THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques Appliquées

Arrêté ministériel : 25 mai 2016

Présentée par

Dmitry GRISHCHENKO

Thèse dirigée par **Jerome MALICK**
et codirigée par **Franck IUTZELER**, Maitre de Conférences, Université Grenoble Alpes
et **Massih-Reza AMINI**, Professeur, Université Grenoble Alpes

préparée au sein du **Laboratoire Laboratoire Jean Kuntzmann** dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Optimisation proximale avec réduction automatique de dimension

Proximal optimization with automatic dimension reduction for large-scale learning

Thèse soutenue publiquement le **3 novembre 2020**,
devant le jury composé de :

Monsieur JEROME MALICK
DIRECTEUR DE RECHERCHE, CNRS DELEGATION ALPES, Directeur de thèse
Monsieur PASCAL BIANCHI
PROFESSEUR, TELECOM PARISTECH, Rapporteur
Monsieur PETER RICHTARIK
PROFESSEUR, UNIV SCIENCES ET TECH. DU ROI ABDALLAH, Rapporteur
Monsieur JULIEN MAIRAL
CHARGE DE RECHERCHE HDR, INRIA CENTRE DE GRENOBLE RHÔNE-ALPES, Examinateur
Monsieur MASSIH-REZA AMINI
PROFESSEUR DES UNIVERSITES, UNIVERSITE GRENOBLE ALPES, Co-directeur de thèse
Monsieur ALEXANDER GASNIKOV
PROFESSEUR ASSOCIE, MIPT - RUSSIE, Examinateur
Monsieur SAMUEL VAITER
CHARGE DE RECHERCHE, CNRS DELEGATION CENTRE-EST, Examinateur

Résumé

Dans cette thèse, nous proposons des algorithmes proximaux, avec réduction de dimension automatique, pour des problèmes d'optimisation avec solutions parcimonieuses. Dans un premier temps, nous proposons une méthode générale de réduction de dimension, exploitant la propriété d'identification proximale, par des projections adaptées à la structure de l'itéré courant. Dans le cas parcimonieux, cet algorithme permet de travailler dans des sous-espaces aléatoires de petites dimensions plutôt que dans l'espace entier, possiblement de très grande dimension. Dans un deuxième temps, nous nous plaçons dans un cadre d'optimisation distribuée asynchrone et utilisons la méthode précédente pour réduire la taille des communications entre machines. Nous montrons tout d'abord, que l'application directe de notre méthode de réduction dimension dans ce cadre fonctionne si le problème est bien conditionné. Pour attaquer les problèmes généraux, nous proposons ensuite un reconditionnement proximal donnant ainsi un algorithme avec garanties théorétiques de convergence et de réduction de communications. Des experiences numériques montrent un gain important pour des problèmes classiques fortement parcimonieux.

Abstract

In this thesis, we develop a framework for reducing the dimensionality of composite optimization problems using sparsity inducing regularizers.Based on the identification property of proximal methods, we first develop a "sketch-and-project" method that uses projections based on the structure of the correct point. This method allows to work with random low-dimensional subspaces instead of considering the full space in the cases when the final solution is sparse. Second, we place ourselves in the context of the delay-tolerant asynchronous proximal methods and use our dimension reduction technique to decrease the total size of communications. However, this technique is proven to converge only for well-conditioned problems both in theory in practice. Thus, we investigate wrapping it up into a proximal reconditioning framework. This leads to a theoretically backed algorithm that is guaranteed to cost less in terms of communications compared with a non-sparsified version. We show in practice that this method enjoys faster runtime convergence when the sparsity of the problem is sufficiently big.

Contents

Introduction

Mathematical optimization is one of the most important tools in machine learning.

Often, learning cames with some specific structure enforced to the final solution and one of the popular ways to enforce it is to add a regularization term to the problem. Such regularizers are often non-smooth, which calls for optimization methods able to handle such non-smooth objectives.

Proximal algorithms are methods of choice for solving optimization problems with explicit non-smooth objectives. Such methods for sparsity-inducing regularizers (e.g. ℓ_1, $\ell_{1,2}$ norms or **TV** regularizer) have the additional property of identifying (near-)optimal subspaces in finite time. This property opens the way for various dimension reduction techniques.

In our work, we focus on the sparsification technique based on the identification property of proximal methods and consider composite optimization problem with sparsity inducing regularizers.

In Chapter 2, we present our sketch-and-project approach for solving composite optimization problems with sparsity inducing regularizers. We propose a randomized proximal gradient method harnessing the underlying structure. This algorithm is the first algorithm that uses identification-based sketches. More precisely, we propose to select the projections based on the properties of the regularizer. We introduce two key components: i) a random subspace proximal gradient algorithm; ii) an identification-based sampling of the subspaces. Their interplay brings a significant performance improvement in terms of dimensions explored on typical learning problems.

In the second part of our work, we consider the centralized setup. We propose different sparsified versions of the asynchronous proximal algorithm [MIM20].

In Chapter 3, we present a general framework `SPY` for sparsification, where we propose sending randomly chosen set of worker's update coordinates. However, both theoretical and practical analysis shows that usage of different probabilities could lead to divergence of the algorithm. On the other hand, in case of uniform selection the algorithm converges and, moreover, identifies the near-optimal support for the ℓ_1 regularized problems that motivate us to look at identification-based selection: some coordinates are selected with probability 1 and other are selected with some fixed probability. This particular selection

allows acceleration in terms of the number of communications when it converges. To guarantee the convergence of SPY with different probabilities the scaling technique should be used; however, the performance of such methods is worse than without sparsification both in theory and in practice.

In Chapter 4, we present a modification of SPY that tackles the divergence issue by using a proximal reconditioning scheme that consists in iteratively regularizing the problem with the squared distance to some center point. We focused on wrapping up the SPY algorithm with different probabilities. This allows us to perform much more aggressive sparsifications. Furthermore, we show that when using a sparsity inducing regularizer, our reconditioned algorithm generates iterates that identify the optimal sparsity pattern of the model in finite time. This progressively uncovered pattern can be used to adaptively update the sparsification distribution of the inner method. All in all, this method only features sparse communications: the downward communications (master-to-worker) consist in sending the (eventually) sparse model, and the upwards communications (worker-to-master) are adaptively and aggressively sparsified. Finally, we show theoretically and numerically that our method has better performance than its non-sparsified version in terms of suboptimality with respect to the quantity of information exchanged.

Corresponding articles The contribution of this manuscript consists of the following articles, prepared in scope of the research made during this Ph.D.

In Chapter 2, we present the contribution from [GIM20].

In Chapters 3, 4, we present the contributions from [GIMA18].

In addition, a couple of articles were prepared outside of the scope of this work.

In [IGGS19], we propose a universal acceleration technique for adaptive methods for smooth convex but not strongly convex objective functions. It allows accelerating well-known methods such as Steepest Descent, Random Adaptive Coordinate Descent, and others where Lipschitz constant of the gradient is either unknown (expensive to compute) or changes a lot along the trajectory of the iterates.

In [BLG+20], we present an asynchronous version of the Progressive Hedging algorithm (a popular strategy in multistage stochastic programming) that is able to compute an update as soon as a scenario subproblem is solved. Based on similar arguments as in Asynchronous ADMM, we prove that the asynchronous version has the same convergence properties as the standard. We release an easy-to-use Julia toolbox[1].

[1] GitHub link: https://github.com/yassine-laguel/RandomizedProgressiveHedging.jl

Chapter 1

Background material: from gradient to identification.

Chapter Contents

Introduction

Mathematical optimization is a tool for choosing some "optimal" parameter x within the *constraint set* X such that it minimizes an *objective* function f. More formally, an optimization problem could be written in the general form

$$\min_{x \in X} f(x).$$

This problem can be used to model many practical applications in areas such as bioinformatics, advertising, and visual object recognition. However, this problem is too difficult in general.

To produce an efficient algorithm with guarantees to solve the problem, some additional assumptions on f and X are commonly made. One of the common assumptions is convexity of the objective function f and of the constraint set X. In our work, we consider convex functions $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $X = \mathbb{R}^n$, and in this chapter, we present some algorithms and approaches that we used to discover new algorithms.

In order to make good predictions, large-scale data is used in practical applications: the number of observations m is large and the dimension of each observation (or the number of features) n is big (see more details in Section 1.2). Such context raises a lot of questions, including, how to make the existing optimization algorithms computationally more efficient? Classical optimization methods like gradient descent and its variations are computationally expensive because every step of these algorithms requires a pass through the full dataset. In contrast, incremental methods relying on using a single data point (or a minibatch) to compute an estimator for the gradient reduce the computational cost of iteration. To accept bigger datasets the state-of-the-art algorithms are distributed. In such algorithms, the communication process between machines is also expensive and methods that can reduce the number of communications or the size of every single update are sought after.

Outline. This chapter is organized as follows. In Section 1.1, we introduce several basic definitions from convex optimization. Furthermore, we recall some first-order optimization methods and theoretical results that we use in later chapters. In Section 1.2, we discuss Machine Learning. In Section 1.3, we present an overview of distributed learning: setting, context, methods. In particular, we recall the asynchronous distributed proximal algorithm [MIMA18]. We present in Section 1.4 an active-set identification property and prove it for this algorithm.

1.1 Convex optimization

This section is organized as folows. In Subsection 1.1.1, we introduce the definitions of convex, strongly-convex, and L-smooth (with L-Lipschitz gradient) functions and prove some important properties that are widely used in our further proofs. In Subsection 1.1.2, we recall the *Gradient Descent* algorithm (see Algorithm 1) and its convergence. In Subsection 1.1.3, we present the notion of *subgradient*, *proximal operator*, and *Moreau-Yosida regularization*. Furthermore, we recall the basic methods for the minimization of non-smooth functions: *Subgradient Descent* (see Algorithm 2) and *Proximal Minimization* (see Algorithm 3). Finally, in Subsection 1.1.3, we overview several optimization methods for solving regularized Empirical Risk Minimization problem: *Coordinate Descent* (see Algorithm 6), *Proximal Gradient Descent* (see Algorithm 4), and *Stochastic Gradient Descent* variations.

1.1.1 Convexity and smoothness

In this section, we recall the basic definitions and properties that are used to analyze optimization methods for smooth and convex objective [HUL12].

In this thesis, we consider that the domain[1] of all functions is the entire space $\mathbb{R}^n$ unless explicitly written otherwise.

Definition 1.1 (Convex function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called convex for any $x, y \in \mathbb{R}^n$ and $\alpha \in [0, 1]$*

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y), \tag{1.1}$$

Moreover, function f is called μ-strongly convex if function $g = f - \frac{\mu}{2}\|\cdot\|_2^2$ is convex.

For continuously differentiable functions the (strongly) convexity to the following first-order lower-bounds.

Lemma 1.2. *[Theorem 2.1.2 [Nes13]] A continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex iff for any $x, y \in \mathbb{R}^n$, we have*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle^2. \tag{1.2}$$

Moreover, function $f : \mathbb{R}^n \to \mathbb{R}$ is μ-strongly convex iff for any $x, y \in \mathbb{R}^n$, we have

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|_2^2. \tag{1.3}$$

[1] $\operatorname{dom} f = \{x : f(x) < \infty\}$.

[2] In this thesis we denote by $\langle \cdot, \cdot \rangle$ the standard Euclidean scalar product.

Convexity implies that any stationary point[3] of a continuously differentiable function f is a global minima of f. In addition, strong convexity implies the existence and uniqueness of $x^\star = \operatorname{argmin}_{x\in\mathbb{R}^n} f(x)$.

Now let us recall the definition of L-smoothness.

Definition 1.3 (L-smoothness)**.** *A continuosly differentiable function f is called L-smooth if its gradient is L-Lipschitz*

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2 \tag{1.4}$$

for any $x, y \in \mathbb{R}^n$.

If function f is L-smooth, then the following upper bound holds.

Lemma 1.4. *[Theorem 2.1.5 [Nes13]] Let us assume that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and L-smooth. Then for any $x, y \in \mathbb{R}^n$ we have*

$$f(y) - f(x) - \langle \nabla f(x), y - x\rangle \leq \frac{L}{2}\|x - y\|_2^2, \tag{1.5}$$

and

$$\frac{1}{L}\|\nabla f(x) - \nabla f(y)\|_2^2 \leq \langle \nabla f(x) - \nabla f(y), x - y\rangle \leq L\|x - y\|_2^2. \tag{1.6}$$

In Figure 1-1 we present a graphical illustration of the lower (1.3) and upper (1.5) bounds for an L-smooth and μ-strongly convex objective function f. As we could see from this figure, the quadratic lower bound provided by (1.3) approximates the functional value much better than the first-order approximation provided by (1.2). The quality of these approximations can be characterized by the **condition number** of the problem, defined as

$$\kappa = \frac{\mu}{L}.$$

When this number is close to 1 (**well-conditioned**), the upper and lower bound are close to each other. As a result, both of these bound are good upper/lower approximations of f. When it is close to 0 (**ill-conditioned**), the difference between the lower and upper bounds is large that drops the quality of such approximations.

The size of condition number impacts on the speed of the first order optimization algorithms: better-conditioned problems are easier to solve (see e.g., Theorem 1.6).

Finally, let us present the following auxiliary lemma (Lemma 3.11 [B+15]) that is widely used in the convergence analysis of first-order methods for L-smooth and μ-strongly convex objectives.

[3] Point x is called stationary or critical if $\nabla f(x) = 0$.

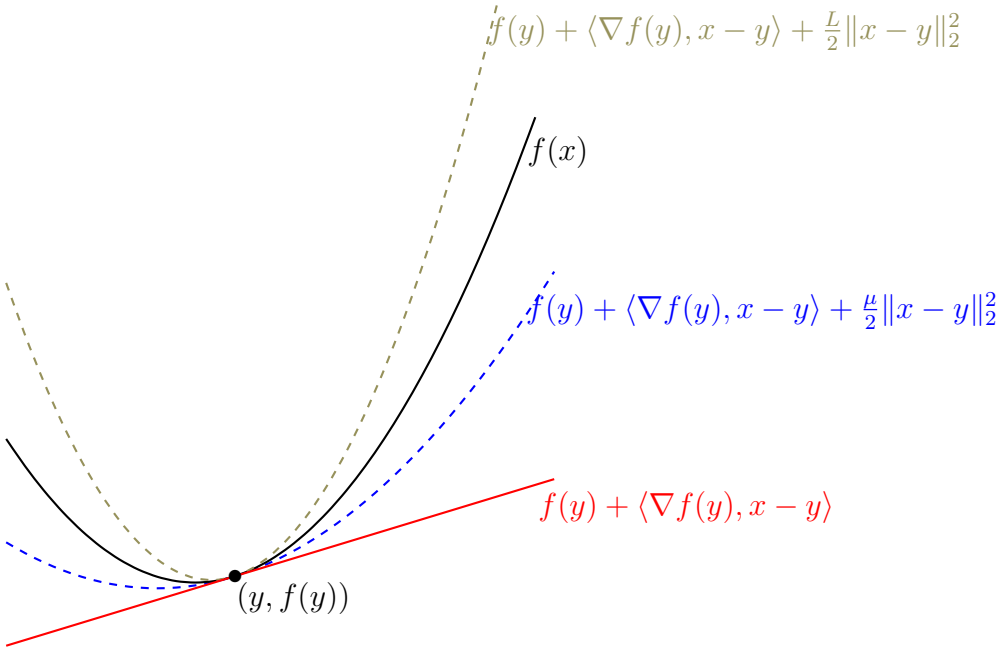


Figure 1-1. Graphical illustration of lower and upper bound for L-smooth and μ-strongly convex function $f : \mathbb{R} \to \mathbb{R}$

Lemma 1.5. *Let us assume that $f : \mathbb{R}^n \to \mathbb{R}$ is L-smooth and μ-strongly convex. Then for any $x, y \in \mathbb{R}^n$ we have*

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \frac{\mu L}{\mu + L}\|x - y\|_2^2 + \frac{1}{\mu + L}\|\nabla f(x) - \nabla f(y)\|_2^2. \tag{1.7}$$

1.1.2 Gradient descent

Let us consider the optimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1.8}$$

where f is convex and differentiable. One of the most important methods in the mathematical optimization literature for solving (1.8) is gradient descent. This method can be traced back at least to the work of Cauchy [Cau47, Extrait 383] and became elaborate after [Pol63]. Thanks to its simplicity, there are many extensions of it [Pol69b, Pol69a, Nes05, BT09].

Consider the ordinary differential equation (ODE)

$$\frac{dx}{dt} = -\nabla f(x).$$

It is well known that the values of $f(x)$ decrease along the trajectories before reaching stationary point $\nabla f(x) = 0$. Another way to see that the anti-gradient is a descent direction is described in the following. From the definition of the gradient, we have that

$$df_x(u) = \nabla f(x)^\top u$$

for any small vector u. Among all vectors u with fixed norm, the scalar product would be the smallest if u is opposite to $\nabla f(x)$. This idea of using anti-gradient direction is exactly the idea of the gradient descent [Pol63]. More precisely, since the anti-gradient direction is the descent direction we can consider the iterative (discretized) version defined as follows.

Algorithm 1 Gradient Descent (`GD`)

Initialize $x^0 \in \mathbb{R}^n$
for $k \geq 0$ **do**
  $x^{k+1} \leftarrow x^k - \gamma^k \nabla f(x^k)$ where γ^k is a stepsize
end for

Let us now present the convergence result for the gradient descent algorithm with the fixed stepsize $\gamma^k = \gamma$ (Theorems 2.1.14 and 2.1.15 [Nes13]).

Theorem 1.6 (Convergence of Gradient Descent)**.** *Let us assume that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and L-smooth. Take $\gamma \in \left(0, \frac{2}{L}\right)$. Then Algorithm 1 generates sequence of points $(x^k)_{k\geq 0}$ such that*

$$f(x^k) - f^\star \leq \frac{2(f(x^0) - f^\star)\|x^0 - x^\star\|_2^2}{2\|x^0 - x^\star\|_2^2 + \gamma k(2 - \gamma L)(f(x^0) - f^\star)}, \tag{1.9}$$

where $f^\star = \min_{x\in\mathbb{R}^n} f$ and $x^\star$ is an optimal solution, i.e., $f(x^\star) = f^\star$. If moreover, f is μ-strongly convex with $\mu > 0$, then taking $\gamma \in \left(0, \frac{2}{\mu+L}\right]$, Algorithm 1 generates sequence of points $(x^k)_{k\geq 0}$ such that

$$\|x^k - x^\star\|_2^2 \leq \left(1 - \frac{2\gamma\mu L}{\mu + L}\right)^k \|x^0 - x^\star\|_2^2, \tag{1.10}$$

where $x^\star$ is the unique minimizer of (1.8).

As we can see from this result, the best theoretical rate for non-strongly convex fuctions is achieved when $\gamma = \text{argmax}\left(\gamma\left(1 - \frac{\gamma L}{2}\right)\right) = \frac{1}{L}$. This leads to the following rate

$$f(x^k) - f^\star \leq \frac{2L(f(x^0) - f^\star)\|x^0 - x^\star\|_2^2}{2L\|x^0 - x^\star\|_2^2 + k(f(x^0) - f^\star)} \leq \frac{2L\|x^0 - x^\star\|_2^2}{k + 4}, \tag{1.11}$$

where for the last inequality we used (1.5). For a μ-strongly convex objective function f, and the optimal stepsize $\gamma = \frac{2}{\mu+L}$, we get the rate

$$\|x^k - x^\star\|_2^2 \leq \left(1 - \frac{4\mu L}{(\mu+L)^2}\right)^k \|x^0 - x^\star\|_2^2 = \left(\frac{1-\kappa}{\kappa+1}\right)^{2k} \|x^0 - x^\star\|_2^2. \tag{1.12}$$

The convergence rate depends on κ, and it is faster if the problem is better conditioned. For example, if the problem is 1-conditioned, then only 1 iteration of `GD` is enough to converge.

1.1.3 Non-smooth optimization

Let us now consider the optimization problem

$$\min_{x\in\mathbb{R}^n} r(x), \tag{1.13}$$

where r is convex but is allowed to be non-smooth (= non-differentiable. These functions are often assumed to be differentiable almost everywhere on their domain. One of the common approaches in different applications is a model of max-type functions

$$r(x) = \max_{1\leq i\leq p} f_i(x),$$

where f_i are convex and differentiable.

Since r is allowed to be non-smooth the Gradient Descent method is not applicable for such problems, since for some points the gradient is not defined. However, for all points where the gradient is defined the anti-gradient direction is still the descent direction. It motivates to find some object to replace gradient that will be equal to the gradient in all smooth points and replaced by something in points of non-smoothness.

Subgradient descent

Let us start with the definition of the object that could replace gradients for non-smooth functions.

Definition 1.7 (Subgradient)**.** *Let $r : \mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$ be convex. Vector $g \in \mathbb{R}^n$ is called subgradient of r at point $y \in \operatorname{dom} r$ if for any $x \in \operatorname{dom} r$ holds*

$$r(x) \geq r(y) + \langle g, x - y\rangle. \tag{1.14}$$

The set of all subgradients of r at y is called the subdifferential of r at point y and denoted by $\partial r(y)$.

Now we are ready to present one of the most basic algorithms [Sho12] to solve (1.13).

Algorithm 2 Subgradient Descent

Initialize $x^0 \in \mathbb{R}^n$
Select the sequence of stepsizes γ^k satisfying $\gamma^k > 0,\ \gamma^k \to 0,\ \sum_0^\infty \gamma^k = \infty$
for $k \geq 0$ **do**
  Compute any subgradient $g^k \in \partial r(x^k)$
  $x^{k+1} \leftarrow x^k - \gamma^k \frac{g^k}{\|g^k\|}$
end for

Let us now present the convergence result for this algorithm.

Theorem 1.8 (Convergence of Subgradient Descent)**.** *Let us assume that* $r : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$. *Take the sequence of stepsizes* γ^k *satisfying* $\gamma^k > 0$, $\gamma^k \to 0$, *and* $\sum_{k=0}^\infty \gamma^k = \infty$. *Then Algorithm 2 generates sequence of points* $(x^k)_{k\geq 0}$ *such that*

$$r(x^k) \to r^\star,$$

where $r^\star = \min_{x\in\mathbb{R}^n} r(x)$.

This method is worse than the gradient descent due to the decreasing stepsize. Moreover, at the points of non-smoothnes the direction of $-g^k$ has no reason to be the descent direction, see Figure 1-2. This explains the usage of decreasing stepsizes to converge to the minimizer and stay in its neighborhood after the next step.

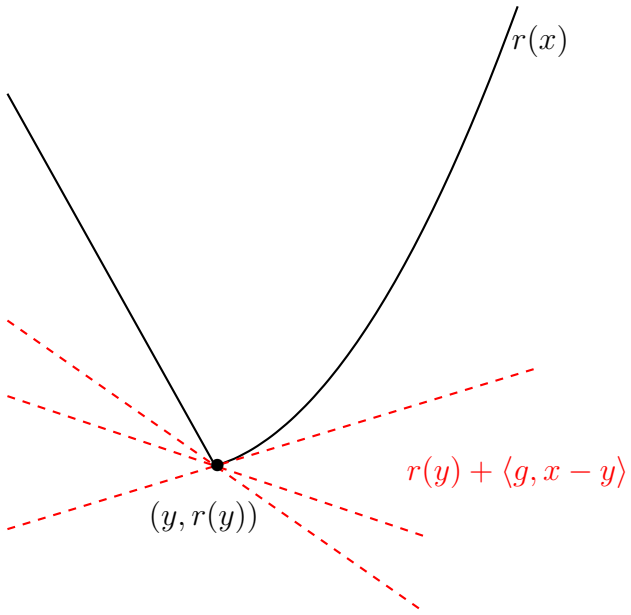


Figure 1-2. Linear approximations of non-smooth function $r = \max\{-2x, (0.2x+1)^2 - 1\}$ from $\mathbb{R}$ to $\mathbb{R}$ at point $y = 0$ with different subgradients. As we could see, $y = 0$ is a minimizer of r, however its subdifferential in this point is $\partial r(0) = [-2, 0.4] \ni 0$. Depending on the subgradient g (we plot for $g = -1, -0.5, 0.4$) different linear approximations of function r at 0 appears and, as a result we will move from the optimal point with probability 1.

Proximal methods

The subgradient descent method is not the only method that can provably solve non-smooth optimization problems. We now present a class of methods called proximal methods. First, let us recall the definition of the proximal operator [Roc76].

Definition 1.9 (Proximal operator)**.** *Given a convex function $r : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, the proximal operator of r is the mapping*

$$\mathbf{prox}_r(x) = \underset{y \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ r(y) + \frac{1}{2}\|x - y\|_2^2 \right\}. \tag{1.15}$$

If function r is non-smooth, it is usually assumed to be proper[4] and lower-semicontinuous [5] thus it implies that the objective of argmin is proper and strongly convex, which means

[4]Proper convex function is a convex function $r : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ such that $\exists\, x \in \mathbb{R}^n \ : \ r(x) < +\infty$ and $r(x) > -\infty \ \forall\, x \in \mathbb{R}^n$.

[5]Function r is called lower-semicontinuous if $\liminf_{x \to x_0} r(x) \geq r(x_0)$ for any $x_0 \in \operatorname{dom} r$.

that the value of the proximal operator at any point is well defined and unique.

In machine learning applications, many regularizers are relatively simple (see Section 1.2 for more details) that makes the computation of proximal operator cheap. More precisely, a proximal operator for such problems usually has a closed-form solution (ℓ_1[6], Group-Lasso[7]) or could be computed iteratively (1-d Total Variation (TV)[8]). Moreover, if r is the indicator function of a convex set[9], the proximal operator is an orthogonal projection onto this set.

Before presenting proximal algorithms, let us recall some important properties of the proximal operator that will be useful. First, let us present the lemma about the optimal solution of (1.13) [PB14, Section 2.3].

Lemma 1.10. *Consider convex, proper, and lower-semicontinuous function $r\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$. Point $x^\star$ is a solution of (1.13) if and only if $x^\star = \mathbf{prox}_r(x^\star)$.*

Now, we present an important property of the proximal operator that is widely used in the analysis of proximal methods [BC11, Prop. 12.27].

Lemma 1.11 (Firm nonexpansiveness of proximal operator)**.** *Consider a convex function $r : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$. Then for any $x, y \in \mathbb{R}^n$ holds*

$$\|\mathbf{prox}_r(x) - \mathbf{prox}_r(y)\|_2^2 \leq \langle x - y, \mathbf{prox}_r(x) - \mathbf{prox}_r(y)\rangle. \tag{1.16}$$

Finally, let us introduce the connection between the proximal operator and the subdifferential of function r [PB14].

Lemma 1.12 (Resolvent)**.** *The proximal operator $\mathbf{prox}_{\gamma r}$ and the subdifferential ∂r are related as follows:*

$$\mathbf{prox}_{\gamma r} = (I + \gamma \partial r)^{-1}[10], \tag{1.17}$$

where I is identity operator. The mapping $(I + \gamma\partial r)^{-1}$ is called the resolvent of operator ∂r with parameter $\gamma > 0$.

In Figures 1-3 we present the proximal operator for the function $r = \lambda\|\cdot\|_1$ that is also known as the soft-thresholding operator [Don95]. It is easy to see that the following

[6]For some $\lambda > 0$ the ℓ_1 regularizer is defined as follows $r(x) = \lambda \sum_{i=1}^n |x_{[i]}|$, where $x_{[i]}$ denotes the i-th coordinate of x.

[7]For some partition G of $[n]$ and some $\lambda > 0$ the group-lasso regularizer is defined as follows $r(x) = \lambda \sum_{\mathbf{g}\in G} \|x_{\mathbf{g}}\|_2$, where $[x_{\mathbf{g}}]_{[i]} = x_{[i]} \Leftrightarrow i \in \mathbf{g}$.

[8]For some $\lambda > 0$ the 1-d TV regularizer is defined as follows $r(x) = \lambda \sum_{i=1}^{n-1} |x_{[i-1]} - x_{[i]}|$.

[9]Indicator function i_C of the convex set C is defined as $r(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{otherwise} \end{cases}$.

[10]By the definition $(I + \gamma\partial r)^{-1}(x) = \{y\colon x \in y + \partial r(y)\}$.

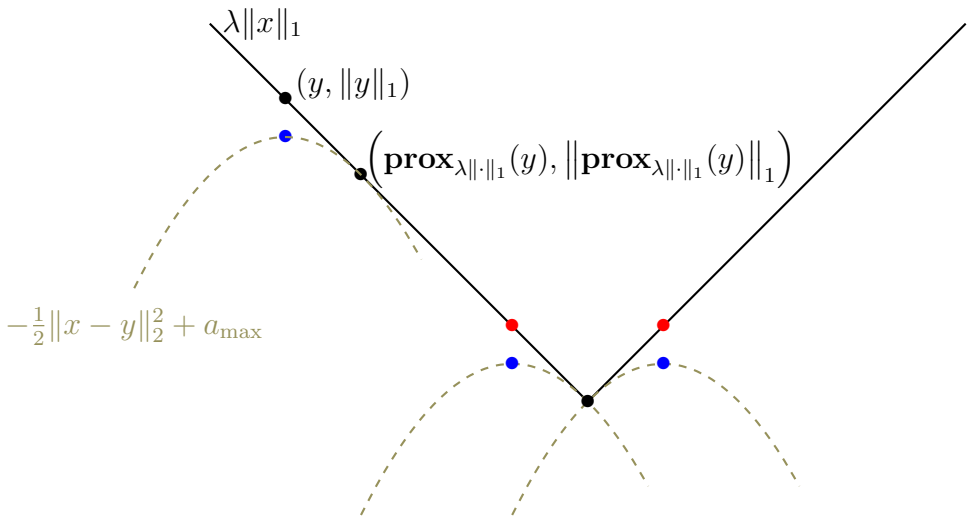


Figure 1-3. Geometrical interpretation of the soft-thresholding operator

problems are equivalent

$$\max_{a-\frac{1}{2\lambda}\|x-y\|_2^2 \le \|y\|_1} a \quad \Leftrightarrow \quad \min_{y\in\mathbb{R}^n} \left\{ \lambda_1 \|y\|_1 + \frac{1}{2}\|x-y\|_2^2 \right\}.$$

It implies that the proximal operator could be interpreted as a point where the lines $r(x)$ and $-\frac{1}{2}\|x-y\|_2^2 - a_{\max}$ touch each other. Moreover, from this figure, we could see the sequence of points

$$x^{k+1} = \mathbf{prox}_{\lambda\|\cdot\|_1}(x^k)$$

is decreasing (in terms of $\|\cdot\|_1$, that motivates the following proximal algorithm [Roc76].

Algorithm 3 Proximal Minimization

Initialize $x^0 \in \mathbb{R}^n$
for $k \ge 0$ **do**
  $x^{k+1} \leftarrow \mathbf{prox}_{\gamma r}(x^k)$, where γ is a stepsize
end for

Algorithm 3 converges to the minimizer if this minimizer exists (see, for example, Theorem 23.41 [BC11]).

Moreau-Yosida regularization

Let us define the Moreau envelope, that is also called Moreau-Yosida regularization [Mor62, Yos12].

Definition 1.13. *Given $\lambda > 0$, the Moreau envelope $M_{\lambda r}$ of the function $r : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ with parameter λ is defined as*

$$M_{\lambda r}(y) = \inf_{x \in \mathbb{R}^n} \left(r(x) + \frac{1}{2\lambda} \|x - y\|_2^2 \right). \tag{1.18}$$

Moreau-Yosida regularization of function r is continuously differentiable, even if r is not [YYY11, Fact 17.17] and its gradient is given by

$$\nabla M_{\lambda r} = \frac{1}{\lambda}(x - \mathbf{prox}_{\lambda r}(x)). \tag{1.19}$$

Moreover, the set of minima of r and M_f are the same. Let us rewrite the proximal operator as

$$\mathbf{prox}_{\lambda r}(x) = x - \lambda \nabla M_{\lambda r}(x).$$

This shows that the proximal operator can be viewed as a gradient step and Algorithm 3 is a gradient descent algorithm with stepsize λ for minimizing $M_{\lambda r}$ [Roc76]. Taking into account that the Moreau envelope has the same minimizers as r, we have the convergence of the proximal point method. In Figure 1-4 we present a geometrical interpretation of Moreau envelope for the non-smooth objective function r.

Let us talk a little bit about the properties of Moreau-Yosida regularization. Since we mentioned that the proximal point method is a gradient descent method on Moreau envelope of the function, let us present smoothness and strong convexity properties of this envelope. From (1.19) we have that

$$L_{M_{\lambda r}} = \lambda^{-1} \tag{1.20}$$

independent on the smoothness parameter of r. If function r is strongly convex the Moreau envelope is strongly convex with strongly convexity parameter

$$\mu_{M_{\lambda r}} = \frac{\mu \lambda^{-1}}{\mu + \lambda^{-1}}, \tag{1.21}$$

where μ is the strongly convexity constant of r (see Theorem 2.2 of [LS97a] for the proof). Thus, the condition number of this Moreau-Yosida regularization of μ-strongly convex function r is

$$\kappa_{M_{\lambda r}} = \frac{\mu_{M_{\lambda r}}}{L_{M_{\lambda r}}} = \frac{\mu}{\mu + \lambda^{-1}}. \tag{1.22}$$

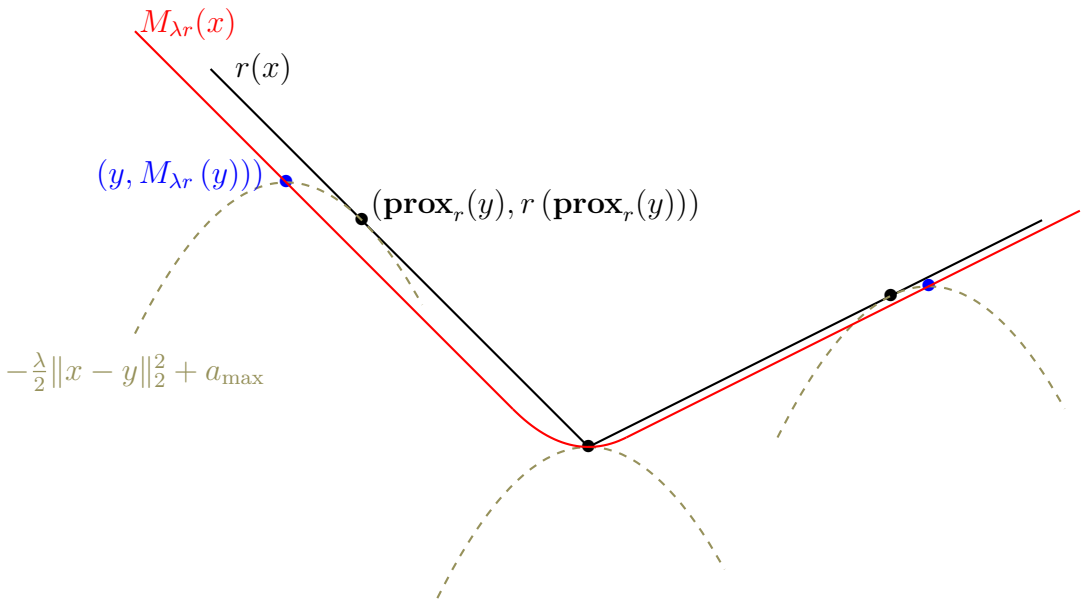


Figure 1-4. Geometrical interpretation of Moreau envelope for $r(x) = \max\{-x, 0.5x\}$ from $\mathbb{R}$ to $\mathbb{R}$.

As a result, this problem becomes exceptionally well-conditioned when λ is selected big.

1.1.4 Composite optimization

Let us consider the composite optimization problem

$$\min_{x\in\mathbb{R}^n} f(x) + r(x), \tag{1.23}$$

where f is smooth and convex and r is convex, non-smooth proper, and lower-semicontinuous. Moreover, we consider r to be a prox-easy function meaning that its proximal operator is easy to compute (a wide class of regularizers used in ML suits to this requirement). As we already mentioned, this formulation corresponds to the regularized empirical loss minimization problem that appears extensively in signal processing and machine learning applications; we refer to e.g. [CWB08, CP11, BJM+12], among a vast literature.

Proximal gradient descent

Let us start from the proximal gradient descent method with constant stepsize that performs the gradient step and the proximal one alternatively. This class of method is

also known as iterative shrinkage-thresholding algorithms (ISTA) [DDDM04], also called forward-backward splitting method [Gab83, CP11, RFP13].

Algorithm 4 Proximal Gradient Descent (ISTA)

Initialize $x^0 \in \mathbb{R}^n$
for $k \geq 0$ **do**
  $x^{k+1} \leftarrow \mathbf{prox}_{\gamma r}(x^k - \gamma \nabla f(x^k))$
end for

Every iteration of this method consists of two steps: shift along the anti-gradient direction of smooth part and proximal mapping to negotiate with the non-smoothness. It could be reformulated as

$$x^{k+1} = \operatorname*{argmin}_{z \in \mathbb{R}^n} \left\{ \underbrace{f(x^k) + \langle \nabla f(x^k), z - x^k \rangle + \frac{1}{2\gamma} \|x^k - z\|_2^2}_{\hat{f}(z, x^k)} + r(z) \right\}. \tag{1.24}$$

If $\gamma \leq \frac{1}{L}$, where L is the smoothness parameter of f then $\hat{f}(z, x^k)$ is a convex upper-bound to f that is tight at x^k. This allows to interpret proximal methods as a majorization-minimization algorithm [LHY00] that is widely used in optimization problems for minimizing an objective function [CM09, NH98, GRC09, WNF09, Mai15]. The idea of such methods in iterative minimizing of surrogate upper-bounds that leads to the objective function value downhill.

From this form it is easy to see that the stationary point of this algorithm is a minimizer of (1.23). Since. the objective in (1.24) is strongly-convex we could write the optimality condition for fixed point

$$\begin{aligned} x^\star &= \operatorname*{argmin}_{z \in \mathbb{R}^n} \left\{ \langle \nabla f(x^\star), z - x^\star \rangle + \frac{1}{2\gamma} \|x^\star - z\|_2^2 + r(z) \right\} \\ &\Updownarrow \\ 0 &\in \partial r(x^\star) + \nabla f(x^\star), \end{aligned}$$

that is an optimality condition for (1.23).

When r is the indicator of a convex set, the proximal operator projects the gradient step back to the constraint set. In terms of convergence, these proximal variant has the same rate as the gradient descent (see for example Theorem 3.1 of [BT09]).

Coordinate descent methods

Coordinate descent methods is a class of iterative methods in which only one coordinate (block) is updated on every iteration. Let us see the interest of such methods for the problem with a least-squares objective function

$$f(x) = \|Ax - b\|_2^2,$$

where A is a matrix of examples, and b is a vector of observations. For such objective function, the computational cost of one coordinate of the gradient is n times smaller than the full gradient computation thanks to the function's separable structure.

The idea of the underlying coordinate descent methods is the following: on every iteration, we compute one (or some) coordinate of gradient and make a move in that direction with some stepsize. There are different ways to select stepsize and the coordinate-based, for example, on the smoothness constants of i-th coordinates of gradient [RT12] or on the maximal absolute value of the gradient's coordinates correspondingly [Nes12].

Let us consider smooth optimization problem (1.8), where function f has coordinate-wise Lipschitz continuous gradient

$$|\nabla_{[i]} f(x + he_i) - \nabla_{[i]} f(x)| \leq M|h|, \tag{1.25}$$

where subscript $[i]$ means the i-th coordinate, e_i is a i-th standard basis vector, and $h \in \mathbb{R}$. For such function, it is easy to see that

Algorithm 5 Coordinate Descent (CD) for (1.8)

Initialize $x^0 \in \mathbb{R}^n$
for $k \geq 0$ **do**
  Select coordinate $i^k = \text{argmax}_{1 \leq i^k \leq n} |\nabla_{i^k} f(x^k)|$
  $x^{k+1} \leftarrow x^k - \frac{1}{M} \nabla_{[i^k]} f(x^k)$
end for

$$f(x^k) - f(x^{k+1}) \geq \frac{1}{2M} |\nabla_{i^k} f(x^k)|^2 \geq \frac{1}{2nM} \|\nabla f(x^k)\|_2^2,$$

where the first inequality follows from (1.25) and the second follows from the coordinate selection. It immediately implies (see the proof of Theorem 1.6) the following rate for the non-strongly convex objective

$$f(x^k) - f^\star \leq \frac{2nM}{k+4} \|x^0 - x^\star\|_2^2,$$

where we use the same notations as before.

As we could see from this result, the rate is n times worse than the rate of gradient descent, and every iteration requires a full gradient computation for coordinate selection. Moreover, it could happen that $M \gg L$ where L is the smoothness constant of f that makes algorithm worse in terms of the amount of gradient evaluations to reach the same accuracy even if only 1 coordinate of the gradient is required.

Together with the greedy strategy, randomized and cyclic variations of this method also appear widely in the literature. Besides, some modifications propose selecting the block of coordinates on every iteration. Thus in practice, it is more popular to use a cyclic or randomized selection of coordinate to be updated (see e.g. [Wri15]).

Now, let us consider composite optimization problem (1.23), where f has coordinate-wise Lipschitz continuous gradient and moreover regularizer r is coordinate-wise separable

$$r(x) = \sum_{i=1}^{n} r_i(x_{[i]}).$$

The best know example of coordinate-separable regularizers is ℓ_1 regularizer with $r_i(x_{[i]}) = \lambda|x_{[i]}|$. This type of regularizers together with block-separable (for example $\ell_{1,2}$ regularization [BJM+12]) is usual assumption for (block) coordinate descent methods, because it implies (block) separability of proximal operator

$$\mathbf{prox}_{\gamma r}(x)_{[i]} = \mathbf{prox}_{\gamma r_i}(x_{[i]}).$$

The key idea of coordinate descent method for (1.23) is to make a majorization-minimization algorithm where in (1.24) the upper bound is based on the coordinate smoothness

$$\hat{f}_i(z, x^k) = f(x^k) + \nabla_{[i]} f(x^k)(z - x^k)_{[i]} + \frac{1}{2\gamma}(x^k - z)^2_{[i]}.$$

The simplest version of the coordinate descent for (1.23) is presented in Algorithm 6 (see Theorem 1 of [RT12] for the proof).

Algorithm 6 Coordinate Descent (`CD`) for (1.23)

Initialize $x^0 \in \mathbb{R}^n$
for $k \geq 0$ **do**
  Select coordinate $i^k \in \{1, \ldots, n\}$
  $x^{k+1}_{[i^k]} \leftarrow \mathbf{prox}_{\gamma r_{i^k}}\left(x^k_{[i^k]} - \gamma \nabla_{[i^k]} f(x^k)\right)$ with $\gamma = \frac{1}{M}$
  $x^{k+1}_{[j]} \leftarrow x^k_{[j]}$ for all $j \neq i^k$
end for

Since a lot of common regularizers are not separable (for example 1-d Total Variation [BJM+12]) coordinate methods Algorithm 6 could not be widely used to solve arbitrary regularized ERM problem (1.29). This calls for some modern modifications that allow to use non-separable proximal term [HMR18].

Part of our research considers the extension of Algorithm 6 with a surprising application to the non-separable regularizers.

Incremental methods

In the big data era, the amount of examples m and the dimension of features n can be very large, which excludes higher-order optimization methods. Under this setting, the computational cost of each gradient could be large because it requires passing through all the m training points. So the `GD` and `CD` (if $m \gg n$) methods that we have presented so far are also expensive. This calls for incremental gradient methods that have the cost of each iteration that does not depend on m. Let us recall that in machine learning, the usual problem is ERM (1.28), where the objective function f has a sum structure (probably infinite).

