^\star, u^\star)$. This upper-bound is in general impossible to evaluate a priori (for an exception, see [19]). Intuitively, its size captures the difficulty of reaching the structure of $x^\star$; see Figure 2.

Note that the structure mentioned here is relative to the solution of $(\mathcal{P})$ to which the algorithm is converging. If there are multiple solutions to the problem, they may have different sparsity patterns, but identification still holds.

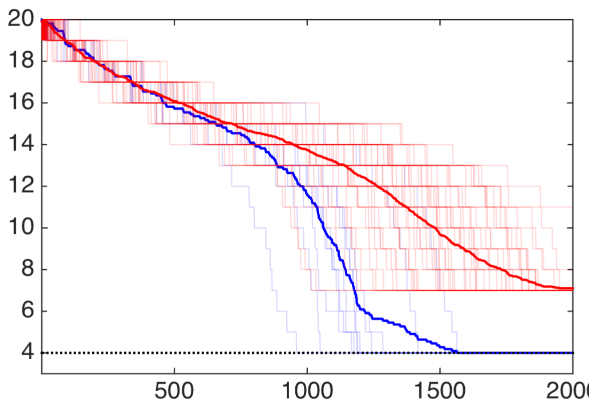


Fig. 2: Illustration of the enlarged identification of the proximal gradient algorithm solving matrix least-squares problems regularized by nuclear norm. We generate many random problems for matrices of size 20×20 and with optimal solutions of rank 4. We plot the decrease of the rank of the iterates for two groups of problems: in blue the problems are well-posed and the algorithm identifies the optimal rank; in red the problems are degenerate and the algorithm identifies something bigger. Each curve is a trajectory and the bold curves are the averaged trajectory for the two groups.

Theorem 1 frames the structure of the iterates; sometimes the terminology *enlarged* or *approximate* identification is used. However, there is no reason for $\mathsf{S}(x_k)$ to be asymptotically: stable, monotonous, or close to $\mathsf{S}(x^\star)$ (or to the right bound), as evidenced numerically in [21,22,5]. This may pose some problems in practice if one uses $\mathsf{S}(x_k)$ as a proxy for the optimal structure. Fortunately, the family of problems for which the identification is *exact*, i.e. $\mathsf{S}(x_k) = \mathsf{S}(x^\star)$, can be characterized, as explained in the next section.

Remark 3 (Relation with Screening) Finding near-optimal structures is an important part of machine learning problems. For instance, when $g = \|\cdot\|_1$ (for instance in the lasso problem [51]), identification brings information about the nullity of the coordinates of the problem solution, enabling feature selection and/or dimension reduction [52]. Discarding null features when solving sparse learning problems is often called *screening*. With the present notation, this means knowing (an upper-bound on) $\mathsf{S}(x^\star)$ before convergence in order to discard null features and optimizing on the non-zeros entries. Theorem 1 implies that $\mathsf{S}(x_k)$ is a valid upper-bound, but after some unknown time.

Observe though that, following the same arguments as in the proof of Theorem 1, we have $\mathsf{S}(x^\star) \leq \max\{\mathsf{S}(\mathbf{prox}_{\gamma g}(u)) : u \in \mathcal{X}\}$ for any set $\mathcal{X} \ni u^\star$. Determining such a *safe region* $\mathcal{X}$ [26] (for instance using duality [23]) and computing the max gives a so-called *safe screening rule*. While this may be computationally intractable in many cases, this proved to be very efficient for solving the the lasso problem and other sparse learning models; for additional details, see e.g. [34,40,35]. $\square$

5.2 Exact identification under a qualifying constraint

The general enlarged identification result of Theorem 1 can be easily strengthened to an exact version, guaranteeing $\mathsf{S}(x_k) = \mathsf{S}(x^\star)$ after some time. A sufficient condition for exact identification is that the proximity operator maps all points u close to $u^\star$ to a point with the same structure as $x^\star$:

$$\exists\, \varepsilon > 0 \text{ such that } \quad \mathsf{S}(x^\star) = \max\left\{\mathsf{S}(\mathbf{prox}_{\gamma g}(u)) : \|u - u^\star\| \leq \varepsilon\right\} \qquad \text{(QC)}$$

Note that this qualifying condition only depends on the optimal pair $(x^\star, u^\star)$ and the proximity operator of function g.

