Contents

List of Figures

List of Tables

Chapter 1

From Gradient to Identification

1.1 Introduction

The goal of machine learning is to learn the model from the provided data using some predefined algorithms. In other words, the goal of machine learners is to propose an algorithm (or a set of algorithms) that will be able to make learning process without further human interventions. For example, in the binary classification problems, we would like to learn the binary function that classified every object of the specific set into one of two possible groups based on the attributes of the object. However, the set of function 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 simply memorizing the correct label for every example. But such functions, generally, are not so good to predict the group of any new object as far as their behaviour 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 to use them for new (unknown during the learning process) examples [Vap13]. One of the first examples was the Rosenblatt Perceptron [Ros60] where author proposed to learn the separated hyperplane for the neurons. 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 learning problem, the search of predictor 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 $h(a, x) = x^\top \Phi(a)$, where features $\Phi(a) \in \mathbb{R}^n$;

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

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 loss. However, we have an access only to the finite amount of examples so we should consider empirical loss minimization (ERM) problem

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} \ell(b_i, h(a_i, x)). \tag{1.1}$$

By the law of big numbers we know, that if the amount of examples $m \to \infty$ then the empirical loss will converge to the real loss. This motivates machine learners to learn on as big training sets as possible that makes the problem extremely hard to solve (since m is big).

Another problem that appears in machine learning is overfitting, when a function is too closely fit to a limited set of data points (training set). One of the common technique to avoid overfitting is regularization that force the model to be simpler.

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} \ell(b_i, h(a_i, x)) + r(x). \tag{1.2}$$

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 to operate with gradient or even higher derivatives to analyse the first(higher)-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 the optimization methods that could handle non-smooth objective. In general, such methods are slower than their convex analogues since subgradients does not allow to make as good approximation of function as gradients, but often, regularizers have a simple geometrical structure that allows to use 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 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 computing full gradient on every iteration some unbiased estimator (stochastic gradient) is computed. These methods appear to be slow out of the box as far as non-zero variance of stochastic gradients force to use decreasing stepsize that leads to the slower rate than for the standart full gradient methods. To figure it out in 2013 in [JZ13] stochastic variance reduced gradient method that allows to use constant stepsize but requires more gradient computations that gives all in all better convergence rate. There is a wide variety of incremental methods and we will make a short review in this section.

Another way to make 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[1]. Together with computational speedup these algorithms bring also a bottleneck: communication. When the dimension 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 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,

[1] we are not going to go into details of these algorithms now as far as it will be the specific section in this work

the structure of the final solution. Let us consider ℓ_1 regularized problem, so we a-priori know that the optimal solution is coordinate sparse. If we could know the set of coordinates that are non-zero in this solution we could send only this coordinates of gradient and lose nothing in terms of 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 specific structure. Unfortunately the exact pattern is unknown but proximal point methods allow to identify it [NSH19]. However, this result shed a light on the moment when the iterate becomes sparse it does not greenlight to switch to a projected mode. In contrast, the result of [FMP18] says that for some proximal algorithms after unknown time this projection technique would be legal. This motivates us to do this research.

1.2 Convexity and smoothness

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

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

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y), \tag{1.3}$$

where the domain of function f is defined as

$$\mathrm{dom} f = \{x : |f(x)| < \infty\}.$$

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

From this definition we could immediately get the following interpretation for smooth functions f that is more practical for further analysis.

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

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle. \tag{1.4}$$

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

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|_2^2. \tag{1.5}$$

Proof of Lemma 1.2. If f is a convex function then for any $\alpha \in [0, 1]$ we have

$$f(y) \geq f(x) + \frac{1}{1-\alpha}\left(f(\alpha x + (1-\alpha)y) - f(x)\right),$$

where the computation of limit for $\alpha \to 1$ will give (1.4).

Let us now prove the convexity of f form (1.4). Summing the following inequalities with multipliers α and $1 - \alpha$ correspondingly

$$f(\alpha x + (1-\alpha)y) \leq f(x) - \langle \nabla f(\alpha x + (1-\alpha)y), (1-\alpha)(x-y) \rangle = f(x) + (1-\alpha)\langle \nabla f(\alpha x + (1-\alpha)y), y - x \rangle$$

$$f(\alpha x + (1-\alpha)y) \leq f(y) - \langle \nabla f(\alpha x + (1-\alpha)y), \alpha(y-x) \rangle = f(x) - \alpha \langle \nabla f(\alpha x + (1-\alpha)y), y - x \rangle$$

we immediately get (1.3).

The proof of (1.5) immediately follows from the definition of the strongly convexity. □

Convexity implies that any stationary point is a global minima of f. In addition, the 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 the L-smoothness.

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

$$\|\nabla f(x) - \nabla f(y)\|_2 \le L\|x - y\|_2 \tag{1.6}$$

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

If function f is L-smooth then the following upper bound takes place.

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

$$f(y) - f(x) - \langle \nabla f(x), y - x\rangle \le \frac{L}{2}\|x - y\|_2^2 \tag{1.7}$$

and

$$\frac{1}{L}\|\nabla f(x) - \nabla f(y)\|_2^2 \le \langle \nabla f(x) - \nabla f(y), x - y\rangle \le L\|x - y\|_2^2. \tag{1.8}$$

Proof of Lemma 1.4. For any $x, y \in \mathbb{R}^n$ we have

$$\begin{aligned} f(y) - f(x) &= \int_0^1 \langle \nabla f(x + \alpha(y - x)), y - x\rangle d\alpha \\ &= \langle \nabla f(x), y - x\rangle + \int_0^1 \langle \nabla f(x + \alpha(y - x)) - \nabla f(x), y - x\rangle d\alpha \end{aligned} \tag{1.9}$$

Now, using convexity of f and Cauchy–Schwarz inequality we have

$$\begin{aligned} f(y) - f(x) - \langle \nabla f(x), y - x\rangle &= |f(y) - f(x) - \langle \nabla f(x), y - x\rangle| \\ &= \left| \int_0^1 \langle \nabla f(x + \alpha(y - x)) - \nabla f(x), y - x\rangle d\alpha \right| \\ &\le \int_0^1 |\langle \nabla f(x + \alpha(y - x)) - \nabla f(x), y - x\rangle| d\alpha \\ &\le \int_0^1 \|\nabla f(x + \alpha(y - x)) - \nabla f(x)\|_2 \|y - x\|_2 d\alpha \\ &\le \int_0^1 \tau L\|x - y\|_2^2 d\tau = \frac{L}{2}\|x - y\|_2^2. \end{aligned} \tag{1.10}$$

To prove the second part let us sum (1.10) for the pairs of points (x, y) and (y, x) and immediately get the upper bound. For the lower bound, let us define function $\varphi(x) = f(x) - \langle \nabla f(x'), x\rangle$ for some fixed $x' \in \mathbb{R}^n$. It is easy to see, that $\varphi(x)$ is L-smooth

$$\|\nabla \varphi(x) - \nabla \varphi(y)\|_2 = \|\nabla f(x) - \nabla f(x') - \nabla f(y) + \nabla f(x')\|_2 \le L\|x - y\|_2.$$

Since x' is a minimizer of $\varphi(x)$ we have

$$\varphi(x') \le \varphi(x - \frac{1}{L}\nabla \varphi(x)) \le \varphi(x) - \frac{1}{2L}\|\nabla \varphi(x)\|_2^2. \tag{1.11}$$

Now setting $x' = y$ we have the corresponding result. □

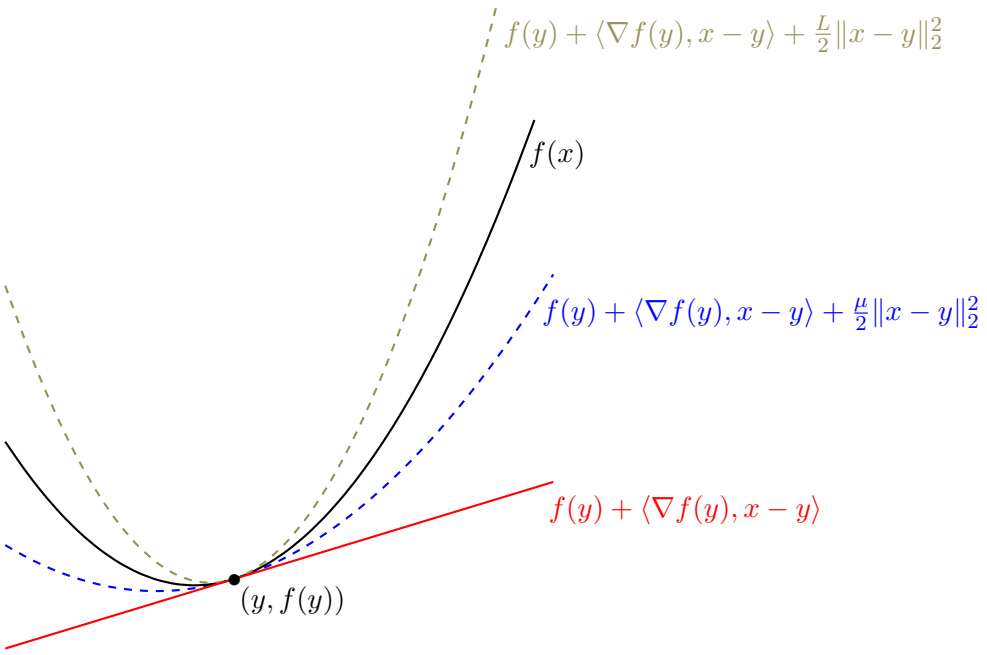

Figure 1-1. Graphical illustration of lower and upper bound for L-smooth and μ-strongly convex function $f : \mathbb{R} \to \mathbb{R}$