In case of a finite sum

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$

if i is selected uniformly at random from $[m] = [1, \dots, m]$ then $\nabla f_i(x)$ is also an unbiased estimator of gradient of f, but its computation is about m times faster than the computation of $\nabla f(x)$

Let us present the general scheme of stochastic gradient descent where by $\mathbb{E}\left[g^k | x^k\right]$

Algorithm 7 Stochastic Gradient Descent (`SGD`)

Initialize $x^0 \in \mathbb{R}^n$
for $k \geq 0$ **do**
  Compute an unbiased estimator g^k $\qquad \mathbb{E}\left[g^k | x^k\right] = \nabla f(x^k)$
  $x^{k+1} \leftarrow x^k - \gamma^k g^k$
end for

we denote conditional expectation [Kol33, Chapter IV, Section 4]. In particular, g^k could be selected as $\nabla f_{i^k}(x^k)$ where i^k is an index uniformly generated among $1, \dots, m$. The vector g^k is a noisy approximation of the real gradient $\nabla f(x^k)$. The difference $g^k - \nabla f(x^k)$ represents this noise and its variance is the important measure of quality

$$\mathbb{E}\left[\|g^k - \nabla f(x^k)\|_2^2 | x^k\right]. \tag{1.26}$$

This variance is usually non-zero for any x^k and even in the optimal point $x^\star$. For example, the full gradient $\nabla f(x^\star) = 0$ but there is no reason for $\nabla f_i(x^\star)$ to vanish. Consider Algorithm 7 with fixed stepsize $\gamma^k = \gamma$. Since, for any point x^k the noise (1.26) is separated from 0 the point $x^\star$ is not fixed point of the algorithm. This again (like for subgradient descent) leads to the slower rate than for the gradient descent, see the usual behaviour of convergence in Figure 1-5.

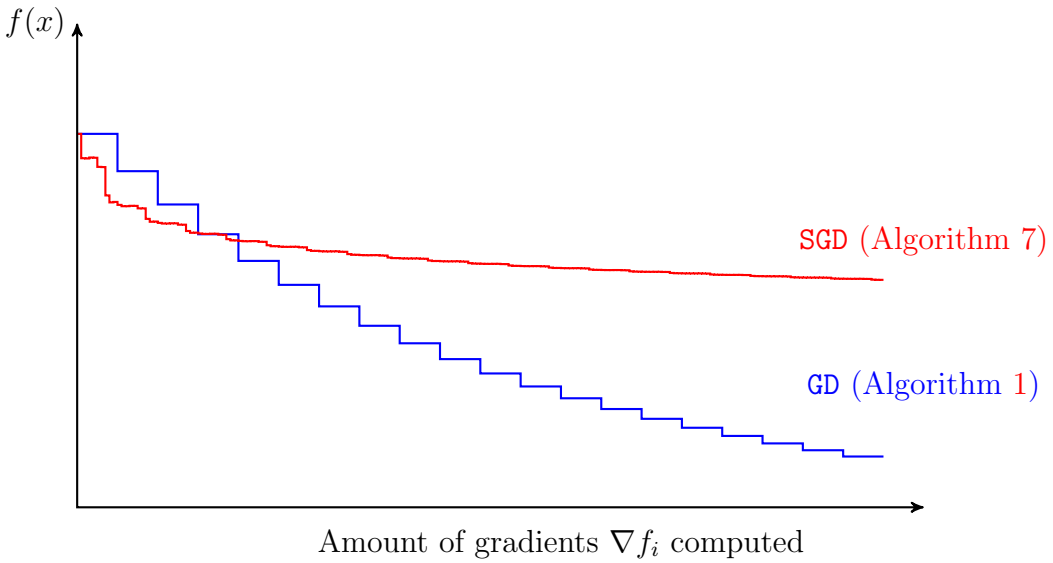


Figure 1-5. Evolution of iterates for GD with fixed stepsize $\gamma = \frac{2}{\mu+L}$ and SGD with decreasing stepsize $\gamma^k = \frac{1}{k}$ for L-smooth and μ-strongly convex objective $f(x) = \|Ax - b\|_2^2$, with random generated $A \in \mathbb{R}^{100\times 100}, b \in \mathbb{R}^{100}$. As we could see from this plot, in the beginning, when the stepsizes in these two algorithm are approximately the same SGD performs better, however when $\gamma^k \ll \frac{1}{L}$ it slows down fast.

In [Sho64] author propose SGD with a constant stepsize; however, this method does not converge to the minimizer. In Proposition 2.4 [NB01] it was proven that such algorithm has a convergence rate that could be split into two parts: first is dependent on the number of iteration and converges linearly (for strongly convex objectives) to 0, though, the second part is independent on k and does not go to 0 that implies a linear convergence but only up to a fixed tolerance. In [Sol98] the linear convergence takes place only under some additional assumptions about the relationship between f_i. In [SSZ13b] Stochastic Dual Coordinate Ascent (SDCA) was proposed, this algorithm has a linear convergence rate; however, it is limited to the finite sum problems with strongly convex f_i. This work was extended to the case when only the strong convexity of the global objective f is required [SS16].

Another possible solution is to decrease the variance of the gradient to guarantee the linear convergence for strongly convex objectives. One of the examples of such methods is `SAG` [SLRB17]. In this paper, authors propose a randomized variant of incremental aggregated gradient (`IAG`) [BHG07] with the following iterations

$$x^{k+1} = x^k - \frac{\gamma^k}{m} \sum_{i=1}^{m} y_i^k,$$

where on every iteration a random training example i^k is selected and

$$y_i^k = \begin{cases} \nabla f_i(x^k) & \text{if } i = i^k, \\ y_i^{k-1} & \text{otherwise.} \end{cases}$$

This algorithm combines the low cost of each iteration with the linear rate in the strongly convex case. Memory allows usage of better approximations of the full gradient on each iteration that allows using constant stepsize. This algorithm was further extended to the composite set up `SAGA` in [DBLJ14]. Another example of variance reduction in stochastic gradient descent methods is method `SVRG` [JZ13]. Every iteration of this algorithm could be written as

$$x^{k+1} = x^k - \gamma^k \left(\nabla f_i(x^k) - y_i^{k-1} + \frac{1}{m} \sum_{i=1}^{m} y_i \right),$$

where y_i is a last stored gradient for f_i. The key difference of this algorithm is in the way of updating gradients in the memory. Unlikely `SAG`/`SAGA` where on every iteration one gradient is updated in `SVRG` gradients are updated simultaneously every p iterations

$$y_i = \nabla f_i(x^d) \quad \text{for all } i,$$

where d is a moment of the last update of stored in memory gradients. It is now understood that these algorithms are conceptually almost the same, and even they could be proven using the same proof [KHR20, HLLJM15]. Another class of methods is majorization-minimization algorithms `Finito` [DDC14], `MISO` [Mai15], `DAve-RPG`[MIM20]. The majorization-minimization approach goes beyond optimization that provides a good start for designing new incremental methods [QSMR19].

1.2 Introduction to machine learning

The goal of machine learning is to learn the model from the provided data. In other words, the goal of machine learners is to propose an algorithm (or a set of algorithms)

that will be able to make the learning process without further human interventions. For example, in binary classification problems, we would like to learn the binary function that classifies every object into one of two possible groups based on the attributes of the object. However, the set of functions from the set of attributes to $\{-1, 1\}$ is so rich that for any given training set of examples, we could generate the function that will ideally suit the classification on this set by merely memorizing the correct label for every example. Nevertheless, such functions, generally, are not so good to predict the group of any new object as far as their behavior on such examples is usually an arbitrary random. Thus it is necessary to add some requirements on the nature of such functions in order to add some learning ability that will allow us to use them for new (unknown during the learning process) examples [Vap13]. One of the first examples was the Rosenblatt Perceptron [Ros60], where the author proposed to learn the separating hyperplane between classes (as we could see in this work, the classification function is assumed to be linear). This assumption is one of the common in machine learning and could be used, for example, in the support vector machines (SVM) problem [SV99]. After fixing of the learning problem, the search for predictors could be formalized as a mathematical optimization problem.

Let us consider m observations $(a_i, b_i) \in \mathcal{A} \times \mathcal{B}$ as an input of our prediction algorithm. The goal of prediction is to find prediction function $h(a, x)$, that belongs to some specific class and is parametrized by $x \in \mathbb{R}^n$. Let us provide some examples of such functions

linear model $h(a, x) = x^\top \Phi(a)$ of features $\Phi(a) \in \mathbb{R}^n$, where $\Phi \colon \mathcal{A} \to \mathbb{R}^n$;

neural networks $h(a, x) = x_l^\top \sigma(x_{l-1}^\top \sigma(\cdots x_1^\top a))$, where σ is an activation function.

Now, let us introduce the loss function that is used as a measure of the quality of the prediction. Loss function $\ell(\cdot, \cdot)$ is a function that represents a distance to between two arguments and as a result is closer to 0 if these arguments are closer to each other. Most widely used loss functions are

quadratic loss $\ell(b, h(a, x)) = \frac{1}{2}(b - h(a, x))^2$,

logistic loss $\ell(b, h(a, x)) = \log(1 + \exp(-bh(a, x)))$,

where the quadratic loss is used in regression problems with $b \in \mathbb{R}$ [B+66] and logistic is used in classification problems with $b \in \{-1, 1\}$ [SCD14, TJSZ04].

As we already mentioned, the goal is to learn the best possible prediction function from the specific class of predictors. In other words, we need to find the vector of parameters $x \in \mathbb{R}^n$ that minimizes the expected loss

$$\min_{x \in \mathbb{R}^n} \mathbb{E}_{P(a,b)} \ell(b, h(a, x)), \tag{1.27}$$

where $P(a, b)$ is unknown joint probability distribution over $\mathcal{A} \times \mathcal{B}$. However, we have access only to the finite amount of examples so infer h under the empirical risk minimization (ERM) principle :

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{m} \ell(b_i, h(a_i, x)). \tag{1.28}$$

The study of consistency of the ERM principle indicates that when the number of examples $m \to \infty$; the empirical loss should converge to the expected loss and also to the infimum expected loss [Vap13]. This study leads to the second principle in Machine Learning that is called structural risk minimization (SRM) and which states that learning is a compromise between accuracy and complexity.

One way to tackle this is to consider a regularized ERM :

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{m} \ell(b_i, h(a_i, x)) + r(x). \tag{1.29}$$

For example, in the variable selection problems it is important to have only few entries of estimator be non-zero in this case ℓ_1 regularization is used ($r = \| \cdot \|_1$) to force the final solution to have few non-zero entries [BJM+12].

Solving (regularized) ERM problem is a complicated mathematical procedure that calls for different optimization techniques. In general, to simplify the optimization problem, loss functions are considered to be convex and smooth. For convex functions, any local minimum is a global one, and smoothness allows us to operate with the gradient or even higher derivatives to analyze the first-order approximation of the function. However, regularizers that enforce the specific structure to the final solution are usually convex but non-smooth. It calls for optimization methods that can handle a non-smooth objective. In general, such methods are slower than their convex analogs since subgradients do not allow to make an equally good approximation of function as gradients, but often, regularizers have a simple geometrical structure that allows using proximal methods which reach the same rate of convergence as smooth ones.

In modern machine learning applications, the dimension of the problem n and the amount of training examples m are usually big. This makes the computation of the full gradient of the loss function expensive and unreliable. It moves state-of-the-art algorithms from gradient methods [Nes13] to the incremental methods [Ber11, Bot10], where instead of full gradient computation on every iteration some unbiased estimator (stochastic gradient) is computed. These methods appear to be slow out of the box as far as a non-zero variance of stochastic gradients force to use decreasing stepsize that leads to a slower rate than for the standard full gradient methods. To figure it out in 2013 in

[JZ13], stochastic variance reduced gradient method that allows using constant stepsize but requires more gradient computations that give all in all better convergence rate.

Another way to make the optimization process faster is to make computations in parallel. Thanks to the sum-structure ERM and its gradient could be computed in parallel on different machines: each machine computes a gradient that corresponds to some subset of examples and after the results are aggregated to the full gradient. Together with computational speedups, these algorithms also bring a bottleneck: communication. When the dimension (n) of the problem is big, the process of sending/receiving of the full gradient could be extremely costly and takes even more time than the computation of the gradient. During the last years, a lot of different algorithms that do data compression before sending were proposed [AGL+17, WWLZ18, HCS+17, WKSZ17]. In such methods, the way of compression does not use the structure of the problem, more precisely, the structure of the final solution. Let us consider ℓ_1 regularized problems, so we a-priori know that the optimal solution is sparse. If we could know the set of coordinates that are non-zero in this solution, we could send only these coordinates of gradient and lose nothing in terms of the rate as far as it would correspond to projected gradient descent. The problem is: we never know this set of coordinates, but we could try to guess.

As we mentioned above, usually regularizers have some strong geometrical properties that enforce the optimal solution to have a specific structure. Unfortunately, the exact pattern is unknown, but proximal methods allow us to identify it [NSH19]. However, this result sheds light on the moment when the iterate becomes sparse; it does not greenlight to switch to a projected mode. In the same view, the result of [FMP18] says that for some proximal algorithms after unknown time, this projection technique would be legal (see Section 1.4 for more details). This motivates us to do this research.

1.3 Distributed learning

Given the tremendous production of data and the ever-growing size of collections, there is a surge of interest for the development of efficient and scalable distributed learning strategies. A distributed system could be interpreted as multiple entities (also mentioned as *nodes*) that communicate in some way between each other, while also performing operations. Training observations are generally split over different computing nodes, and learning is performed simultaneously. This setup is different from shared-memory parallel computing, where each worker machine can potentially have access to all available memory [Val90, Kum02].

1.3.1 Computing setups

All distributed algorithms could be segregated according to the network structure. Some of them use networks with one central node called *master* that has a connection with all the other nodes called *workers* [KMRR16, KMY+16, MIM20]. In such setups usually, all the data is split between worker machines, and worker machines update their parameters simultaneously on a local sub-part of data and send their updated parameters to a master machine. The master integrates all received parameters and broadcasts them back to each computing node for a new local update of their parameters. This setup is called a *centralized* (or *master-worker*) setup. There could be a different amount of layers: master communicates only with some of the other nodes that are the local masters and so on. Graphs of such networks are trees with the root being the master node, leaves being the workers, and all the others being the local masters.

There is another setup called decentralized, when all the nodes are the same and could communicate with neighbor nodes with respect to the network graph [NO09, BPC+11a, DAW11, SLWY15]. In this setup, the approach usually builds on the seminal work of Tsitsikas [Tsi84] (see also [BT97] and [TBA86]) who proposed a framework to analyze the distributed computation models.

In our work, we focus on the centralized distributed setup without shared memory [MIM20] where all the data $\mathcal{D}$ is separated between M workers and all local subsets $\mathcal{D}_i$ are stored on each machine. We also consider the case when there is only one layer, and all worker machines are connected with the central node of the system - master node. This setup could be further categorized into two approaches with different assumptions on the communication rounds - synchronous and asynchronous. In the synchronous approach, every communication round mobilizes all worker machines [BPC+11b, CMBJ16, TR12]. In asynchronous setting, the communication round does not require all workers to participate and could involve only one worker [MIM20].

Synchronous algorithms

The study of distributed SGD in a master-worker setting [Yan13] raises the question of the communication-computation tradeoff. Every communication round in a synchronous setting consists of two phases: *All-Reduce synchronization step* when all the workers send their updates to the master and *Broadcasting step* that shares the updated parameter vector with all workers. This could significantly increase the working time of optimization algorithms when the amount of communication rounds, workers, and bits in the update are big due to the communication overhead. It promotes the paradigm of mini-batch methods [DGBSX12, SSZ13a, SS14, QR16, TRS15]. The main idea of these methods in generalizing stochastic methods to process multiple data points on every iteration to make communication rounds less often. However, when the significant reduction of

communication required, the size of mini-batch becomes big, which revert stochastic methods to the full-batch methods. Another way to look at this idea is local SGD methods that have been investigated in [Sti19, KMR19, KMR20, LSTS19, MKJ+17]. These methods perform well in practice; however, the theoretical understanding is an open area.

Another idea to solve the communication problem in distributed algorithms is in randomly selecting some entries to update. Random selection is used to sparsify local gradients in the synchronous algorithm of [WWLZ18], to sparsify the variance-reducing term in the stochastic algorithm of [LPLJ17] and [PLLJ17], or to sparsify updates in fixed-point iterations [PXYY16]. There are many different techniques of reducing communications that were investigated during the last years: general quantization [AGL+17, HKM+19, KSJ19], ternary quantization [WXY+17], 1-bit quantization [BWAA18], and others [BNH19, LHM+17].

Asynchronous algorithms

Synchronous algorithms perform well when it takes approximately the same time for all machines to make their update. Otherwise, the slower worker machines may slow down the whole process as the faster ones have to wait for all updates in order to terminate their computation and exchange information. As a result, many approaches based on the asynchrony of communications have been recently proposed on distributed optimization methods without shared memory, see e.g., [ZK14, MSJ+15, AFJ16, PXYY16, CR17]. In this case, worker machines update their parameters simultaneously on a local sub-part of data and send their updated parameters to a master machine which broadcasts the integrated updates back to each computing node for a new local update of their parameters [LAS13, KMRR16, MIMA18].

However, these methods generally suffer from communication cost between workers and the master machine and usually rely on restrictive assumptions on the computing system delays, which in turn impact the obtained convergence rates. To overcome these restrictions, asynchronous coordinate descent methods were recently proposed in [HY16, SHY17]. These techniques can handle unbounded delays, but they are based on decreasing stepsizes. In contrast, the recent works [MIMA18, MIM20] provide a delay-independent analysis technique that allows integrating assumptions on the computing system with a constant stepsize.

1.3.2 Notations and preliminaries

We place ourselves in a *totally asynchronous* setting as per the classification of Bertsekas and Tsitsiklis [BT89, Chap. 6.1], meaning that all workers are responsive but without bounded delays.

Let us consider a distributed setup where m observations are split down over M machines, each machine i having a private subset $\mathcal{D}_i$ of the examples. Learning over such scattered data leads to optimization problems with composite objective of the form

$$\min_{x\in\mathbb{R}^n} \; F(x) = \sum_{i=1}^{M} \alpha_i f_i(x) + r(x), \tag{P}$$

with $\alpha_i = |\mathcal{D}_i|/m$ being the proportion of observations locally stored in machine i, hence $\sum_{i=1}^{M} \alpha_i = 1$. $f_i(x) = \frac{1}{|\mathcal{D}_i|}\sum_{j\in\mathcal{D}_i} l_j(x)$ is the local empirical risk at machine i (l_j standing for the smooth loss function for example j) and r is a regularization term.

An asynchronous distributed setting allows the algorithm to carry on computation without waiting for slower machines: the machine performs computations based on outdated versions of the main variable, and the master has to gather the slaves inputs into a productive update. We formalize this framework with the same notation as in [MIM20].

- *For a worker* $i \in \{1,\dots,M\}$. At time k, let d_i^k be the elapsed moment since the last time the master has communicated with worker i ($d_i^k = 0$ iff the master gets updates from worker i at exactly time k, i.e. $i^k = i$). We also consider D_i^k as the elapsed time since the second last update. This means that, at time k, the last two moments that the worker i has communicated with the master are $k - d_i^k$ and $k - D_i^k$ (see Figure 1-6).

- *For the master.* We define k, as the number of updates that the master has received from any of the worker machines. Thus, at time k, the master receives some input from a worker, denoted by i^k, updates its global variables, $\bar{x}^k$ and x^k; and sends x^k back to worker i^k, where $\bar{x}$ is equal to the average of received updates from workers

$$\bar{x}^k = \sum_{i=1}^{M} \alpha_i x_i^k = \sum_{i=1}^{M} \alpha_i x^{k-d_i^k}. \tag{1.30}$$

To handle asynchrony, we define the sequence of stopping times (k_m) iteratively as $k_0 = 0$ and

$$k_{m+1} = \min\left\{k : k - D_i^k \geq k_m \text{ for all } i\right\}. \tag{1.31}$$

The stopping moment k_{m+1} is the first moment of time when $\bar{x}^k$ directly depends only on "new epoch" variables. More precisely, as we could see from (1.30) it depends directly on the $x_i^{k-d_i^k}$ that are the result of some computation on i-th worker while having $x_i^{k-D_i^k}$ and $x^{k-D_i^k}$ as a parameters.

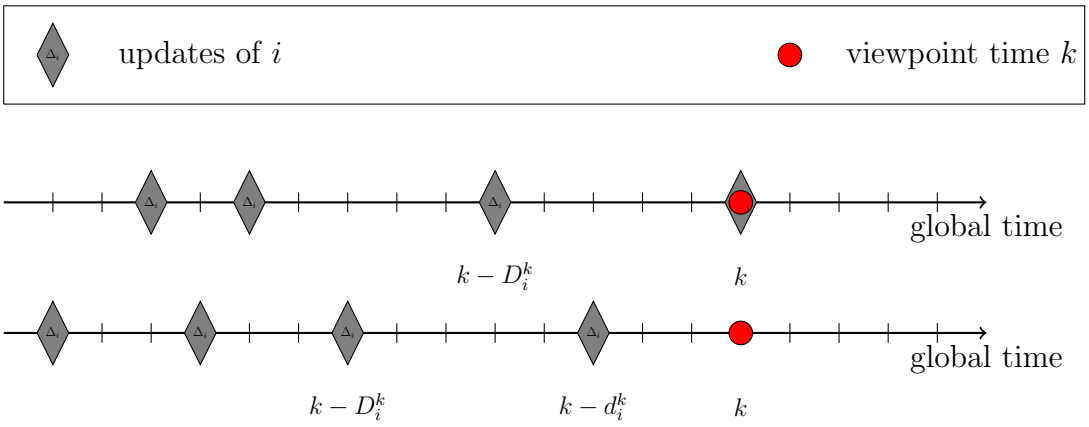


Figure 1-6. Notations of delays at iteration k.

1.3.3 DAve-PG

There exist a lot of distributed optimization algorithms with no shared memory. In [VGO16], the authors propose Proximal Incremental Aggregated Gradient method PIAG that is proven to converge under the assumption of bounded delays. However, the recent [MIM20] provides DAve-PG algorithm with a delay-independent analysis technique that allows integrating assumptions on the computing system with a constant stepsize. In DAve-PG algorithm, the master machine asynchronously gathers delayed updates from workers and sends back the global variable. More specifically, each worker independently computes a gradient step on its local loss, and the master machine keeps track of the weighted average of the most recent worker outputs, computes the proximity operator of the regularizer at this average point, and sends this result back to the updating worker i^k. Maintaining a weighted average of worker outputs is a special feature of DAve-PG; though it may appear conservative, it actually performs well in practice due to the stability of the produced iterations. The intuitive reason is that combining delayed points is more stable than using a combination of delayed directions; see the numerical comparisons of Section 2.4 of [MIM20] and Section 5 of [MIMA18].

Let us recall DAve-PG which solves composite distributed problems of the form P represented by a triplet $((\alpha_i), (f_i), r)$ for (*workers weights*,*workers functions*,*global regularization*) in Algorithm 8.

Convergence and rate for strongly convex objectives Let us assume that all (f_i) are μ-strongly convex and L-smooth with the same constants. This may appear limiting, but actually, this is achieved quickly by *exchanging* ℓ_2 term between functions. This setup allows us to feature a single stepsize in our algorithm, which simplifies the presentation to focus on the contributions.

Algorithm 8 DAve-PG on $((\alpha_i),(f_i),r)$ with stopping criterion C

Master

Initialize $\bar{x}^0$
while *stopping criterion* C *not verified* **do**
  Receive Δ^k from agent i^k
  $\bar{x}^k \leftarrow \bar{x}^{k-1} + \alpha_i \Delta^k$
  $x^k \leftarrow \mathbf{prox}_{\gamma r}(\bar{x}^k)$
  Send x^k to agent i^k
  $k \leftarrow k+1$
end
Interrupt all slaves
Output x^k

Worker i

Initialize $x_i = x_i^+ 0$
while *not interrupted by master* **do**
  Receive x from master
  $x_i^+ \leftarrow x - \gamma \nabla f_i(x)$
  $\Delta \leftarrow x_i^+ - x_i$
  Send Δ to master
  $x_i \leftarrow x_i^+$
end

Under that assumption, we define the *condition number* of the (smooth part of) Problem (P) as

$$\kappa_{(\mathsf{P})} = \frac{\mu}{L}.$$

This definition may be slightly unusual, but it is convenient in the present work as we will consider the case when $\mu = 0$ in the upcoming chapters.

Theorem 1.14 (Th. 3.2 [MIM20])**.** *Let the functions (f_i) be μ-strongly convex ($\mu > 0$) and L-smooth. Let r be convex l.s.c. Using $\gamma \in (0, \frac{2}{\mu+L}]$, DAve-PG converges linearly on the epoch sequence (k_m). More precisely, for all $k \in [k_m, k_{m+1})$*

$$\left\|x^k - x^\star\right\|^2 \leq (1-\gamma\mu)^{2m} \max_i \left\|x_i^0 - x_i^\star\right\|^2,$$

with the shifted local solutions $x_i^\star = x^\star - \gamma_i \nabla f_i(x^\star)$.

Furthermore, using the maximal stepsize $\gamma = \frac{2}{\mu+L}$, we obtain for all $k \in [k_m, k_{m+1})$

$$\left\|x^k - x^\star\right\|^2 \leq \left(\frac{1-\kappa_{(\mathsf{P})}}{1+\kappa_{(\mathsf{P})}}\right)^{2m} \max_i \left\|x_i^0 - x_i^\star\right\|^2.$$

This result is given with respect to the epoch sequence (1.31) to encompass various system behaviors. When we consider bounded delays, which is the standard assumption, the particularization of the previous result leads to a rate depending as expected on the bound on delays, as formalized in the following corollary.

Corollary 1.15 (Special case of bounded delays)**.** *Under the assumptions of Theorem 1.14, suppose that the delays are uniformly bounded over time and machines: $d_i^k \leq D$ for all k and i. Then, the length of epochs is bounded $k_{m+1} - k_m \leq 2D+1$ and we have a rate over k depending on the bound D*

$$\mathbb{E}[\|x^k - x^\star\|^2] \;\leq\; \left(1 - 2\frac{\gamma\,\mu}{2D+1}\right)^k \frac{\max_i \|x_i^0 - x_i^\star\|^2}{1-\gamma\mu}.$$

Proof. Directly from the definition in (1.31), we have that $k_{m+1} - k_m \leq 2D+1$. Therefore, $k_m \leq (2D+1)m$. For all $k \in [k_m, k_{m+1})$, we deduce

$$(1-\gamma\mu)^{2m} \leq \frac{1}{1-\gamma\mu}(1-\gamma\mu)^{\frac{2k}{2D+1}} \leq \frac{1}{1-\gamma\mu}\left(1 - \frac{2\,\gamma\,\mu}{2D+1}\right)^k.$$

by the concavity of $u \mapsto (1-u)^{2/(2D+1)}$ which gives $(1-u)^{2/(2D+1)} \leq 1 - 2u/(2d+1)$ for $u \in (0,1)$. □

Discussion on communication. `DAve-PG` is a delay-tolerant distributed algorithm that has an automatic sparsification of master-to-worker communications for many popular regularizations. For instance, proximal operators of ℓ_0 and ℓ_1 norms correspond respectively to the hard and soft thresholding operators, which set the smallest coordinates to zero, thus sparsifying the output. This kind of proximal sparsification was successfully used in the case of synchronous distributed learning; see e.g., [WKSZ17, SFJJ16]. However, worker-to-master communications do not need to be sparse, which implies that the communication cost is a bottleneck of this algorithm. As we have already mentioned, different techniques to solve this issue were developed for SGD: sparsification/quantization of updates [HKM+19] and mini-batching/local SGD [Yan13, KMR19]. In [MIM20, MIMA18], the authors propose `DAve-RPG` Algorithm that allows making a couple of repetitions p of local proximal gradient steps. However, this requires the knowledge of $\bar{x}$ by all workers that makes communication from master to worker not sparse.

In this work, we focus on another way to resolve the communication bottleneck issue - the sparsification of the updates. We aim at providing a distributed optimization algorithm reducing the size of communications by using the model structure enforced by the regularization r. Our adaptive communication reduction technique would then be complementary to existing compression techniques (reviewed in Subsection 1.3.1).

1.4 Identification

As mentioned in [BJM+12], the use of a proximal operator to handle the nonsmooth part r plays a prominent role, as it typically enforces some "sparsity" structure on the iterates and eventually on optimal solutions, see, e.g., [VGFP15]. For instance, the popular ℓ_1-norm regularization ($r = \|\cdot\|_1$) promotes optimal solutions with a few nonzero elements, and its associated proximity operator (called soft-thresholding, see [Don95]) zeroes entries along the iterations. This is an example of *identification*: in general, the iterates produced by proximal algorithms eventually reach some sparsity pattern close to the one of the optimal solution. For ℓ_1-norm regularization, this means that after a finite but unknown number of iterations, the algorithm "identifies" the final set of non-zero variables. This active-set identification property is typical for constrained convex optimization (see e.g. [Wri93]) and nonsmooth optimization (see e.g. [HL04]).

Related literature

The study of identification dates back at least to [Ber76] who showed that the projected gradient method identifies a sparsity pattern when using non-negative constraints. Such identification has been extensively studied in more general settings; we refer to [BM88], [Lew02], [DL14] or the recent [LL18], among other references. Recent works on this topic include: i) extended identification for a class of functions showing strong primal-dual structure, including TV-regularization and nuclear norm [FMP18]; ii) identification properties of various randomized algorithms, such as coordinate descent [Wri12] and stochastic methods [PLS18, FGMP19, SJNS19a].

The knowledge of the optimal substructure would reduce the optimization problem in this substructure and solve a lower dimension problem. While identification can be guaranteed in special cases (e.g. using duality for ℓ_1-regularized least-squares [OST13, FGS15]), after some substructure identification, one could switch to a more sophisticated method, e.g. updating parameters of first-order methods [LFP17]. Again, since the final identification moment is not known, numerically exploiting identification to accelerate the convergence of first-order methods has to be done with great care.

1.4.1 Active-set identification

In this section, we provide a general identification result for proximal algorithms useful for our developments, using the notion of sparsity vector.

Definition 1.16 (Sparsity vector)**.** *Let $\mathcal{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$ be a family of subspaces of $\mathbb{R}^n$ with m elements. We define the sparsity vector on $\mathcal{M}$ for point $x \in \mathbb{R}^n$ as the*

$\{0,1\}$-*valued*[11] *vector* $\mathsf{S}_{\mathcal{M}}(x) \in \{0,1\}^m$ *verifying*

$$(\mathsf{S}_{\mathcal{M}}(x))_{[i]} = 0 \quad \text{if } x \in \mathcal{M}_i \text{ and } 1 \text{ elsewhere.} \tag{1.32}$$

An identification result is a theorem stating that the iterates of the considered algorithm eventually belong to some – but not all – subspaces in $\mathcal{M}$. We formulate such a result of almost surely converging proximal-based algorithms as follows. This straightforward result is inspired by the extended identification result of [FMP18] (but does not rely on strong primal-dual structures as presented in [FMP18]).

Theorem 1.17 (Enlarged identification)**.** *Let* (u^k) *be an* $\mathbb{R}^n$*-valued sequence converging almost surely to* $u^\star$ *and define sequence* (x^k) *as* $x^k = \mathbf{prox}_{\gamma r}(u^k)$ *and* $x^\star = \mathbf{prox}_{\gamma r}(u^\star)$*. Then* (x^k) *identifies some subspaces with probability one; more precisely for any* $\varepsilon > 0$*, with probability one, after some finite time,*

$$\mathsf{S}_{\mathcal{M}}(x^\star) \;\leq\; \mathsf{S}_{\mathcal{M}}(x^k) \;\leq\; \max\left\{\mathsf{S}_{\mathcal{M}}(\mathbf{prox}_{\gamma r}(u))\colon u \in \mathcal{B}(u^\star, \varepsilon)\right\}. \tag{1.33}$$

Proof of Theorem 1.17. The proof is divided between the two inequalities. We start with the right inequality. As $u^k \to u^\star$ almost surely, for any $\varepsilon > 0$, u^k will belong to a ball centered around $u^\star$ of radius ε in finite time with probability one. Then, trivially, it will belong to a subspace if all points in this ball belong to it, which corresponds to the right inequality

$$\mathsf{S}_{\mathcal{M}}(x^k) \leq \;\max\left\{\mathsf{S}_{\mathcal{M}}(\mathbf{prox}_{\gamma r}(u))\colon u \in \mathcal{B}(u^\star, \varepsilon)\right\}. \tag{1.34}$$

Let us turn now to the proof of the left inequality. Consider the sets to which $x^\star$ belongs i.e. $\mathcal{M}^\star = \{\mathcal{M}_i \in \mathcal{M} : x^\star \in \mathcal{M}_i\}$; as $\mathcal{M}$ is a family of subspaces, there exists a ball of radius $\varepsilon' > 0$ around $x^\star$ such that no point x in it belong to more subspaces than $x^\star$ i.e. $x \notin \mathcal{M} \setminus \mathcal{M}^\star$. 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 $x^\star$. □

Two examples

Let us present a couple of important examples of the regularizers that enforce sparsity of the final solution ([ABMP13], [JAB11], [ZRY06]). The most common types of sparsity are (block) coordinate one, when there are only a few non-zero (blocks of) coordinates and variation one when there are few continuous blocks of coordinates such that the coordinates inside of each are the same.

[11] For two vectors $a, b \in \{0,1\}^m$, we use the following notation and terminology: (1) if $a_{[i]} \leq b_{[i]}$ for all $i = 1, .., m$, we say that b is greater than a, noted $a \leq b$; and (2) we define the union $c = a \cup b$ as $c_{[i]} = 1$ if $a_{[i]} = 1$ or $b_{[i]} = 1$ and 0 elsewhere.

Example 1.18 ($\ell_{1,2}$ regularizer)**.** *It is known that if r is $\ell_{1,2}$ regularizer induced by non-overlapping partition $\mathcal{B}$ reads*

$$r(x) = \sum_{B \in \mathcal{B}} \|x^B\|_2, \tag{1.35}$$

where x^B is the restriction of x to the entries indexed by block B, it enforces block sparsity i.e. it derives all the coefficients in one block to zero together (regularizer ℓ_1 is a particular case of $\ell_{1,2}$ so it leads to coordinate sparsity).

In this case the support of the optimal point $x^\star$ will be small, where

$$\operatorname{supp}(x) \triangleq \left\{ i \in [1, n] \,|\, x_{[i]} \neq 0 \right\}. \tag{1.36}$$

Let us specify a family $\mathcal{M}$ for the case of $\ell_{1,2}$ regularizer (or even ones that enforce (block) coordinate sparsity.)

Let us define collection $\mathcal{M} = \{\mathcal{M}_i\}_{1 \leq i \leq d}$ as the set of all subspaces $\mathcal{M}_i$ with fixed support i.e. $\operatorname{supp}(x) = \operatorname{supp}(y)$ *for all $x, y \in \mathcal{M}_i$. In case of this collection, identification result* (1.33) *can be reformulated as*

$$\operatorname{supp}(x^\star) \subseteq \operatorname{supp}(x^k) \subseteq \max \left\{ \operatorname{supp}(\mathbf{prox}_{\gamma r}(u)) \colon u \in \mathcal{B}(u^\star, \varepsilon) \right\}. \tag{1.37}$$

Proof. To prove this it is enough to show that

$$\mathsf{S}_{\mathcal{M}}(x) \leq \mathsf{S}_{\mathcal{M}}(y) \iff \operatorname{supp}(x) \subseteq \operatorname{supp}(y). \tag{1.38}$$

It follows from the definition that if $\mathcal{M}_i \subseteq \mathcal{M}_j$ then $(\mathsf{S}_{\mathcal{M}}(x))_{[i]} \leq (\mathsf{S}_{\mathcal{M}}(x))_{[j]}$ for any x. Using this fact let us prove this remark.

($\Rightarrow$) Let $\mathsf{S}_{\mathcal{M}}(x) \leq \mathsf{S}_{\mathcal{M}}(y)$ then for all $i = 1, \ldots, d$ it is true that $(\mathsf{S}_{\mathcal{M}}(x))_{[i]} \leq (\mathsf{S}_{\mathcal{M}}(y))_{[i]}$. Choosing j such that $\operatorname{supp}(\mathcal{M}_j) = \operatorname{supp}(y)$ we have $(\mathsf{S}_{\mathcal{M}}(x))_{[j]} = 0$ that means $\operatorname{supp}(x) \subseteq \operatorname{supp}(y)$.

($\Leftarrow$) $(\mathsf{S}_{\mathcal{M}}(y))_{[i]} = 0$ if $\operatorname{supp}(y) \subseteq \operatorname{supp}(\mathcal{M}_i)$. In this case, $\operatorname{supp}(x) \subseteq \operatorname{supp}(y) \subseteq \operatorname{supp}(\mathcal{M}_i)$ that implies $(\mathsf{S}_{\mathcal{M}}(x))_{[i]} = 0 = (\mathsf{S}_{\mathcal{M}}(y))_{[i]}$. On the other hand, if $(\mathsf{S}_{\mathcal{M}}(y))_{[i]} = 1$ then $(\mathsf{S}_{\mathcal{M}}(x))_{[i]} \leq (\mathsf{S}_{\mathcal{M}}(y))_{[i]}$ that finishes the proof of (1.38). □

The identification result in case of regularizer r that enforce (block) coordinate sparsity case can be clearly seen in the example of proximal gradient descent for ℓ_1 regularized problems.

In Figure 1-7 we can see how support changes during proximal gradient descent for LASSO problem

$$\min \ \frac{1}{2}\|Ax - b\|_2^2 + \lambda_1 \|x\|_1$$

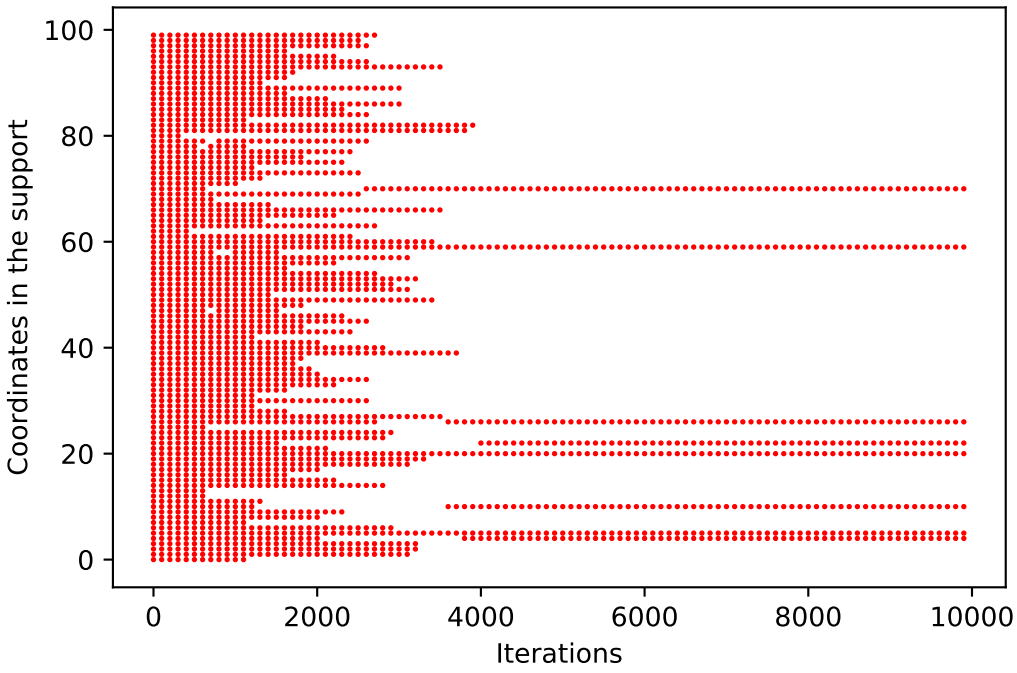


Figure 1-7. ℓ_1 identification

for random generated matrix $A \in \mathbb{R}^{100\times 100}$ and vector $b \in \mathbb{R}^{100}$ and hyperparameter λ_1 chosen to reach 8% of density (amount of non-zero coordinates) of the final solution. Starting from 500^{th} iteration, the support of iterates begins to change and becomes stable after 4000. Moreover, some of the coordinates from the optimal support disappeared after 2000 iterations and re-appears only after the optimal support identification.

Another important example of sparsity inducing regularizer is 1-d **TV** regularizer.

Example 1.19 (**TV** regularizer)**.** *It is known that if r is 1-d* **TV** *regularizer*

$$r(x) = \sum_{i=1}^{n-1} |x_{[i]} - x_{[i+1]}| \tag{1.39}$$

it enforces variation sparsity.

In this case the amount of jumps (blocks) in optimal solution $x^\star$ will be small, where

$$\text{jumps}(x) \triangleq \left\{ i \in [2, n] \,|\, x_{[i]} \neq x_{[i-1]} \right\}. \tag{1.40}$$

Let us define collection $\mathcal{M} = \{\mathcal{M}_i\}_{1\leq i\leq d}$ *as the set of all subspaces* $\mathcal{M}_i$ *with the fixed block structure i.e.* $\operatorname{jumps}(x) = \operatorname{jumps}(y)$ *for all* $x, y \in \mathcal{M}_i$. *In case of this collection, identification result* (1.33) *can be reformulated as*

$$\operatorname{jumps}(x^\star) \subseteq \operatorname{jumps}(x^k) \subseteq \max\left\{\operatorname{jumps}(\mathbf{prox}_{\gamma r}(u))\colon u \in \mathcal{B}(u^\star, \varepsilon)\right\}. \tag{1.41}$$

To prove this it is enough to show that

$$\mathsf{S}_{\mathcal{M}}(x) \leq \mathsf{S}_{\mathcal{M}}(y) \iff \operatorname{jumps}(x) \subseteq \operatorname{jumps}(y), \tag{1.42}$$

which proves exactly the same way as (1.38).

The Theorem 1.17 explains that iterates of any converging proximal algorithm will eventually be sandwiched between two extremes families of subspaces controlled by the pair $(x^\star, u^\star)$.

Under some qualifying constraints, the structure of the iterate $\mathsf{S}_{\mathcal{M}}(x^k)$ will coincide precisely with the one of the solution $\mathsf{S}_{\mathcal{M}}(x^\star)$; as a consequence the left and right terms in (1.33) are the same, and the identification result from the Theorem 1.17 could be specified as follows.

Corollary 1.20 (Exact identification)**.** *Consider the solution* $x^\star$ *of Problem* (1.23) *verifies the qualification constraint*

$$\mathsf{S}_{\mathcal{M}}(x^\star) = \max\left\{\mathsf{S}_{\mathcal{M}}(\mathbf{prox}_{\gamma r}(u))\colon u \in \mathcal{B}(x^\star - \gamma\nabla f(x^\star), \varepsilon)\right\} \tag{QC}$$

for any $\varepsilon > 0$ *small enough. Then, under the same assumptions as in Theorem 1.17 the sequence* (x^k) *identifies an optimal subspace with probability one, after some finite time*

$$\mathsf{S}_{\mathcal{M}}(x^\star) = \mathsf{S}_{\mathcal{M}}(x^k).$$

In general, this corresponds to the relative interior assumption of [Lew02]; see the extensive discussion of [VGFP15].

Proof of Corollary 1.20. Let $u^\star = x^\star - \gamma\nabla f(x^\star)$ and observe from the optimality conditions of (1.23) that $x^\star = \mathbf{prox}_{\gamma r}(u^\star)$. We apply Theorem 1.17 and the qualification condition (QC) ensures that the left and right-hand sides in (1.33) coincide: we get $\mathsf{S}_{\mathcal{M}}(x^k)$ will exactly reach $\mathsf{S}_{\mathcal{M}}(x^\star)$ in finite time. □

Remark 1.21 (Qualification constraint)**.** *The qualifying constraint* (QC) *may seem hard to verify at first glance but for most structure-enhancing regularizers, it simplifies greatly and reduces to usual nondegeneracy assumptions. Broadly speaking, this condition simply means that the point* $u^\star = x^\star - \gamma\nabla f(x^\star)$ *is not* borderline *to be put to an identified value*

by the proximity operator of the regularizer $\mathbf{prox}_{\gamma r}$. *For example, when* $r(x) = \lambda_1\|x\|_1$, *the qualifying constraint* (QC) *simply rewrites* $x_i^\star = 0 \Leftrightarrow \nabla_{[i]} f(x^\star) \in]-\lambda_1, \lambda_1[$; *for* r *is the TV-regularization* (1.39), *the qualifying constraint means that there is no point* u *(in any ball) around* $x^\star - \gamma\nabla f(x^\star)$ *such that* $\mathbf{prox}_{\gamma r}(u)$ *has a jump that* $x^\star$ *does not have.*

Let us present an example of exact identification result for ℓ_1 regularized composite optimization problem (1.23).