Corollary 1 (Exact identification) *Let Assumption 1 hold; suppose that* $(\mathcal{P})$ *has a unique minimizer*[10] *and that* (QC) *holds true. Then, the iterates* (x_k)

[10] In Theorem 1, the result holds for *the* minimizer to which the algorithm converges. Here, for simplicity, we assume furthermore uniqueness of the minimizer. Nevertheless, if there are multiple minimizers and condition (QC) holds for all pairs $(x^\star, u^\star)$, the structure of all optimal points can be proved to be the same under mild conditions. For instance, if f is differentiable and ∇f is L-Lipchitz continuous on the set of optimal solutions $X^\star$, then at optimality $x^\star = \mathbf{prox}_{\gamma g}(u^\star)$ with $u^\star = x^\star - \gamma \nabla f(x^\star)$. If (QC) holds for some $\varepsilon > 0$, take two solutions $x_1^\star, x_2^\star \in X^\star$ such that $\|x_1^\star - x_2^\star\| \leq \varepsilon/(1+\gamma L)$; then, $\|u_1^\star - u_2^\star\| \leq \varepsilon$ and thus $\mathsf{S}(x_1^\star) = \mathsf{S}(x_2^\star)$ by (QC). Then, since $\mathsf{S}(x^\star)$ is the same for all minimizers, Corollary 1 holds.

produced by (**prox** − Update) *identify the optimal structure in the framework* (NS)*. More precisely, for k large enough, we have*

$$\mathsf{S}(x_k) = \mathsf{S}(x^\star) \qquad \textit{with probability } 1$$

where $x^\star$ is the minimizer of ($\mathcal{P}$) *and $u^\star$ is defined as in Assumption 1.*

Remark 4 (Identification time) As mentioned previously, identification time is generally unknown. However, if (QC) holds for some known $\varepsilon > 0$, then any u_k such $\|u_k - u^\star\| \leq \varepsilon$ will lead to a x_k satisfying $\mathsf{S}(x_k) = \mathsf{S}(x^\star)$. Using the convergence rate of the algorithm, an estimate of the identification time can be given. For instance, if a proximal method satisfies $\|u_k - u^\star\| = \mathcal{O}(1/k)$, then we have $\mathsf{S}(x_k) = \mathsf{S}(x^\star)$ after $\mathcal{O}(1/\varepsilon)$ iterations. This rationale is used in [25, 42, 50]. □

Remark 5 (Link between the qualifying constraint and partial smoothness) Condition (QC) depends only weakly on the optimal structure, which is in contrast with part of identification literature that considers g partly smooth [29, 33] relative to the final manifold. As shown in [5, Apx. A], if g is partly smooth relative to the manifold $\mathcal{M}^\star := \bigcap\{\mathcal{M}_i \in \mathsf{C} : x^\star \in \mathcal{M}_i\}$ and the condition $(u^\star - x^\star)/\gamma \in \operatorname{ri} \partial g(x^\star)$ is satisfied, then (QC) holds. Thus the assumption is a consequence of the "irrepresentable condition" which is a classical assumption in machine learning; see e.g. [58] and [2]. Note finally, that this condition simplifies to $-\nabla f(x^\star) \in \operatorname{ri} \partial g(x^\star)$ for the proximal gradient method (PG) (and most derivatives such as (APG), (DAve-PG)). □

6 Nonsmoothness can help computationally

Nonsmooth regularizations are used in machine learning and signal processing for the nice *recovery* or *consistency* properties[11] that they induce; see e.g. [53] and recall Figure 1. The point of this paper is to advocate that nonsmoothness can also be useful to improve the optimization algorithms used in these applications. This is what we illustrate in this section, building on the tools presented so far.

In the previous section, we have properly formalized that proximal methods uncover (some of) the optimal structure of the optimization problem, which corresponds to the priors put in the learning model. Another important thing to keep in mind is that while we are able to observe the structure along the way, we do not know in general if this structure is in fact optimal or if it will remain stable. The key question is then

[11] These properties are intrinsic consequences of the nonsmoothness of regularizations, and directly connected to the stability properties of optimal solutions and identification properties of algorithms. Among a rich literature, we refer to the statistical properties of lasso [51], the recovery in compress sensing literature [10, 18], and the general model consistency result of [22].

How can we harness the identified structure ?

We present below general ideas to exploit the structure brought by proximal identification. These ideas can be seen as good practices or, more generally, important points to keep in mind when designing algorithms on problems with noticeable structure.