In Figure 1-1 we present the graphical illustration of lower (1.5) and upper (1.7) bounds for the L-smooth and μ-strongly convex objective function f. As we could see from this figure, the quadratic lower bound provided by (1.5) approximates the functional value much better than first-order approximation, that leads to the intuition that μ-strongly convex functions are easier to analyse. The quality of these approximations could be characterized by **condition number** of the problem

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

When this number is close to 1 (**well-conditioned**) the problem is extremely well approximated and when it is big (**ill-conditioned**) problems have a weak approximation. It impacts on the speed of optimization algorithms: better-conditioned problems are easier to solve.

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.

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

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \frac{\mu L}{\mu + L}\|x - y\|_2^2 + \frac{1}{\mu + L}\|\nabla f(x) - \nabla f(y)\|_2^2. \tag{1.12}$$

Proof of Lemma 1.5. Consider the case $\mu = L$ then from the μ-strong convexity of f we have

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

that implies

$$\langle \nabla f(y) - \nabla f(x), y - x \rangle \geq \mu\|y - x\|_2^2. \tag{1.13}$$

Now, using L-smoothness we have

$$\langle \nabla f(y) - \nabla f(x), y - x \rangle \geq \frac{\mu}{2}\|y - x\|_2^2 + \frac{\mu}{2L^2}\|\nabla f(x) - \nabla f(y)\|_2^2, \tag{1.14}$$

that proves the statement of the lemma.

Consider the case $L > \mu$. Denote by $\varphi(x) = f(x) - \frac{\mu}{2}\|x\|_2^2$, then $\nabla\varphi(x) = \nabla f(x) - \mu x$. This function is $L - \mu$-smooth, using Cauchy–Schwarz inequality and (1.8) we have

$$\langle \nabla\varphi(x) - \nabla\varphi(y), x - y\rangle \geq \frac{1}{L-\mu}\|\nabla\varphi(x) - \nabla\varphi(y)\|_2^2. \tag{1.15}$$

Now, substituting the expression for $\nabla\varphi$

$$\begin{aligned}\langle \nabla f(x) - \nabla f(y), x - y\rangle - \mu\|x-y\|_2^2 \geq \frac{1}{L-\mu}\left(\|\nabla f(x) - \nabla f(y)\|_2^2 + \mu^2\|x-y\|_2^2\right)\\ - \frac{2\mu}{L-\mu}\langle \nabla f(x) - \nabla f(y), x - y\rangle,\end{aligned} \tag{1.16}$$

that proves the result. □

1.3 Gradient descent

Let us consider the following optimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1.17}$$

where f is convex and smooth. One of the most important methods in the mathematical optimization to minimize (1.17) is a gradient descent method. This method was initially proposed in work of Cauchy [Cau47, Extrait 383] and becomes popular after [Pol63]. Thanks to its simplicity there are many extensions of it [Pol69b, Pol69a, Nes05, BT09].

Consider the ordinary differential equation (ODE)

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

then the values of $f(x)$ are decreasing along the trajectories of it [Cau47]. Another way to see that anti-gradient is a descent direction is the following. From the definition of the gradient we could have that

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

for any small vector. It implies, that in this first-order approximation the best direction to select is anti-gradient, that is exactly the idea of the gradient descent.

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 is convex and L-smooth, take $\gamma \in \left(0, \frac{2}{L}\right)$, then Algorithm 1 generates the sequence of poits $(x^k)_{k\geq 0}$ such that*

$$f(x^k) - f^\star \leq \frac{2(f(x^0) - f^\star)\|x^0 - x^\star\|_2^2}{2\|x^0 - x^\star\|_2^2 + \gamma k(2 - \gamma L)(f(x^0) - f^\star)}, \tag{1.18}$$

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

$$\|x^k - x^\star\|_2^2 \leq \left(1 - \frac{2\gamma\mu L}{\mu + L}\right)^k \|x^0 - x^\star\|_2^2, \tag{1.19}$$

111 *where* $x^\star$ *is a unique minimizer of* (1.17)*.*

Proof of Theorem 1.6. Let us start from the case when $\mu = 0$. Fix some $x^\star \in \operatorname{Argmin}_{x\in\mathbb{R}^n} f(x)$ and denote by $r^k = \|x^k - x^\star\|_2^2$. Then

$$\begin{aligned} r^{k+1} = \|x^k - x^\star - \gamma\nabla f(x^k)\|_2^2 = r^k - 2\gamma\langle\nabla f(x^k), x^k - x^\star\rangle + \gamma^2\|\nabla f(x^k)\|_2^2 \\ \leq r^k - \gamma\left(\frac{2}{L} - \gamma\right)\|\nabla f(x^k)\|_2^2, \end{aligned} \tag{1.20}$$

where in the last inequality we use L-smoothness and $\nabla f(x^\star) = 0$. Now, using (1.7) we get

$$f(x^{k+1}) \leq f(x^k) + \langle\nabla f(x^k), x^{k+1} - x^k\rangle + \frac{L}{2}\|x^{k+1} - x^k\|_2^2 = f(x^k) - \underbrace{\gamma\left(1 - \frac{\gamma L}{2}\right)}_{\omega}\|\nabla f(x^k)\|_2^2.$$

Denote by $\Delta^k = f(x^k) - f^\star$ then using the non-increasing property of $r^{k+1} \leq r^k$ (1.20) we have

$$\Delta^k \leq \langle\nabla f(x^k), x^k - x^\star\rangle \leq \sqrt{r^k}\|\nabla f(x^k)\|_2^2 \leq \sqrt{r^0}\|\nabla f(x^k)\|_2^2.$$

It implies that $\Delta^{k+1} \leq \Delta^k - \frac{\omega}{r^0}(\Delta^k)^2$ and as a result

$$\frac{1}{\Delta^{k+1}} \geq \frac{1}{\Delta^k} + \frac{\omega}{r^0}\frac{\Delta^k}{\Delta^{k+1}} \geq \frac{1}{\Delta^k} + \frac{\omega}{r^0},$$

where in the last inequality the non-increasing property of Δ^k is used. Summing up these inequalities from $k = 0$ we get

$$\frac{1}{\Delta^k} \geq \frac{1}{\Delta^0} + \frac{\omega}{r^0}(k+1).$$

Let us now assume that $\mu > 0$ using the same notation as in the first part

$$\begin{aligned} r^{k+1} = \|x^k - x^\star - \gamma\nabla f(x^k)\|_2^2 = r^k - 2\gamma\langle\nabla f(x^k), x^k - x^\star\rangle + \gamma^2\|\nabla f(x^k)\|_2^2 \\ \leq \left(1 - \frac{2\gamma\mu L}{\mu + L}\right) + \underbrace{\gamma\left(\gamma - \frac{2}{\mu + L}\right)}_{\leq 0}\|\nabla f(x^k)\|_2^2, \end{aligned} \tag{1.21}$$

112 where we use Lemma 1.5 and $\nabla f(x^\star) = 0$. □

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

$$f(x^k) - f^\star \leq \frac{2L(f(x^0) - f^\star)\|x^0 - x^\star\|_2^2}{2L\|x^0 - x^\star\|_2^2 + k(f(x^0) - f^\star)} \leq \frac{2L\|x^0 - x^\star\|_2^2}{k+4}, \tag{1.22}$$

where for the last inequality we used (1.7). For the μ-strongly convex objective function f the

optimal $\gamma = \frac{2}{\mu+L}$, that leads to the following rate

$$\|x^k - x^\star\|_2^2 \leq \left(1 - \frac{4\mu L}{(\mu+L)^2}\right)^k \|x^0 - x^\star\|_2^2 = \left(\frac{\kappa-1}{\kappa+1}\right)^{2k} \|x^0 - x^\star\|_2^2. \tag{1.23}$$

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.

1.4 Non-smooth optimization

Let us now consider the following optimization problem

$$\min_{x \in \mathbb{R}^n} r(x), \tag{1.24}$$

where r is convex but non-smooth. These functions are often assumed to be smooth almost everywhere on their domain since 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 as=re convex and differentiable.

Since r is non-smooth we could not calculate the gradient of it at any arbitrary point that leads to the other type of optimization methods for such methods. Let us start with the definition of the object that could replace gradients for non-smooth functions.