Example 1.22 (Exact identification for ℓ_1 regularizer)**.** *Let us consider that composite optimization problem*

$$\min_{x\in\mathbb{R}^n} f(x) + \lambda\|x\|_1,$$

where f *is* L*-smooth and convex is non-degenerate, that is*

$$-\nabla f(x^\star) \in \operatorname{ri} \partial r(x^\star). \tag{ND}$$

Then for $k > K$ *big enough the iterates* (x^k) *of Algorithm 4 identifies the optimal support with probability one*

$$\operatorname{supp}(x^k) = \operatorname{supp}(x^\star).$$

Proof. First, let us rewrite the non-degeneracy condition for ℓ_1 norm in closed form

$$\left|\nabla f(x^\star)_{[j]}\right| < \lambda \qquad \text{for all } j \in \operatorname{supp}(x^\star). \tag{1.43}$$

Denoting by $u^{k+1} = x^k - \gamma\nabla f(x^k)$, from the convergence of x^k to $x^\star$ we have that u^k converges to $u^\star = x^\star - \gamma\nabla f(x^\star)$.

Now, let us prove the qualification constraint

$$\operatorname{supp}(x^\star) = \max\left\{\operatorname{supp}(\mathbf{prox}_{\gamma\lambda\|\cdot\|_1}(u))\colon u \in \mathcal{B}(x^\star - \gamma\nabla f(x^\star), \varepsilon)\right\}.$$

Using the non-degeneracy of the problem we have that for any coordinate i such that $x^\star_{[i]} = 0$ we have

$$\left|u^\star_{[i]}\right| = \left|x^\star_{[i]} - \gamma\nabla f(x^\star)_{[i]}\right| = \left|\gamma\nabla f(x^\star)_{[i]}\right| = \epsilon_i < \gamma\lambda.$$

Selecting $\varepsilon = \min_{i\notin\operatorname{supp}(x^\star)}(\gamma\lambda - \epsilon_i)/2$ for any $u \in \mathcal{B}(x^\star - \gamma\nabla f(x^\star), \varepsilon)$ we have

$$\left|u_{[i]}\right| \le \left|x^\star_{[i]} - \gamma\nabla f(x^\star)_{[i]}\right| + \varepsilon \le \gamma\lambda \qquad \text{for all } i \notin \operatorname{supp}(x^\star)$$

that implies

$$\max\left\{\operatorname{supp}(\mathbf{prox}_{\gamma\lambda\|\cdot\|_1}(u))\colon u \in \mathcal{B}(x^\star - \gamma\nabla f(x^\star), \varepsilon)\right\} \subseteq \operatorname{supp}(x^\star).$$

Combining it with $x^\star = \mathbf{prox}_{\gamma\lambda\|\cdot\|_1}(x^\star - \gamma\nabla f(x^\star))$ we get the statement of the corollary. □

1.4.2 Identification of DAve-PG

No identification results have been yet reported in the literature for the asynchronous algorithms. However, the delay-tolerant analysis of DAve-PG [MIMA18] allows verifying the active-set identification property for the algorithm.

Corollary 1.23 (Identification of DAve-PG)**.** *Let the functions (f_i) be μ-strongly convex ($\mu > 0$) and L-smooth. Let r be convex lsc. Using $\gamma \in (0, \frac{2}{\mu+L}]$, iterates (x^k) of DAve-PG after finite amount of epochs identifies some subspaces with probability one.*

Proof of 1.23. Let us denote by $u^k = \bar{x}^k$. To use Theorem 1.17 we should find $u^\star$ and prove that u^k converges almost surely to $u^\star$. First, let us notice that the local iterates (x_i) do not converge to a minimizer of the individual functions (f_i) since the algorithm aims at minimizing the global loss but rather to local shifts of the solution $x^\star$ of (P) (unique from the strong convexity assumption):

$$x_i^\star = x^\star - \gamma\nabla f_i(x^\star) \qquad \text{for worker } i.$$

From those, one can define $u^\star = \bar{x}^\star = \sum_{i=1}^M \alpha_i x_i^\star$. First-order optimality conditions

$$0 \in \sum_i \alpha_i \nabla f_i(x^\star) + \partial r(x^\star)$$

imply that

$$\bar{x}^\star = \sum_{i=1}^M \alpha_i x_i^\star = x^\star - \gamma \sum_{i=1}^M \alpha_i \nabla f_i(x^\star) \in x^\star + \gamma\partial r(x^\star)$$

which directly leads to $\mathbf{prox}_{\gamma r}(\bar{x}^\star) = x^\star$ (see Chap. 16 of [BC11]). Now, using the definition of $\bar{x}$ we have

$$\|\bar{x}^k - \bar{x}^\star\|_2^2 = \|\sum_{i=1}^M \alpha_i(x_i^k - x_i^\star)\|_2^2 \leq \sum_{i=1}^M \alpha_i \|x_i^k - x_i^\star\|_2^2 \leq \max_{i=1}^m \|x_i^k - x_i^\star\|_2^2.$$

Now we could control the term $\|x_i^k - x_i^\star\|_2^2$ as follows

$$\begin{aligned}\|x_i^k - x_i^\star\|_2^2 = \|x_i^{k-d_i^k} - x_i^\star\|_2^2 &= \|x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - x^\star + \gamma\nabla f_i(x^\star)\|_2^2 \\ &\leq \left(1 - \frac{2\gamma\mu L}{\mu + L}\right) \|x^{k-D_i^k} - x^\star\|_2^2,\end{aligned}$$

where in the last inequality we use Lemma 1.5. Finally, using the result of Theorem 1.14 we have the convergence of $\bar{x}^k$ to $\bar{x}^\star$. □

Chapter 2

Automatic dimension reduction

Chapter Contents

Introduction

In this chapter, we consider composite optimization problems of the form

$$\min_{x \in \mathbb{R}^n} \; f(x) + r(x), \tag{2.1}$$

where f is convex and differentiable, and r is convex and nonsmooth (see Section 1.1.4).

As we already mentioned in Section 1.4, proximal methods can identify the structure of the final solution.

We propose randomized proximal algorithms leveraging on structure identification: our idea is to sample the variable space according to the structure of r (see Examples 1.18, 1.19). To do so, we first introduce a randomized descent algorithm going beyond separable nonsmoothness and associated coordinate descent methods: we consider "subspace descent" extending "coordinate descent" to generic subspaces. Then, we use a standard identification property of proximal methods to adapt our sampling of the subspaces with the identified structure. This results in a structure-adapted randomized method with automatic dimension reduction, which performs better in terms of dimensions explored compared to standard proximal methods and the non-adaptive version.

Though our main concern is the handling of non-separable nonsmooth functions r, we mention that our identification-based adaptive approach is different from existing adaptation strategies restricted to the particular case of coordinate descent methods. Indeed, adapting coordinate selection probabilities is an important topic for coordinate descent methods as both theoretical and practical rates heavily depend on them (see e.g. [RT14, NP14]). Though the optimal theoretical probabilities, named importance sampling, often depend on unknown quantities, these *fixed* probabilities can sometimes be computed and used in practice, see [ZZ15, RT16b]. The use of *adaptive* probabilities is more limited; some heuristics without convergence guarantees can be found in [LSS11, GD13], and greedy coordinates selection are usually expensive to compute [DRT11, NSL+15, NLS17]. Bridging the gap between greedy and fixed importance sampling, [CQR15] proposes primal-dual coordinate descent with adaptive coordinate sampling based on *dual residue*. More precisely, if the selection of the coordinate happens more often if the dual coordinate is suboptimal. Another way to adapt was proposed in [PCJ17, NSYD17, SRJ17] where probabilities in the coordinate descent methods based on the coordinate-wise Lipschitz constants and current values of the gradient.

The methods proposed in the present chapter, even when specialized in the coordinate descent case, are the first theoretical supported ones where the *iterate structure enforced by a non-smooth regularizer* is used to adapt the selection probabilities. This idea was initially proposed for ℓ_1 regularized problems in [RT14]; however, there was no theoretical guarantees for such probability selection.

Outline. In Section 2.1, we introduce the formalism for subspace descent methods. First, we formalize how to sample subspaces and introduce a first random subspace proximal gradient algorithm. Then, we show its convergence and derive its linear rate in the strongly convex case. Along the way, we make connections and comparisons with the literature on coordinate descent and sketching methods, notably in the special cases of ℓ_1 and total variation regularization. In Section 2.2, we present our identification-based adaptive algorithm. We begin by showing the convergence of an adaptive generalization of our former algorithm; next, we show that this algorithm enjoys some identification property and give practical methods to adapt the sampling, based on generated iterates, leading to refined rates. Finally, in Section 2.3, we report numerical experiments on popular learning problems to illustrate the merits and reach of the proposed methods.

This chapter corresponds to [GIM20].

2.1 Randomized subspace descent

The premise of randomized subspace descent consists in repeating two steps: i) randomly selecting some subspace; and ii) updating the iterate over the chosen subspace. Such algorithms thus extend usual coordinate descent (see Section 1.1.4) to general sampling strategies, which requires algorithmic changes and an associated mathematical analysis. This section presents a subspace descent algorithm along these lines for solving (2.1). In Section 2.1.1, we introduce our subspace selection procedure. We build on it to introduce, in Section 2.1.2, our first subspace descent algorithm, the convergence of which is analyzed in Section 2.1.3. Finally, we put this algorithm into perspective in Section 2.1.4 by connecting and comparing it to related work.

2.1.1 Subspace selection

We begin by introducing the mathematical objects leading to the subspace selection used in our randomized subspace descent algorithms. Though, in practice, most algorithms rely on projection matrices, our presentation highlights intrinsic subspaces associated to these matrices; this opens the way to a finer analysis, especially in Section 2.2.1 when working with adaptive subspaces.

We consider a family $\mathcal{C} = \{\mathcal{C}_i\}_i$ of (linear) subspaces of $\mathbb{R}^n$. Intuitively, this set represents the directions that will be *favored* by the random descent; in order to reach a global optimum, we naturally assume that the sum[1] of the subspaces in a family matches the whole space.

[1] In the definition and the following, we use the natural set addition (sometimes called the Minkowski sum): for any two sets $\mathcal{C}, \mathcal{D} \subseteq \mathbb{R}^n$, the set $\mathcal{C} + \mathcal{D}$ is defined as $\{x + y : x \in \mathcal{C}, y \in \mathcal{D}\} \subseteq \mathbb{R}^n$.

Definition 2.1 (Covering family of subspaces)**.** *Let $\mathcal{C} = \{\mathcal{C}_i\}_i$ be a family of subspaces of $\mathbb{R}^n$. We say that $\mathcal{C}$ is* covering *if it spans the whole space, i.e. if $\sum_i \mathcal{C}_i = \mathbb{R}^n$.*

Example 2.2. *The family of the axes $\mathcal{C}_i = \{x \in \mathbb{R}^n : x_{[j]} = 0 \quad \forall j \neq i\}$ for $i = 1, \dots, n$ is a canonical covering family for $\mathbb{R}^n$.*

From a covering family $\mathcal{C}$, we call *selection* the random subspace obtained by randomly choosing some subspaces in $\mathcal{C}$ and summing them. We call *admissible* the selections that include all directions with some positive probability; or, equivalently, the selections to which no non-zero element of $\mathbb{R}^n$ is orthogonal with probability one.In contrast with the definition of "proper sampling" [RT16c] we allow some probabilities to be equal to 0. In general, any proper sampling from covering family is an admissible selection (see Remark 2.5).

Definition 2.3 (Admissible selection)**.** *Let $\mathcal{C}$ be a covering family of subspaces of $\mathbb{R}^n$. A selection $\mathfrak{S}$ is defined from the set of all subsets of $\mathcal{C}$ to the set of the subspaces of $\mathbb{R}^n$ as*

$$\mathfrak{S}(\omega) = \sum_{j=1}^{s} \mathcal{C}_{i_j} \qquad \text{for } \omega = \{\mathcal{C}_{i_1}, \dots, \mathcal{C}_{i_s}\}.$$

The selection $\mathfrak{S}$ is admissible *if $\mathbb{P}[x \in \mathfrak{S}^\perp] < 1$ for all $x \in \mathbb{R}^n \setminus \{0\}$.*

Admissibility of selections appears on spectral properties of the average projection matrix onto the selected subspaces. For a subspace $F \subseteq \mathbb{R}^n$, we denote by $P_F \in \mathbb{R}^{n \times n}$ the orthogonal projection matrix onto F. The following lemma shows that the average projection associated with an admissible selection is positive definite; this matrix and its extreme eigenvalues will play a major role in our developments.

Lemma 2.4 (Average projection)**.** *If a selection $\mathfrak{S}$ is admissible then*

$$\mathsf{P} := \mathbb{E}[P_{\mathfrak{S}}] \qquad \text{is a positive definite matrix.} \tag{2.2}$$

In this case, we denote by $\lambda_{\min}(\mathsf{P}) > 0$ and $\lambda_{\max}(\mathsf{P}) \leq 1$ its minimal and maximal eigenvalues.

Proof. Note first that for almost all ω, the orthogonal projection $P_{\mathfrak{S}(\omega)}$ is positive semi-definite, and therefore so is P. Now, let us prove that if P is not positive definite, then $\mathfrak{S}$ is not admissible. Take a nonzero x in the kernel of P, then

$$x^\top \mathsf{P} x = 0 \iff x^\top \mathbb{E}[P_{\mathfrak{S}}] x = 0 \iff \mathbb{E}[x^\top P_{\mathfrak{S}} x] = 0.$$

Since $x^\top P_{\mathfrak{S}(\omega)} x \geq 0$ for almost all ω, the above property is further equivalent for almost all ω to

$$x^\top P_{\mathfrak{S}(\omega)} x = 0 \iff P_{\mathfrak{S}(\omega)} x = 0 \iff x \in \mathfrak{S}(\omega)^\perp.$$

Since $x \neq 0$, this yields that $x \in \mathfrak{S}(\omega)^\perp$ for almost all ω which is in contradiction with $\mathfrak{S}$ being admissible. Thus, if a selection $\mathfrak{S}$ is admissible, $\mathsf{P} := \mathbb{E}[P_{\mathfrak{S}}]$ is positive definite (so $\lambda_{\min}(\mathsf{P}) > 0$).

Finally, using Jensen's inequality and the fact that $P_{\mathfrak{S}}$ is a projection, we get $\|\mathsf{P}x\| = \|\mathbb{E}[P_{\mathfrak{S}}]x\| \leq \mathbb{E}\|P_{\mathfrak{S}}x\| \leq \|x\|$, which implies that $\lambda_{\max}(\mathsf{P}) \leq 1$. □

This result holds in the more general context and it is proven in [GR15].

Although the framework, methods, and results presented in this paper allow for infinite subspace families (as in sketching algorithms); the most direct applications of our results only call for finite families for which the notion of admissibility can be made simpler.

Remark 2.5 (Finite Subspace Families)**.** *For a covering family of subspaces $\mathcal{C}$ with a finite number of elements, the admissibility condition can be simplified to $\mathbb{P}[\mathcal{C}_i \subset \mathfrak{S}] > 0$ for all i.*

Indeed, take $x \in \mathbb{R}^n \setminus \{0\}$; then, since $\mathcal{C}$ is covering and $x \neq 0$, there is a subspace $\mathcal{C}_i$ such that $P_{\mathcal{C}_i}x \neq 0$. Observe now that $\mathcal{C}_i \subset \mathfrak{S}$ yields $P_{\mathfrak{S}}x \neq 0$ (since $\mathfrak{S}^\perp \subset \mathcal{C}_i^\perp$, the property $P_{\mathfrak{S}}x = 0$ would give $P_{\mathcal{C}_i}x = 0$ which is a contradiction with $P_{\mathcal{C}_i}x \neq 0$). Thus, we can write

$$\mathbb{P}[x \in \mathfrak{S}^\perp] = \mathbb{P}[P_{\mathfrak{S}}x = 0] = 1 - \mathbb{P}[P_{\mathfrak{S}}x \neq 0] \leq 1 - \mathbb{P}[\mathcal{C}_i \subset \mathfrak{S}] < 1.$$

Building on this property, two natural ways to generate admissible selections from a finite covering family $\mathcal{C} = \{\mathcal{C}_i\}_{i=1,\dots,c}$ are:

- Fixed probabilities: *Selecting each subspace $\mathcal{C}_i$ according to the outcome of a Bernoulli variable of parameter $p_i > 0$. This gives admissible selections as $\mathbb{P}[\mathcal{C}_i \subseteq \mathfrak{S}] = p_i > 0$ for all i;*
- Fixed sample size: *Drawing s subspaces in $\mathcal{C}$ uniformly at random without replacement. This gives admissible selections since $\mathbb{P}[\mathcal{C}_i \subseteq \mathfrak{S}] = s/c$ for all i.*

Example 2.6 (Coordinate-wise projections)**.** *Consider the family of the axes from Example 2.2 and the selection generated with fixed probabilities as described in Remark 2.5. The associated projections amount to zeroing entries at random and the average projection P is the diagonal matrix with entries (p_i); trivially $\lambda_{\min}(\mathsf{P}) = \min_i p_i$ and $\leq \lambda_{\max}(\mathsf{P}) = \max_i p_i$.*

2.1.2 Random subspace proximal gradient algorithm

An iteration of the proximal gradient algorithm ISTA (Algorithm 4) decomposes in gradient and proximal steps. In order to construct a "subspace" version of the proximal gradient, one has to determine which variable will be updated along the randomly chosen subspace (which we will call a projected update). In Section 1.1.4 we already presented Algorithm 6

where a projected update of x^k, i.e., projecting after the proximity operation is used. This choice has limited interest in the general case where the proximity operator is not separable along subspaces and thus a projected update of x^k still requires the computations of the full gradient. In the favorable case of coordinate projection and $r = \|\cdot\|_1$, it was studied in [QR16] using the fact that the projection and the proximity operator commute. Another possible choice is to project an update of $\nabla f(x^k)$, i.e. projecting after the gradient. This option is considered recently in [HMR18] in the slightly different context of sketching. A further discussion on related literature is postponed to Section 2.1.4. In this work, we will consider projecting after the gradient *step*.

This choice inspired by recent works highlighting that combining iterates usually works well in practice (see [MIM20] and references therein). However, taking gradient steps along random subspaces introduces bias and thus such a direct extension fails in practice. In order to retrieve convergence to the optimal solution of (2.1), we slightly modify the proximal gradient iterations by including a correction featuring the inverse square root of the expected projection denoted by $\mathsf{Q} = \mathsf{P}^{-1/2}$ (note that as soon as the selection is admissible, Q is well defined from Lemma 2.4).

Formally, our Random Proximal Subspace Descent algorithm `RPSD`, displayed as Algorithm 9, replaces the gradient step in Algorithm 4 by

$$y^k = \mathsf{Q}\left(x^k - \gamma \nabla f\left(x^k\right)\right) \qquad \text{and} \qquad z^k = P_{\mathfrak{S}^k}\left(y^k\right) + \left(I - P_{\mathfrak{S}^k}\right)\left(z^{k-1}\right). \tag{2.3}$$

That is, we propose to first perform a gradient step followed by a change of basis (by multiplication with the positive definite matrix Q), giving variable y^k; then, variable z^k is updated only in the random subspace $\mathfrak{S}^k$: to $P_{\mathfrak{S}^k}\left(y^k\right)$ in $\mathfrak{S}^k$, and keeping the same value outside. Note that y^k does not actually have to be computed and only the "$P_{\mathfrak{S}^k}\mathsf{Q}$-sketch" of the gradient (i.e. $P_{\mathfrak{S}^k}\mathsf{Q}\nabla f\left(x^k\right)$) is needed. Finally, the final proximal operation in forward-backward algorithm is performed after getting back to the original space (by multiplication with Q^{-1}):

$$x^{k+1} = \mathbf{prox}_{\gamma r}\left(\mathsf{Q}^{-1}\left(z^k\right)\right). \tag{2.4}$$

Contrary to existing coordinate descent methods, our randomized subspace proximal gradient algorithm does not assume that the proximity operator $\mathbf{prox}_{\gamma r}$ is separable with respect to the projection subspaces. In [HMR18] authors also propose a sketch-and-project method that converges without any assumption on the separability of the regularizer that is an uncommon but highly desirable feature to tackle general composite optimization problems. The key difference between our algorithm and the one proposed in [HMR18] is in the moment of projection; we project after the gradient step and in [HMR18] autors propose to make a projection of the gradient.

Let us provide a first example, before moving to the analysis of the algorithm in the

Algorithm 9 Randomized Proximal Subspace Descent - `RPSD`

Input: $\mathsf{Q} = \mathsf{P}^{-\frac{1}{2}}$
Initialize z^0, $x^1 = \mathbf{prox}_{\gamma r}(\mathsf{Q}^{-1}(z^0))$
for $k = 1, \ldots$ **do**
  $y^k = \mathsf{Q}\left(x^k - \gamma \nabla f\left(x^k\right)\right)$
  $z^k = P_{\mathfrak{S}^k}\left(y^k\right) + \left(I - P_{\mathfrak{S}^k}\right)\left(z^{k-1}\right)$
  $x^{k+1} = \mathbf{prox}_{\gamma r}\left(\mathsf{Q}^{-1}\left(z^k\right)\right)$
end for

next section.

Example 2.7 (Interpretation for smooth problems)**.** *In the case where $g \equiv 0$, our algorithm has two interpretations. First, using $\mathbf{prox}_{\gamma r} = I$, the iterations simplify to*

$$z^{k+1} = z^k - \gamma P_{\mathfrak{S}^k}\mathsf{Q}\left(\nabla f\left(\mathsf{Q}^{-1}\left(z^k\right)\right)\right) = z^k - \gamma P_{\mathfrak{S}^k}\mathsf{Q}^2 \underbrace{\mathsf{Q}^{-1}\left(\nabla f\left(\mathsf{Q}^{-1}\left(z^k\right)\right)\right)}_{\nabla f \circ \mathsf{Q}^{-1}(z^k)}.$$

As $\mathbb{E}[P_{\mathfrak{S}^k}\mathsf{Q}^2] = I$, this corresponds to a random subspace descent on $f \circ (\mathsf{Q}^{-1})$ with unbiased gradients. Second, we can write it with the change of variable $u^k = \mathsf{Q}^{-1}z^k$ as

$$u^{k+1} = u^k - \gamma \mathsf{Q}^{-1} P_{\mathfrak{S}^k}\mathsf{Q}\left(\nabla f\left(u^k\right)\right).$$

As $\mathbb{E}[\mathsf{Q}^{-1}P_{\mathfrak{S}^k}\mathsf{Q}] = \mathsf{P}$, this corresponds to random subspace descent on f but with biased gradient. We note that the recent work [FR15] considers a similar set-up and algorithm; however, the provided convergence result does not lead to the convergence to the optimal solution (due to the use of the special semi-norm).

2.1.3 Analysis and convergence rate

In this section, we provide a theoretical analysis for `RPSD`, showing linear convergence for strongly convex objectives.

Assumption 2.8 (On the optimization problem)**.** *The function f is L-smooth and μ-strongly convex and the function r is convex, proper, and lower-semicontinuous.*

Note that this assumption implies that Problem (2.1) has a unique solution that we denote $x^\star$ in the following.

Assumption 2.9 (On the randomness of the algorithm)**.** *Given a covering family $\mathcal{C} = \{\mathcal{C}_i\}$ of subspaces, we consider a sequence $\mathfrak{S}^1, \mathfrak{S}^2, \ldots, \mathfrak{S}^k$ of admissible selections, which is i.i.d.*

In the following theorem, we show that the proposed algorithm converges linearly at a rate that only depends on the function properties and on the smallest eigenvalue of P. We also emphasize that the step size γ can be taken in the usual range for proximal gradient descent.

Theorem 2.10 (RPSD convergence rate)**.** *Let Assumptions 2.8 and 2.9 hold. Then, for any $\gamma \in (0, 2/(\mu + L)]$, the sequence (x^k) of the iterates of RPSD converges almost surely to the minimizer $x^\star$ of (2.1) with rate*

$$\mathbb{E}\left[\|x^{k+1} - x^\star\|_2^2\right] \leq \left(1 - \lambda_{\min}(\mathsf{P})\frac{2\gamma\mu L}{\mu + L}\right)^k C,$$

where $C = \lambda_{\max}(\mathsf{P})\|z^0 - \mathsf{Q}(x^\star - \gamma\nabla f(x^\star))\|_2^2$.

To prove this result, we first demonstrate two intermediate lemmas respectively expressing the distance of z^k towards its fixed points (conditionally to the filtration of the past random subspaces $\mathcal{F}^k = \sigma(\{\mathfrak{S}_\ell\}_{\ell \leq k})$), and bounding the increment (with respect to $\|x\|_{\mathsf{P}}^2 = \langle x, \mathsf{P}x\rangle$ the norm associated to P).

Lemma 2.11 (Expression of the decrease as a martingale)**.** *From the minimizer $x^\star$ of (2.1), define the fixed points $z^\star = y^\star = \mathsf{Q}\left(x^\star - \gamma\nabla f\left(x^\star\right)\right)$ of the sequences (y^k) and (z^k). If Assumption 2.9 holds, then*

$$\mathbb{E}\left[\|z^k - z^\star\|_2^2 \,|\, \mathcal{F}^{k-1}\right] = \|z^{k-1} - z^\star\|_2^2 + \|y^k - y^\star\|_{\mathsf{P}}^2 - \|z^{k-1} - z^\star\|_{\mathsf{P}}^2.$$

Proof of Lemma 2.11. By taking the expectation on $\mathfrak{S}^k$ (conditionally to the past), we get

$$\begin{aligned}
&\mathbb{E}\left[\|z^k - z^\star\|_2^2 | \,\mathcal{F}^{k-1}\right] = \mathbb{E}\left[\|z^{k-1} - z^\star + P_{\mathfrak{S}^k}(y^k - z^{k-1})\|_2^2 | \,\mathcal{F}^{k-1}\right] \\
&\quad = \|z^{k-1} - z^\star\|_2^2 + 2\mathbb{E}\left[\langle z^{k-1} - z^\star, P_{\mathfrak{S}^k}(y^k - z^{k-1})\rangle | \,\mathcal{F}^{k-1}\right] + \mathbb{E}\left[\left\|P_{\mathfrak{S}^k}(y^k - z^{k-1})\right\|^2 | \,\mathcal{F}^{k-1}\right] \\
&\quad = \|z^{k-1} - z^\star\|_2^2 + 2\langle z^{k-1} - z^\star, \mathsf{P}(y^k - z^{k-1})\rangle + \mathbb{E}\left[\langle P_{\mathfrak{S}^k}(y^k - z^{k-1}), P_{\mathfrak{S}^k}(y^k - z^{k-1})\rangle | \,\mathcal{F}^{k-1}\right] \\
&\quad = \|z^{k-1} - z^\star\|_2^2 + 2\langle z^{k-1} - z^\star, \mathsf{P}(y^k - z^{k-1})\rangle + \mathbb{E}\left[\langle y^k - z^{k-1}, P_{\mathfrak{S}^k}(y^k - z^{k-1})\rangle | \,\mathcal{F}^{k-1}\right] \\
&\quad = \|z^{k-1} - z^\star\|_2^2 + \langle z^{k-1} + y^k - 2z^\star, \mathsf{P}(y^k - z^{k-1})\rangle,
\end{aligned}$$

where we used the fact that z^{k-1} and y^k are $\mathcal{F}^{k-1}$-measurable and that $P_{\mathfrak{S}^k}$ is a projection matrix so $P_{\mathfrak{S}^k} = P_{\mathfrak{S}^k}^\top = P_{\mathfrak{S}^k}^2$.

Then, using the fact $y^\star = z^\star$, the scalar product above can be simplified as follows

$$\begin{aligned}\langle z^{k-1} + y^k - 2z^\star, \mathsf{P}(y^k - z^{k-1})\rangle &= \langle z^{k-1} + y^k - z^\star - y^\star, \mathsf{P}(y^k - z^{k-1} + y^\star - z^\star)\rangle \\ &= -\langle z^{k-1} - z^\star, \mathsf{P}(z^{k-1} - z^\star)\rangle + \langle z^{k-1} - z^\star, \mathsf{P}(y^k - y^\star)\rangle \\ &\quad + \langle y^k - y^\star, \mathsf{P}(y^k - y^\star)\rangle - \langle y^k - y^\star, \mathsf{P}(z^{k-1} - z^\star)\rangle \\ &= \langle y^k - y^\star, \mathsf{P}(y^k - y^\star)\rangle - \langle z^{k-1} - z^\star, \mathsf{P}(z^{k-1} - z^\star)\rangle\end{aligned}$$

where we used in the last equality that P is symmetric. □

Lemma 2.12 (Contraction property in P-weighted norm)**.** *From the minimizer $x^\star$ of (2.1), define the fixed points $z^\star = y^\star = \mathsf{Q}\left(x^\star - \gamma\nabla f\left(x^\star\right)\right)$ of the sequences (y^k) and (z^k). If Assumptions 2.8 and 2.9 hold, then*

$$\|y^k - y^\star\|_{\mathsf{P}}^2 - \|z^{k-1} - z^\star\|_{\mathsf{P}}^2 \leq -\lambda_{\min}(\mathsf{P})\frac{2\gamma\mu L}{\mu + L}\|z^{k-1} - z^\star\|_2^2.$$

Proof of Lemma 2.12. First, using the definition of y^k and $y^\star$,

$$\begin{aligned}\|y^k - y^\star\|_{\mathsf{P}}^2 &= \langle \mathsf{Q}(x^k - \gamma\nabla f(x^k) - x^\star + \gamma\nabla f(x^\star)), \mathsf{P}\mathsf{Q}(x^k - \gamma\nabla f(x^k) - x^\star + \gamma\nabla f(x^\star))\rangle \\ &= \langle x^k - \gamma\nabla f(x^k) - x^\star + \gamma\nabla f(x^\star), \mathsf{Q}^\top\mathsf{P}\mathsf{Q}(x^k - \gamma\nabla f(x^k) - x^\star + \gamma\nabla f(x^\star))\rangle \\ &= \left\|x^k - \gamma\nabla f(x^k) - (x^\star - \gamma\nabla f(x^\star))\right\|_2^2.\end{aligned}$$

Using the standard stepsize range $\gamma \in (0, 2/(\mu + L)]$, one has (see Lemma 1.5)

$$\|y^k - y^\star\|_{\mathsf{P}}^2 = \left\|x^k - \gamma\nabla f(x^k) - (x^\star - \gamma\nabla f(x^\star))\right\|_2^2 \leq \left(1 - \frac{2\gamma\mu L}{\mu + L}\right)\|x^k - x^\star\|_2^2.$$

Using the non-expansivity of the proximity operator of convex l.s.c. function r (see Lemma 1.11) along with the fact that as $x^\star$ is a minimizer of (2.1), $x^\star = \mathbf{prox}_{\gamma r}(x^\star - \gamma\nabla f(x^\star)) = \mathbf{prox}_{\gamma r}(\mathsf{Q}^{-1}z^\star)$ [BC11, Th. 26.2], we get

$$\begin{aligned}\|x^k - x^\star\|_2^2 &= \|\mathbf{prox}_{\gamma r}(\mathsf{Q}^{-1}(z^{k-1})) - \mathbf{prox}_{\gamma r}(\mathsf{Q}^{-1}(z^\star))\|_2^2 \leq \|\mathsf{Q}^{-1}(z^{k-1} - z^\star)\|_2^2 \\ &= \langle \mathsf{Q}^{-1}(z^{k-1} - z^\star), \mathsf{Q}^{-1}(z^{k-1} - z^\star)\rangle = \langle z^{k-1} - z^\star, \mathsf{P}(z^{k-1} - z^\star)\rangle = \|z^{k-1} - z^\star\|_{\mathsf{P}}^2\end{aligned}$$

where we used that $\mathsf{Q}^{-\top}\mathsf{Q}^{-1} = \mathsf{Q}^{-2} = \mathsf{P}$. Combining the previous equations, we get

$$\|y^k - y^\star\|_{\mathsf{P}}^2 - \|z^{k-1} - z^\star\|_{\mathsf{P}}^2 \leq -\frac{2\gamma\mu L}{\mu + L}\|z^{k-1} - z^\star\|_{\mathsf{P}}^2.$$

Finally, the fact that $\|x\|_{\mathsf{P}}^2 \geq \lambda_{\min}(\mathsf{P})\|x\|_2^2$ for positive definite matrix P enables to get the

claimed result. □

Relying on these two lemmas, we are now able to prove Theorem 2.10 by showing that the distance of z^k towards the minimizer is a contracting super-martingale.

Proof of Theorem 2.10. Combining Lemmas 2.11 and 2.12, we get

$$\mathbb{E}\left[\|z^k - z^\star\|_2^2 \,|\, \mathcal{F}^{k-1}\right] \leq \left(1 - \lambda_{\min}(\mathsf{P})\frac{2\gamma\mu L}{\mu + L}\right)\|z^{k-1} - z^\star\|_2^2$$

and thus by taking the full expectation and using nested filtrations $(\mathcal{F}^k)$, we obtain

$$\mathbb{E}\left[\|z^k - z^\star\|_2^2\right] \leq \left(1 - \lambda_{\min}(\mathsf{P})\frac{2\gamma\mu L}{\mu + L}\right)^k \|z^0 - z^\star\|_2^2 = \left(1 - \lambda_{\min}(\mathsf{P})\frac{2\gamma\mu L}{\mu + L}\right)^k \|z^0 - \mathsf{Q}(x^\star - \gamma\nabla f(x^\star))\|_2^2.$$

Using the same arguments as in the proof of Lemma 2.12, one has

$$\|x^{k+1} - x^\star\|_2^2 \leq \|z^k - z^\star\|_{\mathsf{P}}^2 \leq \lambda_{\max}(\mathsf{P})\|z^k - z^\star\|_2^2$$

which enables to conclude

$$\mathbb{E}\left[\|x^{k+1} - x^\star\|_2^2\right] \leq \left(1 - \lambda_{\min}(\mathsf{P})\frac{2\gamma\mu L}{\mu + L}\right)^k \lambda_{\max}(\mathsf{P})\|z^0 - \mathsf{Q}(x^\star - \gamma\nabla f(x^\star))\|_2^2.$$

Finally, this linear convergences implies the almost sure convergence of (x^k) to $x^\star$ as

$$\mathbb{E}\left[\sum_{k=1}^{+\infty}\|x^{k+1} - x^\star\|^2\right] \leq C\sum_{k=1}^{+\infty}\left(1 - \lambda_{\min}(\mathsf{P})\frac{2\gamma\mu L}{\mu + L}\right)^k < +\infty$$

implies that $\sum_{k=1}^{+\infty}\|x^{k+1} - x^\star\|^2$ is finite with probability one. Thus we get

$$1 = \mathbb{P}\left[\sum_{k=1}^{+\infty}\|x^{k+1} - x^\star\|^2 < +\infty\right] \leq \mathbb{P}\left[\|x^k - x^\star\|^2 \to 0\right]$$

which in turn implies that (x^k) converges almost surely to $x^\star$. □

2.1.4 Examples and connections with existing work

In this section, we derive specific cases and discuss the relation between our algorithm and related literature.

Projections onto coordinates

A simple instantiation of our setting can be obtained by considering projections onto uniformly chosen coordinates (Example 2.6); with the family

$$\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_n\} \quad \text{with } \mathcal{C}_i = \{x \in \mathbb{R}^n : x_{[j]} = 0 \ \ \forall j \neq i\}$$

and the selection $\mathfrak{S}$ consisting of taking $\mathcal{C}_i$ according to the output of a Bernoulli experiment of parameter p_i. Then, the matrices $\mathsf{P} = \mathrm{diag}([p_1, \dots, p_n])$, $P_{\mathfrak{S}^k}$ and Q commute, and, by a change of variables $\tilde{z}^k = \mathsf{Q}^{-1} z^k$ and $\tilde{y}^k = \mathsf{Q}^{-1} y^k$, Algorithm 9 boils down to

$$\tilde{y}^k = x^k - \gamma \nabla f\left(x^k\right) \qquad \tilde{z}^k = P_{\mathfrak{S}^k}\left(\tilde{y}^k\right) + (I - P_{\mathfrak{S}^k})\left(\tilde{z}^{k-1}\right), \qquad x^{k+1} = \mathbf{prox}_{\gamma r}\left(\tilde{z}^k\right),$$

i.e., no change of basis is needed anymore, even if r is non-separable. Furthermore, the convergence rate simplifies to $(1 - 2\min_i p_i \gamma \mu L/(\mu + L))$, which translates to $(1 - 4\min_i p_i \mu L/(\mu + L)^2)$ for the optimal choice of stepsize $\gamma = 2/(\mu + L)$.

In the special case where r is separable (i.e. $r(x) = \sum_{i=1}^n r_i(x_{[i]})$), we can further simplify the iteration. In this case, projection and proximal steps commute, so that the iteration can be written

$$x^{k+1} = P_{\mathfrak{S}^k} \mathbf{prox}_{\gamma r}\left(x^k - \gamma \nabla f(x^k)\right) + (I - P_{\mathfrak{S}^k}) x^k$$

$$\text{i.e. } x_{[i]}^{k+1} = \begin{cases} \mathbf{prox}_{\gamma r_i}\left(x_{[i]}^k - \gamma \nabla_{[i]} f(x^k)\right) & \text{if } i \in \mathfrak{S}^k \\ x_{[i]}^k & \text{elsewhere} \end{cases}$$

which boils down to the usual (proximal) coordinate descent algorithm (see Algorithm 6), that recently experienced a rebirth in the context of huge-scale optimization, see [Tse01], [Nes12], [RT14] or [Wri15]. In this special case, the theoretical convergence rate of `RPSD` is close to the existing rates in the literature. For clarity, we compare with the uniform randomized coordinate descent of [RT14] (more precisely Th. 6 with $L_i = L$, $B_i = 1$, $\mu L \leq 2$) which can be written as $(1 - \mu L/4n)$ in ℓ_2-norm. The rate of `RPSD` in the same uniform setting (Example 2.6 with $p_i = p = 1/n$) is $\left(1 - \frac{4\mu L}{n(\mu+L)^2}\right)$ with the optimal step-size.

Projections onto vectors of fixed variations

The vast majority of randomized subspace methods consider the coordinate-wise projections treated in 2.1.4. This success is notably due to the fact that most problems onto which they are applied have naturally a coordinate-wise structure; for instance, due to the structure of r (ℓ_1-norm, group lasso, etc). However, many problems in signal processing and machine learning feature a very different structure. A typical example is when r is

the 1D-Total Variation

$$r(x) = \sum_{i=2}^{n} |x_{[i]} - x_{[i-1]}|, \tag{2.5}$$

featured for instance in the fused lasso problem [TSR+05] (see Example 1.19). In order to project onto subspaces of vectors of fixed variation (i.e., vectors for which $x_{[j]} = x_{[j+1]}$ except for a prescribed set of indices), one can define the covering family

$$\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_{n-1}\} \quad \text{with } \mathcal{C}_i = \left\{x \in \mathbb{R}^n : x_{[j]} = x_{[j+1]} \text{ for all } j \in \{1, \dots, n-1\} \setminus \{i\}\right\}$$

and an admissible selection $\mathfrak{S}$ consisting in selecting uniformly s elements in $\mathcal{C}$. Then, if $\mathfrak{S}$ selects $\mathcal{C}_{n_1}, \dots, \mathcal{C}_{n_s}$, the update will live in the sum of these subspaces, i.e. the subspace of the vectors having jumps at coordinates $n_1, n_2, \dots, n_s$. Thus, the associated projection in the algorithm writes

$$P_{\mathfrak{S}} = \begin{pmatrix} \overbrace{\begin{matrix} \frac{1}{n_1} & \dots & \frac{1}{n_1} \\ \vdots & \ddots & \vdots \\ \frac{1}{n_1} & \dots & \frac{1}{n_1} \\ 0 & \dots & 0 \\ \vdots & \ddots & \ddots \\ \vdots & \ddots & \ddots \\ \vdots & \ddots & \ddots \\ 0 & \dots & \dots \end{matrix}}^{n_1} & \begin{matrix} 0 \\ \vdots \\ 0 \\ \ddots \\ \ddots \\ \ddots \\ \ddots \\ \dots \end{matrix} & \begin{matrix} \dots \\ \ddots \\ \ddots \\ \ddots \\ \ddots \\ 0 \\ \vdots \\ 0 \end{matrix} & \overbrace{\begin{matrix} \dots & 0 \\ \ddots & \vdots \\ \ddots & \vdots \\ \ddots & \vdots \\ \ddots & \vdots \\ \dots & 0 \\ \dots & \frac{1}{n-n_s} \\ \ddots & \vdots \\ \dots & \frac{1}{n-n_s} \end{matrix}}^{n-n_s} \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} n_1 \\ \\ \\ \\ \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} n - n_s \end{matrix} \tag{2.6}$$

Note also that $P_{\mathfrak{S}}x$ has the same value for coordinates $[n_i, n_{i+1})$, equal to the average of these values.

As mentioned above, the similarity between the structure of the optimization problem and the one of the subspace descent is fundamental for performance in practice. In Section 2.2.3, we exploit the identification properties of the proximity operator in order to automatically adapt the subspace selection, which leads to a tremendous gain in performance.

Comparison with sketching

In sharp contrast with the existing literature, our subspace descent algorithm handles non-separable regularizers r. A notable exception is the algorithm called `SEGA` [HMR18], a random sketch-and-project proximal algorithm, that can also deal with non-separable regularizers. While the algorithm shares similar components with ours, the main differences between the two algorithms are

- biasedness of the gradient: `SEGA` deals with unbiased gradients while they are biased for `RPSD`;
- projection type: `SEGA` projects the gradient while we project after a gradient step (option (b) vs. option (c) in the discussion starting Section 3.1.1).

These differences are fundamental and create a large gap in terms of target, analysis and performance between the two algorithms. The practical comparison is illustrated in Section 2.3.2.

2.2 Adaptive subspace descent

This section presents an extension of our randomized subspace descent algorithm to the setting in which the projections are iterate-dependent. Our aim is to automatically adapt to the structure identified by the iterates along the run of the algorithm.

The methods proposed here are, up to our knowledge, the first theoretically supported ones where the iterate structure enforced by a non-smooth regularizer is used to adapt the selection probabilities in a randomized first-order method. As discussed in the introduction, even for the special case of coordinate descent, our approach is different from existing techniques that use fixed arbitrary probabilities [RT14, NP14], greedy selection [DRT11, NSL+15, NLS17], or adaptive selection based on dual residue [CQR15] or on the coordinate-wise Lipschitz constant and coordinates [PCJ17, NSYD17, SRJ17].

We present our adaptive subspace descent algorithm in two steps. First, we introduce in Section 2.2.1 a generic algorithm with varying selections and establish its convergence. Second, in Section 2.2.2, we provide a simple general identification result. We then combine these two results to provide an efficient adaptive method in Section 2.2.3.

2.2.1 Random Subspace Descent with time-varying selection

For any randomized algorithm, using iterate-dependent sampling would automatically break down the usual i.i.d. assumption. In our case, adapting to the current iterate structure means that the associated random variable depends on the past. We thus need further analysis and notation.

In the following, we use the subscript ℓ to denote the ℓ-th change in the selection. We denote by L the set of time indices at which an adaptation is made, themselves denoted by $k_\ell = \min\{k > k_{\ell-1} : k \in \mathsf{L}\}$.

In practice, at each time k, there are two decisions to make (see Section 2.2.3): (i) *if* an adaptation should be performed; and (ii) *how* to update the selection. Thus, we replace the i.i.d. assumption of Assumption 2.9 with the following one.