6.1 Compression: using the structure to compress iterates

The first way to harness structure is very simple and direct: since the algorithm's iterates will eventually reach some structure, we may use it to encode them more efficiently. This idea is mainly useful with sparse structures (see the examples of Section 3): the iterates can be stored in a `(key,value)` form which directly saves space when storing the iterates.

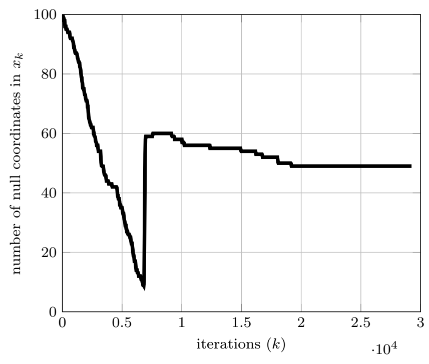
More interestingly, in the case of distributed algorithms involving communications between a master and some workers, such as (DAve-PG), a sparse encoding directly reduces the communication cost, which is often a bottleneck in distributed systems; see Figure 3 and [39, 28].

6.2 Acceleration: promoting structure stability over speed

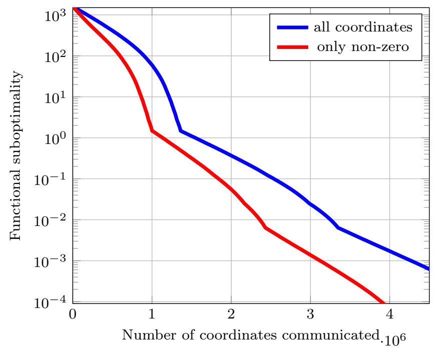
The proximal gradient (PG) can be accelerated using inertia (see (APG)); this acceleration makes the worst case functional convergence rate go from $\mathcal{O}(1/k)$ to $\mathcal{O}(1/k^2)$, which is the optimal complexity for such problems [41]. However, acceleration interferes with identification: we illustrate this graphically on Figure 4 which displays the iterates of (PG) and (APG) on two problems ($\mathcal{P}$) in $\mathbb{R}^2$ with f quadratic and g nonsmooth. This figure highlights two properties:

– Acceleration is faster and has some kind of exploratory behavior which increases its ability to reach the optimal structure (1 iteration = 1 mark in Figure 4).
– Even though the accelerated variant often reaches the optimal structure faster, the additional inertial term in (APG) is proportional to the difference between the last two iterates. Thus, when reaching a new manifold, the inertial term may drive the iterates out of it (In part (a), (APG) arrives to the optimal manifold but then is dragged leftwards). More generally, even if (APG) satisfies Assumption 1 and eventually identifies, it is less stable than the vanilla proximal gradient.

To summarize, accelerated and vanilla proximal gradient both have respective merits in terms of identification. In order to try and combine these merits, an intermediate method, introduced in [5], consists in accelerating only if the structure is preserved. This strategy shows encouraging results by being reasonably fast while preserving a good structure identification. This makes it

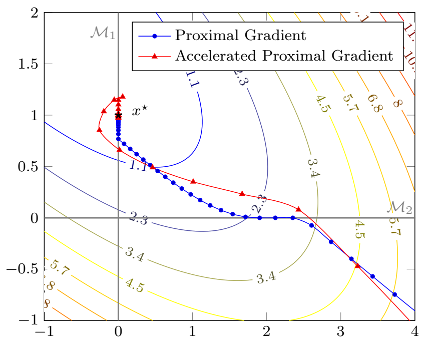


(a) Iterates density along a run of DAve-PG.

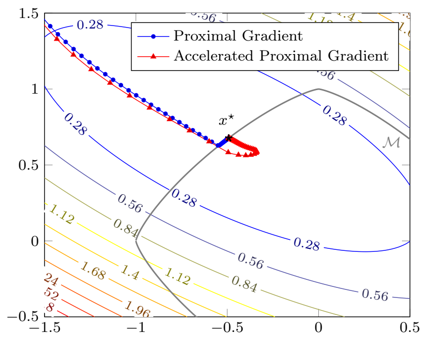


(b) Suboptimality versus number of coordinates exchanged between machines.