Definition 1.7 (Subgradient)**.** *Consinder convex function r. Vector g is called subgradient of r at point $y \in \text{dom}r$ if for any $x \in \text{dom}r$ holds*

$$r(x) \geq r(y) + \langle g, x - y \rangle. \tag{1.25}$$

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

Now we are ready to present one of the basic algorithm [Nes13, Chapter 3] to solve (1.24).

Algorithm 2 Subgradient Descent

Initialize $x^0 \in \mathbb{R}^n$
Select the sequence of stepsizes γ^k such that $\gamma^k > 0,\ \gamma^k \to 0,\ \sum_0^\infty \gamma^k = \infty$
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

As we could see from the definition, the norm of subgradient could be different, that is why in the algorithm used the normalized version. It is clear, that this method is worse than the gradient descent, as far as 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.

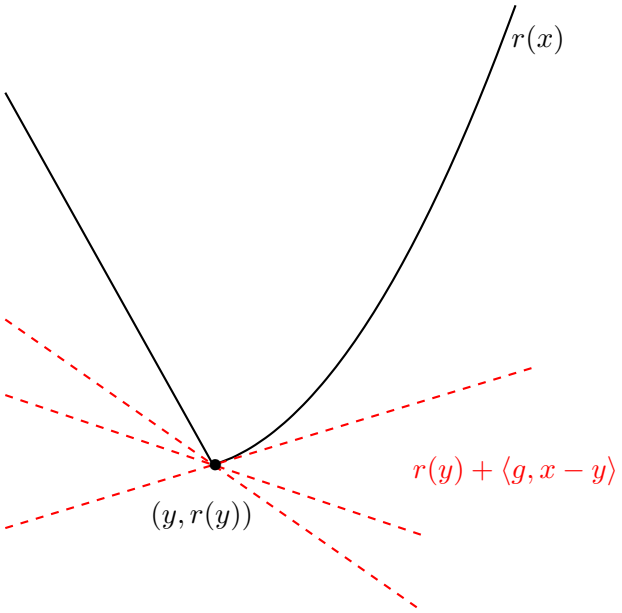

Figure 1-2. Linear approximations of non-smooth function $r = \max\left\{-2x, (0.2x+1)^2 - 1\right\}$ from $\mathbb{R}$ to $\mathbb{R}$ at point $y = 0$ with different subgradients. As we could see, $y = 0$ is a minimizer of r, however its subdifferential in this point is $\partial_0 r = [-2, 0.4] \ni 0$. Depending on the subgradient g (we plot for $g = -1, -0.5, 0.4$) different linear approximations of function r in 0 appears and, as a result, the "descent" step is the descent step for these approximations but not for r.

Proximal methods

The subgradient descent method is not the only method to solve non-smooth problems. Let us present the class of methods called proximal methods and, first, let us recall the definition of a proximal operator.

Definition 1.8 (Proximal operator)**.** *Given a convex function $r : \mathbb{R}^n \to \mathbb{R}$, the proximal operator of r is the function*

$$\mathbf{prox}_r(x) = \operatorname*{argmin}_{y \in \mathbb{R}^n} \left\{ r(y) + \frac{1}{2}\|x - y\|_2^2 \right\}. \tag{1.26}$$

Since r is convex, the objective of argmin is strongly convex that means the proximal operator of any point is well defined and unique.

In machine learning applications many regularizers are relatively simple that makes the computation of proximal operator cheap. More precisely, a proximal operator for such problems usually has a closed-form solution (ℓ_1, Group-Lasso) or could be computed iteratively (TV). Moreover, if r is the indicator function of a convex set, 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 (1.24) that shows the interest of the proximal operator [PB+14, Section 2.3].

Lemma 1.9. *Point $x^\star$ is a solution of (1.24) if and only if $x^\star = \mathbf{prox}_r(x^\star)$.*

Proof of Lemma 1.9. Let $x^\star$ be a minimizer of r then

$$r(x) + \frac{1}{2}\|x - x^\star\|_2^2 \geq r(x) \geq r(x^\star) = r(x^\star) + \frac{1}{2}\|x^\star - x^\star\|_2^2,$$

that implies $x^\star = \mathbf{prox}_r(x^\star)$.

Let $x^\star = \mathbf{prox}_r(x^\star)$, using optimality condition we have

$$0 \in \partial r(y) + (y - x)$$

for any $x \in \mathbb{R}^n$ and $y = \mathbf{prox}_r(x)$. Taking $x = y = x^\star$ we have $0 \in \partial r(x^\star)$. □

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

Lemma 1.10 (Firm nonexpansiveness of proximal operator)**.** *Consider convex function $r : \mathbb{R}^n \to \mathbb{R}$ then for any $x, y \in \mathbb{R}^n$ holds*

$$\|\mathbf{prox}_r(x) - \mathbf{prox}_r(y)\|_2^2 \le \langle x - y, \mathbf{prox}_r(x) - \mathbf{prox}_r(y)\rangle. \tag{1.27}$$

Proof of Lemma 1.10. Let $x' = \mathbf{prox}_r(x)$ and $y' = \mathbf{prox}_r(y)$, then

$$x - x' \in \partial r(x') \qquad \text{and} \qquad y - y' \in \partial r(y').$$

Thus, by the definition of subdifferential we have

$$r(y') \ge r(x') + \langle x - x', y' - x'\rangle \qquad \text{and} \qquad r(x') \ge r(y') + \langle y - y', x' - y'\rangle.$$

Summing up these inequalities we get the statement of the lemma. □

Finally, let us introduce the connection between proximal operator and subdifferential of function r [PB+14].

Lemma 1.11 (Resolvent)**.** *The proximal operator $\mathbf{prox}_{\gamma r}$ and the subdifferential ∂r are related as follows:*

$$\mathbf{prox}_{\gamma r} = (I + \gamma \partial r)^{-1}, \tag{1.28}$$

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

Proof of Lemma 1.11. By the definition if $y \in (I + \gamma\partial r)^{-1}(x)$, Then

$$x \in (I + \gamma\partial r)(y) = y + \partial r(y).$$

It could be rewritten as

$$0 \in \partial r(y) + \frac{1}{\gamma}(y - x) = \partial_y \left(r(y) + \frac{1}{2\gamma}\|x - y\|_2^2 \right).$$

This is the necessary and sufficient condition for

$$y = \operatorname*{argmin}_z \left\{ r(z) + \frac{1}{2\gamma}\|z - x\|_2^2 \right\},$$

that proves the result. □

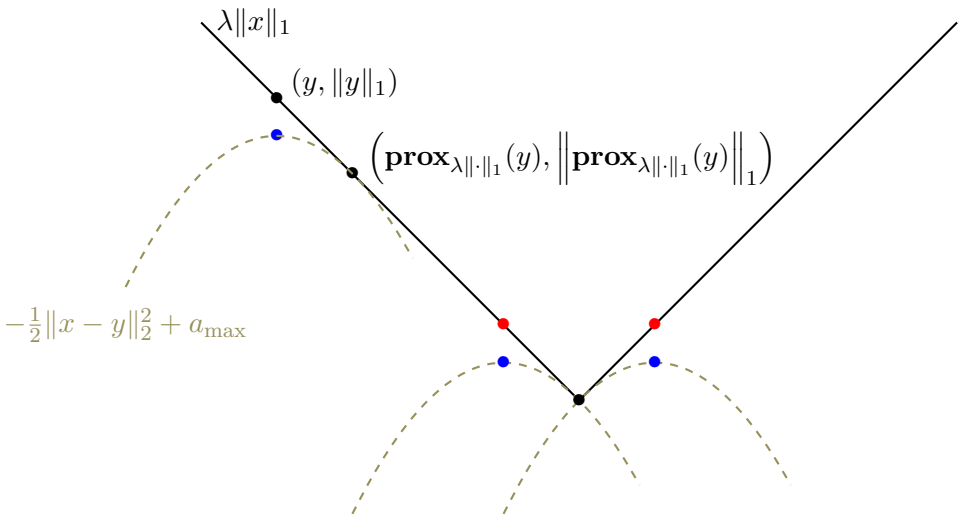

Figure 1-3. Geometrical interpretation of soft-thresholding operator

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

This algorithm converges to the minimizer if this minimizier exists (see for example Theorem 23.41 [BC11]).

1.4.1 Moreau-Yosida regularization

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

Definition 1.12. *Given $\lambda > 0$, the Moreau envelope $M_{\lambda r}$ of the function r with parameter λ is defined as*

$$M_{\lambda r}(y) = \inf_x \left(r(x) + \frac{1}{2\lambda}\|x-y\|_2^2\right). \tag{1.29}$$

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

$$\nabla M_{\lambda r} = \frac{1}{\lambda}(x - \mathbf{prox}_{\lambda r}(x)). \tag{1.30}$$

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

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

which shows that proximal operator could 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 Moreau envelope has the same minimizers as r we have the convergence of proximal point method. In Figure 1-4 we present a geometrical interpretation of Moreau envelope for the non-smooth objective function r.