Assumption 2.13 (On the randomness of the adaptive algorithm)**.** *For all $k > 0$, $\mathfrak{S}^k$ is $\mathcal{F}^k$-measurable and admissible. Furthermore, if $k \notin \mathsf{L}$, $(\mathfrak{S}^k)$ is independent and identically distributed on $[k_\ell, k]$. The decision to adapt or not at time k is $\mathcal{F}^k$-measurable, i.e. $(k_\ell)_\ell$ is a sequence of $\mathcal{F}^k$-stopping times.*

Under this assumption, we can prove the convergence of the varying-selection random subspace descent, Algorithm 10. A generic result is given in Theorem 2.14 and a simple specification in the following example. The rationale of the proof is that the stability of the algorithm is maintained when adaptation is performed sparingly.

Algorithm 10 Adaptive Randomized Proximal Subspace Descent - `ARPSD`

Initialize z^0, $x^1 = \mathbf{prox}_{\gamma g}(\mathsf{Q}_0^{-1}(z^0))$, $\ell = 0$, $\mathsf{L} = \{0\}$.
for $k = 1, \dots$ **do**
  $y^k = \mathsf{Q}_\ell \left(x^k - \gamma \nabla f\left(x^k\right)\right)$
  $z^k = P_{\mathfrak{S}^k}\left(y^k\right) + \left(I - P_{\mathfrak{S}^k}\right)\left(z^{k-1}\right)$
  $x^{k+1} = \mathbf{prox}_{\gamma g}\left(\mathsf{Q}_\ell^{-1}\left(z^k\right)\right)$
  if an adaptation is decided **then**
    $\mathsf{L} \leftarrow \mathsf{L} \cup \{k+1\}$, $\ell \leftarrow \ell + 1$
    Generate a new admissible selection
    Compute $\mathsf{Q}_\ell = \mathsf{P}_\ell^{-\frac{1}{2}}$ and Q_ℓ^{-1}
    Rescale $z^k \leftarrow \mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1} z^k$
  end if
end for

Theorem 2.14 (`ARPSD` convergence)**.** *Let Assumptions 2.8 and 2.13 hold. For any $\gamma \in (0, 2/(\mu + L)]$, let the user choose its adaptation strategy so that:*

- *the* adaptation cost *is upper bounded by a deterministic sequence: $\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\|_2^2 \leq \mathbf{a}_\ell$;*
- *the* inter-adaptation time *is lower bounded by a deterministic sequence: $k_\ell - k_{\ell-1} \geq \mathbf{c}_\ell$;*
- *the* selection uniformity *is lower bounded by a deterministic sequence: $\lambda_{\min}(\mathsf{P}_{\ell-1}) \geq \lambda_{\ell-1}$;*

then, from the previous instantaneous rate $1 - \alpha_{\ell-1} := 1 - 2\gamma\mu L\lambda_{\ell-1}/(\mu + L)$*, the* corrected rate *for cycle ℓ writes*

$$(1 - \beta_\ell) := (1 - \alpha_{\ell-1})\mathbf{a}_\ell^{1/\mathbf{c}_\ell}. \tag{2.7}$$

Then, we have for any $k \in [k_\ell, k_{\ell+1})$

$$\mathbb{E}\left[\|x^{k+1} - x^\star\|_2^2\right] \leq (1-\alpha_\ell)^{k-k_\ell} \prod_{m=1}^{\ell} (1-\beta_m)^{\mathbf{c}_m} \|z^0 - \mathsf{Q}_0\left(x^\star - \gamma \nabla f\left(x^\star\right)\right)\|_2^2.$$

This theorem means that by balancing the magnitude of the adaptation (i.e., $\mathbf{a}_m$) with the time before adaptation (i.e., $\mathbf{c}_m$) from the knowledge of the current rate $(1-\alpha_{m-1})$, one can retrieve the exponential convergence with a controlled degraded rate $(1-\beta_m)$. This result is quite generic, but it can be easily adapted to specific situations. For instance, we provide a simple example with a global rate on the iterates in the forthcoming Example 2.15.

For now, let us turn to the proof of the theorem. To ease its reading, the main notations and measurability relations are depicted in Figure 2-1.

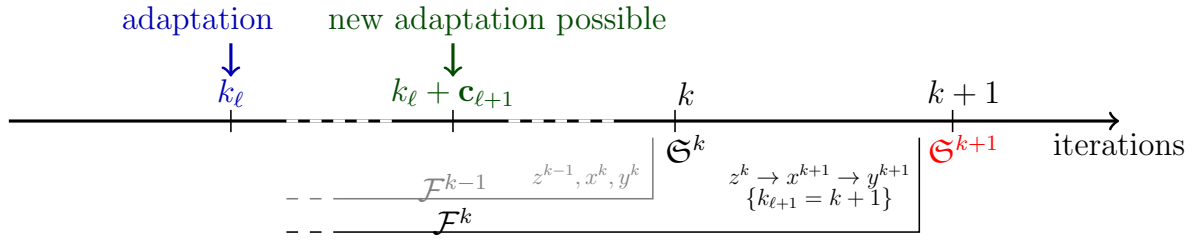


Figure 2-1. Summary of notations about iteration, adaptation and filtration. The filtration $\mathcal{F}^{k-1}$ is the sigma-algebra generated by $\{\mathfrak{S}^\ell\}_{\ell \leq k-1}$ encompassing the knowledge of all variables up to y^k (but not z^k).

Proof of Theorem 2.14. We start by noticing that, for a solution $x^\star$ of (2.1), the proof of Theorem 2.10 introduces the companion variable $z^\star = \mathsf{Q}\left(x^\star - \gamma \nabla f\left(x^\star\right)\right)$ which directly depends on Q, preventing us from a straightforward use of the results of Section 2.1.3. However, defining $z_\ell^\star = \mathsf{Q}_\ell\left(x^\star - \gamma \nabla f\left(x^\star\right)\right)$, Lemmas 2.11 and 2.12 can be directly extended and combined to show for any $k \in [k_\ell, k_{\ell+1})$

$$\mathbb{E}\left[\|z^k - z_\ell^\star\|_2^2 \,|\, \mathcal{F}^{k-1}\right] \leq \underbrace{\left(1 - \frac{2\gamma\mu L \lambda_{\min}(\mathsf{P}_\ell)}{\mu + L}\right)}_{\leq 1-\alpha_\ell} \|z^{k-1} - z_\ell^\star\|_2^2. \tag{2.8}$$

Since the distribution of the selection has not changed since k_ℓ, iterating (2.8) leads to

$$\mathbb{E}\left[\|z^k - z_\ell^\star\|_2^2 \,|\, \mathcal{F}^{k_\ell-1}\right] \leq (1-\alpha_\ell)^{k-k_\ell} \|z^{k_\ell-1} - z_\ell^\star\|_2^2. \tag{2.9}$$

We focus now on the term $\|z^{k_\ell-1} - z_\ell^\star\|_2^2$ corresponding to what happens at the last adaptation step. From the definition of variables in the algorithm and using the deterministic bound on $\|\mathsf{Q}_\ell\mathsf{Q}_{\ell-1}^{-1}\|$, we write

$$\begin{aligned}\mathbb{E}\left[\|z^{k_\ell-1} - z_\ell^\star\|_2^2 \,|\, \mathcal{F}^{k_\ell-2}\right] &\leq \mathbb{E}\left[\|\mathsf{Q}_\ell\mathsf{Q}_{\ell-1}^{-1}(z^{k_\ell-2} + P_{k_\ell-1}(y^{k_\ell-1} - z^{k_\ell-2}) - \mathsf{Q}_\ell\mathsf{Q}_{\ell-1}^{-1}z_{\ell-1}^\star\|_2^2 \,|\, \mathcal{F}^{k_\ell-2}\right] \\ &\leq \mathbb{E}\left[\|\mathsf{Q}_\ell\mathsf{Q}_{\ell-1}^{-1}\|_2^2\|z^{k_\ell-2} + P_{k_\ell-1}(y^{k_\ell-1} - z^{k_\ell-2}) - z_{\ell-1}^\star\|_2^2 \,|\, \mathcal{F}^{k_\ell-2}\right] \\ &\leq \mathbf{a}_\ell(1-\alpha_{\ell-1})\|z^{k_\ell-2} - z_{\ell-1}^\star\|_2^2.\end{aligned} \tag{2.10}$$

Repeating this inequality backward to the previous adaptation step $z^{k_{\ell-1}}$, we get

$$\begin{aligned}\mathbb{E}\left[\|z^{k_\ell-1} - z_\ell^\star\|_2^2 \,|\, \mathcal{F}^{k_{\ell-1}}\right] &\leq \mathbf{a}_\ell(1-\alpha_{\ell-1})^{k_\ell-k_{\ell-1}}\|z^{k_{\ell-1}} - z_{\ell-1}^\star\|_2^2 \\ &\leq \mathbf{a}_\ell(1-\alpha_{\ell-1})^{\mathbf{c}_\ell}\|z^{k_{\ell-1}} - z_{\ell-1}^\star\|_2^2,\end{aligned} \tag{2.11}$$

using the assumption of bounded inter-adaptation times. Combining this inequality and (2.9), we obtain that for any $k \in [k_\ell, k_{\ell+1})$,

$$\mathbb{E}\left[\|z^k - z_\ell^\star\|_2^2\right] \leq (1-\alpha_\ell)^{k-k_\ell}\prod_{m=1}^{\ell}\mathbf{a}_m(1-\alpha_{m-1})^{\mathbf{c}_m}\|z^0 - z_0^\star\|_2^2.$$

Using now (2.7), we get

$$\mathbb{E}\left[\|z^k - z_\ell^\star\|_2^2\right] \;\leq\; (1-\alpha_\ell)^{k-k_\ell}\prod_{m=1}^{\ell}(1-\beta_m)^{\mathbf{c}_m}\|z^0 - z_0^\star\|_2^2$$

Finally, the non-expansiveness of the prox-operator propagates this inequality to x_k, since we have

$$\begin{aligned}\|x^k - x^\star\|_2^2 = \|\mathbf{prox}_{\gamma g}(\mathsf{Q}_\ell^{-1}(z^{k-1})) - \mathbf{prox}_{\gamma g}(\mathsf{Q}_\ell^{-1}(z_\ell^\star))\|_2^2 &\leq \|\mathsf{Q}_\ell^{-1}(z^{k-1} - z_\ell^\star)\|_2^2 \\ \leq \lambda_{\max}(\mathsf{Q}_\ell^{-1})^2\|z^{k-1} - z_\ell^\star\|_2^2 = \lambda_{\max}(\mathsf{P}_\ell)\|z^{k-1} - z_\ell^\star\|_2^2 &\leq \|z^{k-1} - z_\ell^\star\|_2^2.\end{aligned}$$

This concludes the proof. □

Example 2.15 (Explicit convergence rate)**.** *Let us specify Theorem (2.14) with the following simple adaptation strategy. We take a fixed upper bound on the adaptation cost and a fixed lower bound on uniformity:*

$$\|\mathsf{Q}_\ell\mathsf{Q}_{\ell-1}^{-1}\|_2^2 \leq \mathbf{a} \qquad \lambda_{\min}(\mathsf{P}_\ell) \geq \lambda. \tag{2.12}$$

Then from the rate $1-\alpha = 1 - 2\gamma\mu L\lambda/(\mu + L)$*, we can perform an adaptation every*

$$\mathbf{c} = \lceil \log(\mathbf{a}) / \log\left((2-\alpha)/(2-2\alpha)\right) \rceil \tag{2.13}$$

iterations, so that $\mathbf{a}(1-\alpha)^{\mathbf{c}} = (1-\alpha/2)^{\mathbf{c}}$ *and* $k_\ell = \ell\mathbf{c}$*. A direct application of Theorem (2.14) gives that, for any* k*,*

$$\mathbb{E}\left[\|x^{k+1} - x_\ell^\star\|_2^2\right] \leq \left(1 - \frac{\gamma\mu L\lambda}{\mu + L}\right)^k C$$

where $C = \|z^0 - \mathsf{Q}_0(x^\star - \gamma\nabla f(x^\star))\|_2^2$*. That is the same convergence mode as in the non-adaptive case (Theorem 2.10) with a modified rate. Note the modified rate provided here (of the form* $(1-\alpha/2)$ *to be compared with the* $1-\alpha$ *of Theorem 2.10) was chosen for clarity; any rate strictly slower than* $1-\alpha$ *can bring the same result by adapting* $\mathbf{c}$ *accordingly.*

Remark 2.16 (On the adaptation frequency)**.** *Theorem 2.14 and Example 2.15 tell us that we have to respect a prescribed number of iterations between two adaptation steps. We emphasize here that if this inter-adaptation time is violated, the resulting algorithm may be highly unstable. We illustrate this phenomenon on a TV-regularized least squares problem: we compare two versions of* `ARPSD` *with the same adaptation strategy verifying (2.12) but with two different adaptation frequencies*

- *at every iteration (i.e. taking* $\mathbf{c}_\ell = 1$*)*
- *following theory (i.e. taking* $\mathbf{c}_\ell = \mathbf{c}$ *as per Eq. (2.13))*

On Figure 2-2, we observe that adapting every iteration leads to chaotic behavior. Second, even though the theoretical number of iterations in an adaptation cycle is often pessimistic (due to the rough bounding of the rate), the iterates produced with this choice quickly become stable (i.e., identification happens, which will be shown and exploited in the next section) and show a steady decrease in suboptimality.

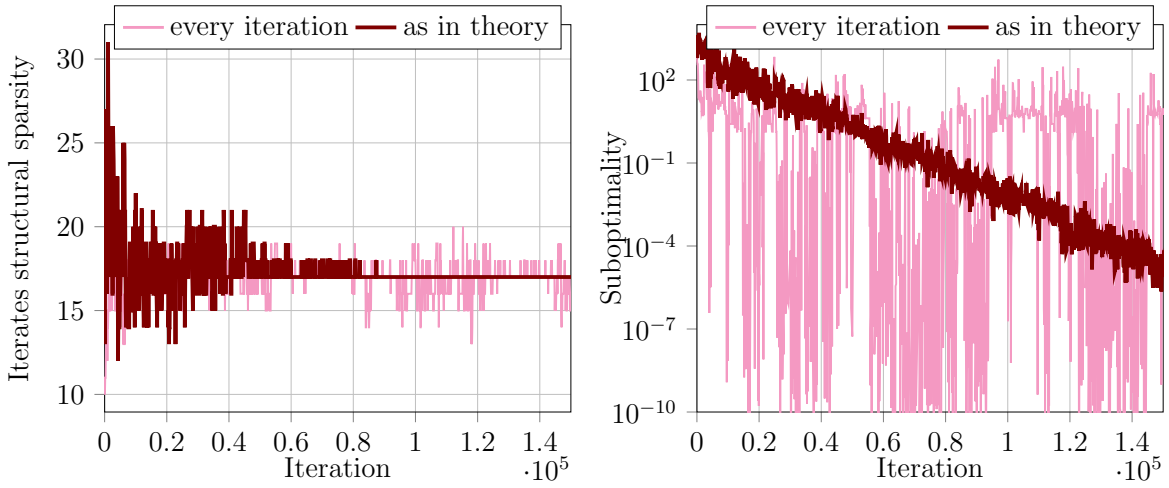


Figure 2-2. Comparisons between theoretical and harsh updating time for `ARPSD`.

A drawback of Theorem 2.14 is that the adaptation cost, inter-adaptation time, and selection uniformity have to be bounded by deterministic sequences. This can be restrictive if we do not have prior knowledge of the problem or if the adaptation cost varies a lot. This drawback can be circumvented to the price of losing the rate *per iteration* to the rate *per adaptation*, as formalized in the following result.

Theorem 2.17 (`ARPSD` convergence: practical version)**.** *Let Assumptions 2.8 and 2.13 hold. Take $\gamma \in (0, 2/(\mu + L)]$, choose $\lambda > 0$, and set $\beta = \gamma\mu L\lambda/(\mu + L)$. Consider the following adaptation strategy:*

1) From the observation of $x^{k_{\ell-1}}$, choose a new sampling with P_ℓ and Q_ℓ, such that

$$\lambda_{\min}(\mathsf{P}_\ell) \geq \lambda;$$

2) Compute $\mathbf{c}_\ell$ so that $\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\|_2^2 (1 - \alpha_{\ell-1})^{\mathbf{c}_\ell} \leq 1 - \beta$ where

$$\alpha_{\ell-1} = 2\gamma\mu L\lambda_{\min}(\mathsf{P}_{\ell-1})/(\mu + L);$$

3) Apply the new sampling after $\mathbf{c}_\ell$ iterations ($k_\ell = k_{\ell-1} + \mathbf{c}_\ell$).

Then, we have for any $k \in [k_\ell, k_{\ell+1})$

$$\mathbb{E}\left[\|x^{k+1} - x^\star\|_2^2\right] \leq (1 - \alpha_\ell)^{k-k_\ell} (1 - \beta)^\ell \left\|z^0 - \mathsf{Q}_0 \left(x^\star - \gamma \nabla f\left(x^\star\right)\right)\right\|_2^2.$$

Proof of Theorem 2.17. The proof follows the same pattern as the one of Theorem 2.14. The only difference is that the three control sequences (adaptation cost, inter-adaptation time, and selection uniformity) are now random sequences since they depend on the iterates of the (random) algorithm. This technical point requires special attention. In (2.10), the adaptation introduces a cost by a factor $\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\|_2^2$, which is not deterministically upper-bounded anymore. However it is $\mathcal{F}^{k_{\ell-1}}$-measurable by construction of Q_ℓ, so we can write

$$
\begin{aligned}
&\mathbb{E}\left[\|z^{k_\ell-1} - z_\ell^\star\|_2^2 \,|\, \mathcal{F}^{k_{\ell-1}}\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\|z^{k_\ell-1} - z_\ell^\star\|_2^2 \,|\, \mathcal{F}^{k_\ell-2}\right] \,|\, \mathcal{F}^{k_{\ell-1}}\right] \\
&\leq \mathbb{E}\left[\mathbb{E}\left[\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}(z^{k_\ell-2} + P_{k_\ell-1}(y^{k_\ell-1} - z^{k_\ell-2}) - \mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1} z_{\ell-1}^\star\|_2^2 \,|\, \mathcal{F}^{k_\ell-2}\right] \,|\, \mathcal{F}^{k_{\ell-1}}\right] \\
&\leq \mathbb{E}\left[\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\|_2^2 (1-\alpha_{\ell-1}) \|z^{k_\ell-2} - z_{\ell-1}^\star\|_2^2 \,|\, \mathcal{F}^{k_{\ell-1}}\right] \\
&= \|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\|_2^2 (1-\alpha_{\ell-1}) \mathbb{E}\left[\|z^{k_\ell-2} - z_{\ell-1}^\star\|_2^2 \,|\, \mathcal{F}^{k_{\ell-1}}\right].
\end{aligned}
$$

Using Eq. (2.8), this inequality yields

$$
\begin{aligned}
\mathbb{E}\left[\|z^{k_\ell-1} - z_\ell^\star\|_2^2 \,|\, \mathcal{F}^{k_{\ell-1}}\right] &\leq \|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\|_2^2 (1-\alpha_{\ell-1})^{k_\ell - k_{\ell-1}} \mathbb{E}\left[\|z^{k_{\ell-1}-1} - z_{\ell-1}^\star\|_2^2 \,|\, \mathcal{F}^{k_{\ell-1}}\right] \\
&\leq (1-\beta)\mathbb{E}\left[\|z^{k_{\ell-1}-1} - z_{\ell-1}^\star\|_2^2 \,|\, \mathcal{F}^{k_{\ell-1}}\right].
\end{aligned}
$$

where we used points 2) and 3) of the strategy to bound the first terms deterministically. Finally, we obtain

$$
\begin{aligned}
\mathbb{E}\left[\|z^{k_\ell-1} - z_\ell^\star\|_2^2\right] &= \mathbb{E}\left[\mathbb{E}\left[\|z^{k_\ell-1} - z_\ell^\star\|_2^2 \,|\, \mathcal{F}^{k_{\ell-1}}\right]\right] \\
&\leq (1-\beta)\mathbb{E}\left[\|z^{k_{\ell-1}-1} - z_{\ell-1}^\star\|_2^2\right]
\end{aligned}
$$

then the rest of the proof follows directly by induction. □

2.2.2 Identification of proximal algorithms

As we mentioned in Section 1.4 proximal algorithms could identify a near optimal subspace before the convegence moment.

This identification can be exploited within our adaptive algorithm `ARPSD` for solving Problem (2.1). Indeed, assuming that the two extreme subspaces of (1.33) coincide, the theorem says that the structure of the iterate $\mathsf{S}_\mathcal{M}(x^k)$ will be the same as the one of the solutions $\mathsf{S}_\mathcal{M}(x^\star)$. In this case, if we choose the adaptation strategy of our adaptive algorithm `ARPSD` deterministically from $\mathsf{S}_\mathcal{M}(x^k)$, then, after a finite time with probability

one, the selection will not be adapted anymore. This allows us to recover the rate of the non-adaptive case (Theorem 2.10), as formalized in the next theorem.

Theorem 2.18 (Improved asymptotic rate)**.** *Under the same assumptions as in Theorems 2.14 and 2.17, if the solution $x^\star$ of (2.1) verifies the qualification constraint (QC) for any $\varepsilon > 0$ small enough, then, using an adaptation deterministically computed from $(\mathsf{S}_{\mathcal{M}}(x^k))$, we have*

$$\mathbb{E}[\|x^k - x^\star\|_2^2] = \mathcal{O}_p\left(\left(1 - \lambda_{\min}(\mathsf{P}^\star)\frac{2\gamma\mu L}{\mu + L}\right)^k\right)$$

where $\mathsf{P}^\star$ is the average projection matrix of the selection associated with $\mathsf{S}_{\mathcal{M}}(x^\star)$ and $\mathcal{O}_p$ denotes big O in probability.

Proof of Theorem 2.18. Let $u^\star = x^\star - \gamma\nabla f(x^\star)$ and observe from the optimality conditions of (2.1) that $x^\star = \mathbf{prox}_{\gamma g}(u^\star)$. We apply Theorem 1.17 and the qualification condition (QC) yields that $\mathsf{S}_{\mathcal{M}}(x^k)$ will exactly reach $\mathsf{S}_{\mathcal{M}}(x^\star)$ in finite time T_i.

Let us first clarify that the rate before the moment of time T_i can be rewritten as following

$$\mathbb{E}\left[\|x^{k+1} - x^\star\|_2^2\right] \leq (1-\beta)^k \|z^0 - \mathsf{Q}_0(x^\star - \gamma\nabla f(x^\star))\|_2^2,$$

where we modify the result of Theorem 2.17 using that $\beta \leq \alpha_\ell$ for any ℓ.

Now, let us discuss what happens after the moment of time $T - i$. Since $\mathsf{S}_{\mathcal{M}}(x^k)$ becomes fixed and equal to $\mathsf{S}_{\mathcal{M}}(x^\star)$ the operators P_ℓ and Q_ℓ become fixed as well. It makes the rescaling step $z^k \leftarrow \underbrace{\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}}_{I} z^k$ trivial and as a result, the adaptations process costs nothing. Thus, the rate after that moment can be written as follows

$$\mathbb{E}\left[\|x^{k+1} - x^\star\|_2^2\right] \leq \left(1 - \lambda_{\min}(\mathsf{P}^\star)\frac{2\gamma\mu L}{\mu + L}\right)\mathbb{E}\left[\|x^k - x^\star\|_2^2\right].$$

Combining these 2 results together for $k \geq T_i$ we have

$$\mathbb{E}\left[\|x^{k+1} - x^\star\|_2^2\right] \leq \left(1 - \frac{\lambda_{\min}(\mathsf{P}^\star)}{\lambda}\beta\right)^k \left\{\left(\frac{1-\beta}{1 - \frac{\lambda_{\min}(\mathsf{P}^\star)}{\lambda}\beta}\right)^{T_i} \|z^0 - \mathsf{Q}_0(x^\star - \gamma\nabla f(x^\star))\|_2^2\right\}.$$

Since T_i is almost sure finite, the second term is almost sure finite as well, that concludes the proof. □

This theorem means that if r, $\mathcal{M}$, and $\mathcal{C}$ are chosen in agreement, the adaptive algorithm `ARPSD` eventually reaches a linear rate in terms of iterations as the non-adaptive

RPSD. In addition, the term $\lambda_{\min}(\mathsf{P}^\star)$ present in the rate now depends on the *final* selection and thus on the optimal structure, which is much better than the structure-agnostic selection of RPSD in Theorem 2.10. In the next section, we develop practical rules for an efficient interlacing of r, $\mathcal{M}$, and $\mathcal{C}$.

2.2.3 Identification-based subspace descent

In this section, we provide practical rules to sample efficiently subspaces according to the structure identified by the iterates of our proximal algorithm. According to Theorem 2.18, we need to properly choose $\mathcal{C}$ with respect to r and $\mathcal{M}$ to have a good asymptotic regime. According to Theorem 2.17, we also need to follow specific interlacing constraints to have good behavior along with the convergence. These two aspects are discussed in Section 2.2.3 and Section 2.2.3, respectively.

How to update the selection

We provide here general rules to sample in the family of subspaces $\mathcal{C}$ according to the structure identified with the family of $\mathcal{M}$. To this end, we need to consider the two families $\mathcal{C}$ and $\mathcal{M}$ that closely related. We introduce the notion of generalized complemented subspaces.

Definition 2.19 (Generalized complemented subspaces)**.** *Two families of subspaces $\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_m\}$ and $\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_m\}$ are said to be (generalized) complemented subspaces if for all $i = 1, \dots, m$*

$$\begin{cases} (\mathcal{C}_i \bigcap \mathcal{M}_i) \subseteq \bigcap_j \mathcal{C}_j \\ \mathcal{C}_i + \mathcal{M}_i = \mathbb{R}^n \end{cases}$$

Example 2.20 (Complemented subspaces and sparsity vectors for axes and jumps)**.** *For the axes subspace set (see Section 2.1.4)*

$$\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_n\} \qquad \text{with } \mathcal{C}_i = \{x \in \mathbb{R}^n : x_{[j]} = 0 \quad \forall j \neq i\}, \tag{2.14}$$

a complemented identification set is

$$\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_n\} \qquad \text{with } \mathcal{M}_i = \{x \in \mathbb{R}^n : x_{[i]} = 0\}, \tag{2.15}$$

as $\mathcal{M}_i \bigcap \mathcal{C}_i = \{0\} = \bigcap_j \mathcal{C}_j$ and $\mathcal{C}_i + \mathcal{M}_i = \mathbb{R}^n$. In this case, the sparsity vector $\mathsf{S}_{\mathcal{M}}(x)$ corresponds to the support *of x (indeed $(\mathsf{S}_{\mathcal{M}}(x))_{[i]} = 0$ iff $x \in \mathcal{M}_i \Leftrightarrow x_{[i]} = 0$). Recall that the support of a point $x \in \mathbb{R}^n$ is defined as the size-n vector* $\mathrm{supp}(x)$ *such that* $\mathrm{supp}(x)_i = 1$

if $x_{[i]} \neq 0$ *and* 0 *otherwise. By a slight abuse of notation, we denote by* $|\mathrm{supp}(x)|$ *the size of the support of* x*, i.e. its number of non-null coordinates and* $|\mathrm{null}(x)| = n - |\mathrm{supp}(x)|$.

For the jumps subspace sets (see Section 2.1.4)

$$\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_{n-1}\} \qquad \text{with } \mathcal{C}_i = \left\{x \in \mathbb{R}^n : x_{[j]} = x_{[j+1]} \text{ for all } j \neq i\right\} \tag{2.16}$$

a complemented identification set is

$$\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_{n-1}\} \qquad \text{with } \mathcal{M}_i = \left\{x \in \mathbb{R}^n : x_{[i]} = x_{[i-1]}\right\}, \tag{2.17}$$

as $\mathcal{M}_i \bigcap \mathcal{C}_i = \mathrm{span}(\{1\}) = \bigcap_j \mathcal{C}_j$ *and* $\mathcal{C}_i + \mathcal{M}_i = \mathbb{R}^n$*. Here* $\mathsf{S}_{\mathcal{M}}(x^k)$ *corresponds to the* jumps *of* x *(indeed* $(\mathsf{S}_{\mathcal{M}}(x^k))_{[i]} = 0$ *iff* $x^k \in \mathcal{M}_i \Leftrightarrow x^k_{[i]} = x^k_{[i+1]}$*). . The jumps of a point* $x \in \mathbb{R}^n$ *is defined as the vector* $\mathrm{jump}(x) \in \mathbb{R}^{(n-1)}$ *such that for all* i *we have:* $\mathrm{jump}(x)_{[i]} = 1$ *if* $x_{[i]} \neq x_{[i+1]}$ *and* 0 *otherwise.*

In this example, we use different collections $\mathcal{M}$ from Examples 1.18, 1.19. More precisely, the collections $\mathcal{M}'$ from Section 1.4 could be interpreted as $\left\{\mathcal{M}_S = \bigcup_{i \in S} \mathcal{M}_i\right\}_{S \in \mathbf{S}}$, where $\mathbf{S}$ is a power set of $\{1, \dots, n\}$ for the axes subspace set and a power set of $\{1, \dots, n-1\}$ for jumps subspace set. Moreover, the big variety of admissible selections makes these two "different" subspace families indistinguishable.

The practical reasoning with using complemented families is the following. If the subspace $\mathcal{M}_i$ is identified at time K (i.e. $(\mathsf{S}_{\mathcal{M}}(x^k))_{[i]} = 0 \Leftrightarrow x^k \in \mathcal{M}_i$ for all $k \geq K$), then it is no use to update the iterates in $\mathcal{C}_i$ in preference, and the next selection $\mathfrak{S}_k$ should not include $\mathcal{C}_i$ anymore. Unfortunately, the moment after which a subspace is definitively identified is unknown in general; however, subspaces $\mathcal{M}_i$ usually show specific stability, and thus $\mathcal{C}_i$ may be "less included" in the selection. This is the intuition behind our adaptive subspace descent algorithm: when the selection $\mathfrak{S}^k$ is adapted to the subspaces in $\mathcal{M}$ to which x^k belongs, this gives birth to an automatically adaptive subspace descent algorithm, from the generic `ARPSD`.

Table 2.1 summarizes the common points and differences between the adaptive and non-adaptive subspace descent methods. Note that the two options introduced in this table are examples of how to generate reasonably performing admissible selections. Their difference lies in the fact that for Option 1, the *probability* of sampling a subspace outside the support is controled, while for Option 2, the *number* of subspaces is controlled (this makes every iteration computationally similar which can be interesting in practice). Option 2 will be discussed in Section 2.2.3 and illustrated numerically in Section 2.3.

Notice that, contrary to the importance-like adaptive algorithms of [SRJ17], for instance, the purpose of these methods is not to adapt each subspace probability to local *steepness* but rather to adapt them to the current *structure*. This is notably due to the fact that local steepness-adapted probabilities can be difficult to evaluate numerically

		(non-adaptive) subspace descent `RPSD`	adaptive subspace descent `ARPSD`
Subspace family		$\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_c\}$	
Algorithm		$\left\{ \begin{array}{rl} y^k = & \mathsf{Q}\left(x^k - \gamma \nabla f\left(x^k\right)\right) \\ z^k = & P_{\mathfrak{S}^k}\left(y^k\right) + \left(I - P_{\mathfrak{S}^k}\right)\left(z^{k-1}\right) \\ x^{k+1} = & \mathbf{prox}_{\gamma g}\left(\mathsf{Q}^{-1}\left(z^k\right)\right) \end{array} \right.$	
Selection	Option 1	$\mathcal{C}_i \in \mathfrak{S}^k$ with probability p	$\mathcal{C}_i \in \mathfrak{S}^k$ with probability $\left\{ \begin{array}{ll} p & \text{if } x^k \in \mathcal{M}_i \Leftrightarrow [\mathsf{S}_{\mathcal{M}}(x^k)]_i = 0 \\ 1 & \text{elsewhere} \end{array} \right.$
	Option 2	Sample s elements uniformly in $\mathcal{C}$	Sample s elements uniformly in $\{\mathcal{C}_i : x^k \in \mathcal{M}_i \text{ i.e. } [\mathsf{S}_{\mathcal{M}}(x^k)]_i = 0\}$ and add *all* elements in $\{\mathcal{C}_j : x^k \notin \mathcal{M}_j \text{ i.e. } [\mathsf{S}_{\mathcal{M}}(x^k)]_j = 1\}$

Table 2.1. Strategies for non-adaptive vs. adaptive algorithms

and that in heavily structured problems, adapting to an ultimately very sparse structure already reduces the number of explored dimensions drastically, as suggested in [GIMA18] for the case of coordinate-wise projections.

Practical examples and discussion

We discuss further the families of subspaces of Example 2.20 when selected with Option 2 of Table 2.1.

Coordinate-wise projections Using the subspaces (2.14) and (2.15), a practical adaptative coordinate descent can be obtained from the following reasoning at each adaptation time $k = k_{\ell-1}$:

- Observe $\mathsf{S}_{\mathcal{M}}(x^k)$ i.e. the support of x^k.
- Take all coordinates in the support and randomly select s coordinates outside the support. Compute[2] associated P_ℓ, Q_ℓ, and Q_ℓ^{-1}. Notice that $\lambda_{\min}(\mathsf{P}_\ell) = p_\ell = s/|\text{null}(x^k)|$.

[2] Let us give a simple example in $\mathbb{R}^4$:

$$\text{for } x^k = \begin{pmatrix} 1.23 \\ -0.6 \\ 0 \\ 0 \end{pmatrix}, \mathsf{S}_{\mathcal{M}}(x^k) = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \text{ then } \begin{array}{l} \mathbb{P}[\mathcal{C}_1 \subseteq \mathfrak{S}^k] = \mathbb{P}[\mathcal{C}_2 \subseteq \mathfrak{S}^k] = 1 \\ \mathbb{P}[\mathcal{C}_3 \subseteq \mathfrak{S}^k] = \mathbb{P}[\mathcal{C}_4 \subseteq \mathfrak{S}^k] = p_\ell := s/|\text{null}(x^k)| = s/2 \end{array}$$

- Following the rules of Theorem 2.17, compute

$$\mathbf{c}_\ell = \left\lceil \frac{\log\left(\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\|_2^2\right) + \log(1/(1-\beta))}{\log(1/(1-\alpha_{\ell-1}))} \right\rceil \qquad \text{with } \alpha_{\ell-1} = 2p_{\ell-1}\gamma\mu L/(\mu+L)$$

  for some small fixed $0 < \beta \leq 2\gamma\mu L/(n(\mu+L)) \leq \inf_\ell \alpha_\ell$.

  Apply the new sampling after $\mathbf{c}_\ell$ iterations (i.e. $k_\ell = k_{\ell-1} + \mathbf{c}_\ell$).

Finally, we notice that the above strategy with Option 2 of Table 2.1 produces moderate adaptations as long as the iterates are rather dense. To see this, observe first that $\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}$ is a diagonal matrix, the entries of which depend on the support of the corresponding coordinates at times $k_{\ell-1}$ and $k_{\ell-2}$. More precisely, the diagonal entries are described in the following table:

i is in the support at		
$k_{\ell-1}$	$k_{\ell-2}$	$\left[\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\right]_{ii}$
yes	yes	1
no	yes	$\frac{1}{p_\ell} = \frac{\lvert\mathrm{null}(x^{k_{\ell-1}})\rvert}{s}$
yes	no	$p_{\ell-1} = \frac{s}{\lvert\mathrm{null}(x^{k_{\ell-2}})\rvert}$
no	no	$\frac{p_{\ell-1}}{p_\ell} = \frac{\lvert\mathrm{null}(x^{k_{\ell-1}})\rvert}{\lvert\mathrm{null}(x^{k_{\ell-2}})\rvert}$

Thus, as long as the iterates are not sparse (i.e. in the first iterations, when $|\mathrm{null}(x^k)| \approx s$ is small), the adaptation cost is moderate so the first adaptations can be done rather frequently. Also, in the frequently-observed case when the support only decreases ($\mathsf{S}_{\mathcal{M}}(x^{k_{\ell-2}}) \leq \mathsf{S}_{\mathcal{M}}(x^{k_{\ell-1}})$), the second line of the table is not active and thus $\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\| = 1$, so the adaptation can be done without waiting.

Vectors of fixed variations The same reasoning as above can be done for vectors of fixed variation by using the families (2.16) and (2.17). At each adaptation time $k = k_{\ell-1}$:

- Observe $\mathsf{S}_{\mathcal{M}}(x^k)$ i.e. the *jumps* of x;
- The adapted selection consists in selecting all jumps present in x^k and randomly selecting s jumps that are not in x^k. Compute P_ℓ, Q_ℓ, and Q_ℓ^{-1} (to the difference of coordinate sparsity they have to be computed numerically).

$$\mathsf{P}_\ell = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & p_\ell & \\ & & & p_\ell \end{pmatrix} \quad \mathsf{Q}_\ell = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1/\sqrt{p_\ell} & \\ & & & 1/\sqrt{p_\ell} \end{pmatrix} \quad \mathsf{Q}_\ell^{-1} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \sqrt{p_\ell} & \\ & & & \sqrt{p_\ell} \end{pmatrix}$$

- For a fixed $\beta > 0$, compute

$$\mathbf{c}_\ell = \left\lceil \frac{\log\left(\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\|_2^2\right) + \log(1/(1-\beta))}{\log(1/(1-\alpha_{\ell-1}))} \right\rceil.$$

  Apply the new sampling after $\mathbf{c}_\ell$ iterations (i.e. $k_\ell = k_{\ell-1} + \mathbf{c}_\ell$).

Practical consideration for TV

Before going to the numerical section, let us discuss the problem of computation of Q_ℓ for 1-d Total Variation.

First, to compute P_ℓ for arbitrary admissible selection $\mathfrak{S}$, we need to calculate the sum of $2^d - 1$ matrices $n \times n$ where d is the size of the subspace family $\mathcal{C}$. Second, the computation of the inverse matrix Q_ℓ is also expensive. To figure out both of these problems, let us propose the following adaptive selection based on Option 2 Table 2.1.

Consider the set of artificial jumps $\mathcal{S} = \{n_1, n_2, \dots, n_{l-1}\}$ and denote by $\mathcal{R} = \{i \notin \mathcal{S} : [\mathsf{S}_{\mathcal{M}}(x^k)]_i = 0\}$ the set of possible random entries. Fix the amount of sampled elements s and sample "first" element $\mathcal{R}_0$ uniformly in $\mathcal{R} = \{\mathcal{R}_i\}_{1 \le i \le r}$. Select "first s" elements starting from $\mathcal{R}_f$ considering the cyclic structure of the list of elements ($\mathcal{R}_{r+1} = \mathcal{R}_1$).

This selection allows us to solve both issues stated above if set $\mathcal{S}$ is chosen as described in this paragraph. If l is small enough, it will not change the sparsity property of the random projection $P_{\mathfrak{S}^k}$; however, this modification will force all the projections to be block-diagonal with blocks' ends on positions $n_1, \dots n_{l-1}$. In contrast with jumps(x^k) that we could not control, by adding l artificial jumps, we could guarantee that each block of the $P_{\mathfrak{S}^k}$ has at most $\lceil n/l \rceil$ rows. Since every random projection has end of the block on positions $\{n_i\}_{1 \le i \le l-1}$[3] P_ℓ also has such block structure and we could split the computation of Q_ℓ^{-1} and Q_ℓ into l independent parts and could be done in parallel.

Second, the total amount of possible projections is equal to the r that decrease the amount of terms in the sum; however, if l is small and the final solution is sparse $r = \mathcal{O}(n)$ and the computational cost of P_ℓ is $\mathcal{O}(n^3)$. Due to our specific selection of $\mathcal{S}$ we could make this computation in parallel block by block, and moreover, the amount of projections to be considered for every block is at most the size of the block. All in all, it leads to the same complexity as the inversion procedure and selecting $\mathcal{S}$ such that $n_i = \lceil \frac{ni}{l} \rceil$, the computational cost of update of P_ℓ and Q_ℓ is $\mathcal{O}(n^3 l^{-2})$.

[3] Of course, any projection also has the ends of the blocks on positions $\{i : [\mathsf{S}_{\mathcal{M}}(x^k)]_i = 1\}$ but we will skip them for simplicity.

2.3 Numerical illustrations

We report preliminary numerical experiments illustrating the behavior of our randomized proximal algorithms on standard problems involving ℓ_1/TV regularizations. We provide an empirical comparison of our algorithms with the standard proximal (full and coordinate) gradient algorithms and a recent proximal sketching algorithm.

2.3.1 Experimental setup

We consider the standard regularized logistic regression with three different regularization terms, which can be written for given $(a_i, b_i) \in \mathbb{R}^{n+1}$ $(i = 1, \dots, m)$ and parameters $\lambda_1, \lambda_2 > 0$

$$\min_{x \in \mathbb{R}^n} \quad \frac{1}{m} \sum_{i=1}^{m} \log\left(1 + \exp\left(-b_i a_i^\top x\right)\right) + \frac{\lambda_2}{2}\|x\|_2^2 + \lambda_1 \|x\|_1 \tag{2.18a}$$

$$+ \lambda_1 \|x\|_{1,2} \tag{2.18b}$$

$$+ \lambda_1 \mathbf{TV}(x) \tag{2.18c}$$

We use two standard data-sets from the LibSVM repository: the *a1a* data-set ($m = 1,605$ $n = 123$) for the **TV** regularizer the *rcv1_train* data-set ($m = 20,242$ $n = 47,236$) for the ℓ_1 and $\ell_{1,2}$ regularizers. We fix the parameters $\lambda_2 = 1/m$ and λ_1 to reach a final sparsity of roughly 90%.

The subspace collections are taken naturally adapted to the regularizers: by coordinate for (2.18a) and (2.18b), and by variation for (2.18c). The adaptation strategies are the ones described in Section 2.2.3.

We consider five algorithms:

Name	Reference	Description	Randomness
`PGD`		vanilla proximal gradient descent	None
x[4] `RPCD`	[Nes12]	standard proximal coordinate descent	x coordinates selected for each update
x `SEGA`	[HMR18]	Algorithm `SEGA` with coordinate sketches	$\text{rank}(S^k) = \text{x}$
x `RPSD`	Algorithm 9	(non-adaptive) random subspace descent	Option 2 of Table 2.1 with $s = \text{x}$
x `ARPSD`	Algorithm 10	adaptive random subspace descent	Option 2 of Table 2.1 with $s = \text{x}$

For the produced iterates, we measure the sparsity of a point x by $\|\mathsf{S}_{\mathcal{M}}(x_k)\|_1$, which correspond to the size of the supports for the ℓ_1 case and the number of jumps for the TV

[4] In the following, x is often given in percentage of the possible subspaces, i.e. x% of $|\mathcal{C}|$, that is x% of n for coordinate projections and x% of $n-1$ for variation projections.

case. We also consider the quantity:

$$\text{Number of subspaces explored at time } k \;=\; \sum_{t=1}^{k} \|\mathsf{S}_{\mathcal{M}}(x^t)\|_1.$$

We then compare the performance of the algorithms on three criteria:

- functional suboptimality vs iterations;
- size of the sparsity pattern vs iterations (showing the identification properties);
- functional suboptimality vs number of subspaces explored (showing the gain of adaptivity).

2.3.2 Illustrations for coordinate-structured problems

Comparison with standard methods

We consider first ℓ_1-regularized logistic regression (2.18a); in this setup, the non-adaptive `RPSD` boils down to the usual randomized proximal gradient descent (see Section 2.1.4). We compare the proximal gradient to its adaptive and non-adaptive randomized counterparts.

First, we observe that the iterates of `PGD` and `ARPSD` coincide. This is due to the fact that the sparsity of iterates only decreases ($\mathsf{S}_{\mathcal{M}}(x_k) \leq \mathsf{S}_{\mathcal{M}}(x_{k+1})$) along the convergence, and according to Option 2 all the non-zero coordinates are selected at each iteration and thus set to the same value as with `PGD`. However, a single iteration of 10%-`ARPSD`costs less in terms of number of subspaces explored, leading the speed-up of the right-most plot. Contrary to the adaptive `ARPSD`, the structure-blind `RPSD` identifies much later then `PGD` and shows poor convergence.