Fig. 3: Identification of null coordinates and its use for reducing the exchanges for (DAve-PG) on a 200×100 lasso problem ran on one machine/master coordinating 10 oracles/workers. We plot on the subfigure (b) the decrease of objective function with respect, not to iterations as usual, but to number of coordinates exchanged between the master and the workers. We see exchanging only non-zero coordinates is directly beneficial in terms of communication cost. The three regimes in the curves correspond to the three behaviors shown on subfigure (a): first the algorithm gives priority to sparsity against the quadratic term, until around iteration 7000, where the huge change in sparsity depicts an approximate identification; and finally exact identification takes place after 2e4 iterations. Recall from Section 4 that an iteration here corresponds to an update from one of the 10 workers. In this distributed framework, the bottleneck is the communications; we show that identification would automatically reduce communications. Finally, we mention that this example also shows that identification is not monotone in general: contrary to Figure 2, subfigure (a) shows huge variations in the current support.



(a) $g(x) = \|x\|_1$ $\quad \mathsf{C} = \{\mathcal{M}_1, \mathcal{M}_2\}$
$\mathcal{M}_1 = y$-axis, $\mathcal{M}_2 = x$-axis



(b) $g(x) = \max(0, \|x\|_{1.3} - 1)$ $\quad \mathsf{C} = \{\mathcal{M}\}$
$\mathcal{M}$ is the unit sphere for $\|\cdot\|_{1.3}$

Fig. 4: Iterates behavior for the Proximal Gradient (with and without acceleration) when minimizing a function of the type $\|Ax - b\|^2 + g(x)$ with two different functions g, the ℓ_1-norm (left) and the distance to the unit ball of the 1.3 norm (right).

interesting in machine learning problems where recovering (partial) structure is almost as important as convergence.

6.3 Dimension reduction: updating with respect to the identified structure

Let us make a simple remark. Once a manifold has been identified (e.g. some coordinates are null), there is some hope that it will stay identified for the remainder of the algorithm (e.g. the coordinates will remain null for all subsequent iterations). It is thus natural to update preferentially outside of this manifold (e.g. update preferentially the non-null coordinates). Mathematically, this leads to the following conceptual algorithm:

– Observe $\mathsf{S}(x_k)$, which gives $\mathsf{C}_k = \{\mathcal{M}_i \in \mathsf{C} : x_k \in \mathcal{M}_i\}$, the identified collection, and $\mathcal{M}_k = \bigcap_{\mathcal{M}\in\mathsf{C}_k} \mathcal{M}$, the identified manifold.
– Compute a *working manifold* $\mathcal{W}_k$ from C_k and $\mathcal{M}_k$.
– Update iterates preferentially[12] along $\mathcal{W}_k$.

Two types of strategies for choosing $\mathcal{W}_k$ and updates are presented in the next two subsections: (i) alternating between updates along $\mathcal{W}_k$ and updates on the whole space; (ii) drawing randomly $\mathcal{W}_k$ as a larger manifold so that, in expectation, the full space is covered.

6.3.1 Predictor-corrector methods

After the observation of the current collection C_k, a simple and natural choice of working manifold is to consider

$$\mathcal{W}_k = \bigcap_{\mathcal{M}\in\mathsf{C}_k} \mathcal{M} \; .$$

In the ideal situation where we know that our working manifold $\mathcal{W}_k$ is optimal (i.e. $x^\star \in \mathcal{W}_k$), we would just have to solve our optimization problem restricted to $\mathcal{W}_k$. However, as already mentioned, we never know if $\mathcal{W}_k$ is optimal or not. An idea is therefore to interlace efficient steps along $\mathcal{W}_k$ (as if we knew the optimal manifold) and proximal steps in the whole space (to keep on identifying). This type of methods is called predictor-corrector in [16].

Interestingly, the nonsmooth function g restricted to the manifold $\mathcal{W}_k$ is often smooth, and the problem restricted to $\mathcal{W}_k$ is then a Riemannian optimization problem [1]. In words: on top of reducing dimension, identification may also allow us to get rid of nonsmoothness; at the price of the additional constraint of lying in $\mathcal{W}_k$. See [37] for discussions on the connections between nonlinear programming, nonsmooth optimization, and Riemannian optimization.

We write the generic predictor-corrector method exploiting this identification of smooth substructure, as follows

$$\begin{cases} u_k = \text{(Gradient or Newton) step tangent to } \mathcal{W}_k & \text{(Riemannian step)} \\ x_{k+1} = \mathbf{prox}_{\gamma g}(u_k) & \text{(prox)} \end{cases}$$

[12] Preferentially but not only. Since the user never knows if the optimal structure is attained, the directions outside the identified manifold $\mathcal{M}_k$ should still be updated at times.