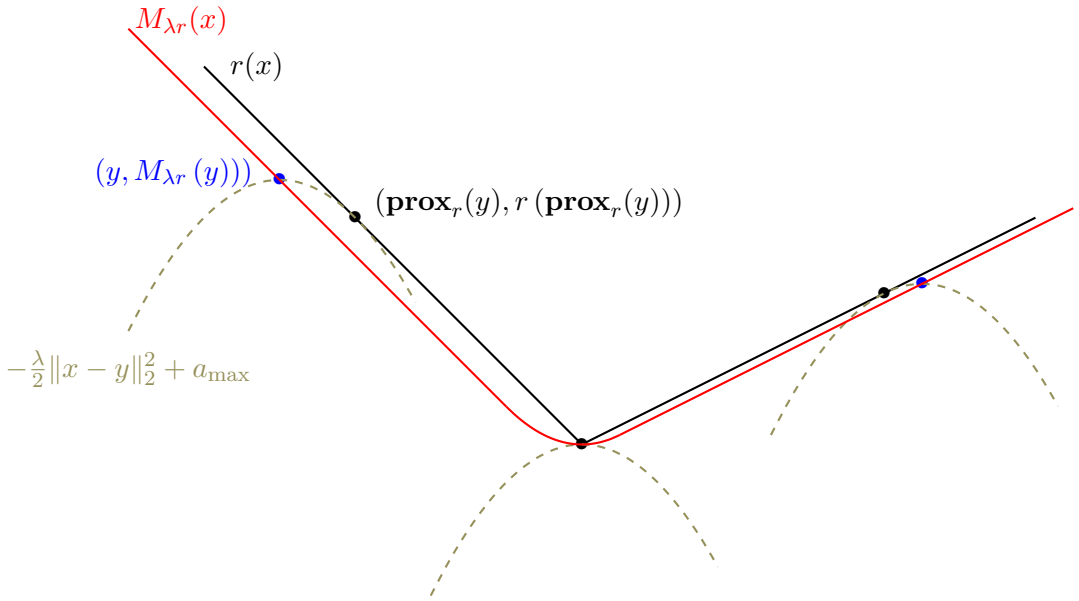

Figure 1-4. Geometrical interpretation of Moreau envelope for $r = \max\{-x, 0.5x\}$ from $\mathbb{R}$ to $\mathbb{R}$.

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 (1.30) we have that

$$L_{M_{\lambda r}} = \lambda^{-1} \tag{1.31}$$

independent on the smoothness parameter of r. However, Moreau-Yosida regularization is strongly convex if the objective function r is strongly convex, moreover, their strongly convexity constants relates as

$$\mu_{M_{\lambda r}} = \frac{\mu\lambda^{-1}}{\mu + \lambda^{-1}}, \tag{1.32}$$

where μ is the strongly convexity constant of r (see Theorem 2.2 of [LS97] for the proof). Thus, the condition number of this Moreau-Yosida regularization of μ-strongly convex function r is

$$\kappa_{M_{\lambda r}} = \frac{L_{M_{\lambda r}}}{\mu_{M_{\lambda r}}} = \frac{\mu + \lambda^{-1}}{\mu}. \tag{1.33}$$

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

1.4.2 Composite optimization

Let us consider composite optimization problem

$$\min_{x\in\mathbb{R}^n} f(x) + r(x), \tag{1.34}$$

where f is smooth and convex and r is convex, non-smooth. Moreover, we consider r be a prox-easy function meaning that its proximal operator is easy to compute (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 alternatively the gradient step and the proximal one. 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

At every iterate of this method, additionally to the step of gradient descent algorithm performs an additional proximal step on top of it that could be reformulated as

$$x^{k+1} = \operatorname*{argmin}_{z\in\mathbb{R}^n} \left\{ \underbrace{f(x^k) + \langle \nabla f(x^k), z - x^k\rangle + \frac{1}{2\gamma}\|x^k - z\|_2^2}_{\hat{f}(z,x^k)} + r(z) \right\}. \tag{1.35}$$

If $\gamma \leq \frac{1}{L}$, where L is a 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 [Mai15].

From this form it is easy to see that the stationary point of this algorithm is a minimizer of (1.34). Since objective in (1.35) 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 (1.34).

When r is the indicator of a convex set, the proximal operator projects the gradient step back to the constrained set, that yields to the generalization of the projected gradient method. 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 have a big history since the idea of simplifying the gradient descent update as old as a gradient descent scheme. 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 features vector. 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 separable structure of the function.

The idea of the basic 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 (1.17), where function f has coordinate-wise Lipschitz continuous gradient

$$|\nabla_{[i]} f(x + \Delta e_i) - \nabla_{[i]} f(x)| \leq M|h|,$$

where subscript $[i]$ means the i-th coordinate, e_i is a i-th standart basis vector, and $h \in \mathbb{R}$. For

Algorithm 5 Coordinate Descent (CD) for (1.17)

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

such function it is easy to see, that

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

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 the full gradient computation for coordinate selection. Moreover, it could happen that $M \gg L$ where L is a smoothness constant of f that makes algorithm worse 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. Moreover, in such algorithms instead of selecting only one coordinate the block of coordinates is selected every time. Thus in practice, it is more popular to use a cyclic or randomized selection of coordinate to be updated.

Now, let us consider composite optimization problem (1.34), 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) separaboility of proximal operator

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

The key idea of coordinate descent method for (1.34) is to make a majorization-minimization algorithm where in (1.35) instead of full upper bound $\hat{f}(z, x^k)$ the coordinate one used

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

The simplest version of the coordinate descent for (1.34) is presented in Algorithm 6 (see Theorem 1 of [RT12] for the proof).

Algorithm 6 Coordinate Descent (CD) for (1.34)

Initialize $x^0 \in \mathbb{R}^n$
for $k \geq 0$ **do**
  Select coordinate i^k i.i.d.
  $x^{k+1}_{[i^k]} \leftarrow \mathbf{prox}_{\gamma r_{i^k}}\left(x^k_{[i^k]} - \gamma \nabla_{[i^k]} f(x^k)\right)$ 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 (1.2). This calls for some modern modifications that allow to use non-separable proximal term [HMR18].

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

Incremental methods

In the big data era, both 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 gradient and coordinate (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 independent on m. Let us recall that in machine learning the usual problem is ERM (1.1), where the objective function f has a sum structure (probably infinite).

In case of finite sum

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

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

219 $\nabla f(x)$

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

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

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 $1, \ldots, m$. 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

$$\mathbb{E}\left[\|g^k - \nabla f(x^k)\|_2^2 | x^k\right]. \tag{1.36}$$

220 This variance is usually non-zero for any x^k and even in the optimal point $x^\star$. For example,
221 the full gradient $\nabla f(x^\star) = 0$ but there is no reason for $\nabla f_i(x^\star)$ to vanish. Consider Algorithm
222 7 with fixed stepsize $\gamma^k = \gamma$. Since, for any point x^k the noise (1.36) is separated from 0 the
223 point $x^\star$ is not fixed point of the algoithm. This again (like for subgradient descent) leads to the
224 slower rate than for the gradient descent, see the usual behaviour of convergence in Figure 1-5.

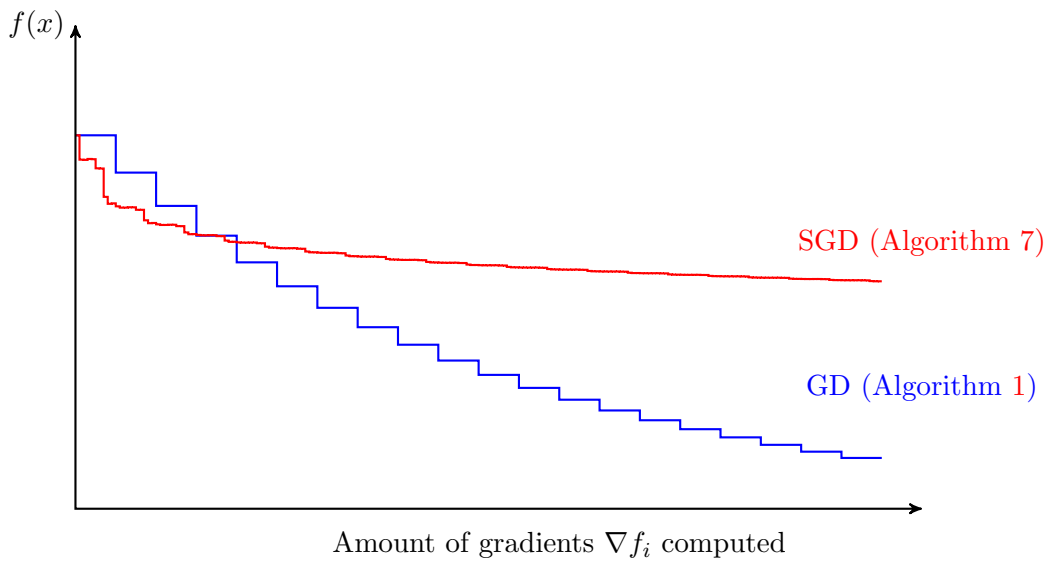

Figure 1-5. Evolution of iterates for GD with fixed stepsize $\gamma = \frac{2}{\mu+L}$ and SGD with decreasing stepsize $\gamma^k = \frac{1}{k}$ for L-smooth and μ-strongly convex objective $f(x) = \|Ax - b\|_2^2$, with random generated $A \in \mathbb{R}^{100\times 100}, b \in \mathbb{R}^{100}$. As 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.