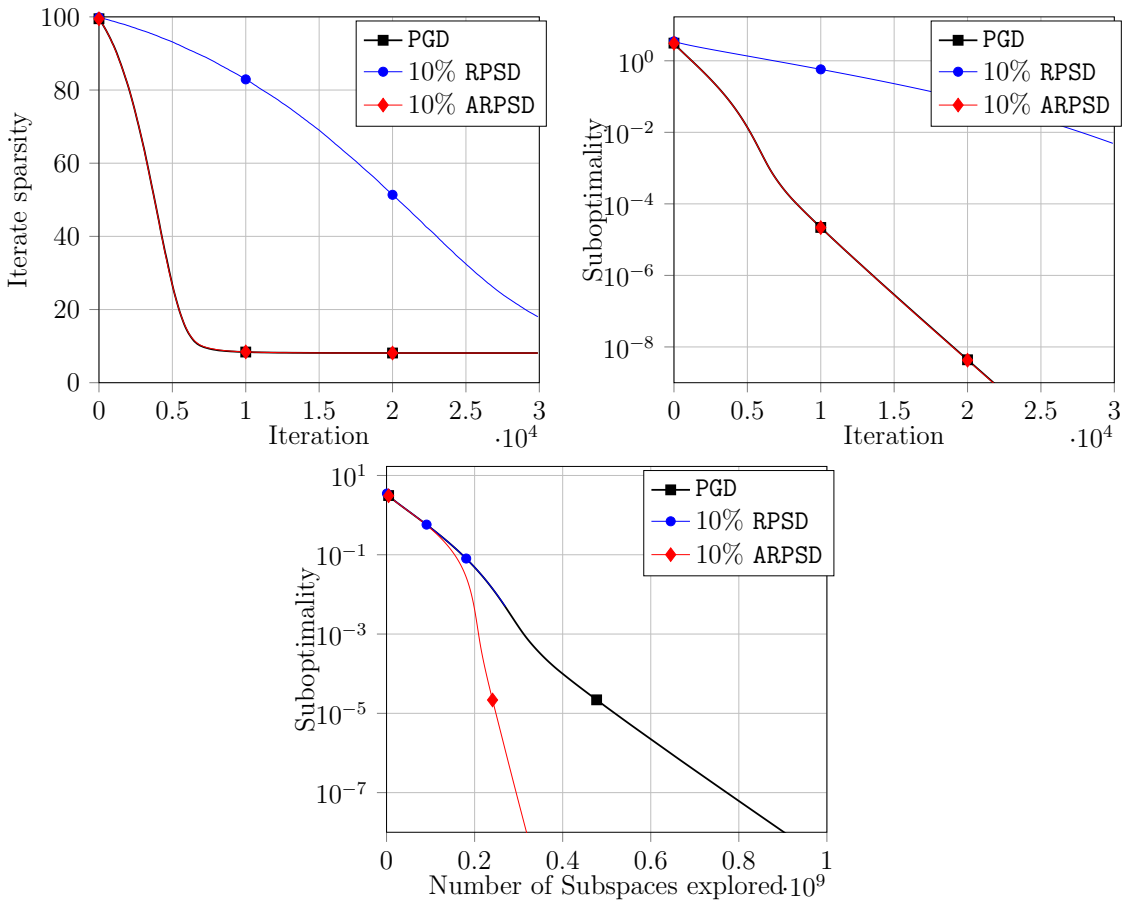


Figure 2-3. ℓ_1-regularized logistic regression (2.18a)

Comparison with SEGA

In Figure 2-4, we compare ARPSD algorithm with SEGA algorithm featuring coordinate sketches [HMR18]. While the focus of SEGA is not to produce an efficient coordinate descent method but rather to use sketched gradients, SEGA and RPSD are similar algorithmically and reach similar rates (see Section 2.1.4). As mentioned in [HMR18, Apx. G2], SEGA is slightly slower than plain randomized proximal coordinate descent (10% RPSD) but still competitive, which corresponds to our experiments. Thanks to the use of identification, ARPSD shows a clear improvement over other methods in terms of efficiency with respect to the number of subspaces explored.

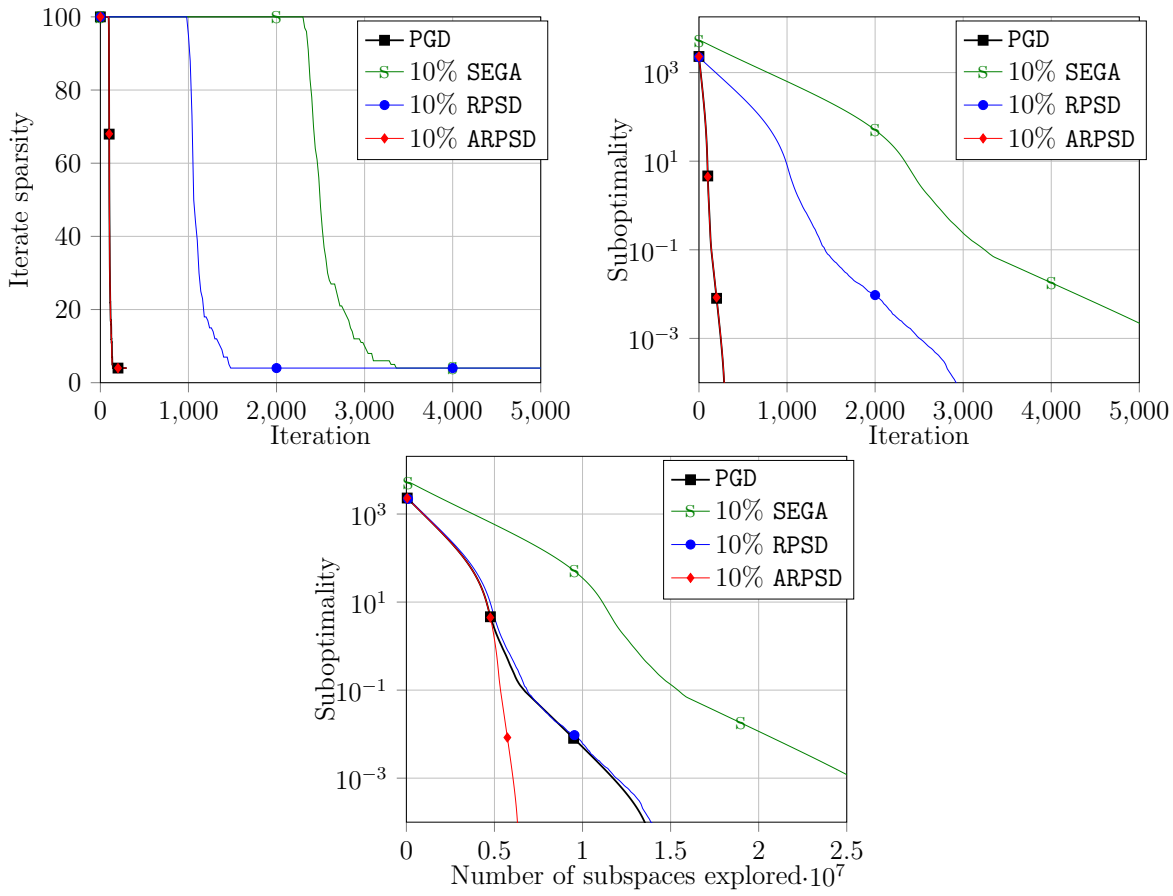


Figure 2-4. $\ell_{1,2}$ regularized logistic regression (2.18b)

2.3.3 Illustrations for total variation regularization

We focus here on the case of total variation (2.18c) which is a typical usecase for our adaptive algorithm and subspace descent in general. Figure 2-5 displays a comparison between the vanilla proximal gradient and various versions of our subspace descent methods.

We observe first that `RPSD`, not exploiting the problem structure, fails to reach satisfying performances as it identifies lately and converges slowly. In contrast, the adaptive versions `ARPSD` perform similarly to the vanilla proximal gradient in terms of sparsification and suboptimality with respect to iterations. As a consequence, in terms of number of subspaces explored, `ARPSD` becomes much faster once a near-optimal structure is identified. More precisely, all adaptive algorithms (except 1 `ARPSD`, see the next paragraph) identify

a subspace of size $\approx 8\%$ (10 jumps in the entries of the iterates) after having explored around 10^5 subspaces. Subsequently, each iteration involves a subspace of size 22,32,62 (out of a total dimension of 123) for 10%,20%,50% ARPSD respectively, resulting in the different slopes in the red plots on the rightmost figure.

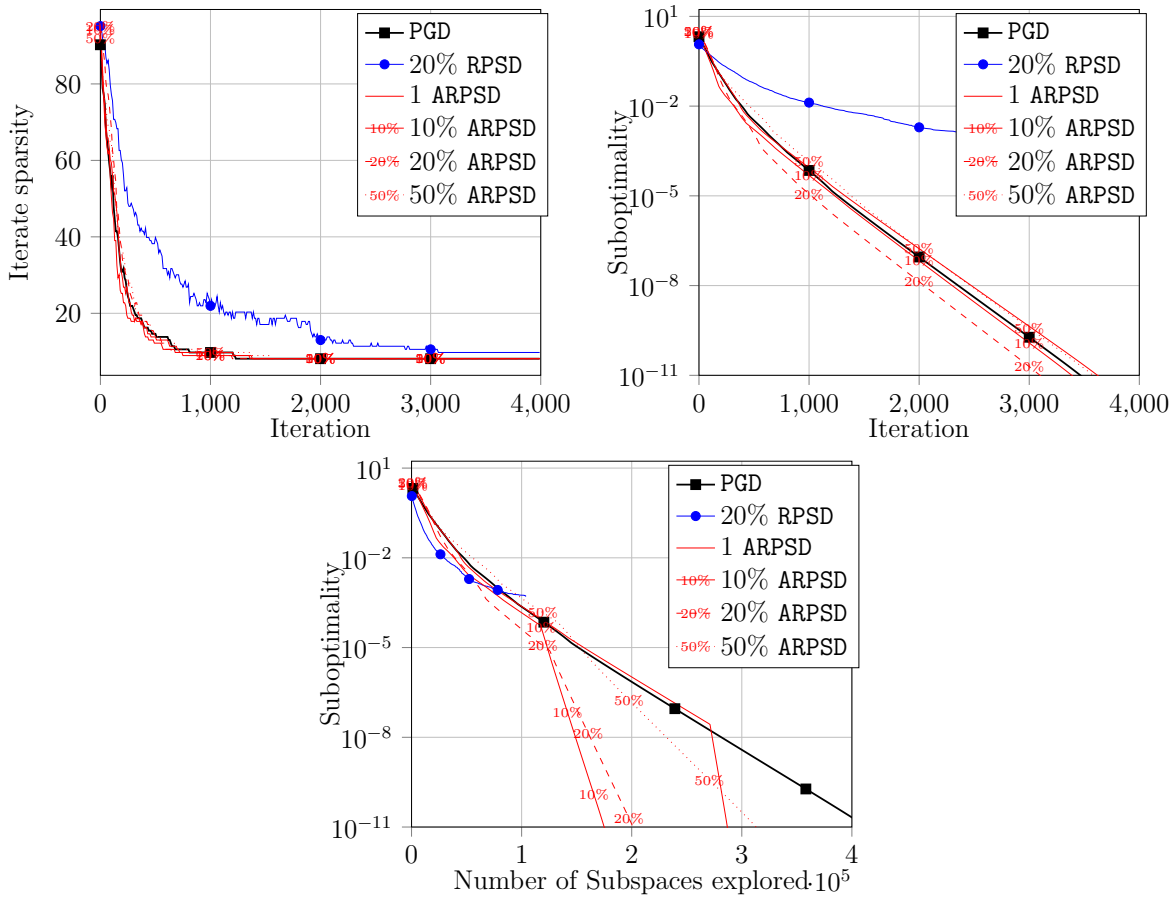


Figure 2-5. 1D-TV-regularized logistic regression (2.18c)

Finally, Figure 2-6 displays 20 runs of 1 and 20% ARPSD as well as the median of the runs in bold. We notice that more than 50% of the time, a low-dimensional structure is quickly identified (after the third adaptation) resulting in a dramatic speed increase in terms of subspaces explored. However, this adaptation to the lower-dimensional subspace might take some more time (either because of poor identification in the first iterates or because a first heavy adaptation was made early and a pessimistic bound on the rate prevents a new adaptation in theory). Yet, one can notice that these adaptations are more

stable for the 20% than for the 1 ARPSD, illustrating the "speed versus stability" tradeoff in the selection.

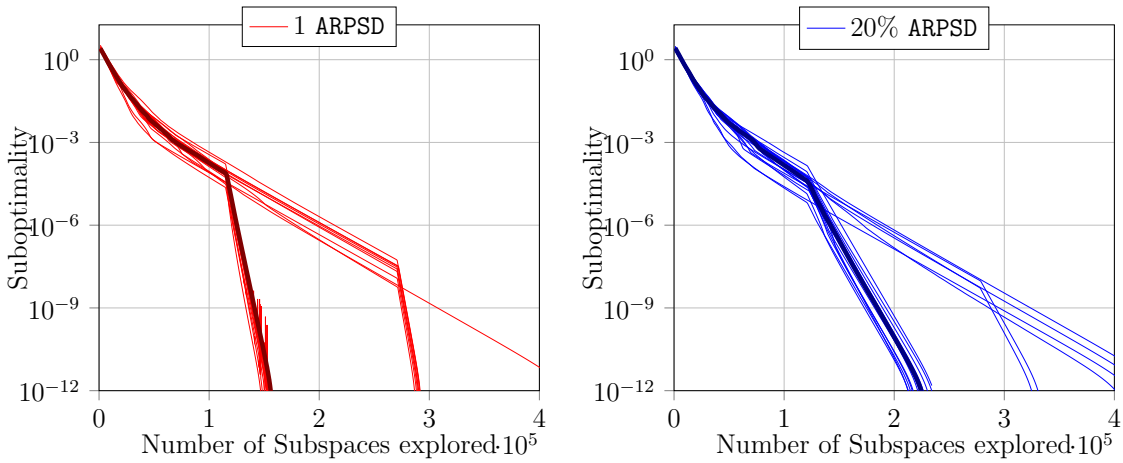


Figure 2-6. 20 runs of ARPSD and their median (in bold) on 1D-TV-regularized logistic regression (2.18c)

2.4 Conclusion

In this chapter, we presented a sketch-and-project algorithm with automatic dimension reduction. The key feature of our approach is an identification-based way to select projections using the geometric structure of the regularizer in contrast with the common approaches that propose taking into account only properties of the smooth term. We shown both in theory and in practice, that after some moment of time the iteration complexity of our method is the same as for vanilla proximal gradient descent; however, the dimension of the problem solved on every iteration is sufficiently smaller. This leads to acceleration in terms of the "number of subspaces explored".

Chapter 3

Adaptive sparsification of distributed methods

Chapter Contents

Introduction

In this chapter, we propose an asynchronous distributed algorithm SPY featuring a sparsification of upward communications (slave-to-master) of DAve-PG (see Section 1.3.3). The sparsification mechanism consists in randomly zeroing of the local update entries. This randomized technique maintains the linear convergence in the mean-squared error sense for strongly convex objectives when difference in probabilities of coordinates to be selected is small enough and is adjustable to various levels of communication costs, machines' computational powers, and data distribution evenness. An attractive and original property of this algorithm is the possibility to use a fixed stepsize that depends neither on communication delays nor on the number of machines.

We propose several options for the zeroing of coordinates. First, we analyze U-SPY: the version of SPY, where all the coordinates are selected independently and uniformly. We show that such sparsification performs worse than DAve-PG.

Considering the identification property of DAve-PG (see Subsection 1.4.2), we furthermore leverage on it to improve our sparsification technique by preferably sampling the entries in the support of the master model in the case of ℓ_1-regularized problems performing I-SPY. The good point about I-SPY is that when it converges, this approach can be seen as an automatic dimension reduction procedure, resulting in better performance in terms of *quantity of information exchanged* that is better than for the DAve-PG. Nevertheless, the theoretical analysis prevents us from using different probabilities in the case when the problem is ill-conditioned, and we show that in practice such algorithm could diverge.

To handle the divergence, we propose S-SPY Algorithm that allows using different probabilities; however, it requires scaling. This algorithm is proven to have the same theoretical rate as U-SPY, but with the identification based selection (I-SPY with scaling) it performs better in practice.

Finally, we investigate the performance of I-SPY in terms of *expected convergence time* and show empirically that it performs better than DAve-PG both in terms of the *amount of communications between machines* and *time*.

Name	Reference	Description
DAve-PG	[MIMA18]	delay-tolerant asynchronous proximal gradient descent
SPY	Algorithm 11	general sparsification framework for DAve-PG
U-SPY	Algorithm 12	SPY with uniform selection
I-SPY	Algorithm 13	SPY with identification-based selection
S-SPY	Algorithm 14	SPY with scaled updates
IS-SPY	Algorithm 15	SPY with identification-based selection and scaled updates

Outline. This chapter is organized as follows. In Section 3.1, we present the general sparsification framework SPY and investigate the convergence. In Section 3.2, we present

the `U-SPY` and `S-SPY`, prove the identification result for these algorithms, and show numerically that the performance is weaker than for `DAve-PG`. In Section 3.3, we present `I-SPY` and show numerically that the algorithm could both converge and diverge in practice. Furthermore, we prove the identification result and better rate than for `DAve-PG` under the additional assumption on convergence. Finally in Subsection 3.3.2, we present our empirical investigation on the performance of `I-SPY`.

3.1 General sparsification framework

In this section, we present our distributed algorithm for solving (P) with sparse upward communications.

3.1.1 Sparsification of local updates

In the presented methods, the master machine asynchronously gathers *sparsified* delayed gradient updates from slaves and sends them the current point. More specifically, each slave independently computes a gradient step on its local loss for a randomly drawn subset of coordinates only. The master machine keeps track of the weighted average of the most recent slave outputs, computes the proximity operator of the regularizer at this average point, and sends this result back to the updating worker i^k.

Thus the k-th iteration of our algorithm is the following. The randomly drawn subset of entries that worker i^k updates at iteration k is called *mask* and is denoted by $\mathbf{S}_p^k$ (in bold, to emphasize that it is the *only* random variable in the algorithm). Iteration k of our algorithm thus writes

$$x_{i[j]}^k = \begin{cases} \left(x^{k-D_i^k} - \gamma \nabla f_i(x^{k-D_i^k})\right)_{[j]} & \text{if } \left| \begin{array}{l} i = i^k \\ j \in \mathbf{S}_p^{k-D_i^k} \end{array} \right. \\ x_{i[j]}^{k-1} & \text{otherwise} \end{cases}$$

$$x^k = \mathbf{prox}_{\gamma r}\left(\bar{x}^k\right) \qquad \text{with} \qquad \bar{x}^k = \sum_{i=1}^M \alpha_i x_i^k,$$

Assumption 3.1 (On the random sparsification)**.** *The sparsity mask selectors* $(\mathbf{S}_p^k)$ *are independent and identically distributed random variables. We select a coordinate in the mask as follows:*

$$\mathbb{P}[j \in \mathbf{S}_p^k] = p_j > 0 \qquad \textit{for all } j \in \{1, \dots, n\},$$

with $p = (p_1, \dots, p_n) \in (0, 1]^n$*. We denote* $p_{\max} = \max_i p_i$ *and* $p_{\min} = \min_i p_i$.

The mask selectors ($\mathbf{S}_p^k$) being the only random variables of the algorithm, it is natural to define the filtration $\mathcal{F}^k = \sigma(\{\mathbf{S}_p^\ell\}_{\ell<k})$ so that all variables at time k (x_i^k, $\bar{x}^k$, x^k, d_i^k, D_i^k) are $\mathcal{F}^k$-measurable but $\mathbf{S}_p^k$ is not. This filtration will be used in the proofs.

With the sparsification, this iteration corresponds to one iteration of randomized (block-)coordinate descent, locally at the worker. However, our algorithm does not correspond to an asynchronous stochastic block-coordinate descent algorithm [LWR+15, SHY17, PXYY16, RT16a], since our method maintains a variable, $\bar{x}^k$, aggregating all the workers' contributions asynchronously.

3.1.2 Distributed implementation

The proposed algorithm SPY is generic as none of its ingredients (including the stepsize choice) depend on the computing system (data distribution, agents delays). A unique feature of this algorithm is that although each master update relies on only one agent (and thus part of the data), all the data is always implicitly involved in the master variable, with even proportions. This allows the algorithm to cope with the heterogeneity of the computing system. Its presentation uses the following notation: for a vector of $x \in \mathbb{R}^n$ and a subset S of $\{1, \dots, n\}$, $[x]_S$ denotes the sparse size-n vector where S is the set of non-null entries, for which they match those of x, i.e. $([x]_S)_{[i]} = x_{[i]}$ if $i \in S$ and 0 otherwise.

The algorithm SPY has the same arguments as DAve-PG plus a probability vector (see Assumption 3.1).

Algorithm 11 SPY on $((\alpha_i), (f_i), r\ ;\ p)$ with stopping criterion C

Master

Initialize $\bar{x}^0$
while *stopping criterion* C *not verified* **do**
  Receive $[\Delta^k]_{\mathbf{S}_p^{k-D_{i^k}^k}}$ from agent i^k
  $\bar{x}^k \leftarrow \bar{x}^{k-1} + \alpha_i [\Delta^k]_{\mathbf{S}_p^{k-D_{i^k}^k}}$
  $x^k \leftarrow \mathbf{prox}_{\gamma r}(\bar{x}^k)$
  Send x^k to agent i^k
  $k \leftarrow k+1$
end
Interrupt all slaves
Output x^k

Worker i

Initialize $x_i = x_i^+ 0$
while *not interrupted by master* **do**
  Receive x from master
  Draw sparsity mask $\mathbf{S}_p$ as
  $$\mathbb{P}[j \in \mathbf{S}_p] = p_j$$
  $[x_i^+]_{\mathbf{S}_p} \leftarrow [x - \gamma \nabla f_i(x)]_{\mathbf{S}_p}$
  $\Delta \leftarrow x_i^+ - x_i$
  Send $[\Delta]_{\mathbf{S}_p}$ to master
  $[x_i]_{\mathbf{S}_p} \leftarrow [x_i^+]_{\mathbf{S}_p}$
end

The communications per iteration are (i) a blocking send/receive from a slave to the master (in blue) of size $|\mathbf{S}|$, and (ii) a blocking send/receive from the master to the last updating slave (in red) of the current iterate. The upward communication is thus made sparse by the algorithm, and the downward communication cost depends on the structure of x^k, which is the output of a proximal operator on r. In the case of ℓ_1-regularization, x^k will become sparse after some iterations, leading to a two-way sparse algorithm. This particular case will be investigated in details in Section 3.2.

Without sparsification (i.e. $\mathbf{S}_1^k = \{1 \dots, n\}$ at any time k), this iteration corresponds to the one of the asynchronous proximal-gradient algorithm `DAve-PG` [MIMA18].

This distinguishing feature is inspired by `DAve-PG`: though it may appear conservative, it performs well in practice due to the stability of the produced iterations. The intuitive reason is that combining delayed points is more stable than using a combination of delayed directions; see the numerical comparisons of Section 2.4 of [MIM20] and Section 5 of [MIMA18].

3.1.3 Convergence analysis

We study the convergence properties of our algorithm under standard assumptions on the learning problem (P) and no apriori assumption on the system (neither on delays nor on data distribution).

We emphasize here that we do not put assumptions on the delays; for instance they do not need to be bounded or independent of the previous mask selectors ($\mathbf{S}_p^k$).

Theorem 3.2 (Reach and Limits of Sparsification)**.** *Let the functions (f_i) be μ-strongly convex ($\mu > 0$) and L-smooth. Let r be convex lsc.*

Suppose that Assumption 3.1 holds for the probability vector p, and take $\gamma \in (0, \frac{2}{\mu+L}]$, then `SPY` on $((\alpha_i), (f_i), r\ ;\ p)$ verifies for all $k \in [k_m, k_{m+1})$

$$\mathbb{E}\left\|x^k - x^\star\right\|^2 \leq \mathbb{E}\left\|\bar{x}^k - \bar{x}^\star\right\|^2 \leq \left(p_{\max}(1-\gamma\mu)^2 + 1 - p_{\min}\right)^m \max_i \left\|x_i^0 - x_i^\star\right\|^2, \tag{3.2}$$

with the shifted local solutions $x_i^\star = x^\star - \gamma_i \nabla f_i(x^\star)$.

Furthermore, using the maximal stepsize $\gamma = \frac{2}{\mu+L}$, we obtain for all $k \in [k_m, k_{m+1})$

$$\mathbb{E}\left\|x^k - x^\star\right\|^2 \leq \left(p_{\max}\left(\frac{1-\kappa_{(\mathsf{P})}}{1+\kappa_{(\mathsf{P})}}\right)^2 + 1 - p_{\min}\right)^m \max_i \left\|x_i^0 - x_i^\star\right\|^2. \tag{3.3}$$

Proof of Theorem 3.2. The co-existence of both deterministic and stochastic delays in the algorithm calls for a fundamental mathematical analysis using the notation introduced in Section 1.3.2, and in particular the notion of epoch sequence (1.31).

For a time k and a worker i, we have that $x_i^k = x_i^{k-d_i^k}$ depends on i) $x^{k-D_i^k}$ which is $\mathcal{F}^{k-D_i^k}$-measurable; and ii) $\mathbf{S}_p^{k-D_i^k}$ which is i.i.d.. First, we are going to control the term $\|x_i^k - x_i^\star\|^2$, using the same notations as in the proof of Corollary 1.23 .

Let us define $\|x\|_p^2 = \sum_{i=1}^d p_i x_{[i]}^2$ where $(p_1, \dots, p_d)$ is the vector of probabilities of Assumption 3.1. The conditional expectation can be developed as follows:

$$\begin{aligned}\mathbb{E}[\|x_i^k - x_i^\star\|^2|\mathcal{F}^{k-D_i^k}] &= \mathbb{E}[\|x_i^{k-d_i^k} - x_i^\star\|^2|\mathcal{F}^{k-D_i^k}] = \sum_{j=1}^d \mathbb{E}[(x_{i[j]}^{k-d_i^k} - x_{i[j]}^\star)^2|\mathcal{F}^{k-D_i^k}] \\ &= \|x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - (x^\star - \gamma\nabla f_i(x^\star))\|_p^2 + \|x_i^{k-D_i^k} - x_i^\star\|_{1-p}^2.\end{aligned}$$

Let us now bound both terms of this sum above using $p_{\max} = \max_i p_i$ and $p_{\min} = \min_i p_i$.

$$\begin{aligned}&\|x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - (x^\star - \gamma\nabla f_i(x^\star))\|_p^2 + \|x_i^{k-D_i^k} - x_i^\star\|_{1-p}^2 \\ &\quad\le p_{\max}\|x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - (x^\star - \gamma\nabla f_i(x^\star))\|^2 + (1-p_{\min})\|x_i^{k-D_i^k} - x_i^\star\|^2.\end{aligned}$$

We now use the μ-strong convexity and L-smoothness of f_i to write (see Lemma 1.5),

$$\begin{aligned}&\|x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - (x^\star - \gamma\nabla f_i(x^\star))\|^2 \\ &\quad\le \left(1 - \frac{2\gamma\mu L}{\mu+L}\right)\left\|x^{k-D_i^k} - x^\star\right\|^2 - \gamma\left(\frac{2}{\mu+L} - \gamma\right)\left\|\nabla f_i(x^{k-D_i^k}) - \nabla f_i(x^\star)\right\|^2 \\ &\quad\le \left[\left(1 - \frac{2\gamma\mu L}{\mu+L}\right) - \mu^2\gamma\left(\frac{2}{\mu+L} - \gamma\right)\right]\left\|x^{k-D_i^k} - x^\star\right\|^2 \\ &\quad= (1-\gamma\mu)^2\left\|x^{k-D_i^k} - x^\star\right\|^2.\end{aligned}$$

Thus, for any $\gamma \in (0, 2/(\mu+L)]$, one can drop the last non-negative term,

$$\begin{aligned}\mathbb{E}[\|x_i^k - x_i^\star\|^2|\mathcal{F}^{k-D_i^k}] &\le p_{\max}(1-\gamma\mu)^2\left\|x^{k-D_i^k} - x^\star\right\|^2 + (1-p_{\min})\|x_i^{k-D_i^k} - x_i^\star\|^2 \\ &\le p_{\max}(1-\gamma\mu)^2\left\|\overline{x}^{k-D_i^k} - \overline{x}^\star\right\|^2 + (1-p_{\min})\|x_i^{k-D_i^k} - x_i^\star\|^2,\end{aligned}$$

where we used that $\|x^{k-D_i^k} - x^\star\|^2 = \|\mathbf{prox}_{\gamma r}(\overline{x}^{k-D_i^k}) - \mathbf{prox}_{\gamma r}(\overline{x}^\star)\|^2 \le \|\overline{x}^{k-D_i^k} - \overline{x}^\star\|^2$ by definition and non-expansiveness of the proximity operator of r.

Taking full expectation on both sides, we get

$$\mathbb{E}\|x_i^k - x_i^\star\|^2 \le p_{\max}(1-\gamma\mu)^2\,\mathbb{E}\left\|\overline{x}^{k-D_i^k} - \overline{x}^\star\right\|^2 + (1-p_{\min})\mathbb{E}\|x_i^{k-D_i^k} - x_i^\star\|^2.$$

Then, using that $\overline{x}^{k-D_i^k} - \overline{x}^\star = \sum_{i=1}^M \alpha_i (x_i^{k-D_i^k} - \overline{x}_i^\star)$ and the convexity of $\|\cdot\|^2$, we get

$$\begin{aligned}
\mathbb{E}\|x_i^k - x_i^\star\|^2 \leq\ & p_{\max}(1-\gamma\mu)^2 \sum_{j=1}^M \alpha_j \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2 + (1-p_{\min})\mathbb{E}\|x_i^{k-D_i^k} - x_i^\star\|^2 \\
\leq\ & p_{\max}(1-\gamma\mu)^2 \max_{j=1,\ldots,M} \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2 + (1-p_{\min}) \max_{j=1,\ldots,M} \mathbb{E}\|x_j^{k-D_i^k} - x_j^\star\|^2 \\
\leq\ & \left(p_{\max}(1-\gamma\mu)^2 + 1 - p_{\min}\right) \max_{j=1,\ldots,M} \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2.
\end{aligned}$$

Let $c_k = \max_{i=1,\ldots,M} \mathbb{E}\left\|x_j^k - x_j^\star\right\|^2$ and $\beta = \left(p_{\max}(1-\gamma\mu)^2 + 1 - p_{\min}\right)$, then the above result implies that

$$c_k \leq \beta \max_{j=1,\ldots,M} c_{k-D_j^k}$$

and using the definition of the sequence (k_m), we get

$$\begin{aligned}
c_{k_m} &\leq \beta \max_j c_{k_m - D_j^{k_m}} \leq \beta \max_{\ell \in [k_{m-1}, k_m)} c_\ell \\
c_{k_m+1} &\leq \beta \max(c_{k_m}, \max_{\ell \in [k_{m-1}, k_m)} c_\ell) \leq \beta \max_{\ell \in [k_{m-1}, k_m)} c_\ell.
\end{aligned}$$

Thus for all $k \geq k_m$, $c_k \leq \beta \max_{\ell \in [k_{m-1}, k_m)} c_\ell$. This implies that the sequence $\widetilde{c}_m$ defined by $\widetilde{c}_m = \max_{\ell \in [k_m, k_{m+1})} c_\ell$ has an exponential bound:

$$\widetilde{c}_m \leq \beta\, \widetilde{c}_{m-1} \leq \beta^m\, \widetilde{c}_0 \leq \beta^m \max_{i=1,\ldots,M} \|x_i^0 - x_i^\star\|^2.$$

Finally, it suffices to use once again the non-expansivity of the proximity operator of r and the definitions to get that for all $k \in [k_m, k_{m+1})$,

$$\mathbb{E}\|x^k - x^\star\|^2 \ \leq\ \mathbb{E}\|\overline{x}^k - \overline{x}^\star\|^2 \ \leq\ \sum_{i=1}^M \alpha_i \mathbb{E}\|x_i^k - x_i^\star\|^2 \ \leq\ c_k \ \leq\ \beta^m \max_{i=1,\ldots,M} \|x_i^0 - x_i^\star\|^2, \tag{3.4}$$

which concludes the proof. □

This result establishes bounds that may or may not lead to convergence, depending on the selection probabilities. First, if all probabilities are equal to 1, the algorithm boils down to `DAve-PG` and Theorem 3.2 coincides with Theorem 1.14. However, in more general cases, this result has to be interpreted more carefully as developed in the following section.

Algorithm 12 `U-SPY` on $((\alpha_i),(f_i),r\ ;\ \pi)$

Worker i

Initialize $x_i = x_i^+ = x = \bar{x}^0$
while *not interrupted by master* **do**
  Receive x from master
  Draw sparsity mask $\mathbf{S}_\pi$ as
  $$\mathbb{P}\left[j \in \mathbf{S}_\pi\right] = \pi$$
  $[x_i^+]_{\mathbf{S}_\pi} \leftarrow [x - \gamma \nabla f_i(x)]_{\mathbf{S}_\pi}$
  $\Delta \leftarrow x_i^+ - x_i$
  Send $[\Delta]_{\mathbf{S}_\pi}$ to master
  $[x_i]_{\mathbf{S}_\pi} \leftarrow [x_i^+]_{\mathbf{S}_\pi}$
end

3.2 On the sparsification choice for ℓ_1 regularized problems

In this section, we present 3 different options of sparsity mask selectors for ℓ_1 regularized problems.

Assuming that all machines are responsive (i.e. $m \to \infty$ when $k \to \infty$), the inequality (3.2) gives linear convergence of the mean squared error in terms of epochs if

$$\frac{p_{\min}}{p_{\max}} > (1-\gamma\mu)^2 \overset{\gamma=\frac{2}{\mu+L}}{\geq} \left(\frac{1-\kappa_{(\mathsf{P})}}{1+\kappa_{(\mathsf{P})}}\right)^2 \tag{3.5}$$

and thus the behavior of the algorithm depends if the selection is performed uniformly (and thus structure-blind) or non-uniformly (to encompass some prior information).

3.2.1 Inefficiency of uniform sparsification

If the selection is *uniform* `U-SPY` (see Algorithm 12), i.e. if $p_i = \pi \in (0,1]$ for all i, $x^k \to x^\star$ in probability. Furthermore, the mean squared error vanishes linearly in terms of epochs with a rate $(1 - \pi\gamma\mu(2-\gamma\mu))$. As a result, under the mild assumption over the delays (Assumption 3.3) we could prove an almost sure convergence $\bar{x}^k \to \bar{x}^\star$ and as a corollary the identification property (see Theorem 1.17) of `U-SPY`.

Assumption 3.3 (Additional assumption for identification)**.** *The number of iterations between two full updates cannot grow exponentially, i.e.* $k_{m+1} - k_m = o(\exp(m))$*. This assumption is rather mild and subsumes the usual bounded delay assumption.*

This assumption, together with an exponential decrease in terms of mean squared error, implies the following result.

Lemma 3.4 (Almost-sure convergence of `SPY`)**.** *Let the functions* (f_i) *be* μ*-strongly convex* ($\mu > 0$) *and* L*-smooth and assumptions 3.1 and 3.3 hold. Select the probability vector* p *such that* (3.5) *holds then* $\bar{x}^k$ *converges almost surely to* $\bar{x}^k$*.*

Proof of Lemma 3.4. Recall first from the end of the proof of Theorem 3.2 that we have for some C and $\delta \in]0, 1]$

$$\mathbb{E}\|\bar{x}^k - \bar{x}^\star\|^2 \leq C(1-\delta)^m. \tag{3.6}$$

Notice that the exponent is not the iteration number but rather the number of stopping times (k_m) before k. Thus, we can decompose the sum of the $(\|x^k - x^\star\|^2)_k$ by epochs and get in expectation

$$\mathbb{E}\left[\sum_{k=1}^{+\infty} \|\bar{x}^k - \bar{x}^\star\|^2\right] \leq \mathbb{E}\left[\sum_{m=1}^{+\infty} \sum_{k=k_m}^{k_{m+1}-1} \|\bar{x}^k - \bar{x}^\star\|^2\right] \leq C \sum_{m=1}^{+\infty} (k_{m+1} - 1 - k_m)(1-\delta)^m.$$

Assumption 3.3 now allows us to bound $k_{m+1} - 1 - k_m$ and to get that this sum is finite almost surely:

$$1 = \mathbb{P}\left[\sum_{k=1}^{+\infty} \|\bar{x}^k - \bar{x}^\star\|^2 < +\infty\right] \leq \mathbb{P}\left[\|\bar{x}^k - \bar{x}^\star\|^2 \to 0\right],$$

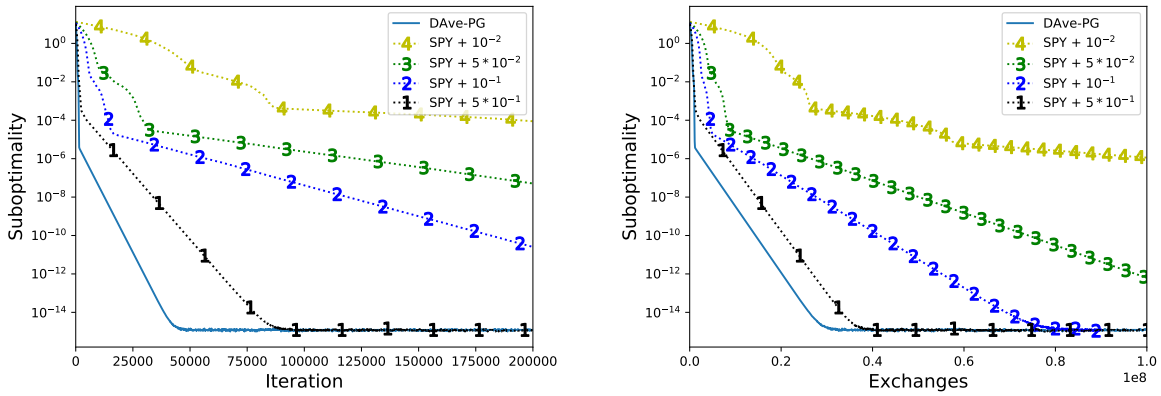
which means that $(\bar{x}^k)$ converges almost surely to $\bar{x}^\star$. □

Almost sure convergence result guarantees the identification of the near-optimal support by iterates of the algorithms in the finite time (see Theorem 1.17) with $u^k = \bar{x}^k$ and (1.38).

Unfortunately, uniform selection usually results in poor performance in practice, as displayed in Figure 3-1. Hence the need for a more adaptive sparsification manner.

3.2.2 Efficiency of adaptive sparsification

The idea of adaptively using the identified structure in coordinate descent for ℓ_1-regularized problems methods was recently developed in [GIM20] (see Section 2.2). Mathematically, this means select coordinates in the mask as follows performing Algorithm 13



(a) The functional suboptimality versus the amount of iterations/gradient computations/communication rounds done.

(b) The functional suboptimality versus amount of coordinates sent during communication rounds.

Figure 3-1. Evolution of `U-SPY` iterates comparing to `DAve-PG`. We consider logistic regression objective function with elastic net regularizer on madelon dataset from LibSVM library [CL11]. We denote by "SPY + π" the `U-SPY` with probability π.

Assumption 3.5 (Sparsification with support identification)**.** *The sparsity mask selectors* $(\mathbf{S}_\pi^k)$ *are random variables such that* $\mathbb{P}[j \in \mathbf{S}_\pi^k] = 1$ *if* $j \in \operatorname{supp}(x^k)$ *and either:*

Option I. $\mathbb{P}[j' \in \mathbf{S}_\pi^k] = \pi \in (0, 1]$ *for all* $j' \notin \operatorname{supp}(x^k)$.

Option II. $|\mathbf{S}_\pi^k| = |\operatorname{supp}(x^k)| + \pi(n - |\operatorname{supp}(x^k)|) \in \{1, .., n\}$ *almost surely and* $\mathbb{P}[j' \in \mathbf{S}_\pi^k] = \mathbb{P}[j'' \in \mathbf{S}^k]$ *for all* $j', j'' \notin \operatorname{supp}(x^k)$.

In words, this means updating/communicate all coordinates in the support of the last computed master point x^k and selecting coordinates outside the support with *exploration probability* π for the Option I and select the set of fixed size for the Option II. These two different approaches have no difference in terms of theoretical proofs since the expectation $\mathbb{E}\left[x_{[\mathbf{S}_\pi^k]}|x^k\right]$ is the same for both of them and defined as

$$\left[\mathbb{E}\left[x_{[\mathbf{S}_\pi^k]}|x^k\right]\right]_j = \begin{cases} x_{[j]} & \text{if } j \in \operatorname{supp}(x^k) \\ \pi x_{[j]} & \text{otherwise} \end{cases}$$

This kind of sampling showed tremendous gains compared to uniform sampling; however, it adds two difficulties with respect to Lemma 3.2:

Algorithm 13 `I-SPY` on $((\alpha_i), (f_i), \pi)$

Worker i

Initialize $x_i = x_i^+ = \bar{x}^0$
while not interrupted **do**
  Receive x from master
  Draw sparsity mask $\mathbf{S}_\pi$ as

$$\mathbb{P}[j \in \mathbf{S}_\pi] = \begin{cases} 1, & \text{if } j \in \text{supp}(x) \\ \pi, & \text{if } j \notin \text{supp}(x) \end{cases}$$

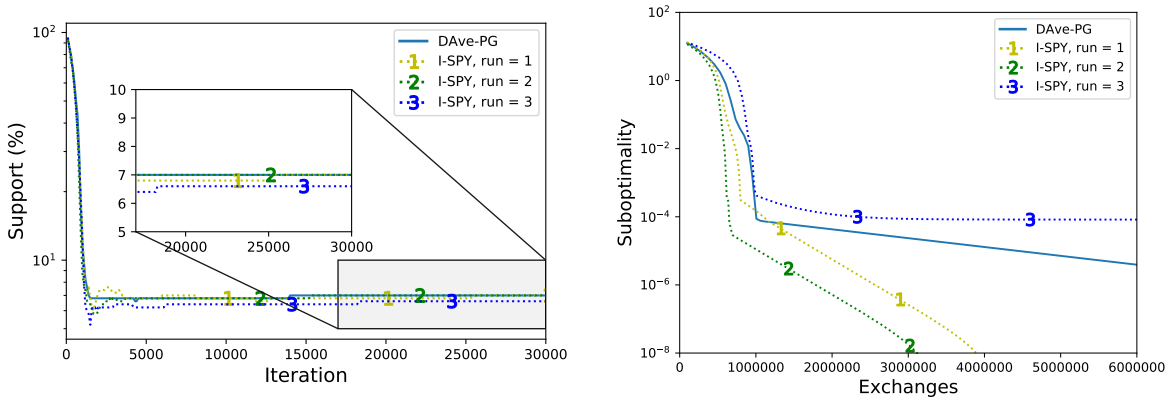
  $\left[x_i^+\right]_{\mathbf{S}_\pi} \leftarrow [x - \gamma \nabla f_i(x)]_{\mathbf{S}_\pi}$
  $\Delta = x_i^+ - x_i$
  $x_i = x_i^+$
  Send Δ to master
end while

- a *good conditioning* is necessary to allow for a small $p_{\min} = \pi$ and $p_{\max} = 1$. More precisely, from Eq. (3.5), we get the condition

$$\kappa_{(\mathsf{P})} > \kappa_{\min} = \frac{1 - \sqrt{\pi}}{1 + \sqrt{\pi}} \tag{3.7}$$

  on the minimal conditioning to allow for a probability π of selection outside the support. As we aim at taking π quite small in order not to communicate much more than needed, this is a stringent condition.

- the sampling is *not i.i.d.* anymore since the probabilities depend on the points generated by the algorithm.