When a Riemannian Newton step is used, this method offers a geometrical interpretation of the so-called $\mathcal{VU}$ algorithms (introduced in [32] and showed to be superlinearly convergent in [36]).

6.3.2 Randomized structured descent

We focus here on ($\mathcal{P}$) for f convex and smooth with a Lipschitz gradient, g convex lower semi-continuous, and linear manifolds, as for instance for sparse (Section 3.1) or flat (Section 3.2) structures.

After the observation of the current collection C_k, [27] proposes to update on the random working subspace

$$\mathcal{W}_k = \bigcap_{\mathcal{M}\in\mathsf{C}_k} (\xi_{\mathcal{M},k}\mathcal{M} + (1-\xi_{\mathcal{M},k})\mathbb{R}^n)$$

where the $(\xi_{\mathcal{M},k})$ are independent identically distributed Bernoulli random variables with success probability $p \in (0,1)$. This means that the working subspace is obtained from the current collection, by randomly removing some identified spaces. This allows any direction in $\mathbb{R}^n$ to be explored with non-zero probability, which prevents the algorithm against mis-identification. For instance, in the case of sparsity manifolds, it consists in working on variables with the same support as the current variable, except for some randomly drawn coordinates that we allow to be non-zero as well.

Working in the subspace $\mathcal{W}_k$ involves projecting the gradient, which unfortunately adds a bias in the averaged decent direction. To compensate, [27] proposes to debias the iterates with the inverse squared root of the averaged projection $Q_k = \left(\mathbb{E}\ \mathrm{proj}_{\mathcal{W}_k}\right)^{-\frac{1}{2}}$, which leads to the following algorithm.

$$\begin{cases} y_k = x_k - \gamma\nabla f(x_k) & \text{(gradient)}\\ u_k = Q_k^{-1}\left(\mathrm{proj}_{\mathcal{W}_k}(Q_k y_k) + \mathrm{proj}^{\perp}_{\mathcal{W}_k}(u_{k-1})\right) & \text{(debiased projection)}\\ x_{k+1} = \mathbf{prox}_{\gamma g}(u_k) & \text{(prox)}\end{cases}$$

Unfortunately, as such, this method does not work if the collection C_k (or equivalently the distribution of subspaces $\mathcal{W}_k$) changes too much, which goes against the progressive structure identification. To overcome this, [27] proposes to change the current collection C_k (onto which $\mathcal{W}_k$ is based) not at every iteration but after some waiting time depending on the amount of change in the iterates' structure (the more changes, the longer the wait). Thus, by adapting the collection at the right frequency, the method identifies the optimal structure and benefits from a competitive rate of convergence compared to the vanilla proximal gradient while using a smaller part of the gradient at each iteration; see [27] for details and illustrations.

7 Conclusions

In this paper, we introduced a framework to formalize nonsmooth optimization problems with noticeable structure. These problems often arise in machine learning applications and are prone to be solved by proximal methods. These algorithms are known to identify the problem structure, but going one step further, we show that the structure progressively uncovered by these methods can be harnessed in practice to improve their performances. Among many possible applications, we emphasize the automatic compression, the interplay with acceleration, and some algorithmic options offered by dimension reduction.

Acknowledgements

The authors would like to warmly thank the whole *DAO team* and especially our PhD students Gilles Bareilles, Mathias Chastan, Sélim Chraibi, Dmitry Grishchenko, Yu-Guan Hsieh, and Yassine Laguel. FI benefited from the support of the ANR JCJC project *STROLL* (ANR-19-CE23-0008). This work has been partially supported by MIAI@Grenoble Alpes (ANR-19-P3IA-0003).