225 In [Sol98, NB01] authors propose SGD with a constant stepsize. In Proposition 2.4 [NB01] it
226 was proven that such algorithm has a convergence rate that could be split into two parts: first is
227 dependent on the number of iteration and converges linearly (for the strongly convex objective)
228 to 0, however 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 sum 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 to 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 [KHR19, HLLJM15]. Another class of methods is majorization-minimization algorithms Finito [DDC14], MISO [Mai15], DAve-RPG (asynchronous version of MISO) [MIM18]. The majorization-minimization approach goes beyond optimization that provides a good start for designing new incremental methods [QSMR19].

1.5 Useful tools for stochastic processes analysis.

In this section, we recall some definitions and results from the probability theory that we will use in our further proofs.

Let us start from the law of total expectation [KOL33, Chapter V, Section 4].

Lemma 1.13 (Tower property). *For any random variables X and Y, we have $\mathbb{E}[X] = \mathbb{E}\left[\mathbb{E}[X|Y]\right]$.*

Proof of Lemma 1.13. Let us provide the proof for discrete random variables only.

$$\begin{aligned}\mathbb{E}[X] &= \sum_x xP(X=x) = \sum_x x \sum_y P(X=x)\&P(Y=y) = \sum_x \sum_y xP(X=x)\&P(Y=y) \\ &= \sum_x \sum_y P(Y=y)x\frac{P(X=x\&Y=y)}{P(Y=y)} = \sum_y P(Y=y)\sum_x xP(X=x\&Y=y) \\ &= \sum_y P(Y=y)\mathbb{E}[X|Y=y] = \mathbb{E}\left[\mathbb{E}[X|Y]\right], \end{aligned} \tag{1.37}$$

which finishes the proof. □

Now, let us present the notion of almost surely convergence [Sto74].

Definition 1.14 (Almost sure convergence)**.** *We say that the sequence $\{X_t\}$ of random variables $X_t : \Omega \to \mathcal{S}$ converges almost surely (a.s.) to $X \in \mathcal{S}$ if*

$$\mathbf{P}\left[\omega \in \Omega : \lim_{n\to\infty} X_n(s) = X\right] = 1.$$

The following theorem allows proving almost sure convergence for almost supermartingales [RS71, Theorem 1].

Theorem 1.15. *Consider probability space $(\Omega, \mathcal{F}, \mathcal{P})$ with the sequence of σ-algebras $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \ldots$. For each $n = 1, 2, \ldots$ let z_n, β_n, ξ_n, and ζ_n be non-negative $\mathcal{F}_n$-measurable random variables such that*

$$\mathbb{E}[z_{n+1}|\mathcal{F}_n] \leq z_n(1+\beta_n) + \xi_n - \zeta_n. \tag{1.38}$$

Assume that $\sum_{i=1}^{\infty} \beta_i < \infty$ and $\sum_{i=1}^{\infty} \xi_i < \infty$ then $\lim_{n\to\infty} z_n$ exist and is finite and $\sum_{i=1}^{\infty} \zeta_i < \infty$ a.s.

1.6 Identification

As we mentioned, the use of a proximity 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 of constrained convex optimization (see e.g. [Wri93]) and nonsmooth optimization (see e.g. [HL04]).

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, FGMP18, SJNS19].

The knowledge of the optimal substructure would allow to 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]), it is usually unknown beforehand and proximal algorithms can be exploited to obtain approximations of this substructure. 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.

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 $\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_m\}$ 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[2] vector $\mathsf{S}_{\mathcal{M}}(x) \in \{0,1\}^m$ verifying*

$$(\mathsf{S}_{\mathcal{M}}(x))_{[i]} = 0 \quad \text{if } x \in \mathcal{M}_i \text{ and } 1 \text{ elsewhere.} \tag{1.39}$$

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 for almost surely converging proximal-based algorithms as follows. This very simple result is inspired from 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 $x^k = \mathbf{prox}_{\gamma g}(u^k)$ and $x^\star = \mathbf{prox}_{\gamma g}(u^\star)$. Then (x^k) identifies some subspaces with probability one; more precisely for any $\varepsilon > 0$, with probability one, after some finite time,*

$$\mathsf{S}_{\mathcal{M}}(x^\star) \;\leq\; \mathsf{S}_{\mathcal{M}}(x^k) \;\leq \bigcup_{u \in \mathcal{B}(u^\star, \varepsilon)} \mathsf{S}_{\mathcal{M}}(\mathbf{prox}_{\gamma g}(u)). \tag{1.40}$$

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 second inequality.

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 $x^\star$. □

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.

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*

$$r(x) = \sum_{B \in \mathcal{B}} \|x^B\|_2, \tag{1.41}$$

[2] 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.

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

$$\operatorname{supp}(x) \stackrel{\triangle}{=} \left\{ i \in [1,n] \,|\, x_{[i]} \neq 0 \right\}. \tag{1.42}$$

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 (1.40) can be reformulated as

$$\operatorname{supp}(x^\star) \subseteq \operatorname{supp}(x^k) \subseteq \bigcup_{u \in \mathcal{B}(u^\star, \varepsilon)} \operatorname{supp}(\mathbf{prox}_{\gamma r}(u)). \tag{1.43}$$

To prove this it is enough to show that

$$\mathsf{S}_{\mathcal{M}}(x) \leq \mathsf{S}_{\mathcal{M}}(y) \iff \operatorname{supp}(x) \subseteq \operatorname{supp}(y). \tag{1.44}$$

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 (1.44).

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.

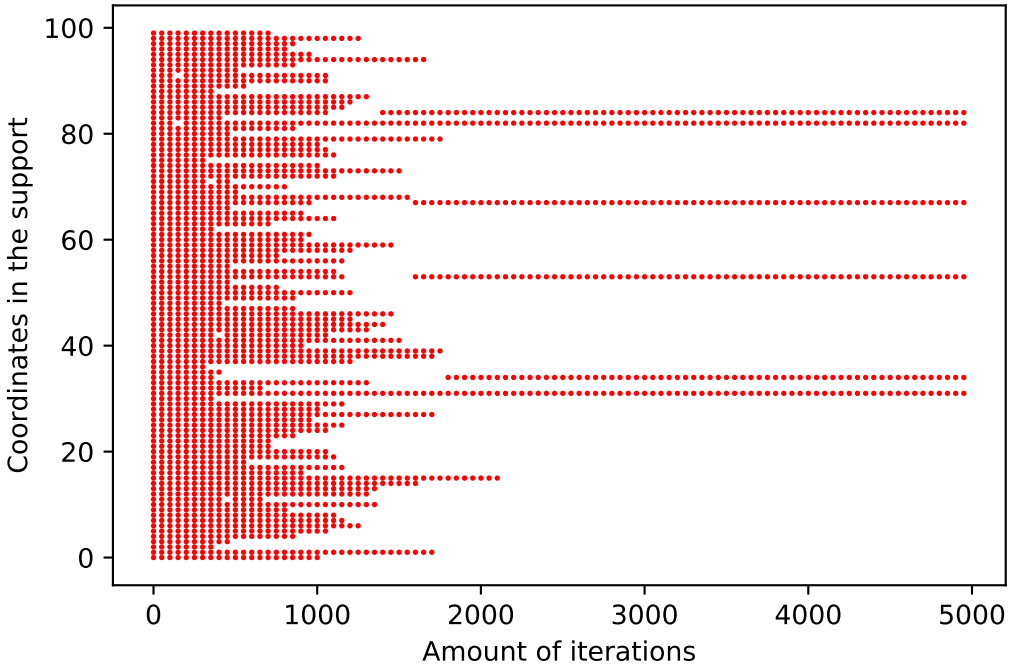

Figure 1-6. ℓ_1 identification

In Figure 1-6 we can see how support changes during proximal gradient descent for LASSO problem

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

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 6% 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 2000.

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*

$$r(x) = \sum_{i=1}^{n-1} |x_{[i]} - x_{[i+1]}| \tag{1.45}$$

it enforces variation sparsity.