(a) The size of the support versus the amount of iterations/gradient computations/communication rounds done.

(b) The functional suboptimality versus amount of coordinates sent during communication rounds.

Figure 3-2. Evolution of `I-SPY` iterates comparing to `DAve-PG`. We consider logistic regression objective function with elastic net regularizer on madelon dataset from LibSVM library [CL11]. We present 3 different runs of `I-SPY` with adaptive selection as in Option I of Assumption 3.5 with probability $\pi = 0.1$ showing both convergence and divergence of `I-SPY`.

In Figure 3-2, we see that the algorithm with such selection (`I-SPY`) could misidentify and diverge as a result; however, if it converges, it identifies the near-optimal support and has the same linear rate as `DAve-PG` with fewer communication cost. We further investigate this algorithm in Section 3.3.

3.2.3 Scaled adaptive sparsification

One of the possible modifications of `SPY` is a scaled version of it `S-SPY` see Algorithm 14. This is often used in sparsification methods for `SGD` (see e.g. [WKSZ17]) to have the unbiased estimator of the sparsified object. In our case, scaling helps to figure out the difference in probabilities and keep the "probability-gap" equal to 0. More precisely, every worker performs the following update

$$x_{i[j]}^k = \begin{cases} \frac{p_{\min}}{p_j}\left(x^{k-D_i^k} - \gamma \nabla f_i(x^{k-D_i^k})\right)_{[j]} + \left(1 - \frac{p_{\min}}{p_j}\right) x_{i[j]}^{k-1} & \text{if } \left| \begin{array}{l} i = i^k \\ j \in \mathbf{S}_p^{k-D_i^k} \end{array} \right. \\ x_{i[j]}^{k-1} & \text{otherwise} \end{cases} \tag{3.8}$$

Algorithm 14 S-SPY on $((\alpha_i), (f_i), r\ ;\ p)$

Worker i

Initialize $x_i = x_i^+ = x = \bar{x}^0$

Calculate scaled probability vector $q = \left(\frac{p_{\min}}{p_1}, \frac{p_{\min}}{p_2}, \ldots, \frac{p_{\min}}{p_n}\right)$

while *not interrupted by master* **do**

Receive x from master

Draw sparsity mask $\mathbf{S}_p$ as

$$\mathbb{P}\left[j \in \mathbf{S}_p\right] = p_j$$

$[x_i^+]_{\mathbf{S}_p} \leftarrow [q]_{\mathbf{S}_p} * [x - \gamma \nabla f_i(x)]_{\mathbf{S}_p} + [\mathbf{1^n} - q]_{\mathbf{S}_p} * [x_i]_{\mathbf{S}_p}$ [a]

$\Delta \leftarrow x_i^+ - x_i$

Send $[\Delta]_{\mathbf{S}_p}$ to master

$[x_i]_{\mathbf{S}_p} \leftarrow [x_i^+]_{\mathbf{S}_p}$

end

[a] Here we denote by $\mathbf{1^n} \in \mathbb{R}^n$ the identity vector and by $*$ we denote the coordinate-wise vector-to-vector multiplication.

The learning rate of the update is "$\frac{1}{p_{\min}}$ times smaller" than γ because of the scaling. On the other hand, this algorithm is proven to converge for any L-smooth and μ-strongly convex problem independent of its conditioning.

Theorem 3.6 (Convergence of S-SPY)**.** *Let the functions (f_i) be μ-strongly convex ($\mu > 0$) and L-smooth. Let r be convex lsc. Suppose that Assumption 3.1 holds for the probability vector p, and take $\gamma \in (0, \frac{2}{\mu+L}]$. Then S-SPY on $((\alpha_i), (f_i), r\ ;\ p)$ verifies for all $k \in [k_m, k_{m+1})$*

$$\mathbb{E}\left\|x^k - x^\star\right\|^2 \leq \mathbb{E}\left\|x^k - x^\star\right\|^2 \leq \left(1 - p_{\min}\gamma\mu(2 - \gamma\mu)\right)^m \max_i \left\|x_i^0 - x_i^\star\right\|^2, \tag{3.9}$$

with the shifted local solutions $x_i^\star = x^\star - \gamma_i \nabla f_i(x^\star)$.

Proof of Theorem 3.6. The proof follows the same arguments as the one of Theorem 3.2, expect for one crucial inequality. Using the same notations as in the proof of Theorem 3.2,

we have

$$\begin{aligned}
\mathbb{E}[\|x_i^k - x_i^\star\|^2 | \mathcal{F}^{k-D_i^k}] = \; & \mathbb{E}[\|x_i^{k-D_i^k} - x_i^\star\|^2 | \mathcal{F}^{k-D_i^k}] = \sum_{j=1}^n \mathbb{E}[([x_i^{k-D_i^k} - x_i^\star]_{[j]})^2 | \mathcal{F}^{k-D_i^k}] \\
&= \sum_{j=1}^n p_j \left(\frac{p_{\min}}{p_j} \left[x^{k-D_i^k} - \gamma \nabla f_i(x^{k-D_i^k}) \right]_{[j]} + \left(1 - \frac{p_{\min}}{p_j}\right) \left[x_i^{k-D_i^k} \right]_{[j]} - [x_i^\star]_{[j]} \right)^2 \\
&\qquad + (1-p_j) \left(\left[x_i^{k-D_i^k} \right]_{[j]} - [x_i^\star]_{[j]} \right)^2 \\
&\leq \sum_{j=1}^n p_j \frac{p_{\min}}{p_j} \left(\left[x^{k-D_i^k} - \gamma \nabla f_i(x^{k-D_i^k}) \right]_{[j]} - [x_i^\star]_{[j]} \right)^2 \\
&\quad + p_j \left(1 - \frac{p_{\min}}{p_j}\right) \left(\left[x_i^{k-D_i^k} \right]_{[j]} - [x_i^\star]_{[j]} \right)^2 + (1-p_j) \left(\left[x_i^{k-D_i^k} \right]_{[j]} - [x_i^\star]_{[j]} \right)^2 \\
&= \; p_{\min} \|x^{k-D_i^k} - \gamma \nabla f_i(x^{k-D_i^k}) - (x^\star - \gamma \nabla f_i(x^\star))\|^2 + (1-p_{\min}) \|x_i^{k-D_i^k} - x_i^\star\|^2
\end{aligned}$$

where the inequality follows from convexity of $\|\cdot\|^2$. The rest of the proof follows unchanged. □

As we could see, the theoretical rate depends only on the minimal probability of coordinate to be selected; however, the size of communication is the smallest if all the probabilities are the same.

Adaptive coordinate selection could be combined with this scaling technique to perform `IS-SPY` see Algorithm 15.

On the one hand, this algorithm still has the dimension-reduction property - after the moment of identification (see Lemma 3.4), all the critical coordinates update every iteration. However, as we could see from the theorem, the rate of Algorithm 15 is the same as for Algorithm 11 with uniform sampling; however, in practice it performs better in case $r = \lambda\|\cdot\|_1$ thanks to the identification property (Lemma 3.4). More precisely, while uniform sampling takes place, coordinates from the support are selected with probability π that makes it worse than selecting all of them but scaling. Unfortunately, even the identification property of the `IS-SPY` algorithm does not help it to perform at least as good as `DAve-PG` both in terms of iterations and communications (see Figure 3-3).

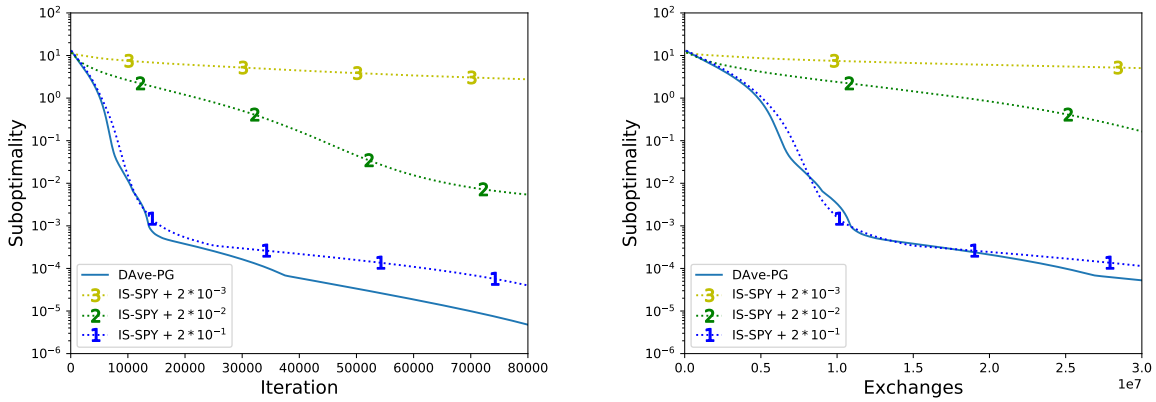
Algorithm 15 `IS-SPY` on $((\alpha_i), (f_i), r\ ;\ \pi)$

Worker i

Initialize $x_i = x_i^+ = x = \bar{x}^0$
while *not interrupted by master* **do**
  Receive x from master
  Draw sparsity mask $\mathbf{S}_\pi$ as

$$\mathbb{P}[j \in \mathbf{S}_\pi] = \begin{cases} 1, & \text{if } j \in \text{supp}(x) \\ \pi, & \text{if } j \notin \text{supp}(x) \end{cases}$$

  $[x_i^+]_{\mathbf{S}_\pi \setminus \text{supp}(x)} \leftarrow [x - \gamma \nabla f_i(x)]_{\mathbf{S}_\pi \setminus \text{supp}(x)}$
  $[x_i^+]_{\text{supp}(x)} \leftarrow \pi [x - \gamma \nabla f_i(x)]_{\text{supp}(x)} + (1-\pi)[x_i]_{\text{supp}(x)}$
  $\Delta \leftarrow x_i^+ - x_i$
  Send $[\Delta]_{\mathbf{S}_\pi}$ to master
  $[x_i]_{\mathbf{S}_\pi} \leftarrow [x_i^+]_{\mathbf{S}_\pi}$
end



(a) The functional suboptimality versus the amount of iterations/gradient computations/communication rounds done.

(b) The functional suboptimality versus amount of coordinates sent during communication rounds.

Figure 3-3. Evolution of `IS-SPY` iterates comparing to `DAve-PG`. We consider logistic regression objective function with elastic net regularizer on madelon dataset from LibSVM library [CL11]. We denote by "IS-SPY + π" the `IS-SPY` with selected as in Option I of Assumption 3.5 with probability π.

3.3 Better analysis for I-SPY in case of ℓ_1 regularized problems

In this section, we present some additional analysis of I-SPY see Algorithm 13.

3.3.1 Identification and better rate

Let us consider the optimization problem (P) with ℓ_1 regularization term, i.e.,

$$\min_{x\in\mathbb{R}^n} \left\{ F(x) = \sum_{i=1}^{M} \alpha_i f_i(x) + \lambda \|x\|_1 \right\}. \tag{P_{ℓ_1}}$$

As before, we consider the distributed setup, where m observations are split down over M machines, each machine i having a private subset $\mathcal{D}_i$ of the examples. We also denote by $\alpha_i = |\mathcal{D}_i|/m$ the proportion of observations locally stored in machine i, hence $\sum_{i=1}^{M} \alpha_i = 1$. $f_i(x) = \frac{1}{|\mathcal{D}_i|} \sum_{j\in\mathcal{D}_i} l_j(x)$ is the local empirical risk at machine i (l_j standing for the smooth loss function for example j).

Let us recall the theoretical result (see Theorem 3.2), specified for this selection of I-SPY. Using the same notations as in Theorem 3.2, the iterates of I-SPY satisfy

$$\mathbb{E}\left\|x^k - x^\star\right\|^2 \leq \mathbb{E}\left\|\bar{x}^k - \bar{x}^\star\right\|^2 \leq \left((1-\gamma\mu)^2 + 1 - \pi\right)^m \max_i \left\|x_i^0 - x_i^\star\right\|^2, \tag{3.10}$$

This general result deserves several comments:

- We retrieve the convergence results of [MIM20] in the case where there is no sparsification (i.e., $\pi = 1$), and if there is no delay, we recover the rate of vanilla proximal-gradient algorithm.
- Assuming that all machines are responsive (i.e. $s \to \infty$ when $k \to \infty$), the inequality (3.10) gives convergence if $\beta > 0$, i.e. $\pi > 1 - \alpha$. In other words, when we sample entries non-uniformly, we still have convergence if the probability of selection is big enough.
- Finally, when the problem is well-conditioned (i.e., $\mu \simeq L$ and thus $\alpha \simeq 1$), the algorithm is guaranteed to converge for any reasonable choice of π.

Taking into account that there is no almost sure convergence in our case, we could not talk about identification result [FMP18] (see Section 1.4) out of the box. Let us introduce the following assumption, that will help us t obtain an identification result.

Assumption 3.7 (On convergence). *Let us assume that for any $\varepsilon > 0$ there exists iterate number K such that for any $k > K$, the average point $\|\bar{x}^k - \bar{x}^\star\|_2^2 < \varepsilon$ is ε-close to the final solution.*

As we could see from the Theorem 1.17, if this assumption holds for any particular run of the Algorithm 13, then the iterates (x^k) identifies the near-optimal support in finite time. Furthermore, under the additional non-degeneracy assumption, the iterates will identify the optimal support.

Lemma 3.8 (Identification). *Suppose that Assumption 3.7 holds; furthermore, let us assume that problem (P_{ℓ_1}) is non-degenerate (ND). Then for any $k > K$, we have :*

$$\operatorname{supp}(x^k) = \operatorname{supp}(x^\star). \tag{3.11}$$

Proof of Lemma 3.8. First, using nonexpansiveness of the proximal operator we could have that for any $\varepsilon > 0$ there exists K big enough such that for any $k > K$:

$$\|x^k - x^\star\|_2^2 = \|\mathbf{prox}_{\gamma\lambda_1\|\cdot\|_1}(\bar{x}^k) - \mathbf{prox}_{\gamma\lambda_1\|\cdot\|_1}(\bar{x}^\star)\|_2^2 \leq \|\bar{x}^k - \bar{x}^\star\|_2^2 \leq \varepsilon.$$

Now, using the Corollary 1.20 with $u^k = \bar{x}^k$ we get the result of the lemma.

□

As a result, under these assumptions, ant together with the "convergence" of the algorithm, we could even get the same rate as for non-sparsified `DAve-PG` [MIMA18].

Theorem 3.9 (Better rate of `I-SPY`). *Suppose that all functions $\{f_i\}_{i=1,..,M}$ in (P) are μ-strongly convex and L-smooth In addition suppose that Assumption 3.7 holds. For any $\gamma \in (0, 2/(\mu + L)]$ and for any $k \in [k_s, k_{s+1})$ we have:*

$$\|x^k - x^\star\|^2 = \mathcal{O}_p\left(\left(1 - \frac{2\gamma\mu L}{\mu + L}\right)^s\right), \tag{3.12}$$

where $\mathcal{O}_p$ denotes big O in probability.

Proof. First, let us define the identification moment $k_{\mathbf{i}}$ as follows

$$k_{\mathbf{i}} = \max\{k : \operatorname{supp}(x^{k-1}) \neq \operatorname{supp}(x^\star)\},$$

In addition, let us denote by $s_{\mathbf{i}}$ the epoch number such that

$$k_i \in [k_{s_{\mathbf{i}}}, k_{s_{\mathbf{i}}+1}).$$

Finally, let us denote by

$$C_{\mathbf{i}} = \left(1 - \frac{2\gamma\mu L}{\mu + L}\right)^{-s_{\mathbf{i}}} \max_{i=1,\dots,M} \|x_i^{k_{s_{\mathbf{i}}+1}} - x_i^\star\|_2^2.$$

Using the result of Lemma 3.8 we have that $C_{\mathbf{i}}$ is almost sure finite since s_i is almost sure finite.

Let us consider k big enough, such that $\text{supp}(x^k) = \text{supp}(x^\star)$, and let $\mathcal{C} = \{x \in \mathbb{R}^n : \text{supp}(x) \subseteq \text{supp}(x^\star)\}$ and let $\mathbf{proj}_{\mathcal{C}}(\cdot)$ be an orthogonal projection onto $\mathcal{C}$.

Now, using the stability of the support of the algorithm run, let us consider an iterate of the `I-SPY` algorithm.

$$\begin{aligned} x^k &= \mathbf{prox}_{\gamma\lambda_1\|\cdot\|_1}(\bar{x}^{k-1} + \alpha_i[\Delta^k]_{\mathbf{S}_\pi^{k-D_{ik}^k}}) \\ &= \mathbf{proj}_{\mathcal{C}}(\mathbf{prox}_{\gamma\lambda_1\|\cdot\|_1}(\bar{x}^{k-1} + \alpha_i[\Delta^k]_{\mathbf{S}_\pi^{k-D_{ik}^k}})). \end{aligned}$$

Using the fact that both $\mathbf{proj}_{\mathcal{C}}(\cdot)$ and $\mathbf{prox}_{\gamma\lambda_1\|\cdot\|_1}(\cdot)$ are coordinate-wise separable we can change the order :

$$\begin{aligned} x^k &= \mathbf{prox}_{\gamma\lambda_1\|\cdot\|_1}(\mathbf{proj}_{\mathcal{C}}(\bar{x}^{k-1} + \alpha_i\Delta^k)) \\ &= \mathbf{prox}_{\gamma\lambda_1\|\cdot\|_1}(\mathbf{proj}_{\mathcal{C}}(\bar{x}^{k-1}) + \mathbf{proj}_{\mathcal{C}}(\alpha_i\Delta^k)) \\ &= \mathbf{prox}_{\gamma\lambda_1\|\cdot\|_1}(\mathbf{proj}_{\mathcal{C}}(\bar{x}^{k-1}) + \mathbf{proj}_{\mathcal{C}}(\alpha_i[x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - x_i^{k-D_i^k}]_{\mathbf{S}_\pi})) \\ &= \mathbf{prox}_{\gamma\lambda_1\|\cdot\|_1}(\mathbf{proj}_{\mathcal{C}}(\bar{x}^{k-1}) + \underbrace{\mathbf{proj}_{\mathcal{C}}(\alpha_i(x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - x_i^{k-D_i^k}))}_{\text{supp}(x^\star)\subseteq\mathbf{S}_\pi^{k-D_{ik}^k}}) \\ &= \mathbf{prox}_{\gamma\lambda_1\|\cdot\|_1}(\mathbf{proj}_{\mathcal{C}}(\bar{x}^{k-1} + \alpha_i([x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - x_i^{k-D_i^k}]_{\mathbf{S}_1^{k-D_{ik}^k}})). \end{aligned}$$

Hence the sequence of parameters generated by projected `I-SPY` algorithm with $\pi = 1$ (`DAve-PG` algorithm) when starting from $x^0 = x^{k_{s_{\mathbf{i}}+1}}$ and $x_j^0 = x_j^{k_{s_{\mathbf{i}}+1}}$ for the worker machines, satisfy Lemma 3.2 which gives :

$$\|x^k - x^\star\|_2^2 \leq \left(1 - \frac{2\gamma\mu L}{\mu + L}\right)^s \max_{i=1,\dots,M} \|x_i^0 - x_i^\star\|_2^2, \quad \forall\ k \in [k_s, k_{s+1})$$

where, the expectation can be dropped as far as we have a deterministic sequence of points.

Finally, combining two parts we have:

$$\|x^k - x^\star\|_2^2 \leq \left(1 - \frac{2\gamma\mu L}{\mu + L}\right)^s C_{\mathbf{i}}, \quad \forall\ k \in [k_s, k_{s+1})$$

that finishes the proof. □

We can see from this theorem that after identification, the convergence rate does not depend on probability π. This means that communications become smaller with the same rate. On the other hand, as identification depends on this probability π, in practice, the selection of π should be a trade-off between speed of identification and the size of sparsification.

3.3.2 Numerical experiments

In the previous sections we proved the convergence of `I-SPY` in the case where the mask is formed with a high probability π. In this section, we present numerical results providing empirical evidence on a faster execution of `I-SPY` with lower communication cast than if the mask is not used.

Experimental setup

In our experiments, we consider $\ell_1 - \ell_2$-regularized Logistic Regression surrogate loss that is common to many machine learning and signal processing applications and which can be minimized in a distributed way. With respect to our composite learning problem (P_{ℓ_1}), that is :

$$\forall i \in \{1, \ldots, M\}, f_i(x) = \frac{1}{|\mathcal{S}_i|} \sum_{(\mathbf{x}_j, y_j) \in \mathcal{D}_i} \left[\log\left(1 + e^{-y_j \mathbf{x}_j^\top x}\right) + \frac{\lambda_2}{2}\|x\|_2^2\right] \tag{3.13}$$

where $\mathcal{D}_i = (\mathbf{x}_j, y_j)_{j \in \{1,\ldots,|\mathcal{S}_j|} \in (\mathbb{R}^n \times \{-1, +1\})^{|\mathcal{D}_j|}$ is the sub-part of the training set stored in the worker machine $i \in \{1, \ldots, M\}$.

We performed experiments on three publicly available datasets[1]. Each dataset is normalized by dividing each feature characteristic by the maximum of the absolute value in the column using the scikit-learn Transformer API.[2] In Table 3.1, we present some statistics for these datasets as well as the percentage of no-zero entries of the final parameter (supp($x^\star$)). For the communications between the master and the workers, we

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[2] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html

used the message passing interface for Python (MPI4py)[3]. We compared our approach I-SPY with its direct competitor DAve-PG [MIMA18] which is also a delay-independent asynchronous technique with constant stepsize but does not use a sparsification mask. For comparisons, we plot objective values as their relative distance to the optimum, referred to as suboptimality, with respect to time, and also with respect to iterations and the number of exchanges for different values of the probability π used in the mask. We also present the dependence of sparsity of iterates to the number iterates.

Speed of convergence

Figure 3-4 (top) presents suboptimality, as the difference between the objective function and its minimum with respect to time for the I-SPY algortihm with four values of probability π, to form the mask, and the DAve-PG algorithm [MIMA18] with $M = 20$ workers on real-sim and rcv1_train datasets. To this end, the minimum of the loss function (P_{ℓ_1}), using (3.13), is first obtained with a precision $\epsilon = 10^{-15}$. As it can be observed, for larger values of the probability π; I-SPY converges much faster than DAve-PG. This is mainly because that I-SPY passes through the whole data (iteration) in lesser time than DAve-PG as it can be seen in Figure 3-4 (down). For both datasets we notice that for reaching the same number of epochs, at the beginning of each plateau on the left hand side of the plots, I-SPY put ten times less time than DAve-PG for all values of the probability π. In the supplementary, we provide more plots on all datasets for different number of workers and parameter λ_1.

Cost of communication

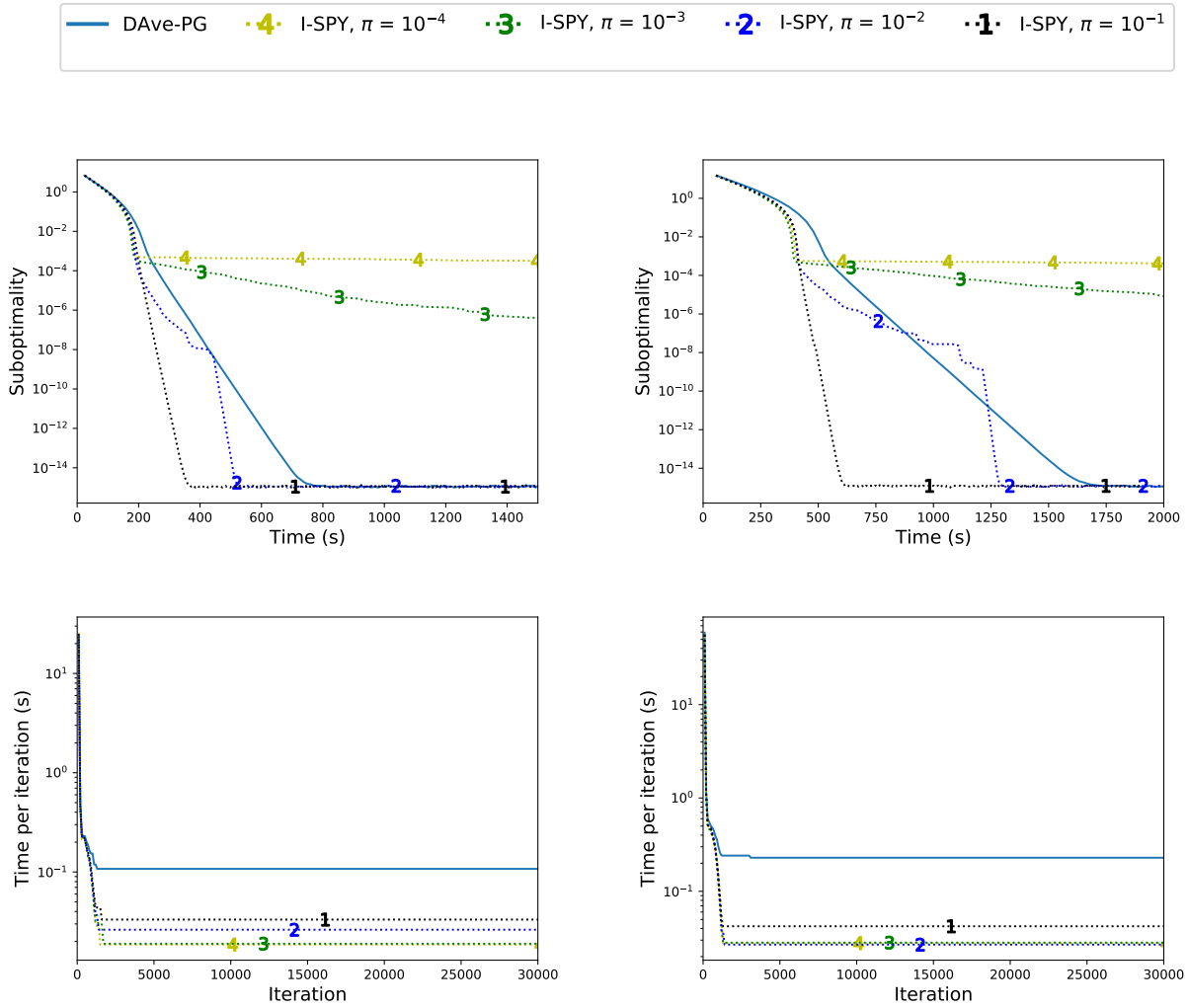
We have computed the cost of communication, as the number of exchanges, between the master and the worker machines until convergence for different values of the probability π

Dataset	m	n	λ_1	λ_2	$\lvert\text{supp}(x^\star)\rvert$ in (%)
madelon	2000	500	2×10^{-2}	10^{-3}	7
real-sim	72309	20958	10^{-4}	10^{-3}	8.6
rcv1_train	20242	47236	10^{-4}	10^{-3}	4.1

Table 3.1. Statistics of datasets used in our experiments; m, n, λ_1, λ_2 and support are respectively the size of the training set, the number of parameters, the hyperparmeters corresponding to ℓ_1 and ℓ_2 regularization terms and the percentage of non-zero entries of the final solution; $\text{supp}(x^\star)$.

[3] https://mpi4py.readthedocs.io/en/stable/citing.html

to form the mask. In the case where π is low, we know from the previous section that there is no guarantee that the algorithm `I-SPY` converges. However, note that as all workers are minimizing their local convex objectives, after one round of communication between the master and all workers, it is easy to detect when the global objective (P_{ℓ_1}) does not decrease at the master level and the algorithm can be stopped and restarted in this case. In Figure 3-5 we plot the amount of exchanges between the master and the



(a) real-sim (b) rcv1_train

Figure 3-4. Objective loss (P_{ℓ_1}) suboptimality versus time in second (top) and epochs with respect to time (down), for $M = 20$ workers and $\lambda_1 = 10^{-4}$ on Real_sim and RCV1 datasets.

workers for different values of the probability π used to form the mask on the madelon dataset. For each value of π; we run the algorithm 10 times. Blue dots correspond to successful runs where the algorithm converged to the minimum of the objective (P_{ℓ_1}) up to the precision 10^{-15}. Red numbers at the bottom of the figure mention the number of times when the algorithm diverged and expected amount of exchanges are shown by orange stars. In addition, we plot the line (in cyan blue) for the number of exchanges of the `DAve-PG` algorithm [MIMA18]. As it can be seen, in mostly all the cases the `I-SPY` algorithm converges to the minimum of (P_{ℓ_1}) with much less exchanges between the master and the workers than in the `DAve-PG` algorithm. For lower values of π, the number of times where the algorithm converges is low and for larger values of π; the expected number of exchanges tends to the one of the `DAve-PG` algorithm. This figure suggests that for this dataset the best compromise between the number of convergence and number

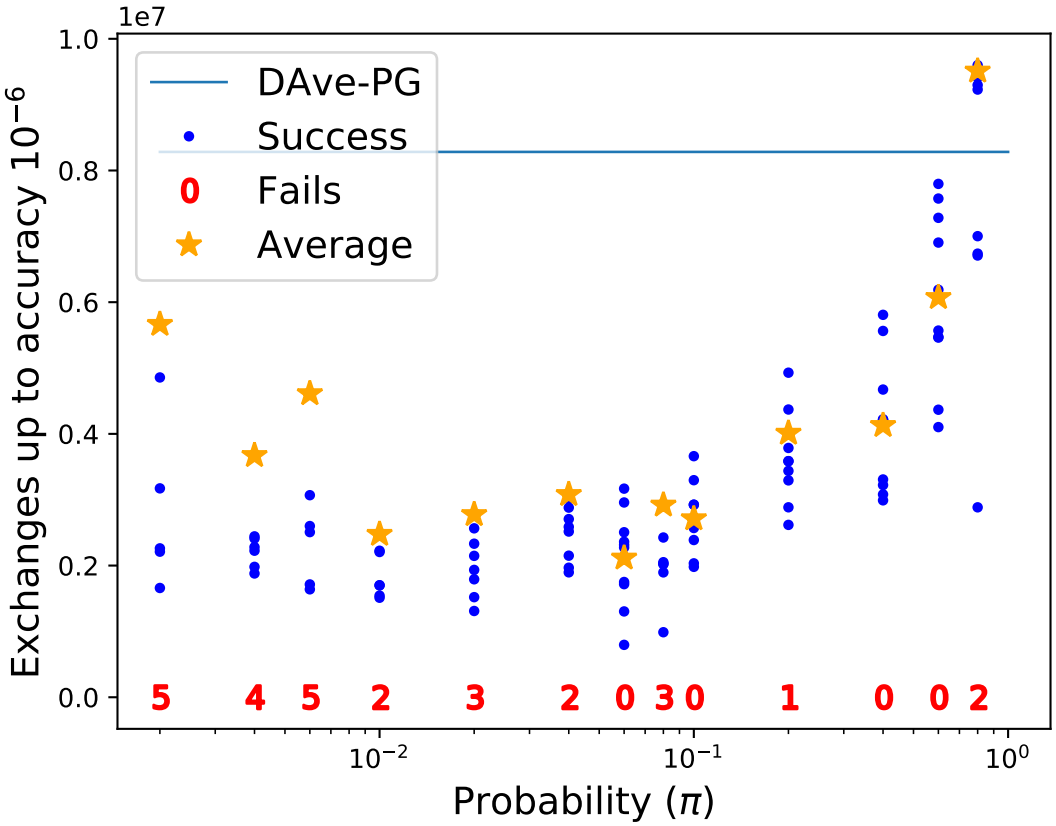
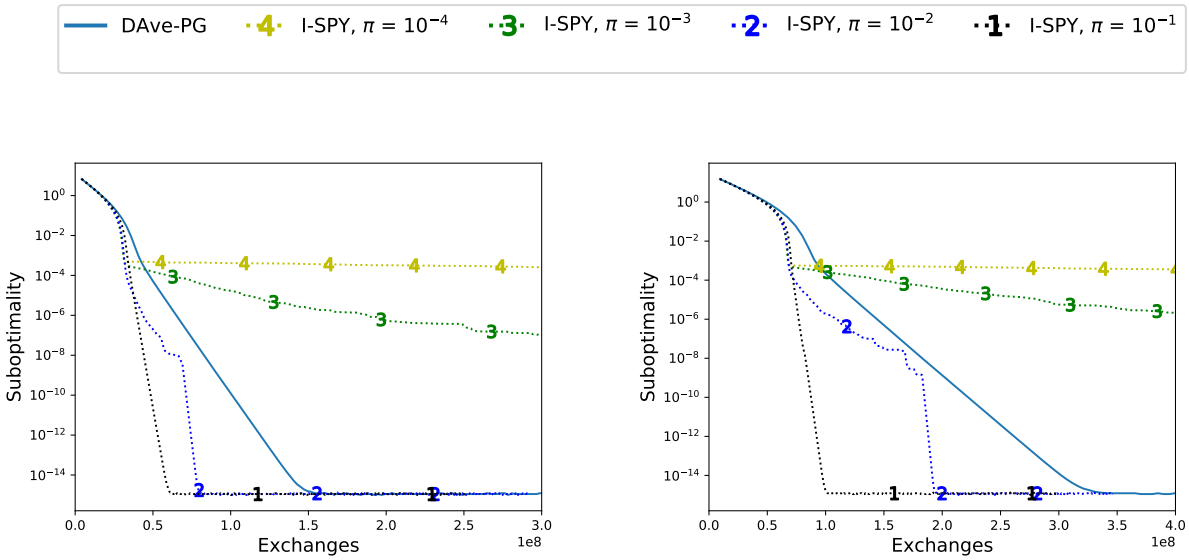


Figure 3-5. madelon dataset, $M = 10$ workers, $\lambda_1 = 0.02$.



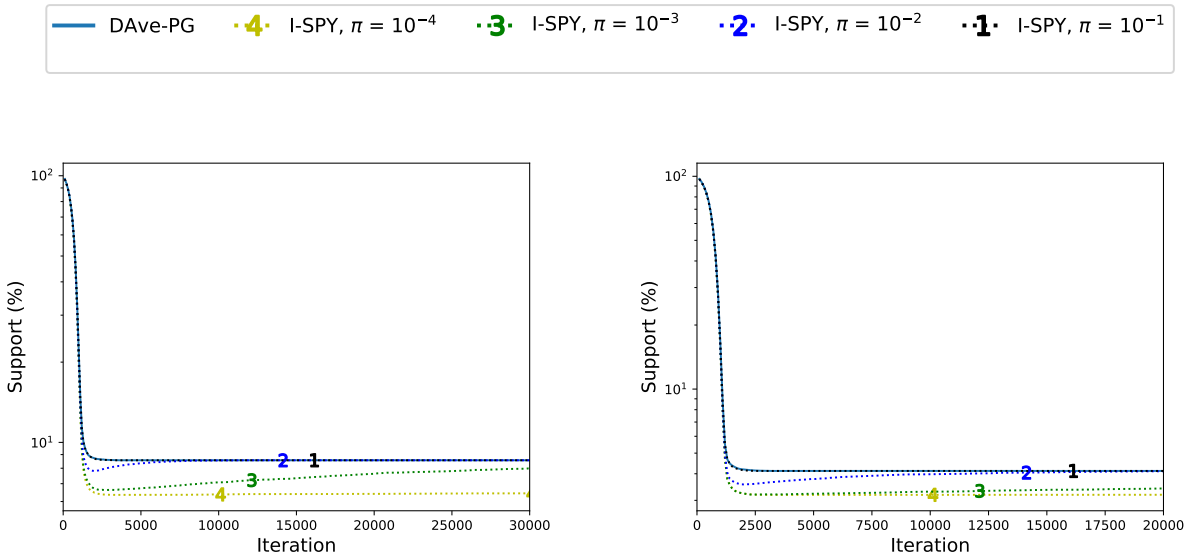
(a) real-sim (b) rcv1_train

Figure 3-6. Objective loss (P_{ℓ_1}) suboptimality versus number of exchanges, for $M = 20$ workers and $\lambda_1 = 10^{-4}$ on real-sim and rcv1_train datasets.

of exchanges is reached for the values of $\pi \in [0.01, 0.6]$. In Figure 3-6, we present the number of exchanges with respect to suboptimality for the `DAve-PG` algorithm [MIMA18] and `I-SPY` with different values of the probability π to form the mask on real-sim and rcv1_train datasets. As it can be observed; for larger values of π, `I-SPY` converges faster with much less exchanges. These plots with the ones of Figure 3-4, suggest that if the mask is formed with a large enough probability π, the proposed algorithm converges faster with fewer exchanges and epochs to the optimum of the global objective function than without sparsification.

Evolution of sparsity

Let us finally discuss the importance of sparsity, as the number of no-zero entries, of the final solution. Figure 3-7, shows the evolution of the percentage of no-zero entries of the parameter with respect to epochs on real-sim and rcv1_train datasets for $M = 20$ workers and $\lambda_1 = 10^{-5}$. The sparsity of the solution increases over epochs for both `DAve-PG` and `I-SPY` algorithms. This is mainly due to the use of the ℓ_1 in both algorithms. This sparsification is accentuated for `I-SPY` by the use of the mask. From previous plots, we observed that for higher values of the probability π, the proposed algorithm converges faster to the minimum of the composite objective. From Figure 3-7, it comes out that in this case, the `I-SPY` algorithm is able to identify the same set of informative non-zero



(a) real-sim (b) rcv1_train

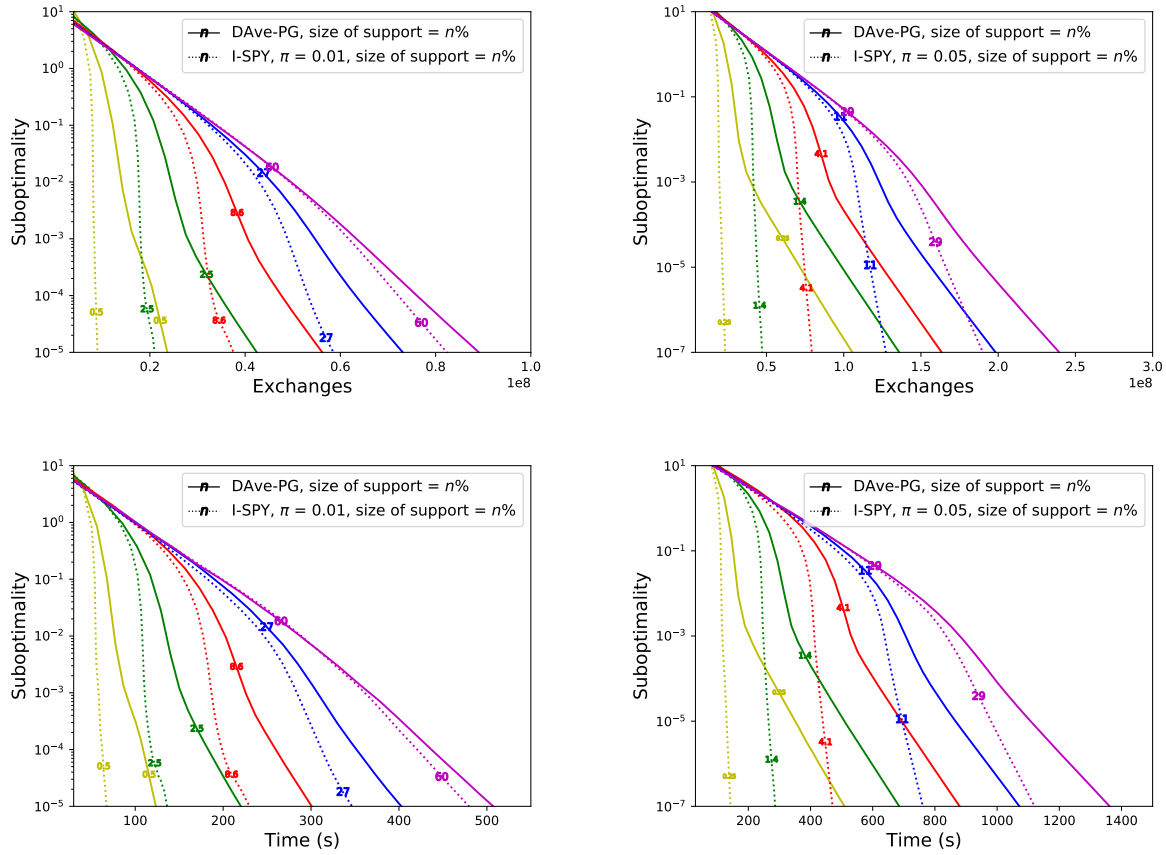
Figure 3-7. Evolution of sparsity versus iterations on real-sim and rcv1_train collections with, $M = 20$ workers and $\lambda_1 = 10^{-4}$.

entries, than `DAve-PG`, at convergence for higher values of the probability π.

In Figure 3-8, we present the evolution of the functional suboptimality for two different datasets: rcv1_train and real-sim. For both datasets we see that the gain is bigger if the final sparsity is smaller. It corresponds to the theoretical rate of Theorem 3.9.

3.4 Conclusion

In this chapter, we proposed an asynchronous distributed learning algorithm with sparsification. The sparsification is induced through a mask that selects a subpart of the model parameters constituted with all non-zero entries and some others chosen randomly with a fixed probability π. We analyzed the convergence property of the algorithm by showing that when π is moderately high, the algorithm is ensured to converge for strongly convex composite objectives. In the case of small values of π, we have empirically shown on three benchmarks that when the algorithm converges, it reaches faster the minimum with much fewer communications between the master and the worker machines than if the mask is not used. This algorithm has no guarantee, and we propose its modification with theoretical proofs of convergence in the next chapter.



(a) real-sim (b) rcv1_train

Figure 3-8. Dependence of gain on the sparsity of the final solution

Chapter 4

A provably convergent adaptively sparsified methods through reconditioning

Chapter Contents

Introduction

In Chapter 3, we presented the general framework SPY as a version of DAve-PG [MIMA18] with sparsified communication. However, this random sparsification technique provably works only for i.i.d. sparsifications with either almost-uniform distributions, or well-conditioned problems. This makes aggressive sparsification or adaptation to the sparsity structure of the model impossible with such an algorithm.

The proximal point algorithm is a standard regularization approach in optimization. It was presented in [BKL66, Chap. 5] to recondition a convex quadratic objective, for which computing the proximal operator (1.15) is easy (it is the unique solution of a linear system, well-conditioned by construction). The general proximal algorithm was then popularized by the seminal works [Mar70, Roc76]. The study of these algorithms, and especially their inexact variants, has attracted a lot of attention; see e.g. [Gül92, SS00, FML12, LMH17, LMH19].

Practical implementations of such methods require an inner algorithm to compute the proximal point and a rule to stop this algorithm. Several papers consider the key question of inner stopping criteria for inexact proximal methods in various contexts; see e.g. [FML12] in smooth optimization, [LS97b] in nonsmooth optimization, and [SS01] in operator theory.

Proximal reconditioning scheme wrapping up the previously mentioned algorithm as an inner minimization method allows us to perform much more aggressive sparsifications. Furthermore, we show that when using a sparsity-inducing regularizer, our reconditioned algorithm generates iterates that identify the optimal sparsity pattern of the model in finite time. This progressively uncovered pattern can be used to adaptively update the sparsification distribution of the inner method. All in all, this method only features sparse communications: the downward communications (master-to-worker) consists in sending the (eventually) sparse model, and the the upwards communications (worker-to-master) are adaptively and aggressively sparsified.

Finally, we show theoretically and numerically that our method has better performance than its non-sparsified version, in terms of suboptimality with respect to the quantity of information exchanged.

Outline This chapter is organized as follows. First, in Section 4.1, we present a proximal reconditioning framework that allows wrapping up any optimization algorithm. We specify it to the case of I-SPY (see Algorithm 13) and investigate the convergence of such algorithm with three different stopping criterion. In Section 4.2, we present an identification result for this method under the standard non-degeneracy assumption. It keeps the automatic-dimension reduction property of the inner algorithm while it converges; furthermore, we present an improved convergence rate based on this property. Finally, in Section 4.3, we

present the numerical experiments to prove that in practice algorithm performs better than `DAve-PG`.

This chapter corresponds to [GIMA18].