References

1. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization Algorithms on matrix manifolds. Princeton University Press (2008)
2. Bach, F.: Consistency of trace norm minimization. The Journal of Machine Learning Research **9**(Jun), 1019–1048 (2008)
3. Bach, F., Jenatton, R., Mairal, J., Obozinski, G., et al.: Convex optimization with sparsity-inducing norms. Optimization for Machine Learning **5**, 19–53 (2011)
4. Bach, F., Jenatton, R., Mairal, J., Obozinski, G., et al.: Optimization with sparsity-inducing penalties. Foundations and Trends® in Machine Learning **4**(1), 1–106 (2012)
5. Bareilles, G., Iutzeler, F.: On the interplay between acceleration and identification for the proximal gradient algorithm. Computational Optimization and Applications (2020)
6. Bauschke, H.H., Combettes, P.L.: Convex analysis and monotone operator theory in Hilbert spaces. Springer Science & Business Media (2011)
7. Bauschke, H.H., Goebel, R., Lucet, Y., Wang, X.: The proximal average: basic theory. SIAM Journal on Optimization **19**(2), 766–785 (2008)
8. Beck, A.: First-order methods in optimization, vol. 25. SIAM (2017)
9. Benfenati, A., Chouzenoux, E., Pesquet, J.C.: A proximal approach for a class of matrix optimization problems. arXiv preprint arXiv:1801.07452 (2018)
10. Candès, E.J., Romberg, J., Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Transactions on information theory **52**(2), 489–509 (2006)
11. Chambolle, A., Dossal, C.: On the convergence of the iterates of the "fast iterative shrinkage/thresholding algorithm". Journal of Optimization theory and Applications **166**(3), 968–982 (2015)
12. Chartrand, R., Yin, W.: Nonconvex sparse regularization and splitting algorithms. In: Splitting methods in communication, imaging, science, and engineering, pp. 237–249. Springer (2016)
13. Combettes, P.L., Pesquet, J.C.: Proximal splitting methods in signal processing. In: Fixed-point algorithms for inverse problems in science and engineering, pp. 185–212. Springer (2011)

14. Condat, L.: A direct algorithm for 1-d total variation denoising. IEEE Signal Processing Letters **20**(11), 1054–1057 (2013)
15. Condat, L.: Discrete total variation: New definition and minimization. SIAM Journal on Imaging Sciences **10**(3), 1258–1290 (2017)
16. Daniilidis, A., Hare, W., Malick, J.: Geometrical interpretation of the predictor-corrector type algorithms in structured optimization problems. Optimization **55**(5-6), 481–503 (2006)
17. Defazio, A., Bach, F., Lacoste-Julien, S.: Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In: Advances in Neural Information Processing Systems, pp. 1646–1654 (2014)
18. Donoho, D.L.: Compressed sensing. IEEE Transactions on information theory **52**(4), 1289–1306 (2006)
19. Duval, V., Peyré, G.: Sparse regularization on thin grids i: the lasso. Inverse Problems **33**(5), 055008 (2017)
20. Eckstein, J., Bertsekas, D.P.: On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators. Mathematical Programming **55**(1-3), 293–318 (1992)
21. Fadili, J., Malick, J., Peyré, G.: Sensitivity analysis for mirror-stratifiable convex functions. SIAM Journal on Optimization **28**(4), 2975–3000 (2018)
22. Fadili, J.M., Garrigos, G., Malick, J., Peyré, G.: Model consistency for learning with mirror-stratifiable regularizers. In: International Conference on Artificial Intelligence and Statistics (AISTATS) (2019)
23. Fercoq, O., Gramfort, A., Salmon, J.: Mind the duality gap: safer rules for the lasso. In: International Conference on Machine Learning, pp. 333–342 (2015)
24. Friedrich, F., Kempe, A., Liebscher, V., Winkler, G.: Complexity penalized m-estimation: fast computation. Journal of Computational and Graphical Statistics **17**(1), 201–224 (2008)
25. Garrigos, G., Rosasco, L., Villa, S.: Thresholding gradient methods in Hilbert spaces: support identification and linear convergence. arXiv preprint arXiv:1712.00357 (2017)
26. Ghaoui, L.E., Viallon, V., Rabbani, T.: Safe feature elimination for the lasso and sparse supervised learning problems. Pacific Journal of Optimization **8**(4), 667–698 (2012)
27. Grishchenko, D., Iutzeler, F., Malick, J.: Proximal gradient methods with adaptive subspace sampling. Mathematics of Operations Research (2020)
28. Grishchenko, D., Iutzeler, F., Malick, J., Amini, M.R.: Asynchronous distributed learning with sparse communications and identification. arXiv preprint arXiv:1812.03871 (2018)
29. Hare, W., Lewis, A.S.: Identifying active constraints via partial smoothness and prox-regularity. Journal of Convex Analysis **11**(2), 251–266 (2004)
30. Hare, W., Sagastizábal, C.: Computing proximal points of nonconvex functions. Mathematical Programming **116**(1-2), 221–258 (2009)
31. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms. Springer Verlag, Heidelberg (1993). Two volumes
32. Lemaréchal, C., Oustry, F., Sagastizábal, C.: The $\mathcal{U}$-Lagrangian of a convex function. Transactions of the AMS **352**(2), 711–729 (2000)
33. Liang, J., Fadili, J., Peyré, G.: Activity identification and local linear convergence of forward–backward-type methods. SIAM Journal on Optimization **27**(1), 408–437 (2017)
34. Mairal, J.: Sparse coding for machine learning, image processing and computer vision. Ph.D. thesis, École Normale Supérieure de Cachan (2010)
35. Massias, M., Salmon, J., Gramfort, A.: Celer: a fast solver for the lasso with dual extrapolation. In: International Conference on Machine Learning, pp. 3321–3330 (2018)
36. Mifflin, R., Sagastizábal, C.: A $\mathcal{VU}$-algorithm for convex minimization. Mathematical programming **104**(2-3), 583–608 (2005)
37. Miller, S.A., Malick, J.: Newton methods for nonsmooth convex minimization: connections among-lagrangian, riemannian newton and sqp methods. Mathematical programming **104**(2-3), 609–633 (2005)
38. Mishchenko, K., Iutzeler, F., Malick, J.: A distributed flexible delay-tolerant proximal gradient algorithm. SIAM Journal on Optimization **30**(1), 933–959 (2020)