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

$$\text{jumps}(x) \stackrel{\triangle}{=} \left\{ i \in [2, n] \,|\, x_{[i]} \neq x_{[i-1]} \right\}. \tag{1.46}$$

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. $\text{jumps}(x) = \text{jumps}(y)$ *for all $x, y \in \mathcal{M}_i$. In case of this collection, identification result* (1.40) *can be reformulated as*

$$\text{jumps}(x^\star) \subseteq \text{jumps}(x^k) \subseteq \bigcup_{u\in\mathcal{B}(u^\star,\varepsilon)} \text{jumps}(\mathbf{prox}_{\gamma r}(u)). \tag{1.47}$$

To prove this it is enough to show that

$$\mathsf{S}_{\mathcal{M}}(x) \leq \mathsf{S}_{\mathcal{M}}(y) \iff \text{jumps}(x) \subseteq \text{jumps}(y), \tag{1.48}$$

which proves exactly the same way as (1.44).

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 $(x^\star, u^\star)$.

Chapter 2

Automatic dimension reduction

2.1 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. This type of problem appears extensively in signal processing and machine learning applications; we refer to e.g. [CWB08], [CP11], [BJM+12], among a vast literature. Large scale applications in these fields call for first-order optimization, such as proximal gradient methods (see e.g. the recent [Teb18]) and coordinate descent algorithms (see e.g. the review [Wri15]).

As we already mentioned in Section 1.6 proximal methods could 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. 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 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, RT16]. 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, [PCJ17, NSYD17, SRJ17] propose adaptive coordinate descent methods based on the coordinate-wise Lipschitz constants and current values of the gradient. The methods proposed in the present paper, even when specialized in the coordinate descent case, are the first ones where the *iterate structure enforced by a non-smooth regularizer* is used to adapt the selection probabilities.

Outline of the chapter. In Section 2.2, 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 2.3, 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 2.4, we report numerical experiments on popular learning problems to illustrate the merits and reach of the proposed methods.

2.2 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 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 2.2.1, we introduce our subspace selection procedure. We build on it to introduce, in Section 2.2.2, our first subspace descent algorithm, the convergence of which is analyzed in Section 2.2.3. Finally, we put this algorithm into perspective in Section 2.2.4 by connecting and comparing it to related work.

2.2.1 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 2.3.1 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.

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 $i = 1, .., n$ 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.

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*

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

[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 $\{x + y : x \in \mathcal{C}, y \in \mathcal{D}\} \subseteq \mathbb{R}^n$.

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 $\lambda_{\max}(\mathsf{P}) \leq 1$. □

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. 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* $\leq \lambda_{\max}(\mathsf{P}) = \max_i p_i$.

2.2.2 Random Subspace Proximal Gradient Algorithm

An iteration of the proximal gradient algorithm (Algorithm 4) decomposes in two steps (sometimes called "forward" and "backward"):

$$z^k = x^k - \gamma \nabla f(x^k) \tag{2.3a}$$
$$x^{k+1} = \mathbf{prox}_{\gamma r}(z^k) \tag{2.3b}$$

where the proximal operator is well-defined when r is a proper, lower semi-continuous convex function [BC11, Def. 12.23]. Furthermore, it is computationally cheap to compute in several cases, either from a closed form (e.g. for ℓ_1-norm, ℓ_1/ℓ_2-norm, see among others [CP08] and references therein), or by an efficient procedure (e.g. for the 1D-total variation, projection on the simplex, see [YLY11, Con13]).

In order to construct a "subspace" version of the proximal gradient (2.3), one has to determine which variable will be updated along the randomly chosen subspace (which we will call a projected update). Three choices are possible:

(a) a projected update of x^k, i.e. projecting after the proximity operation;

(b) a projected update of $\nabla f(x^k)$, i.e. projecting after the gradient;

(c) a projected update of z^k, i.e. projecting after the gradient *step*.

Choice (a) 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. Choice (b) is considered recently in [HMR18] in the slightly different context of sketching. A further discussion on related literature is postponed to Section 2.2.4.

In this paper, we will consider Choice (c), inspired by recent works highlighting that combining iterates usually works well in practice (see [MIM18] and references therein). However, taking gradient steps along random subspaces introduce 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 8, replaces (2.3a) by

$$y^k = \mathsf{Q}\left(x^k - \gamma \nabla f\left(x^k\right)\right) \qquad \text{and} \qquad z^k = P_{\mathfrak{S}^k}\left(y^k\right) + \left(I - P_{\mathfrak{S}^k}\right)\left(z^{k-1}\right). \tag{2.4}$$

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 (2.3b) is performed after getting back to the original space (by multiplication with Q^{-1}):

$$x^{k+1} = \mathbf{prox}_{\gamma r}\left(\mathsf{Q}^{-1}\left(z^k\right)\right). \tag{2.5}$$

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. Apart from the algorithm of [HMR18] in a different setting, this is an uncommon but highly desirable feature to tackle general composite optimization problems.

Algorithm 8 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 $k = 1, \dots$ **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

Let us provide a first example, before moving to the analysis of the algorithm in the 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 $f \circ \left(\mathsf{Q}^{-1}\right)$ 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).

2.2.3 Analysis and convergence rate

In this section, we provide a theoretical analysis for `RPSD`, showing linear convergence for strongly convex objectives. Tackling the non-strongly convex case requires extra-technicalities; we thus choose to postpone the corresponding convergence result to the appendix for clarity.

Assumption 2.8 (On the optimization problem)**.** *The function f is L-smooth and μ-strongly convex and the function g 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 $\mathfrak{S}^1, \mathfrak{S}^2, .., \mathfrak{S}^k$ 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 the 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

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

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

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

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 1.10) along with the fact that as $x^\star$ is a minimizer of (2.1) so $x^\star = \mathbf{prox}_{\gamma r}(x^\star - \gamma\nabla f(x^\star)) = \mathbf{prox}_{\gamma r}(\mathsf{Q}^{-1}z^\star)$ [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 minimizers 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 $x^\star$. □

2.2.4 Examples and Connections with the existing work

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

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

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 $\mathsf{P} = \mathrm{diag}([p_1, .., p_n])$, $P_{\mathfrak{S}^k}$ 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 8 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 rates 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 $\gamma = 2/(\mu + L)$.

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

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

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

which boils down to the usual (proximal) coordinate descent algorithm (see Algorithm 6), that recently knew 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 2.2.4. 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

$$r(x) = \sum_{i=2}^{n} |x_{[i]} - x_{[i-1]}| \tag{2.6}$$

featured for instance in the fused lasso problem [TSR+05]. 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 following covering family

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

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

$$P_{\mathfrak{S}} = \begin{pmatrix} \overbrace{\frac{1}{n_1} & \dots & \frac{1}{n_1}}^{n_1} & 0 & \dots & \overbrace{\dots & \dots & 0}^{n-n_s} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \frac{1}{n_1} & \dots & \frac{1}{n_1} & 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 & \frac{1}{n-n_s} & \dots & \frac{1}{n-n_s} \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & 0 & \frac{1}{n-n_s} & \dots & \frac{1}{n-n_s} \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \end{matrix}}\right\} n_1 \\ \\ \\ \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \end{matrix}}\right\} n - n_s \end{matrix} \tag{2.7}$$

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 2.3.3, 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 2.2.2).

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 2.4.2.

2.3 Adaptive subspace descent

This section presents an extension of our randomized subspace descent algorithm where 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 ones where the iterate structure enforced by a nonsmooth 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 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 2.3.1 a generic algorithm with varying selections and establish its convergence. Second, in Section 2.3.2, we provide a simple general identification result. We then combine these two results to provide an efficient adaptive method in Section 2.3.3.

2.3.1 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 2.3.3): (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 9. 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 9 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*

$$(1 - \beta_\ell) := (1 - \alpha_{\ell-1}) \mathbf{a}_\ell^{1/\mathbf{c}_\ell}. \tag{2.8}$$

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.

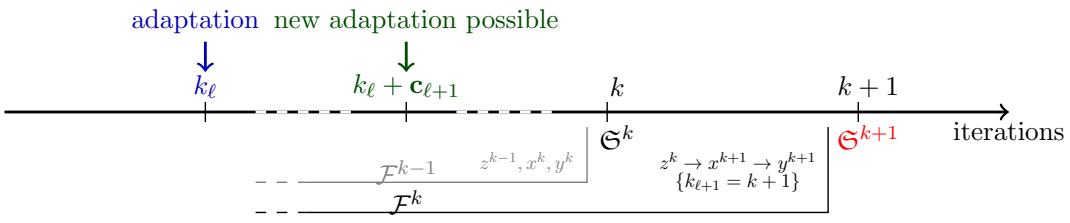

Figure 2-1. Summary of notations about iteration, adaptation and filtration. The filtration $\mathcal{F}^{k-1}$ is the sigma-algebra generated by $\{\mathfrak{S}^\ell\}_{\ell \leq k-1}$ encompassing the knowledge of all variables up to y^k (but not z^k).