4.1 Proximal reconditioning for adaptive sparsification

The learning problem (P) should be well-conditioned to safely apply the random sparsification technique of `I-SPY` with reasonable exploration probability π. The idea is then not to apply `I-SPY` directly to (P) but rather to a modified problem for which we control the condition number and thus the sparsification potential.

We thus propose to recondition the learning problem (P) using the standard proximal algorithm (see e.g. [Roc76]), as described in Section 4.1.1. We present in Section 4.1.2 our algorithmic choices and the resulting algorithm, called Reconditioned–`I-SPY`.

4.1.1 Proximal reconditioning

This type of methods consist in iteratively regularizing the problem with the squared distance to some center point. We call *outer iteration* the process of (approximatively) solving such a reconditioned problem. At outer loop[1] ℓ, we define the worker i's regularized function as

$$h_{i,\ell} = f_i + \frac{\rho}{2}\| \cdot - x_\ell\|_2^2$$

where ρ is the regularization factor and x_ℓ the center point at outer loop ℓ. The *reconditioned problem for loop* ℓ then writes

$$\min_{x\in\mathbb{R}^n} \quad H_\ell(x) := \sum_{i=1}^{M} \alpha_i \underbrace{\left(f_i(x) + \frac{\rho}{2}\|x - x_\ell\|_2^2\right)}_{h_{i,\ell}(x)} + r(x). \tag{R_ℓ}$$

For μ-strongly convex L-smooth (f_i), the regularized functions $h_{i,\ell}$ are $(\mu+\rho)$-strongly convex and $(L+\rho)$-smooth. Hence, the condition number of the smooth part of (R_ℓ) writes

$$\kappa_{(\mathsf{R}_\ell)} = \frac{\mu+\rho}{L+\rho} \quad \left(\geq \kappa_{(\mathsf{P})} = \frac{\mu}{L}\right).$$

[1] The quantities related to outer loop ℓ are denoted with a subscript ℓ.

The optimal solution of (R_ℓ) is exactly the proximal point (1.15) of F/ρ at x_ℓ (see Section 1.1.3)

$$\mathbf{prox}_{F/\rho}(x_\ell) = \operatorname*{argmin}_{x\in\mathbb{R}^n} \underbrace{\sum_{i=1}^M \alpha_i f_i(x) + r(x)}_{=F(x)} + \frac{\rho}{2}\|x - x_\ell\|_2^2.$$

Thus, for solving (P), each outer iteration consists in an (inexact) proximal step:

$$x_{\ell+1} \approx \mathbf{prox}_{F/\rho}(x_\ell) \tag{4.1}$$

4.1.2 Reconditioned–I-SPY

We present our main algorithm, which consists in applying the inexact proximal scheme (4.1) with I-SPY as inner algorithm to solve (P).

At the outer iteration ℓ, we run I-SPY for solving (R_ℓ) with i.i.d. non-uniform sparsification probabilities given, for a fixed $0 < c \leq n$, by

$$p_{j,\ell} = \begin{cases} \pi_\ell := \min\left(\dfrac{c}{|\mathrm{null}(x_\ell)|}; 1\right) & \text{if } (x_\ell)_{[j]} = 0 \\ 1 & \text{if } (x_\ell)_{[j]} \neq 0 \end{cases} \qquad \text{for all } j \in \{1,\dots,n\}. \tag{4.2}$$

The sparsification level over outer iterations is then bounded from below by

$$\pi := \frac{c}{n} \;\leq\; \inf_\ell \pi_\ell.$$

We now choose the reconditioning parameter ρ from π so that SPY converges linearly to the solution of the reconditioned problem (R_ℓ). We know, from Section 2.2.3, that this is the case as soon as

$$\kappa_{(\mathsf{R}_\ell)} = \frac{\mu+\rho}{L+\rho} > \kappa_{\min} \iff \rho > \frac{\kappa_{\min} L - \mu}{1-\kappa_{\min}} \quad \text{with } \kappa_{\min} = \frac{1-\sqrt{\pi}}{1+\sqrt{\pi}} \text{ as in (3.7)}.$$

To properly handle the strict inequality above, we propose to choose a conditioning which guarantees a $(1-\alpha)$ rate for SPY on the reconditioned problems uniformly over ℓ. Mathematically, for $0 < \alpha < \pi$ (for instance $\alpha = \pi/2$), we choose

$$\rho = \frac{\kappa_{(\mathsf{R}_\ell)} L - \mu}{1 - \kappa_{(\mathsf{R}_\ell)}} \quad \text{with} \quad \kappa_{(\mathsf{R}_\ell)} = \frac{1-\sqrt{\pi-\alpha}}{1+\sqrt{\pi-\alpha}}. \tag{4.3}$$

Then, the contraction factor of SPY (3.3) for the reconditioned problem (R_ℓ) becomes

$$\left(\left(\frac{1-\kappa_{(\mathsf{R}_\ell)}}{1+\kappa_{(\mathsf{R}_\ell)}}\right)^2 + 1 - \pi_\ell\right) = (\pi - \alpha + 1 - \pi_\ell) = \underbrace{(1-\alpha-(\pi_\ell-\pi))}_{=:(1-\alpha_\ell)} \leq 1-\alpha < 1. \quad (4.4)$$

This means that I-SPY is linearly convergent on the reconditioned problem (R_ℓ). Thus, it can safely be used as an inner method in the inexact proximal algorithm (4.1) to solve the original problem (P).

The remaining part is to the choice of a stopping criterion for the inner loop. We propose to use three different criteria: epoch budget, absolute accuracy, and relative accuracy (called $\mathsf{C}_1, \mathsf{C}_2$, and C_3, respectively). Stopping criteria based on accuracy are usually much more stringent to enforce (see e.g. [LMH17, Sec. 2.3] and references therein), however they may bring significant performance improvement when the instantaneous rate is much better than the theoretical one.

The resulting algorithm, called Reconditioned–I-SPY, is presented as Algorithm 16. Under any of the three stopping criteria, we recover the same convergence result, formalized in the next theorem.

Theorem 4.1. *Let the functions (f_i) be μ-strongly convex ($\mu \geq 0$) and L-smooth. Let r be convex lsc. If $\mu = 0$ and C_3 is used, we furthermore require that F has a unique minimizer $x^\star$ and that $\liminf_{x\to x^\star}(F(x)-F(x^\star))/\|x-x^\star\|^2 > 0$.*

Then, the sequence generated by Reconditioned–I-SPY on $((\alpha_i),(f_i),r)$ with stopping criterion $\mathsf{C}_1, \mathsf{C}_2$, or C_3 converges almost surely to a minimizer of F. Furthermore, if $\mu > 0$, then we have[2]

$$\mathbb{E}\left[\|x_{\ell+1}-x^\star\|_2^2\right] = \tilde{\mathcal{O}}\left(\left(1-\frac{\mu}{\mu+\rho/2}\right)^\ell\right) \quad \text{for criterion } \mathsf{C}_1;$$

$$\|x_{\ell+1}-x^\star\|_2^2 = \tilde{\mathcal{O}}\left(\left(1-\frac{\mu}{\mu+\rho/2}\right)^\ell\right) \quad \text{for criteria } \mathsf{C}_2, \mathsf{C}_3.$$

Even though the final result is similar for the three cases, the proof techniques are rather different. We thus present them separated.

[2] We use the standard notation: $a_\ell = \tilde{\mathcal{O}}((1-r)^\ell)$ denotes that there exists C, p such that $a_\ell \leq C\ell^p(1-r)^\ell$.

Algorithm 16 Reconditioned–`I-SPY` on $((\alpha_i),(f_i),r)$

Initialize x_1, $n \geq c > 0$, and $\delta \in (0,1)$.

$$\text{Set } \rho = \frac{\kappa L - \mu}{1-\kappa} \text{ and } \gamma \in \left(0, \frac{2}{\mu + L + 2\rho}\right] \text{ with } \kappa = \frac{1-\sqrt{\pi-\alpha}}{1+\sqrt{\pi-\alpha}}; \pi = \frac{c}{n} \text{ and } \alpha = \frac{c}{2n}. \tag{4.5}$$

while *the desired accuracy is not achieved* **do**

Observe the support of x_ℓ, compute p_ℓ as

$$p_{j,\ell} = \begin{cases} \pi_\ell := \min\left(\frac{c}{|\text{null}(x_\ell)|}; 1\right) & \text{if } [x_\ell]_j = 0 \\ 1 & \text{if } [x_\ell]_j \neq 0 \end{cases} \qquad \text{for all } j \in \{1,\dots,n\}. \tag{4.6}$$

Compute an approximate solution of the reconditioned problem

$$x_{\ell+1} \approx \mathbf{prox}_{F/\rho}(x_\ell) = \operatorname*{argmin}_{x\in\mathbb{R}^n} \left\{ \sum_{i=1}^{M} \alpha_i \underbrace{\left(f_i(x) + \frac{\rho}{2}\|x - x_\ell\|_2^2\right)}_{h_{i,\ell}(x)} + r(x) \right\} \tag{4.7}$$

with `I-SPY` on $\Big((\alpha_i),(h_{i,\ell}),r \ ; \ p_\ell\Big)$ with x_ℓ as initial point and with the stopping criterion:

C_1 (epoch budget): Run `I-SPY` with the maximal stepsize for

$$\mathsf{M}_\ell = \left\lceil \frac{(1+\delta)\log(\ell)}{\log\left(\frac{1}{1-\alpha+\pi-\pi_\ell}\right)} + \frac{\log\left(\frac{2\mu+\rho}{(1-\delta)\rho}\right)}{\log\left(\frac{1}{1-\alpha+\pi-\pi_\ell}\right)} \right\rceil \text{ epochs.}$$

or C_2 (absolute accuracy): Run `I-SPY` until it finds $x_{\ell+1}$ such that

$$\|x_{\ell+1} - \mathbf{prox}_{F/\rho}(x_\ell)\|_2^2 \leq \frac{(1-\delta)\rho}{(2\mu+\rho)\ell^{1+\delta}}\|x_\ell - \mathbf{prox}_{F/\rho}(x_\ell)\|_2^2.$$

or C_3 (relative accuracy): Run `I-SPY` until it finds $x_{\ell+1}$ such that

$$\|x_{\ell+1} - \mathbf{prox}_{F/\rho}(x_\ell)\|_2^2 \leq \frac{\rho}{4(2\mu+\rho)\ell^{2+2\delta}}\|x_{\ell+1} - x_\ell\|_2^2.$$

end

4.1.3 Proof of Theorem 4.1

Two basic lemmas

First, let us present two simple lemmas that we have not found as such in the literature and that are required in proofs of the theorem. They use the fact that the unique minimum $x^\star$ of a strongly convex function F is a fixed point of $\mathbf{prox}_{F/\rho}$ for any $\rho > 0$; see [BC11, Prop. 12.28]).

Lemma 4.2. *Let $F : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be μ-strongly convex lsc and $\rho > 0$. Then, for any $x \in \mathbb{R}^n$,*

$$\|\mathbf{prox}_{F/\rho}(x) - \mathbf{prox}_{F/\rho}(x^\star)\|_2^2 \leq \frac{\rho}{2\mu+\rho}\|x - x^\star\|_2^2 - \frac{\rho}{2\mu+\rho}\|\mathbf{prox}_{F/\rho}(x) - x\|_2^2.$$

Proof. The proof simply consists in developing norms as follows:

$$\begin{aligned}
&\|\mathbf{prox}_{F/\rho}(x) - x\|_2^2 \\
&\quad = \|\mathbf{prox}_{F/\rho}(x) - \mathbf{prox}_{F/\rho}(x^\star) + x^\star - x\|_2^2 \\
&\quad = \|\mathbf{prox}_{F/\rho}(x) - \mathbf{prox}_{F/\rho}(x^\star)\|_2^2 + \|x - x^\star\|_2^2 - 2\langle \mathbf{prox}_{F/\rho}(x) - \mathbf{prox}_{F/\rho}(x^\star); x - x^\star\rangle \\
&\quad \leq \|\mathbf{prox}_{F/\rho}(x) - \mathbf{prox}_{F/\rho}(x^\star)\|_2^2 + \|x - x^\star\|_2^2 - 2(1+\mu/\rho)\|\mathbf{prox}_{F/\rho}(x) - \mathbf{prox}_{F/\rho}(x^\star)\|_2^2 \\
&\quad = \|x - x^\star\|_2^2 - (1+2\mu/\rho)\|\mathbf{prox}_{F/\rho}(x) - \mathbf{prox}_{F/\rho}(x^\star)\|_2^2;
\end{aligned}$$

and a reordering concludes the proof. In words, we formalize the fact that the resolvent of the μ/ρ strongly monotone operator $\partial F/\rho$ is $(1+\mu/\rho)$-cocoercive; see [BC11] for definitions. □

Lemma 4.3. *Let $F : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be μ-strongly convex lsc, $\rho > 0$. Then, for any $x, x' \in \mathbb{R}^n$ such that*

$$\mathbb{E}\left[\left\|x' - \mathbf{prox}_{F/\rho}(x)\right\|_2^2 | x\right] \leq \nu \left\|x - \mathbf{prox}_{F/\rho}(x)\right\|_2^2$$

for some $\nu > 0$, we have for any $\varepsilon \in (0,1)$

$$\begin{aligned}
\mathbb{E}\left[\|x' - x^\star\|_2^2 | x\right] \leq (1+\varepsilon)\frac{\rho}{2\mu+\rho}\|x - x^\star\|_2^2 \\
- \left[(1+\varepsilon)\frac{\rho}{2\mu+\rho} - \left(1+\frac{1}{\varepsilon}\right)\nu\right]\|x - \mathbf{prox}_{F/\rho}(x)\|_2^2.
\end{aligned} \tag{4.8}$$

Proof. Using Young's inequality, we get that for any $\varepsilon \in (0,1)$,

$$\begin{aligned}
&\mathbb{E}\left[\|x' - x^\star\|_2^2 \,|x\right] \\
&\leq \left(1+\frac{1}{\varepsilon}\right)\mathbb{E}\left[\left\|x' - \mathbf{prox}_{F/\rho}(x)\right\|_2^2 |x\right] + (1+\varepsilon)\left\|\mathbf{prox}_{F/\rho}(x) - \mathbf{prox}_{F/\rho}(x^\star)\right\|_2^2 \\
&\leq \left(1+\frac{1}{\varepsilon}\right)\nu\|x - \mathbf{prox}_{F/\rho}(x)\|_2^2 + (1+\varepsilon)\frac{\rho}{2\mu+\rho}\|x-x^\star\|_2^2 \\
&\quad - (1+\varepsilon)\frac{\rho}{2\mu+\rho}\left\|x - \mathbf{prox}_{F/\rho}(x)\right\|_2^2 \\
&= (1+\varepsilon)\frac{\rho}{2\mu+\rho}\|x-x^\star\|_2^2 - \left[(1+\varepsilon)\frac{\rho}{2\mu+\rho} - \left(1+\frac{1}{\varepsilon}\right)\nu\right]\|x - \mathbf{prox}_{F/\rho}(x)\|_2^2.
\end{aligned}$$

where we used Lemma 4.2. □

Proof of Theorem 4.1 for criterion C_1 *(epoch budget)*

We start by noticing that, at outer loop ℓ, SPY solves the reconditioned problem (4.7) over which it has a contraction factor of $(1-\alpha_\ell)$ with $1-\alpha_\ell := 1-\alpha+\pi-\pi_\ell$; see (4.4) and (3.3).This means that SPY initialized with x_ℓ verifies after m epochs with the maximal stepsize

$$\mathbb{E}\left\|x^{k_m} - x_\ell^\star\right\|_2^2 \leq (1-\alpha_\ell)^m \max_i \left\|x_i^0 - x_{i,\ell}^\star\right\|_2^2 \leq (1-\alpha_\ell)^m \|x_\ell - x_\ell^\star\|_2^2$$

where $x_\ell^\star$ is the unique solution of (R_ℓ), and $x_{i,\ell}^\star = x_\ell^\star - \gamma\nabla h_{i,\ell}(x_\ell^\star)$ are its local shifts.

We now apply Lemma 4.3 with the following input: $x' = x_{\ell+1}$; $x = x_\ell$; $F = F$ (which is μ strongly convex); $x^\star = x^\star$ (minimizer of F); and $\nu = (1-\alpha_\ell)^{\mathsf{M}_\ell}$ (noting that this term only depends on x_ℓ). We get for $\varepsilon = \frac{1}{\ell^{1+\delta}} \in (0,1)$

$$\begin{aligned}
\mathbb{E}\left[\|x_{\ell+1} - x^\star\|_2^2 |x_\ell\right] &\leq \left(1+\frac{1}{\ell^{1+\delta}}\right)\frac{\rho}{2\mu+\rho}\|x_\ell - x^\star\|_2^2 \\
&\quad - \underbrace{\left[\left(1+\frac{1}{\ell^{1+\delta}}\right)\frac{\rho}{2\mu+\rho} - \left(1+\ell^{1+\delta}\right)(1-\alpha_\ell)^{\mathsf{M}_\ell}\right]}_{:=b_\ell}\|x_\ell - \mathbf{prox}_{F/\rho}(x_\ell)\|_2^2.
\end{aligned}$$

Choosing M_ℓ as per C_1 guarantees that

$$b_\ell \geq \delta\left(1+\frac{1}{\ell^{1+\delta}}\right)\frac{\rho}{2\mu+\rho} \geq \frac{\delta\rho}{2\mu+\rho}.$$

Thus, for any $\mu \geq 0$, we have

$$\mathbb{E}\left[\|x_{\ell+1} - x^\star\|_2^2 \,|x_\ell\right] \leq \left(1 + \frac{1}{\ell^{1+\delta}}\right) \frac{\rho}{2\mu+\rho} \|x_\ell - x^\star\|_2^2 - \frac{\delta\rho}{2\mu+\rho} \|x_\ell - \mathbf{prox}_{F/\rho}(x_\ell)\|_2^2. \tag{4.9}$$

Convergence. By using $\mu \geq 0$ in (4.9), we get

$$\mathbb{E}\left[\|x_{\ell+1} - x^\star\|_2^2 \,|x_\ell\right] \leq \left(1 + \frac{1}{\ell^{1+\delta}}\right) \|x_\ell - x^\star\|_2^2 - \delta \|x_\ell - \mathbf{prox}_{F/\rho}(x_\ell)\|_2^2. \tag{4.10}$$

By Robbins-Siegmund theorem [RS71, Th. 1] (and [IBCH13, BHI15, CP15] for applications to optimization), we have that i) $(\|x_\ell - x^\star\|_2^2)$ converges almost surely to a random variable with finite support; and ii) $\sum_{\ell=1}^{\infty} \|x_\ell - \mathbf{prox}_{F/\rho}(x_\ell)\|_2^2 < \infty$. This means that we can extract a subsequence (x_{ℓ^n}) that converges almost surely to some $\overline{x}^\star$ which is necessarily a minimizer from ii). Using (4.10) again with $x^\star = \overline{x}^\star$, we see that (x_ℓ) converges to $\overline{x}^\star$ almost surely.

Rate. Now, if $\mu > 0$, we get by dropping the last term in (4.9) and successively taking expectations that

$$\mathbb{E}\left[\|x_{\ell+1} - x^\star\|_2^2\right] \leq (\ell+1)^{1+\delta} \left(\frac{\rho}{2\mu+\rho}\right)^\ell \|x_1 - x^\star\|_2^2 = \tilde{\mathcal{O}}\left(\left(1 - \frac{\mu}{\mu+\rho/2}\right)^\ell\right). \tag{4.11}$$

Proof of Theorem 4.1 for criterion C_2 *(absolute accuracy)*

We apply Lemma 4.3 with the following input: $x' = x_{\ell+1}$; $x = x_\ell$; $F = F$ (which is μ strongly convex); $x^\star = x^\star$ (minimizer of F); and $\nu = (1-\delta)\rho/((2\mu+\rho)\ell^{1+\delta})$ (noting that the condition on $x_{\ell+1}$ is almost sure). We get for $\varepsilon = \frac{1}{\ell^{1+\delta}} \in (0,1)$

$$\begin{aligned}
\|x_{\ell+1} - x^\star\|_2^2 &\leq \left(1 + \frac{1}{\ell^{1+\delta}}\right) \frac{\rho}{2\mu+\rho} \|x_\ell - x^\star\|_2^2 \\
&\quad - \underbrace{\left[\left(1 + \frac{1}{\ell^{1+\delta}}\right) \frac{\rho}{2\mu+\rho} - \left(1+\ell^{1+\delta}\right) \frac{1}{\ell^{1+\delta}} \frac{(1-\delta)\rho}{2\mu+\rho}\right]}_{\leq 0} \|x_\ell - \mathbf{prox}_{F/\rho}(x_\ell)\|_2^2 \\
&\leq (\ell+1)^{1+\delta} \left(\frac{\rho}{2\mu+\rho}\right)^\ell \|x_1 - x^\star\|_2^2 .
\end{aligned}$$

This directly gives the rate of convergence when $\mu > 0$. When $\mu = 0$, the inequality can be simplified to

$$\begin{aligned}\|x_{\ell+1} - x^\star\|_2^2 &\leq \left(1 + \frac{1}{\ell^{1+\delta}}\right) \|x_\ell - x^\star\|_2^2 \\ &\quad - \left[\left(1 + \frac{1}{\ell^{1+\delta}}\right) - \left(1 + \ell^{1+\delta}\right) \frac{1}{\ell^{1+\delta}}(1-\delta)\right] \|x_\ell - \mathbf{prox}_{F/\rho}(x_\ell)\|_2^2 \\ &\leq \left(1 + \frac{1}{\ell^{1+\delta}}\right) \|x_\ell - x^\star\|_2^2 - \delta \|x_\ell - \mathbf{prox}_{F/\rho}(x_\ell)\|_2^2.\end{aligned}$$

In this case, the same arguments as for criterion C_1 enable to get almost sure convergence.

Proof of Theorem 4.1 for criterion C_3 (relative accuracy)

Denoting $\beta := \sqrt{\rho/(2\mu + \rho)} \in (0, 1]$, the stopping criterion C_3 writes

$$\|x_{\ell+1} - \mathbf{prox}_{F/\rho}(x_\ell)\|_2 \leq \varepsilon_\ell \|x_{\ell+1} - x_\ell\|_2 \qquad \text{with } \varepsilon_\ell = \frac{\beta}{2\ell^{1+\delta}}. \tag{4.12}$$

Convergence. The condition (4.12) matches condition (B) of [Roc76, Th. 2]. We also have clearly $\sum_\ell \varepsilon_\ell < +\infty$ and the regularity assumption of the operator is verified, by our extra assumption and [Roc76, Prop. 7]. Thus [Roc76, Th. 2] directly gives us that (x_ℓ) converges to a minimizer of F, that we denote by $x^\star$.

Rate. When F is μ-strongly convex, we can furthermore develop:

$$\begin{aligned}\|x_{\ell+1} - x^\star\|_2 &\leq \left\|x_{\ell+1} - \mathbf{prox}_{F/\rho}(x_\ell)\right\|_2 + \left\|\mathbf{prox}_{F/\rho}(x_\ell) - x^\star\right\|_2 \\ &\leq \varepsilon_\ell \left\|x_{\ell+1} - x_\ell\right\|_2 + \beta \left\|x_\ell - x^\star\right\|_2 \\ &\leq \varepsilon_\ell \left\|x_{\ell+1} - x^\star\right\|_2 + \varepsilon_\ell \left\|x_\ell - x^\star\right\|_2 + \beta \left\|x_\ell - x^\star\right\|_2\end{aligned}$$

where the first inequality used both condition C_3 and Lemma 4.2. This implies that

$$\|x_{\ell+1} - x^\star\|_2^2 \leq \left(\frac{\varepsilon_\ell + \beta}{1 - \varepsilon_\ell}\right)^2 \|x_\ell - x^\star\|_2^2 .$$

Finally, denoting $d_\ell := \ell^{1+\delta}/(\ell^{1+\delta} - 1) > 1$, we have[3]

$$\varepsilon_\ell \leq \beta \frac{(d_\ell - 1)}{(1 + \beta d_\ell)} \quad \Longrightarrow \quad \frac{\varepsilon_\ell + \beta}{1 - \varepsilon_\ell} \leq d_\ell \beta \leq \ell^{1+\delta}/((\ell-1)^{1+\delta})\beta.$$

[3]Note that $\varepsilon_\ell^2 = \frac{\rho}{4(2\mu+\rho)\ell^{2+2\delta}} \leq \left(\frac{\beta}{2}\right)^2 \frac{(d_\ell-1)^2}{d_\ell^2} \leq \beta^2 \frac{(d_\ell-1)^2}{(1+\beta d_\ell)^2}$.

This yields

$$\|x_{\ell+1} - x^\star\|_2^2 \leq \left(\frac{\ell}{\ell-1}\right)^{2+2\delta} \frac{\rho}{2\mu+\rho} \|x_\ell - x^\star\|_2^2$$

which gives the result.

General comments on the result

This result thus establishes that Reconditioned–`I-SPY` converges linearly to a solution of (P). This means that Reconditioned–`I-SPY` has qualitatively the same behavior as `DAve-PG`, with the additional feature of having sparse local updates and therefore sparse upward communications. In other words, our algorithm is similar to the baseline in terms of iterations, but it is expected to be faster in terms of communications (more precisely in terms of quantity of information exchanged between master and workers) which would result in a wallclock gain in practice, as shown in Section 4.3. Before this, we further investigate in the next section the theoretical gain of our sparsification technique in the case of sparse optimal solutions.

Remark 4.4 (Acceleration (with respect to iterations))**.** *In this chapter, we are interested in sparsifying communications and we primarily consider the reconditioning aspect of proximal methods, leaving aside other aspects including acceleration. As proposed in [Gül92], the iterations of the inexact proximal algorithm can indeed be accelerated using Nesterov's method [Nes83]. The recent works [LMH17, LMH19] also propose accelerated and quasi-Newton variants of the inexact proximal point algorithm as a meta-algorithm to improve the convergence of optimization methods (driven by machine learning applications [LMH15]). In this chapter, we investigate the complexity in terms of communications rather than iterations, so we do not insist much on these accelerated variants. However the developments of this section could be extended with accelerated proximal algorithm, following the meta-algorithm of [LMH17].*

4.2 Identification for two-way sparse communications

In the previous section, we present an adaptive sparsification of upward communications (worker-to-master) and show that the resulting algorithm converges after proximal reconditioning. By construction, the downward communications (master-to-workers) depends on the structure of x^k (the master point of the inner method), which is the output of a proximal operator on r. In the case of ℓ_1-regularization or other sparsity-promoting regularization [BJM+12], we show in Section 4.2.1 that the x^k eventually become sparse after some iterations. This automatically makes our algorithm a two-way sparse algorithm.

Finally, in Section 4.2.2, we take a closer look to the complexity of our algorithm with respect to the communication cost.

For this study, we make an additional assumption that our problem has a strongly sparse solution. This assumption is divided into two parts: i) the regularizer r should induce a *stable* support at the optimum (through its proximity operator); and ii) this optimal support $\mathrm{supp}(x^\star)$ should be small with respect to the ambient dimension.

Assumption 4.5 (Strongly sparse optimal solution)**.** *Problem* (P) *is μ-strongly convex ($\mu > 0$) and its solution $x^\star$ verifies*

i) $\exists\ \varepsilon > 0$ *such that*

$$\mathrm{supp}(x^\star) = \mathrm{supp}\left(\mathbf{prox}_r\left(x^\star - \sum_{i=1}^{M} \alpha_i \nabla f_i(x^\star) + \mathbf{e}\right)\right) \quad \forall \mathbf{e} \in \mathcal{B}(0, \varepsilon);$$

ii) the size $s^\star = |\mathrm{supp}(x^\star)|$ of the optimal support is small compared to n: $s^\star \ll n$.

While part ii) is rather explicit, part i) is quite abstract; this condition matches the nondegeneracy condition (ND) for sparse solutions commonly admitted for exact recovery in machine learning; see e.g. [NSH19, SJNS19b]. The interest of the general assumption i) is that it accounts for a variety of sparsity-inducing regularizations, including weighted ℓ_1-norms, "group" ℓ_1/ℓ_q-norms; see [BJM+12, Sec. 3.3].

4.2.1 Identification and consequences

The iterates of proximal algorithms usually *identify* the optimal structure; see Section 1.4. Unfortunately, randomness may break this identification property. For instance, it is well-known that for the proximal stochastic gradient descent, the sparse structure may not be identified with probability one; see e.g. [LW12] and a counter-example in [PLS18]. We first establish that our algorithm does identify the optimal support under the non-degeneracy assumption 4.5.

Theorem 4.6 (Identification)**.** *Let the functions (f_i) be μ-strongly convex and L-smooth. Let r be convex lsc. Under Assumption 4.5, the* outer and inner *iterates of Reconditioned–* ***I-SPY*** *identify the optimal structure in finite time with probability one: there exists a finite time $\Lambda < \infty$ such that*

$$\mathrm{supp}(x_\ell^k) = \mathrm{supp}(x_\ell) = \mathrm{supp}(x^\star) \qquad \text{for any } k \text{ and all } \ell \geq \Lambda$$

where x_ℓ^k denotes the k-th iterate produced by ***SPY*** *during the ℓ-th outer loop.*

Proof. We proved in the previous section that Reconditioned–I-SPY converges almost surely, but it is not enough to guarantee identification in general[4]. Here it is the fact that the inner algorithm I-SPY features a proximity operator with a non-vanishing stepsize that yields the following identification property.

More precisely, Reconditioned–I-SPY is of the form

$$x_{\ell+1} \approx \mathbf{prox}_{F/\rho}(x_\ell)$$

and verify for all ℓ, k

$$\mathbb{E}\|x_\ell - x^\star\| \leq C(1-\rho)^\ell \qquad \text{and} \qquad \mathbb{E}\|\overline{x}_\ell^k - \overline{x}_\ell^\star\| \leq C'\|x_\ell - x_\ell^\star\|$$

for some $C, C' > 0$ and $\rho \in (0,1)$, where

$$\overline{x}_\ell^\star := x_\ell^\star - \gamma \sum_{i=1}^M \alpha_i \nabla h_{i,\ell}(x_\ell^\star).$$

As in the proof in Section 3.1.3, we also consider $\overline{x}^\star$ given by (1.30). Then we have

$$\begin{aligned}
\|\overline{x}_\ell^\star - \overline{x}^\star\| &= \|x_\ell^\star - \gamma \sum_{i=1}^M \alpha_i \nabla h_{i,\ell}(x_\ell^\star) - x^\star + \gamma \sum_{i=1}^M \alpha_i \nabla f_i(x^\star)\| \\
&= \|x_\ell^\star - \gamma \sum_{i=1}^M \alpha_i \nabla f_i(x_\ell^\star) - \gamma\rho(x_\ell^\star - x_\ell) - x^\star + \gamma \sum_{i=1}^M \alpha_i \nabla f_i(x^\star)\| \\
&\leq \|x_\ell^\star - x^\star\| + \gamma \sum_{i=1}^M \alpha_i \|\nabla f_i(x_\ell^\star) - \nabla f_i(x^\star)\| + \gamma\rho\|x_\ell^\star - x_\ell\| \\
&\leq \|x_\ell^\star - x^\star\| + \gamma \sum_{i=1}^M \alpha_i L\|x_\ell^\star - x^\star\| + \gamma\rho\|x_\ell^\star - x_\ell\| \\
&\leq D\|x_\ell^\star - x^\star\| + D'\|x_\ell - x_\ell^\star\|
\end{aligned}$$

[4]Take $n = 1$, $F(x) = |x|$, and $x_{\ell+1} = \mathbf{prox}_{|\cdot|}(x_\ell) + 1/\ell^2$. The minimum of F is 0 but we have $x_\ell = 1/(\ell-1)^2 > 0$ for all $\ell > 1$.

For any ℓ, k, we then have

$$\begin{aligned}
&\mathbb{E}\|\overline{x}_\ell^k - \overline{x}^\star\|^2 \\
&\quad\leq 2\mathbb{E}\left[\|\overline{x}_\ell^k - \overline{x}_\ell^\star\|^2 + \|\overline{x}_\ell^\star - \overline{x}^\star\|^2\right] \\
&\quad\leq 2C'\mathbb{E}\|x_\ell - x_\ell^\star\|^2 + 2D\mathbb{E}\|x_\ell^\star - x^\star\|^2 + 2D'\mathbb{E}\|x_\ell - x_\ell^\star\|^2 \\
&\quad\leq 4C'\mathbb{E}\|x_\ell - x^\star\|^2 + (4C' + 2D)\mathbb{E}\|x_\ell^\star - x^\star\|^2 + 2D'\mathbb{E}\|x_\ell - x_\ell^\star\|^2 \\
&\quad\leq (8C' + 2D + 2D')\mathbb{E}\|x_\ell - x^\star\|^2 \\
&\quad\leq (8C' + 2D + 2D')\mathbb{E}\|(1+\beta_\ell)(x_\ell - x^\star) - \beta_\ell(x_{\ell-1} - x^\star)\|^2 + 2D'\mathbb{E}\|x_\ell^\star - x_\ell\|^2 \\
&\quad\leq 2(8C' + 2D + 2D')(1+\beta_\ell)^2\mathbb{E}\|x_\ell - x^\star)\|^2 + 2(8C' + 2D + 2D')(1+\beta_\ell)^2\mathbb{E}\|x_{\ell-1} - x^\star)\|^2 \\
&\quad\leq 2(8C' + 2D + 2D')(1+\beta_\ell)^2 C(1-\rho)^\ell + 2(8C' + 2D + 2D')(1+\beta_\ell)^2 C(1-\rho)^{\ell-1} \\
&\quad\leq 16(8C' + 2D + 2D')C(1-\rho)^{\ell-1}.
\end{aligned}$$

Hence, by Markov's inequality and Borel-Cantelli's lemma, $\overline{x}_\ell^k \to \overline{x}^\star$ almost surely. As a direct result, we get

$$x_\ell^k = \mathbf{prox}_{\gamma r}(\overline{x}_\ell^k) \to x^\star = \mathbf{prox}_{\gamma r}(\overline{x}^\star).$$

Together with Assumption 4.5, this convergence implies identification of optimal support (see e.g. the recent survey [IM20, Cor. 1]): there is a $\Lambda < \infty$ such that for all $\ell \geq \Lambda$,

$$\mathrm{null}(x_\ell^k) = \mathrm{null}(x^\star) \qquad \text{and} \qquad \mathrm{supp}(x_\ell^k) = \mathrm{supp}(x^\star).$$

Finally, it suffices to notice that $x_{\ell+1} = x_\ell^k$ for some k to conclude the proof. □

This identification has two consequences on communications in our distributed setting. First, identification implies that the variables communicated by the master to the workers will eventually be sparse. Second, this sparsity is also leveraged in the sparsification strategies of Reconditioned–`I-SPY` where only the coordinates in (x_ℓ) are randomly zeroed. Thus, for sparsity inducing problems such as ℓ_1-regularized learning problems, our distributed algorithm has, structurally, *two-way sparse communications*.

Even better, once this identification occurs, the rate of the inner algorithm `SPY` dramatically improves to match the rate of its non-sparsified version `DAve-PG`.

Theorem 4.7 (Improved rate)**.** *Let the functions (f_i) be μ-strongly convex and L-smooth. Let r be convex lsc. Under Assumption 4.5, the* inner *iterates of Reconditioned–`I-SPY` benefit from an improved rate after identification. There is $\Lambda < \infty$ such that for all $\ell > \Lambda$ and $k \in [k_m, k_{m+1})$*

$$\left\|x_\ell^k - x_\ell^\star\right\|_2^2 \leq \Big(1 - \gamma(\mu+\rho)\Big)^{2m} \|x_\ell - x_\ell^\star\|_2^2$$

where x_ℓ^k denotes the k-th iterate produced by `I-SPY` *during the ℓ-th outer loop.*

Furthermore, using the maximal stepsize $\gamma = \frac{2}{\mu+L+2\rho}$, we obtain for all $k \in [k_m, k_{m+1})$

$$\left\|x_\ell^k - x_\ell^\star\right\|_2^2 \leq \left(\frac{1-\kappa_{(\mathsf{R}_\ell)}}{1+\kappa_{(\mathsf{R}_\ell)}}\right)^{2m} \|x_\ell - x_\ell^\star\|_2^2 .$$

Proof of Theorem 4.7. From Theorem 4.6 we know that identification takes place i.e. that we have $\mathrm{null}(x_\ell^k) = \mathrm{null}(x^\star) = n^\star$ for all k, ℓ ($\ell \geq \Lambda$). In this case,

$$[x_\ell^k]_{n^\star} = 0 \qquad \text{and} \qquad [x_\ell^k]_{\overline{n^\star}} = x_\ell^k$$

where we denote by $\overline{n^\star}$ the complementary of $n^\star$. Then, we have

$$\begin{aligned}[x_\ell^k]_{\overline{n^\star}} = \left[\mathbf{prox}_{\gamma r}([\overline{x}_\ell^k]_{\overline{n^\star}})\right]_{\overline{n^\star}} &= \left[\mathbf{prox}_{\gamma r}\left(\sum_{i=1}^M \alpha_i [x^{k-D_i^k}]_{\overline{n^\star}} - \gamma \sum_{i=1}^M \alpha_i [\nabla f_i(x^{k-D_i^k})]_{\overline{n^\star}}\right)\right]_{\overline{n^\star}} \\ &= \left[\mathbf{prox}_{\gamma r}\left(\sum_{i=1}^M \alpha_i x^{k-D_i^k} - \gamma \sum_{i=1}^M \alpha_i [\nabla f_i(x^{k-D_i^k})]_{\overline{n^\star}}\right)\right]_{\overline{n^\star}} .\end{aligned}$$

This exactly coincides with a non-sparsified update on the restriction of f_i to the subspace of vectors with null coordinates in $\overline{n^\star}$. More specifically, let $S^\star = \{x \in \mathbb{R}^n : \mathrm{null}(x) = n^\star\}$ be the subspace of vectors with null coordinates in $\overline{n^\star}$, and $f_{i|\overline{n^\star}}$ be the restriction of f_i to $S^\star$. Then the above iteration coincides with non-sparsified update on $((\alpha_i), (f_{i|\overline{n^\star}}), r)$. In other words, after identification, `I-SPY` is no longer random and has the same iterates as `DAve-PG` on $S^\star$ (while in $S^{\star\perp}$, the algorithm has converged to 0). Theorem 1.14 therefore guarantees that `I-SPY` benefits from a $(1-\gamma(\mu+\rho))^2$ rate in terms of epochs (since $(\mu+\rho)$ is the modulus of strong convexity).

□

Thus our algorithm eventually has the practical interest of having sparse two-way communication, at almost no additional cost.

4.2.2 Communication complexity

We study in this section the asymptotic communication complexity of our method in terms of *number of coordinates (*real *numbers) exchanged between the master and the workers.*

To do so, we combine the controlled number of (inner) iterations with the communication cost by iteration. The convergence rate analysis relies on the epoch sequence through the number of iterations (and thus communications) per epoch, which is unknown since it

depends on the computing system. We thus make the assumption that the computing setup is able to perform an epoch within a predefined number of iterations, which is verified in many practical contexts e.g. when the delays are bounded (recall Corollary 1.15). Note that this assumptions is only needed for the communication complexity analysis but not by any means for the algorithm nor the previous results.

Assumption 4.8 (Tamed epochs)**.** *The number of iterations between two epochs is bounded as* $k_m - k_{m-1} \leq K$*.*

We now define the *communication complexity* of our method as the number $\mathbf{C}(\varepsilon)$ of coordinates exchanged between master and workers in order to reach an accuracy ε. For Reconditioned–`I-SPY`, we get that

$$\mathbf{C}(\varepsilon) = K(\mathbf{c}^{\text{up}} + \mathbf{c}^{\text{down}})\mathbf{M}^{\mathsf{C}}\mathbf{L}(\varepsilon) \tag{4.13}$$

where

- $\mathbf{c}^{\text{up}}$ (resp. $\mathbf{c}^{\text{down}}$) is the (expected) number of coordinates communicated from the master to the active worker (resp. from the active worker to the master);
- $\mathbf{M}^{\mathsf{C}}$ is the (expected) number of epochs of the inner method to reach stopping criterion C_2 and C_3 (Note that, for C_1, the complexity is different, in $\tilde{O}(1)$ by construction);
- $\mathbf{L}(\varepsilon)$ is the number of outer loops to reach accuracy ε:

$$\mathbf{L}(\varepsilon) = \min\left\{\ell : \|x_\ell - x^\star\|^2 \leq \varepsilon\right\}.$$

Focusing on the final regime of the algorithm when identification has taken place (as per Section 4.2.1), we get

$$\mathbf{c}^{\text{up}} = |\text{supp}(x_\ell^k)| = |\text{supp}(x^\star)| = s^\star \quad \text{and} \quad \mathbf{c}^{\text{down}} = s^\star + c$$

which leads to the following communication complexity for Reconditioned–`I-SPY`.

Theorem 4.9 (Communication complexity of Reconditioned–`I-SPY`)**.** *Let the functions* (f_i) *be* μ*-strongly convex and* L*-smooth* ($\mu > 0$)*. Let* r *be convex lsc. Let assumptions 4.5 and 4.8 hold. If the parameter* c *is of the same order as* $s^\star$ *compared to* n ($c \approx s^\star \ll n$)*, then the communication complexity* (4.13) *of Reconditioned–`I-SPY` with criteria* C_2 *or* C_3 *is:*

$$\mathbf{C}(\varepsilon) = \tilde{O}\left(\frac{L-\mu}{\mu}\sqrt{n\,s^\star}\ \max\left\{\sqrt{\frac{c}{s^\star}}; \sqrt{\frac{s^\star}{c}}\right\}\log\left(\frac{1}{\varepsilon}\right)\right).$$

Proof of Theorem 4.7. The proof consists in evaluating the terms in (4.13), one by one, in the right regime. From Theorem 4.6 we know that identification takes place i.e. that we have $\mathrm{null}(x_\ell^k) = \mathrm{null}(x^\star) := n^\star$ for all k, ℓ ($\ell \geq \Lambda$). In this case,

$$[x_\ell^k]_{n^\star} = 0 \quad \text{and} \quad [x_\ell^k]_{\overline{n^\star}} = x_\ell^k$$

where we denote by $\overline{n^\star}$ the complementary of $n^\star$. Then, we have

$$\begin{aligned}[x_\ell^k]_{\overline{n^\star}} &= \left[\mathbf{prox}_{\gamma r}([\overline{x}_\ell^k]_{\overline{n^\star}})\right]_{\overline{n^\star}} = \left[\mathbf{prox}_{\gamma r}\left(\sum_{i=1}^M \alpha_i [x^{k-D_i^k}]_{\overline{n^\star}} - \gamma \sum_{i=1}^M \alpha_i [\nabla f_i(x^{k-D_i^k})]_{\overline{n^\star}}\right)\right]_{\overline{n^\star}} \\ &= \left[\mathbf{prox}_{\gamma r}\left(\sum_{i=1}^M \alpha_i x^{k-D_i^k} - \gamma \sum_{i=1}^M \alpha_i [\nabla f_i(x^{k-D_i^k})]_{\overline{n^\star}}\right)\right]_{\overline{n^\star}}.\end{aligned}$$

This exactly coincides with a non-sparsified update on the restriction of f_i to the subspace of vectors with null coordinates in $\overline{n^\star}$. More specifically, let $S^\star = \{x \in \mathbb{R}^n : \mathrm{null}(x) = n^\star\}$ be the subspace of vectors with null coordinates in $\overline{n^\star}$, and $f_{i|\overline{n^\star}}$ be the restriction of f_i to $S^\star$. Then the above iteration coincides with non-sparsified update on $((\alpha_i), (f_{i|\overline{n^\star}}), r)$. In other words, after identification, SPY is no longer random and has the same iterates as DAve-PG on $S^\star$ (while in $S^{\star\perp}$, the algorithm has converged to 0). Theorem 1.14 therefore guarantees that SPY benefits from a $(1 - \gamma(\mu + \rho))^2$ rate in terms of epochs (since $(\mu + \rho)$ is the modulus of strong convexity). □

Comparing this result with the communication complexity of our baseline DAve-PG shows the interest of our adaptive sparsification technique. For DAve-PG, the communication complexity writes as

$$\mathbf{C}(\varepsilon) = K(\mathbf{c}^{\mathrm{up}} + \mathbf{c}^{\mathrm{down}})\mathbf{M}(\varepsilon)$$

where $\mathbf{M}(\varepsilon)$ is the number of epochs to reach accuracy ε. Theorem 1.14 gives us $\mathbf{M}(\varepsilon) = \mathcal{O}((\mu + L)/\mu \log(1/\varepsilon))$. Noticing that DAve-PG also identifies the optimal support (apply e.g. [IM20, Cor. 1]), we get $\mathbf{c}^{\mathrm{up}} = s^\star$ as previously. However without the sparsification of SPY, the cost of an upward communication is $\mathbf{c}^{\mathrm{down}} = n$. This yields the following gain in communication complexity of our algorithm (the greater, the more performing Reconditioned–I-SPY compared to DAve-PG):

$$\tilde{O}\left(\frac{1 + \kappa_{(\mathsf{P})}}{1 - \kappa_{(\mathsf{P})}} \min\left\{\sqrt{\frac{c}{s^\star}}; \sqrt{\frac{s^\star}{c}}\right\} \frac{n + s^\star}{\sqrt{n s^\star}}\right).$$

The gain thus shows a product of three terms. The first term depends on the condition

number of the problem and indicates that our method dominates DAve-PG for all problems. The second term is in $(0, 1]$ but should be not too far from 1, provided that the final sparsity is not poorly estimated. Finally, the last term fully exhibits the merits of adaptive sparsification with a term in $nr + s^\star$ for DAve-PG much greater than the $\sqrt{ns^\star}$ for Reconditioned–I-SPY. This last term really shows a nice dependence in the dimension of the problem and optimal solution for the proposed method. The theoretical gain of Reconditioned–I-SPY compared to DAve-PG is confirmed in the next numerical illustrations.