39. Mishchenko, K., Iutzeler, F., Malick, J., Amini, M.R.: A delay-tolerant proximal-gradient algorithm for distributed learning. In: International Conference on Machine Learning, pp. 3584–3592 (2018)
40. Ndiaye, E., Fercoq, O., Gramfort, A., Salmon, J.: Gap safe screening rules for sparsity enforcing penalties. The Journal of Machine Learning Research **18**(1), 4671–4703 (2017)
41. Nesterov, Y.E.: A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In: Dokl. Akad. Nauk SSSR, vol. 269, pp. 543–547 (1983)
42. Nutini, J., Schmidt, M., Hare, W.: "active-set complexity" of proximal gradient: How long does it take to find the sparsity pattern? Optimization Letters **13**(4), 645–655 (2019)
43. Parikh, N., Boyd, S.P.: Proximal algorithms. Foundations and Trends in optimization **1**(3), 127–239 (2014)
44. Poon, C., Liang, J., Schönlieb, C.B.: Local convergence properties of saga/prox-svrg and acceleration. arXiv:1802.02554 (2018)
45. Rockafellar, R., Wets, R.B.: Variational Analysis. Springer Verlag, Heidelberg (1998)
46. Rockafellar, R.T.: Monotone operators and the proximal point algorithm. SIAM journal on control and optimization **14**(5), 877–898 (1976)
47. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D: nonlinear phenomena **60**(1-4), 259–268 (1992)
48. Shalev-Shwartz, S., Ben-David, S.: Understanding machine learning: From theory to algorithms. Cambridge university press (2014)
49. Stewart, G.W.: Perturbation theory for the singular value decomposition. Tech. rep. (1998)
50. Sun, Y., Jeong, H., Nutini, J., Schmidt, M.: Are we there yet? manifold identification of gradient-related proximal methods. In: The 22nd International Conference on Artificial Intelligence and Statistics, pp. 1110–1119 (2019)
51. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) pp. 267–288 (1996)
52. Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J., Tibshirani, R.J.: Strong rules for discarding predictors in lasso-type problems. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **74**(2), 245–266 (2012)
53. Vaiter, S., Peyré, G., Fadili, J.: Low complexity regularization of linear inverse problems. In: Sampling Theory, a Renaissance, pp. 103–153. Springer (2015)
54. Weinmann, A., Storath, M., Demaret, L.: The l^1-Potts functional for robust jump-sparse reconstruction. SIAM Journal on Numerical Analysis **53**(1), 644–673 (2015)
55. Weyl, H.: Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung). Mathematische Annalen **71**(4), 441–479 (1912)
56. Wright, S.J.: Identifiable surfaces in constrained optimization. SIAM Journal on Control and Optimization **31**(4), 1063–1079 (1993)
57. Yu, Y.L.: On decomposing the proximal map. In: Advances in Neural Information Processing Systems, pp. 91–99 (2013)
58. Zhao, P., Yu, B.: On model selection consistency of Lasso. The Journal of Machine Learning Research **7**, 2541–2563 (2006)