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 2.2.3. 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 $k \in [k_\ell, k_{\ell+1})$

$$\mathbb{E}\left[\|z^k - z_\ell^\star\|_2^2 \,|\, \mathcal{F}^{k-1}\right] \leq \underbrace{\left(1 - \frac{2\gamma\mu L \lambda_{\min}(\mathsf{P}_\ell)}{\mu + L}\right)}_{\leq 1-\alpha_\ell} \|z^{k-1} - z_\ell^\star\|_2^2. \tag{2.9}$$

Since the distribution of the selection has not changed since k_ℓ, iterating (2.9) leads to

$$\mathbb{E}\left[\|z^k - z_\ell^\star\|_2^2 \,|\, \mathcal{F}^{k_\ell-1}\right] \leq (1-\alpha_\ell)^{k-k_\ell} \|z^{k_\ell-1} - z_\ell^\star\|_2^2. \tag{2.10}$$

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

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

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

$$\begin{aligned}
\mathbb{E}\left[\|z^{k_\ell-1} - z_\ell^\star\|_2^2 \,|\, \mathcal{F}^{k_{\ell-1}}\right] &\leq \mathbf{a}_\ell(1-\alpha_{\ell-1})^{k_\ell - k_{\ell-1}} \|z^{k_{\ell-1}} - z_{\ell-1}^\star\|_2^2 \\
&\leq \mathbf{a}_\ell(1-\alpha_{\ell-1})^{\mathbf{c}_\ell} \|z^{k_{\ell-1}} - z_{\ell-1}^\star\|_2^2,
\end{aligned} \tag{2.12}$$

using the assumption of bounded inter-adaptation times. Combining this inequality and (2.10), 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 (2.8), 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:

$$\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\|_2^2 \leq \mathbf{a} \qquad \lambda_{\min}(\mathsf{P}_\ell) \geq \lambda. \tag{2.13}$$

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

$$\mathbf{c} = \lceil \log(\mathbf{a}) / \log\left((2-\alpha)/(2-2\alpha)\right) \rceil \tag{2.14}$$

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 (2.13) 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. (2.14))*

On Figure 2-2, we observe that adapting every iteration leads to a 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.

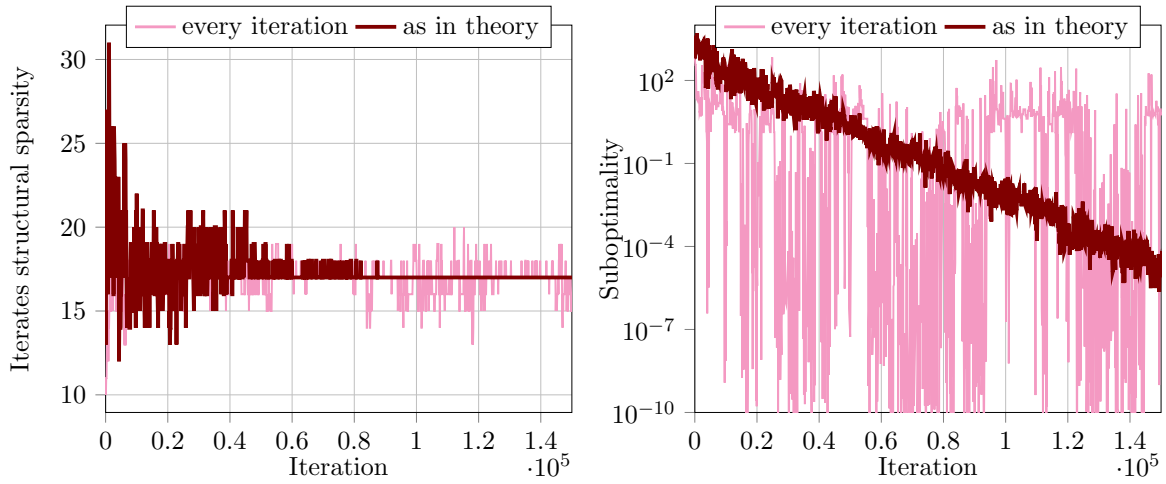

Figure 2-2. Comparisons between theoretical and harsh updating time for `ARPSD`.

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 on the problem or if the adaptation cost varies a lot. This drawback can be circumvented to the price of loosing 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 $k \in [k_\ell, k_{\ell+1})$

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

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 a special attention. In (2.11), 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. (2.9), 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. □

2.3.2 Identification of proximal algorithms

As we mentioned in Section 1.6 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 (1.40) 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 solution $\mathsf{S}_{\mathcal{M}}(x^\star)$. 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*[2]

$$\mathsf{S}_{\mathcal{M}}(x^\star) \;=\; \bigcup_{u\in\mathcal{B}(x^\star-\gamma\nabla f(x^\star),\varepsilon)} \mathsf{S}_{\mathcal{M}}(\mathbf{prox}_{\gamma g}(u)) \tag{QC}$$

for any $\varepsilon > 0$ small enough, then, using an adaptation deterministically computed from $(\mathsf{S}_{\mathcal{M}}(x^k))$, we have

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

where $\mathsf{P}^\star$ is the average projection matrix of the selection associated with $\mathsf{S}_{\mathcal{M}}(x^\star)$.

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. Now we go back to the proof of Theorem 2.17 to see that the random variable defined by

$$X^k = \begin{cases} x^{k_\ell} & \text{if } k \in (k_\ell, k_\ell + \mathbf{c}_\ell] \\ x^k & \text{if } k \in (k_\ell + \mathbf{c}_\ell, k_{\ell+1}] \end{cases} \quad \text{for some } \ell$$

also converges almost surely to $x^\star$. Intuitively, this sequence is a replica of (x^k) except that it stays fixed at the beginning of adaptation cycles when no adaptation is admitted. This means that $\mathsf{S}_{\mathcal{M}}(X^k)$ which can be used for adapting the selection will exactly reach $\mathsf{S}_{\mathcal{M}}(x^\star)$ in finite time. From that point on, since we use an adaptation technique that deterministically relies on $\mathsf{S}_{\mathcal{M}}(x^k)$, there are no more adaptations and thus the rate matches the non-adaptive one of Theorem 2.10. □

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 $\lambda_{\min}(\mathsf{P})$ 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] 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 $g(x) = \lambda_1\|x\|_1$, 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 (2.6), 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. In general, this corresponds to the relative interior assumption of [Lew02]; see the extensive discussion of [VGFP15].

2.3.3 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 a good behavior along the convergence. These two aspects are discussed in Section 2.3.3 and Section 2.3.3, 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 2.2.4)*

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

a complemented identification set is

$$\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_n\} \qquad \text{with } \mathcal{M}_i = \{x \in \mathbb{R}^n : x_{[i]} = 0\}, \tag{2.16}$$

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 2.2.4)

$$\mathcal{C} = \{\mathcal{C}_1, .., \mathcal{C}_{n-1}\} \qquad \text{with } \mathcal{C}_i = \left\{x \in \mathbb{R}^n : x_{[j]} = x_{[j+1]} \text{ for all } j \neq i\right\} \tag{2.17}$$

a complemented identification set is

$$\mathcal{M} = \{\mathcal{M}_1, .., \mathcal{M}_{n-1}\} \qquad \text{with } \mathcal{M}_i = \left\{x \in \mathbb{R}^n : x_{[i]} = x_{[i-1]}\right\}, \tag{2.18}$$

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.*

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 a certain 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 on 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 2.3.3 and illustrated numerically in Section 2.4.

		(non-adaptive) subspace descent `RPSD`	adaptive subspace descent `ARPSD`
Subspace family		$\mathcal{C} = \{\mathcal{C}_1, .., \mathcal{C}_c\}$	
Algorithm		$\begin{cases} y^k = & \mathsf{Q}\left(x^k - \gamma \nabla f\left(x^k\right)\right) \\ z^k = & P_{\mathfrak{S}^k}\left(y^k\right) + \left(I - P_{\mathfrak{S}^k}\right)\left(z^{k-1}\right) \\ x^{k+1} = & \mathbf{prox}_{\gamma g}\left(\mathsf{Q}^{-1}\left(z^k\right)\right) \end{cases}$	
Selection	Option 1	$\mathcal{C}_i \in \mathfrak{S}^k$ with probability p	$\mathcal{C}_i \in \mathfrak{S}^k$ with probability $\begin{cases} p & \text{if } x^k \in \mathcal{M}_i \Leftrightarrow [\mathsf{S}_{\mathcal{M}}(x^k)]_i = 0 \\ 1 & \text{elsewhere} \end{cases}$
	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

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 and that in heavily structured problems, adapting to an ultimately very sparse structure already reduces drastically the number of explored dimensions, 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 (2.15) and (2.16), 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[3] associated P_ℓ, Q_ℓ, and Q_ℓ^{-1}. Notice that $\lambda_{\min}(\mathsf{P}_\ell) = p_\ell = s/|\mathrm{null}(x^k)|$.

- 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 $k_{\ell-1}$	i is in the support at $k_{\ell-2}$	$\left[\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\right]_{ii}$
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 (2.17) and (2.18). 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).