4.3 Numerical illustrations

In this section, we illustrate the communication gain provided by our random sparsification algorithms on two classic ℓ_1 regularized empirical risk minimization problems.

4.3.1 Experimental setup

Problem

We first consider a synthetic LASSO problem

$$\min_{x\in\mathbb{R}^n} \quad \|Ax - b\|_2^2 + \lambda_1\|x\|_1 \tag{4.14}$$

with $n = 1000$ features and two different sizes of example set $m = 500$ and $m = 10,000$. Data matrix A is generated from the standard normal distribution, $b = Ax_0 + e$ where x_0 is a 99% sparse vector and e is taken from the normal distribution with standard deviation 0.01. We take λ_1 to reach the density of the final solution approximately 1.2%.

We also examine the regularized logistic regression with elastic net

$$\min_{x\in\mathbb{R}^n} \quad \frac{1}{m}\sum_{j=1}^{m}\log(1+\exp(-y_j z_j^\top x)) + \lambda_1\|x\|_1 + \frac{\lambda_2}{2}\|x\|_2^2 \tag{4.15}$$

on two data-sets from the LibSVM repository: the *madelon* data-set ($n = 500$ $m = 2000$) with hyperparameters $\lambda_2 = 0.03$ and $\lambda_1 = 0.001$, chosen to reach a 99% sparsity; the *rcv1_train* dataset ($n = 47236$ $m = 20242$) with parameters $\lambda_1 = 0.001$ and $\lambda_2 = 0.0001$ to reach a 99.7% sparsity.

Setup details

We used Python and MPI (Message Passing Interface) for the distributed communications framework. To communicate sparse vectors, we send a list of coordinates then their values,

as usual in sparse communications.

We run our experiments on a machine with 32 cores and 256 Gb of RAM: one core plays the role the master, M cores are the slaves ($M = 5$ for LASSO problems, $M = 10$ for madelon, and $M = 20$ for rcv1_train). The data sets are split evenly between the M slaves, each having access only to its own part.

Restart technique Let us precise the way we perform restarts for Reconditioned–`I-SPY`. Since, after the restart only two things changes for every worker: the prox-center and the probability vector, the first update received by master from any worker after the "restart" could be modified by master to the "correct update" (with a new prox-center but with a previous probability vector) by adding the weighted difference of old and new prox-centers to it. Furthermore, after sending the new prox-center to the worker, this "shift" could also be performed with the worker's local variable x_i that allows making "sliding" restart without any synchronization rounds.

Algorithms

We compare three algorithms:

- 'DAve-PG': `DAve-PG` algorithm without any sparsification;
- 'Reco-I-SPY; (xxx;n)': Reconditioned–`I-SPY` with simplified stopping criteria C_1 that consider $\mathsf{M}_\ell = n$. Where 'xxx' corresponds to the algorithm parameter c - the amount of randomly chosen coordinates;
- 'Reco-I-SPY; (xxx;cn)': Reconditioned–`I-SPY` with stopping criteria C_n.

We display the performance of the algorithms in four ways: i) size of support vs number of iterations, showing the identification properties; functional suboptimality vs ii) communication cost, modeled as the number of couples (coordinate, value) sent from and to the master, iii) amount of epochs (1.31), and iv) computational time (only for rcv1_train dataset, where the dimension of the problem is big enough to see the communication bottleneck in practice.

Performance of fixed budget criterion

Let us discuss the performance of different stopping criteria in practice. For this, let us consider two different criteria: theoretical (epoch budget) C_1 with constant budget n and practical (fixed budget).

In Figure 4-1, we present the convergence of Reconditioned–`I-SPY` with fixed-budget stopping criteria on synthetic LASSO problem (4.14) with $m = 500$ and $n = 1000$. As

we could see, the performance of the Reconditioned–I-SPY with C_1 with fixed budget $\mathsf{M}_\ell \equiv 1$ is much better than for ones with inner cycle of bigger size. On the one hand, the theoretical budget criterion C_1 propose the non-decreasing sequence $\mathsf{M}_\ell \geq 1$. On the other hand, as we could see from the plots, the slope is constant after some time, that means the rate of real C_1 is expected to be slower than the algorithm run with 1-epoch fixed budget and this is shown in Figure 4-2. More precisely, we consider problem with $m = 10000$ and $n = 1000$ and as we could see, that the rate of C_1 is worse and since lines contains horizontal parts the size of inner loop is bigger than it should be. We could see that algorithm with C_1 is faster in identification in the beginning; however it needs much more time to identify the correct active-set in the end.

Comparing with relative accuracy criterion

Let us now consider the theoretical criterion C_3. From the practical point of view this criterion is impossible to use since it requires the knowledge of the full dataset to verify. However, for us, it is important to make a comparison with it so we could see if the fixed budget mode is good enough in practice.

Criterion C_3 for LASSO To verify the stopping criterion, we use the duality gap and the fact that the inner problem is convex and lower semi-continuous, we could bound the distance to the solution using the strong duality. We consider synthetic LASSO problem for which the solution of the dual problem could be explicitly computed from the primal solution.

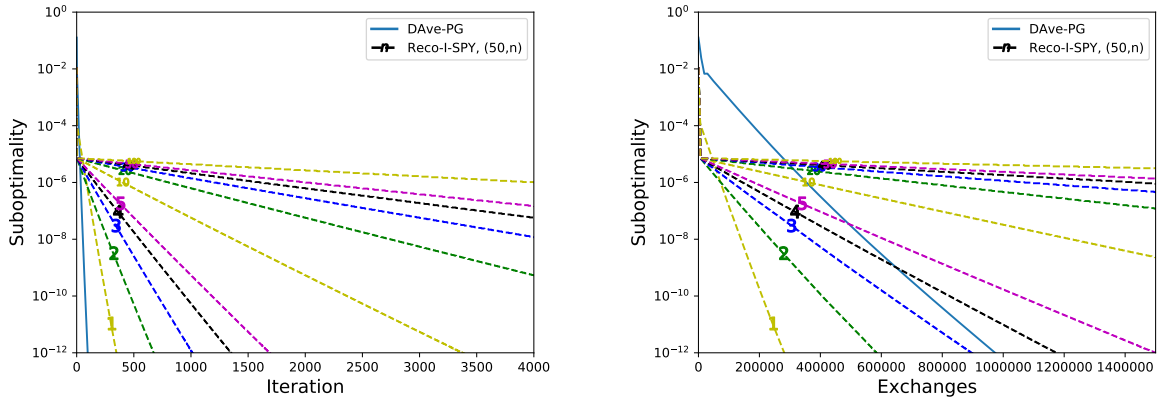


Figure 4-1. Dependence between convergence and the size of fixed budget on synthetic LASSO (4.14).

In Figure 4-3, we see that the theoretical criterion C_3 performs better in practice; however, the performance of 1-epoch mode is worse only in the beginning since the duality gap does not give a tight approximation of the distance to the solution in general.

Criterion C_3 for logistic regression Let us now consider logistic regression problem (4.15) where the dual solution is hard to compute. In this case, we use the following bound

$$\|x_{\ell+1} - \mathbf{prox}_{F/\rho}(x_\ell)\|_2 \leq \frac{1}{\mu+\rho}\|\partial H_\ell(x_{\ell+1}) - \partial H_\ell(x_\ell^\star)\|_2 \leq \frac{1}{\mu+\rho}\|\partial H_\ell(x_{\ell+1})\|_2.$$

Since we use ℓ_1 regularized problem the distance of subdifferential to 0 is easy to compute; however, it requires full dateset to calculate it. This bound is less tight than duality gap one, so for practical experiments we use the stopping criterion

$$\|\partial H_\ell(x_{\ell+1})\|_2^2 \leq \frac{(\mu+\rho)^2\rho}{4(2\mu+\rho)(0.001\ell)^{2+2\delta}}\|x_{\ell+1} - x_\ell\|_2^2 \tag{4.16}$$

In Figure 4-4, we present the performance of 1-epoch mode versus C_3 with the practical consideration as in (4.16). We consider logistic loss with elastic-net regularization on madelon dataset with $\lambda_1 = 0.03$ and $\lambda_2 = 0.001$. As we could see from the plots, the performance of such version of stopping criterion is worse than 1-epoch mode in practice since the distance from the subdifferential to 0 is not the best approximator; however, in the beginning, the performance of the theoretical run C_3 is still better.

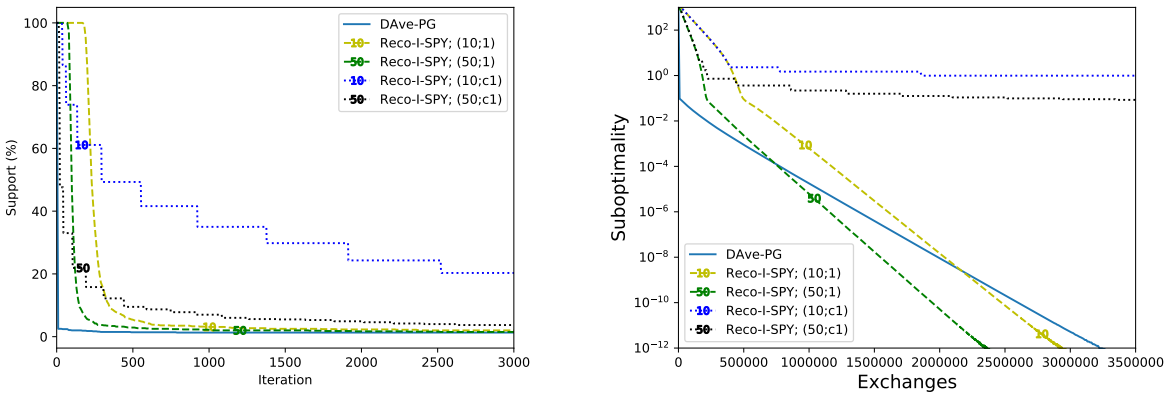


Figure 4-2. Synthetic LASSO (4.14): 1-epoch vs C_1.

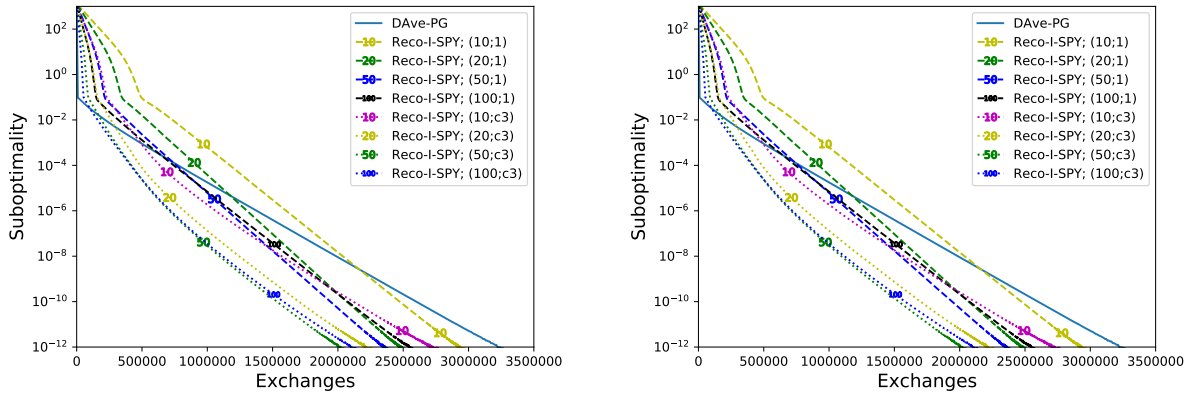


Figure 4-3. Synthetic LASSO (4.14): 1-epoch vs C_3.

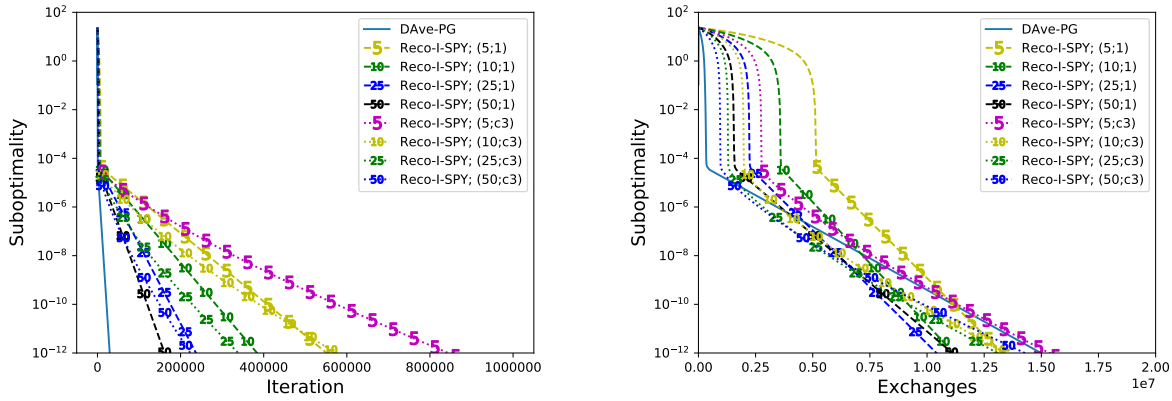


Figure 4-4. Logistic regression with elastic net regularization and madelon dataset: 1-epoch vs C_3.

4.3.2 Warm start

As we could see from the Figure 4-2, the identification moment takes much more iterates for Reconditioned–`I-SPY`; however, there is no need to sparsify updates from the beginning since master-to-worker communication is not sparse. Considering this, we propose to run `DAve-PG` Algorithm first, for some amount of iterations and switch to Reconditioned–`I-SPY` when the current master point is sparse enough.

In Figure 4-5, we present the experimental result for rcv1_train dataset that has much more coordinates so that we could see the communication bottleneck in practice. More

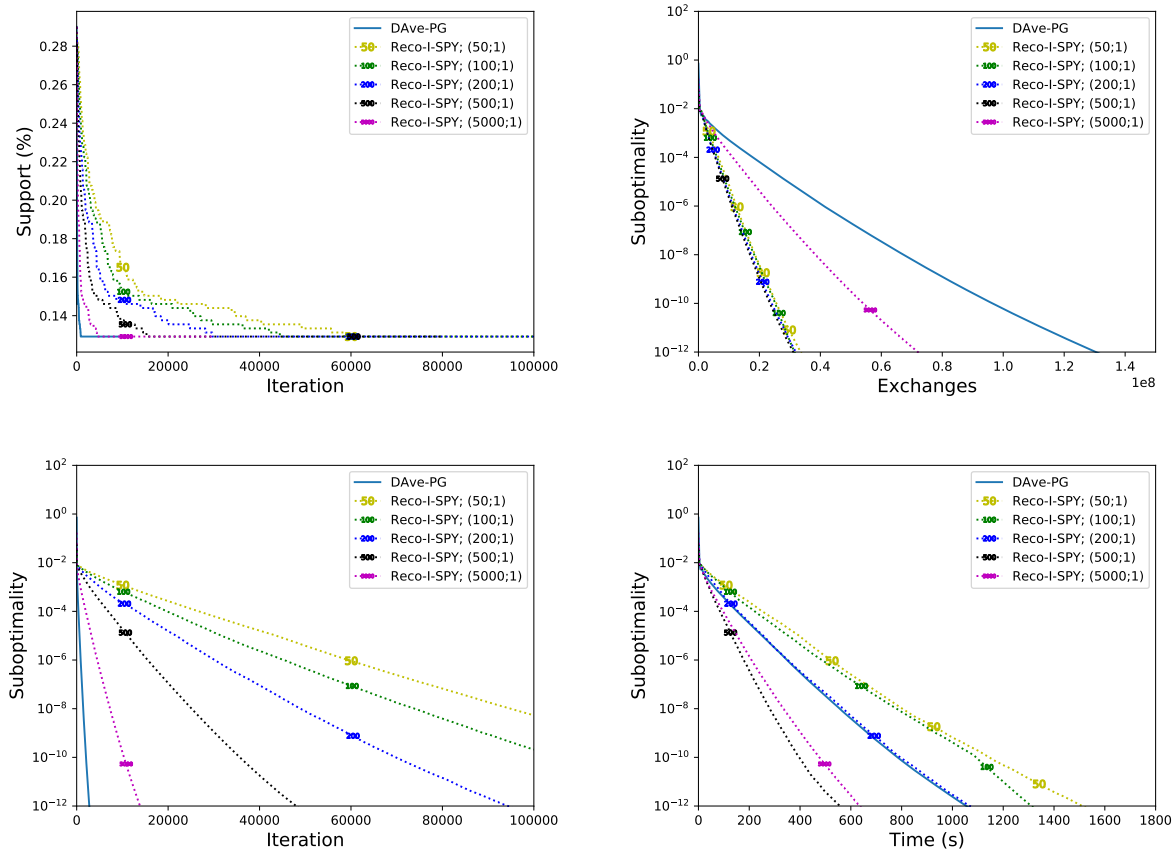


Figure 4-5. Logistic regression with rcv1_train dataset: 1-epoch.

precisely, sparsified algorithm performs much better in terms of communication; however, the total amount of iterations is also sufficiently bigger for the very small c. As a result, the total time complexity is not well approximated by exchanges for $c \ll n$, that prevents us from using extreme sparsification that makes identification process and, as a result, convergence slower.

4.4 Conclusion

In this chapter, we present Reconditioned–`I-SPY` Algorithm that consists of two key components: `I-SPY` algorithm, that allows solving well-conditioned problems efficiently in terms of data exchanges and proximal reconditioning technique that allows solving the ill-conditioned problem via solving a sequence of well-conditioned subproblems iteratively.

We present both the theoretical result and the performance in practice that both show that the sparsification technique helps to save runtime of the minimization process by decreasing the total amounts of bits sent.

Conclusion and perspectives

We propose a couple of techniques to reduce the dimensionality of the problem to reach a better convergence result both for distributed and non-distributed setups. In both cases, we focused on composite optimization problems with sparsity inducing regularizers.

First, we consider the non-distributed setting where on every iteration of algorithm, we have an access to the whole dataset. It allows us to use vanilla proximal algorithm as a basic method for our algorithm. Using the identification property of proximal methods for a wide class of regularizers, we propose a "sketch-and-project" technique with specific identification-based selections of projections. This allows the acceleration of proximal gradient descent in terms of the amount of dimensions explored that we show both in theory and practice. Moreover, our technique allows using non-separable regularizers.

Second, we present our approach for an asynchronous distributed master-worker setup. We consider the problem of sparsification of distributed proximal algorithm by using the same identification-based idea. However, asynchronicity brings additional restrictions, so we consider only ℓ_1 regularized problems in our study. Starting from a random sparsification that performs worse both in theory and practice, we propose two different approaches. In the first one, we investigate the practical performance of the algorithm with identification-based sparsification that is proven to converge only for well-conditioned problems. We show both that this algorithm could diverge, but in the same time, if it converges, it brings a significant performance profit. In the second one, we propose a proximal reconditioning technique that allows minimization of ill-conditioned problem via iterative approximate minimization of well-conditioned ones. It allows using our algorithm with theoretical guarantees to converge and good performance in practice.

This work opens several perspectives for future development. As we could see from the Figures 4-5 and 3-6, 3-4 the correlation between runtime and the amount of exchanges is better for `I-SPY` and it corresponds to the logic of the algorithm. As we could see from the theoretical results, the amount of iterations to solve the problem is larger if the sparsification is stronger. However, thanks to the identification property, the speed becomes faster and performance becomes exactly the same as for `DAve-PG`. The situation with Reconditioned–`I-SPY` is not the same. On the one hand, identification takes place for this algorithm that allows solving the inner problem as fast as `DAve-PG`. On the other

hand, the proximal parameter ρ is large for small c, which makes the profit in terms of exchanges less significant than in `I-SPY` case. We could see from the numerical result, the amount of iterations grows faster than the amount of communications decreases. This property brings us to the "sparsification limit" (the smallest possible c that leads to the acceleration) and it could be further discovered in future works.

Furthermore, in Remark 4.4, we mention the acceleration of algorithms in Nesterov's sense and Catalyst algorithm as an example of the accelerated proximal reconditioning technique. Since the dependence of amount of coordinates c (as well as of proximal parameter ρ) in case of acceleration would be as a square root it could make the amount of required iterations for `I-SPY` to converge much smaller and closer to `DAve-PG`, or even smaller that could lead to the better convergence both in theory and practice. In addition, it would be interesting to compare performance of such accelerated method with performance of `I-SPY` when it converges.

Finally, one of the important open question is to design an efficient distributed version of Adaptive Randomized Proximal Subspace Descent, when each machine selects the subspace to project independently like in [MHR20]. Or even to investigate if the asynchronous option is possible.

In recent [BI19] authors propose a method that is much more stable in terms of identification than accelerated proximal gradient descent and it is interesting to combine this method with our sparsification technique.

Bibliography

[ABMP13] Andreas Argyriou, Luca Baldassarre, Charles A Micchelli, and Massimiliano Pontil. On sparsity inducing regularization methods for machine learning. In *Empirical Inference*, pages 205–216. Springer, 2013.

[AFJ16] Arda Aytekin, Hamid Reza Feyzmahdavian, and Mikael Johansson. Analysis and implementation of an asynchronous optimization algorithm for the parameter server. *arXiv preprint arXiv:1610.05507*, 2016.

[AGL+17] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

[B+66] PK Bhattacharya et al. Estimating the mean of a multivariate normal population with general quadratic loss function. *The Annals of Mathematical Statistics*, 37(6):1819–1824, 1966.

[B+15] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

[BC11] Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer Science & Business Media, 2011.

[Ber76] Dimitri P Bertsekas. On the goldstein-levitin-polyak gradient projection method. *IEEE Transactions on automatic control*, 21(2):174–184, 1976.

[Ber11] Dimitri P Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical Programming*, 129(2):163, 2011.

[BHG07] Doron Blatt, Alfred O Hero, and Hillel Gauchman. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1):29–51, 2007.

[BHI15] Pascal Bianchi, Walid Hachem, and Franck Iutzeler. A coordinate descent primal-dual algorithm and application to distributed asynchronous optimization. *IEEE Transactions on Automatic Control*, 61(10):2947–2957, 2015.

[BI19] Gilles Bareilles and Franck Iutzeler. On the interplay between acceleration and identification for the proximal gradient algorithm. *arXiv preprint arXiv:1909.08944*, 2019.

[BJM+12] Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[BKL66] R.E. Bellman, R.E. Kalaba, and J. Lockett. *Numerical Inversion of the Laplace Transform*, pages 143–144. Elsevier, New York, 1966.

[BLG+20] Gilles Bareilles, Yassine Laguel, Dmitry Grishchenko, Franck Iutzeler, and Jérôme Malick. Randomized Progressive Hedging methods for Multi-stage Stochastic Programming. *Annals of Operations Research*, 2020.

[BM88] James V Burke and Jorge J Moré. On the identification of active constraints. *SIAM Journal on Numerical Analysis*, 25(5):1197–1211, 1988.

[BNH19] Tal Ben-Nun and Torsten Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR)*, 52(4):1–43, 2019.

[Bot10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[BPC+11a] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning*, 3(1):1–122, 2011.

[BPC+11b] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[BT89] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[BT97] Dimitri P Bertsekas and John N Tsitsiklis. Parallel and distributed computation: numerical methods. 1989. *Englewood Clifffs, New Jersey: Printice-Hall*, 1997.

[BT09] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[BWAA18] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

[Cau47] Augustin Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[CL11] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[CM09] Olivier Cappé and Eric Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

[CMBJ16] Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. In *International Conference on Learning Representations Workshop Track*, 2016.

[CP11] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer, 2011.

[CP15] Patrick L Combettes and Jean-Christophe Pesquet. Stochastic quasi-fejér block-coordinate fixed point iterations with random sweeping. *SIAM Journal on Optimization*, 25(2):1221–1248, 2015.

[CQR15] Dominik Csiba, Zheng Qu, and Peter Richtárik. Stochastic dual coordinate ascent with adaptive probabilities. In *International Conference on Machine Learning*, pages 674–683, 2015.

[CR17] Clément Calauzènes and Nicolas Le Roux. Distributed SAGA: Maintaining linear convergence rate with limited communication. *arXiv preprint arXiv:1705.10405*, 2017.

[CWB08] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted l 1 minimization. *Journal of Fourier Analysis and Applications*, 14(5-6):877–905, 2008.

[DAW11] John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 57(3):592–606, 2011.

[DBLJ14] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[DDC14] Aaron Defazio, Justin Domke, and Tiberio Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1125–1133, 2014.

[DDDM04] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

[DGBSX12] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202, 2012.

[DL14] Dmitriy Drusvyatskiy and Adrian S Lewis. Optimality, identifiability, and sensitivity. *Mathematical Programming*, 147(1-2):467–498, 2014.

[Don95] David L Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

[DRT11] Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Nearest neighbor based greedy coordinate descent. In *Advances in Neural Information Processing Systems*, 2011.

[FGMP19] Jalal M Fadili, Guillaume Garrigos, Jérôme Malick, and Gabriel Peyré. Model consistency for learning with mirror-stratifiable regularizers. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

[FGS15] Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Mind the duality gap: safer rules for the lasso. In *International Conference on Machine Learning*, pages 333–342, 2015.

[FML12] Marc Fuentes, Jérôme Malick, and Claude Lemaréchal. Descentwise inexact proximal algorithms for smooth optimization. *Computational Optimization and Applications*, 53(3):755–769, 2012.

[FMP18] Jalal Fadili, Jerome Malick, and Gabriel Peyré. Sensitivity analysis for mirror-stratifiable convex functions. *SIAM Journal on Optimization*, 28(4):2975–3000, 2018.

[FR15] Rafael Frongillo and Mark D Reid. Convergence analysis of prediction markets via randomized subspace descent. In *Advances in Neural Information Processing Systems*, pages 3034–3042, 2015.

[Gab83] Daniel Gabay. Chapter IX applications of the method of multipliers to variational inequalities. In *Studies in Mathematics and its Applications*, volume 15, pages 299–331. Elsevier, 1983.

[GD13] Tobias Glasmachers and Urun Dogan. Accelerated coordinate descent with adaptive coordinate frequencies. In *Asian Conference on Machine Learning*, pages 72–86, 2013.

[GIM20] Dmitry Grishchenko, Franck Iutzeler, and Jérôme Malick. Proximal gradient methods with adaptive subspace sampling. *Mathematics of Operations Research*, 2020.

[GIMA18] Dmitry Grishchenko, Franck Iutzeler, Jérôme Malick, and Massih-Reza Amini. Asynchronous distributed learning with sparse communications and identification. *arXiv preprint arXiv:1812.03871*, 2018.

[GR15] Robert M Gower and Peter Richtárik. Randomized iterative methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1660–1690, 2015.

[GRC09] Gilles Gasso, Alain Rakotomamonjy, and Stéphane Canu. Recovering sparse signals with a certain family of nonconvex penalties and dc programming. *IEEE Transactions on Signal Processing*, 57(12):4686–4698, 2009.

[Gül92] Osman Güler. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

[HCS+17] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[HKM+19] Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, Sebastian Stich, and Peter Richtárik. Stochastic distributed learning with gradient quantization and variance reduction. *arXiv preprint arXiv:1904.05115*, 2019.

[HL04] Warren L Hare and Adrian S Lewis. Identifying active constraints via partial smoothness and prox-regularity. *Journal of Convex Analysis*, 11(2):251–266, 2004.

[HLLJM15] Thomas Hofmann, Aurelien Lucchi, Simon Lacoste-Julien, and Brian McWilliams. Variance reduced stochastic gradient descent with neighbors. In *Advances in Neural Information Processing Systems*, pages 2305–2313, 2015.

[HMR18] Filip Hanzely, Konstantin Mishchenko, and Peter Richtárik. Sega: Variance reduction via gradient sketching. In *Advances in Neural Information Processing Systems*, pages 2082–2093, 2018.

[HUL12] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of convex analysis*. Springer Science & Business Media, 2012.

[HY16] Robert Hannah and Wotao Yin. On unbounded delays in asynchronous parallel fixed-point algorithms. *Journal of Scientific Computing*, pages 1–28, 2016.

[IBCH13] Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 3671–3676. IEEE, 2013.

[IGGS19] Anastasiya Ivanova, Dmitry Grishchenko, Alexander Gasnikov, and Egor Shulgin. Adaptive catalyst for smooth convex optimization. *arXiv preprint arXiv:1911.11271*, 2019.

[IM20] Franck Iutzeler and Jérôme Malick. Nonsmoothness in machine learning: specific structure, proximal identification, and applications. *arXiv preprint arXiv:2010.00848*, 2020.

[JAB11] Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, 12(Oct):2777–2824, 2011.

[JZ13] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[KHR20] Dmitry Kovalev, Samuel Horváth, and Peter Richtárik. Don't jump through hoops and remove those loops: Svrg and katyusha are better without the outer loop. In *Algorithmic Learning Theory*, pages 451–467, 2020.

[KMR19] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. First analysis of local gd on heterogeneous data. *NeurIPS, Federated Learning for Data Privacy and Confidentiality workshop*, 2019.

[KMR20] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local SGD on identical and heterogeneous data. In *The 23rd International Conference on Artificial Intelligence and Statistics (AISTATS 2020)*, 2020.

[KMRR16] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[KMY+16] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv:1610.05492*, 2016.

[Kol33] A.N. Kolmogotrov. Grundbegriffe der wahrscheinlichkeitreichnung. *Ergebnisse der Mathematik*, 1933.

[KSJ19] Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International Conference on Machine Learning*, pages 3478–3487, 2019.

[Kum02] Vipin Kumar. *Introduction to Parallel Computing*. Addison-Wesley Longman, 2002.

[LAS13] Mu Li, David G Andersen, and Alexander Smola. Distributed delayed proximal gradient methods. In *NIPS Workshop on Optimization for Machine Learning*, 2013.

[Lew02] Adrian S Lewis. Active sets, nonsmoothness, and sensitivity. *SIAM Journal on Optimization*, 13(3):702–725, 2002.

[LFP17] Jingwei Liang, Jalal Fadili, and Gabriel Peyré. Activity identification and local linear convergence of forward–backward-type methods. *SIAM Journal on Optimization*, 27(1):408–437, 2017.

[LHM+17] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[LHY00] Kenneth Lange, David R Hunter, and Ilsoon Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1–20, 2000.

[LL18] Adrian S Lewis and Jingwei Liang. Partial smoothness and constant rank. *arXiv preprint arXiv:1807.03134*, 2018.

[LMH15] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pages 3384–3392, 2015.

[LMH17] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. Catalyst acceleration for first-order convex optimization: from theory to practice. *The Journal of Machine Learning Research*, 18(1):7854–7907, 2017.

[LMH19] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. An inexact variable metric proximal point algorithm for generic quasi-Newton acceleration. *SIAM Journal on Optimization*, 29(2):1408–1443, 2019.

[LPLJ17] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. Asaga: Asynchronous parallel saga. In *Artificial Intelligence and Statistics*, pages 46–54, 2017.

[LS97a] Claude Lemaréchal and Claudia Sagastizábal. Practical aspects of the moreau–yosida regularization: Theoretical preliminaries. *SIAM Journal on Optimization*, 7(2):367–385, 1997.

[LS97b] Claude Lemaréchal and Claudia Sagastizábal. Variable metric bundle methods: from conceptual to implementable forms. *Mathematical Programming*, 76(3):393–410, 1997.

[LSS11] Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Adaptive coordinate descent. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pages 885–892. ACM, 2011.

[LSTS19] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

[LW12] Sangkyun Lee and Stephen J Wright. Manifold identification in dual averaging for regularized stochastic online learning. *Journal of Machine Learning Research*, 13, 2012.

[LWR+15] Ji Liu, Stephen J Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *The Journal of Machine Learning Research*, 16(1):285–322, 2015.

[Mai15] Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

[Mar70] B. Martinet. Régularisation d'inéquations variationelles par approximation successives. *Revue Française d'Informatique et de Recherche Opérationelle*, R3:154–158, 1970.

[MHR20] Konstantin Mishchenko, Filip Hanzely, and Peter Richtárik. 99% of parallel optimization is inevitably a waste of time. *UAI*, 2020.

[MIM20] Konstantin Mishchenko, Franck Iutzeler, and Jérôme Malick. A distributed flexible delay-tolerant proximal gradient algorithm. *SIAM Journal on Optimization*, 30(1):933–959, 2020.

[MIMA18] Konstantin Mishchenko, Franck Iutzeler, Jérôme Malick, and Massih-Reza Amini. A delay-tolerant proximal-gradient algorithm for distributed learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 3587–3595, 2018.

[MKJ+17] Chenxin Ma, Jakub Konečný, Martin Jaggi, Virginia Smith, Michael I Jordan, Peter Richtárik, and Martin Takáč. Distributed optimization with arbitrary local solvers. *Optimization Methods and Software*, 32(4):813–848, 2017.

[Mor62] Jean Jacques Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. 1962.

[MSJ+15] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael Jordan, Peter Richtárik, and Martin Takáč. Adding vs. averaging in distributed primal-dual optimization. In *International Conference on Machine Learning*, pages 1973–1982, 2015.

[NB01] Angelia Nedić and Dimitri Bertsekas. Convergence rate of incremental subgradient algorithms. In *Stochastic optimization: algorithms and applications*, pages 223–264. Springer, 2001.

[Nes83] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547, 1983.

[Nes05] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[Nes12] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[Nes13] Yu. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[NH98] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[NLS17] Julie Nutini, Issam Laradji, and Mark Schmidt. Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence. *arXiv preprint arXiv:1712.08859*, 2017.

[NO09] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[NP14] Ion Necoara and Andrei Patrascu. A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints. *Computational Optimization and Applications*, 57(2):307–337, 2014.

[NSH19] Julie Nutini, Mark Schmidt, and Warren Hare. "Active-set complexity" of proximal gradient: How long does it take to find the sparsity pattern? *Optimization Letters*, 13(4):645–655, 2019.

[NSL+15] Julie Nutini, Mark Schmidt, Issam Laradji, Michael Friedlander, and Hoyt Koepke. Coordinate descent converges faster with the gauss-southwell rule than random selection. In *International Conference on Machine Learning*, 2015.

[NSYD17] Hongseok Namkoong, Aman Sinha, Steve Yadlowsky, and John C Duchi. Adaptive sampling probabilities for non-smooth optimization. In *International Conference on Machine Learning*, pages 2574–2583, 2017.

[OST13] Kohei Ogawa, Yoshiki Suzuki, and Ichiro Takeuchi. Safe screening of non-support vectors in pathwise SVM computation. In *International Conference on Machine Learning*, pages 1382–1390, 2013.

[PB14] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[PCJ17] Dmytro Perekrestenko, Volkan Cevher, and Martin Jaggi. Faster coordinate descent via adaptive importance sampling. *AISTATS*, 2017.

[PLLJ17] Fabian Pedregosa, Rémi Leblond, and Simon Lacoste-Julien. Breaking the nonsmooth barrier: A scalable parallel method for composite optimization. In *Advances in Neural Information Processing Systems*, pages 55–64, 2017.

[PLS18] Clarice Poon, Jingwei Liang, and Carola Schoenlieb. Local convergence properties of SAGA/prox-SVRG and acceleration. In *International Conference on Machine Learning*, pages 4124–4132, 2018.

[Pol63] Boris Teodorovich Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963.

[Pol69a] Boris Teodorovich Polyak. The conjugate gradient method in extremal problems. *USSR Computational Mathematics and Mathematical Physics*, 9(4):94–112, 1969.

[Pol69b] Boris Teodorovich Polyak. Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics*, 9(3):14–29, 1969.

[PXYY16] Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.

[QR16] Zheng Qu and Peter Richtárik. Coordinate descent with arbitrary sampling i: Algorithms and complexity. *Optimization Methods and Software*, 31(5):829–857, 2016.

[QSMR19] Xun Qian, Alibek Sailanbayev, Konstantin Mishchenko, and Peter Richtárik. Miso is making a comeback with better proofs and rates. *arXiv preprint arXiv:1906.01474*, 2019.

[RFP13] Hugo Raguet, Jalal Fadili, and Gabriel Peyré. A generalized forward-backward splitting. *SIAM Journal on Imaging Sciences*, 6(3):1199–1226, 2013.

[Roc76] R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

[Ros60] Frank Rosenblatt. Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309, 1960.

[RS71] Herbert Robbins and David Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Optimizing Methods in Statistics*, pages 233–257. Elsevier, 1971.

[RT12] Peter Richtárik and Martin Takáč. Efficient serial and parallel coordinate descent methods for huge-scale truss topology design. In *Operations Research Proceedings 2011*, pages 27–32. Springer, 2012.

[RT14] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

[RT16a] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *The Journal of Machine Learning Research*, 17(1):2657–2681, 2016.

[RT16b] Peter Richtárik and Martin Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, 10(6):1233–1243, 2016.

[RT16c] Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1-2):433–484, 2016.

[SCD14] Yuan-Hai Shao, Wei-Jie Chen, and Nai-Yang Deng. Nonparallel hyperplane support vector machine for binary classification problems. *Information Sciences*, 263:22–35, 2014.

[SFJJ16] Virginia Smith, Simone Forte, Michael I Jordan, and Martin Jaggi. L1-regularized distributed optimization: A communication-efficient primal-dual framework. *arXiv preprint arXiv:1512.04011, presented at the ML Systems Workshop of ICML*, 2016.

[Sho64] Naum Z Shor. On the structure of algorithms for the numerical solution of optimal planning and design problems. In *Diss. kand. fiz.-matem. n.* IK Akad. Nauk UkSSR Kiev, 1964.

[Sho12] Naum Zuselevich Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

[SHY17] Tao Sun, Robert Hannah, and Wotao Yin. Asynchronous coordinate descent under more realistic assumptions. In *Advances in Neural Information Processing Systems*, 2017.

[SJNS19a] Yifan Sun, Halyun Jeong, Julie Nutini, and Mark Schmidt. Are we there yet? manifold identification of gradient-related proximal methods. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1110–1119, 2019.

[SJNS19b] Yifan Sun, Halyun Jeong, Julie Nutini, and Mark Schmidt. Are we there yet? manifold identification of gradient-related proximal methods. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1110–1119. PMLR, 16–18 Apr 2019.

[SLRB17] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[SLWY15] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[Sol98] Mikhail V Solodov. Incremental gradient algorithms with stepsizes bounded away from zero. *Computational Optimization and Applications*, 11(1):23–35, 1998.

[SRJ17] Sebastian U Stich, Anant Raj, and Martin Jaggi. Safe adaptive importance sampling. In *Advances in Neural Information Processing Systems*, pages 4381–4391, 2017.

[SS00] Mikhail V Solodov and Benar Fux Svaiter. Error bounds for proximal point subproblems and associated inexact proximal point algorithms. *Mathematical programming*, 88(2):371–389, 2000.

[SS01] Mikhail V Solodov and Benar Fux Svaiter. A unified framework for some inexact proximal point algorithms. *Numerical functional analysis and optimization*, 22:1013–1035, 2001.

[SS14] Ohad Shamir and Nathan Srebro. Distributed stochastic optimization and learning. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 850–857. IEEE, 2014.

[SS16] Shai Shalev-Shwartz. SDCA without duality, regularization, and individual convexity. In *International Conference on Machine Learning*, pages 747–754, 2016.

[SSZ13a] Shai Shalev-Shwartz and Tong Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 378–385, 2013.

[SSZ13b] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

[Sti19] Sebastian U Stich. Local SGD converges fast and communicates little. *ICLR*, 2019.

[SV99] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[TBA86] John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

[TJSZ04] Yuchun Tang, Bo Jin, Yi Sun, and Yan-Qing Zhang. Granular support vector machines for medical binary classification problems. In *2004 Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 73–78. IEEE, 2004.

[TR12] K.I. Tsianos and M.G. Rabbat. Revisiting distributed synchronous sgd. In *American Control Conference*, pages 1067–1072, 2012.

[TRS15] Martin Takáč, Peter Richtárik, and Nathan Srebro. Distributed mini-batch sdca. *arXiv preprint arXiv:1507.08322*, 2015.

[Tse01] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[Tsi84] John Nikolas Tsitsiklis. Problems in decentralized decision making and computation. Technical report, Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, 1984.

[TSR+05] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[Val90] Leslie G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

[Vap13] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[VGFP15] Samuel Vaiter, Mohammad Golbabaee, Jalal Fadili, and Gabriel Peyré. Model selection with low complexity priors. *Information and Inference: A Journal of the IMA*, 4(3):230–287, 2015.

[VGO16] Nuri Denizcan Vanli, Mert Gurbuzbalaban, and Asu Ozdaglar. A stronger convergence result on the proximal incremental aggregated gradient method. *arXiv preprint arXiv:1611.08022*, 2016.

[WKSZ17] Jialei Wang, Mladen Kolar, Nathan Srebro, and Tong Zhang. Efficient distributed learning with sparsity. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3636–3645. JMLR. org, 2017.

[WNF09] Stephen J Wright, Robert D Nowak, and Mário AT Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[Wri93] Stephen J Wright. Identifiable surfaces in constrained optimization. *SIAM Journal on Control and Optimization*, 31(4):1063–1079, 1993.

[Wri12] Stephen J Wright. Accelerated block-coordinate relaxation for regularized optimization. *SIAM Journal on Optimization*, 22(1):159–186, 2012.

[Wri15] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[WWLZ18] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pages 1306–1316, 2018.

[WXY+17] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems*, pages 1509–1519, 2017.

[Yan13] Tianbao Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 629–637, 2013.

[Yos12] Kösaku Yosida. *Functional analysis*. Springer Science & Business Media, 2012.

[YYY11] Isao Yamada, Masahiro Yukawa, and Masao Yamagishi. Minimizing the moreau envelope of nonsmooth convex functions over the fixed point set of certain quasi-nonexpansive mappings. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 345–390. Springer, 2011.

[ZK14] Ruiliang Zhang and James Kwok. Asynchronous distributed admm for consensus optimization. In *International Conference on Machine Learning*, pages 1701–1709, 2014.

[ZRY06] Peng Zhao, Guilherme Rocha, and Bin Yu. Grouped and hierarchical model selection through composite absolute penalties. *Department of Statistics, UC Berkeley, Tech. Rep*, 703, 2006.

[ZZ15] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *International Conference on Machine Learning*, pages 1–9, 2015.