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

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

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

- 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

$$\mathbb{P}[\mathcal{C}_i \in \mathfrak{S}^k] = \begin{cases} 1 & \text{if } i \in \mathcal{S} \\ 1 & \text{if } [\mathsf{S}_{\mathcal{M}}(x^k)]_i = 1 \\ p & \text{elsewhere} \end{cases}$$

where $\mathcal{S}$ is a set of artificial jumps $\{n_1, n_2, \ldots, n_{l-1}\}$. It is easy to see that small enough l will not change the sparsisty property of the random projection $P_{\mathfrak{S}^k}$ however this modification will force all the projection to be block-diagonal with blocks' ends on positions $n_1, \ldots n_{l-1}$. 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 $\{n_i\}_{1 \le i \le l-1}$[4] P_ℓ also has such block sructure and we could split the computation of Q_ℓ^{-1} and Q_ℓ into l independent parts and could be done in parallel. Consider $n_i = \lceil \frac{ni}{l} \rceil$, then the computational cost of inversion cost will decrease by factor of l^2.

2.4 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.

2.4.1 Experimental setup

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

$$\min_{x \in \mathbb{R}^n} \quad \frac{1}{m} \sum_{i=1}^{m} \log\left(1 + \exp\left(-b_i a_i^\top x\right)\right) + \frac{\lambda_2}{2}\|x\|_2^2 + \lambda_1 \|x\|_1 \tag{2.19a}$$

$$+ \lambda_1 \|x\|_{1,2} \tag{2.19b}$$

$$+ \lambda_1 \mathbf{TV}(x) \tag{2.19c}$$

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

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

$\ell_{1,2}$ 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 (2.19a) and (2.19b), and by variation for (2.19c). The adaptation strategies are the ones described in Section 2.3.3.

We consider five algorithms:

Name	Reference	Description	Randomness
`PGD`		vanilla proximal gradient descent	None
x`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 8	(non-adaptive) random subspace descent	Option 2 of Table 2.1 with $s = \text{x}$
x `ARPSD`	Algorithm 9	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 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 (standard comparison);
- size of the sparsity pattern vs iterations (showing the identification properties);
- functional suboptimality vs number of subspaces explored (showing the gain of adaptivity).

2.4.2 Illustrations for coordinate-structured problems

Comparison with standard methods

We consider first ℓ_1-regularized logistic regression (2.19a); in this setup, the non-adaptive `RPSD` boils down to the usual randomized proximal gradient descent (see Section 2.2.4). 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.

[5] 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.

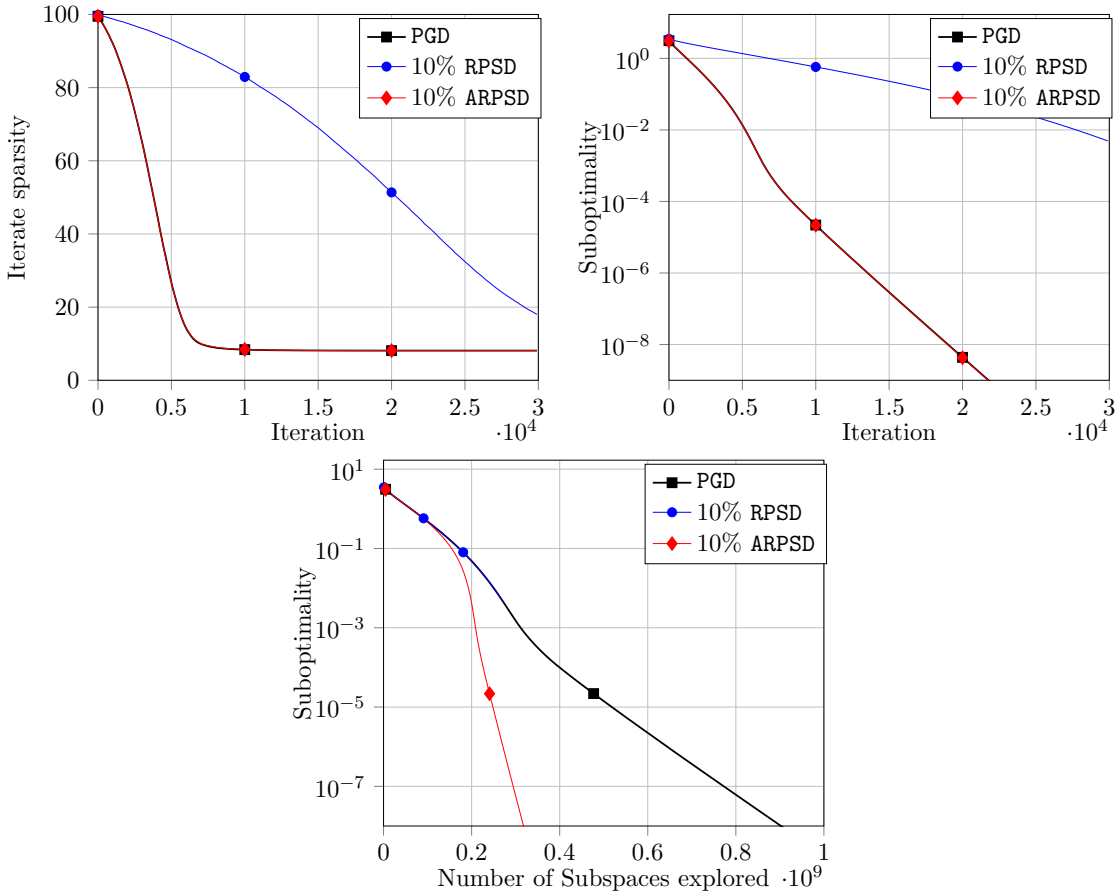

Figure 2-3. ℓ_1-regularized logistic regression (2.19a)

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 2.2.4). 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.

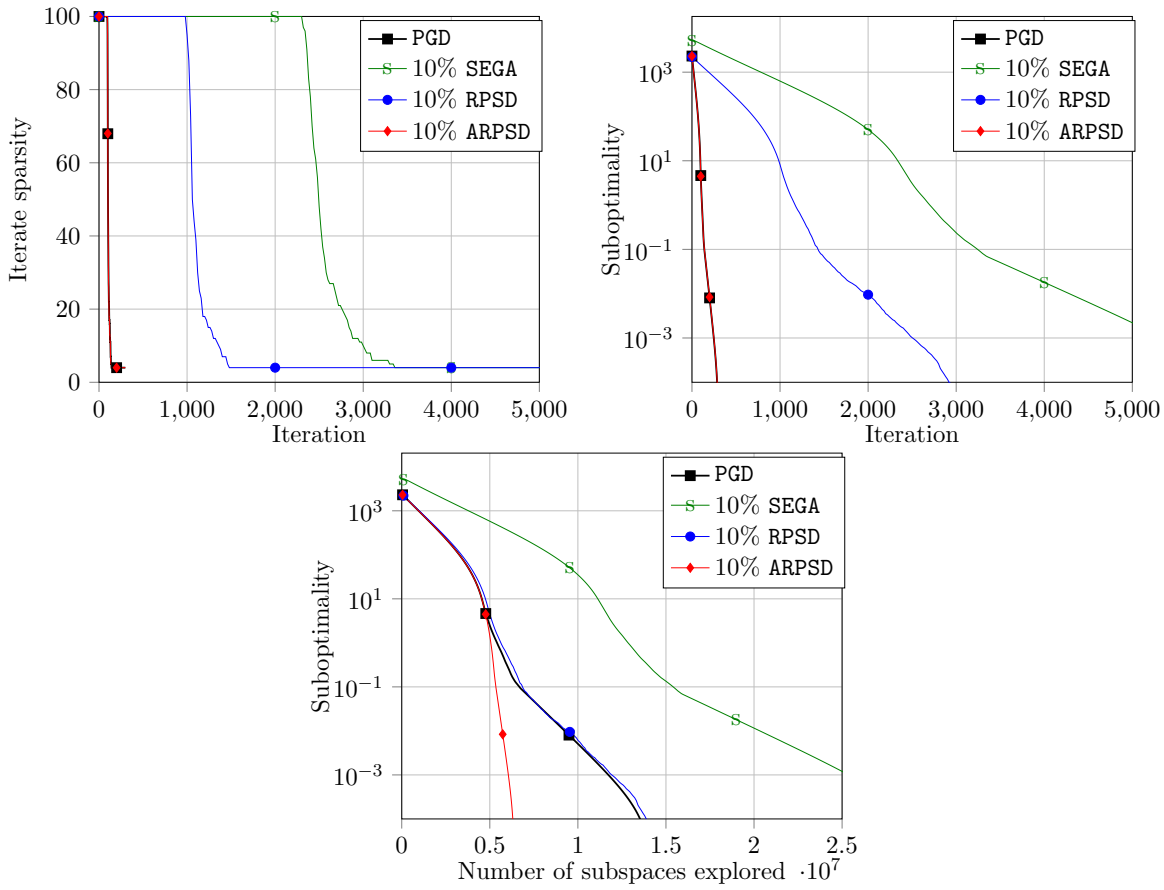

Figure 2-4. $\ell_{1,2}$ regularized logistic regression (2.19b)

2.4.3 Illustrations for total variation regularization

We focus here on the case of total variation (2.19c) 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 sub-optimality 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.

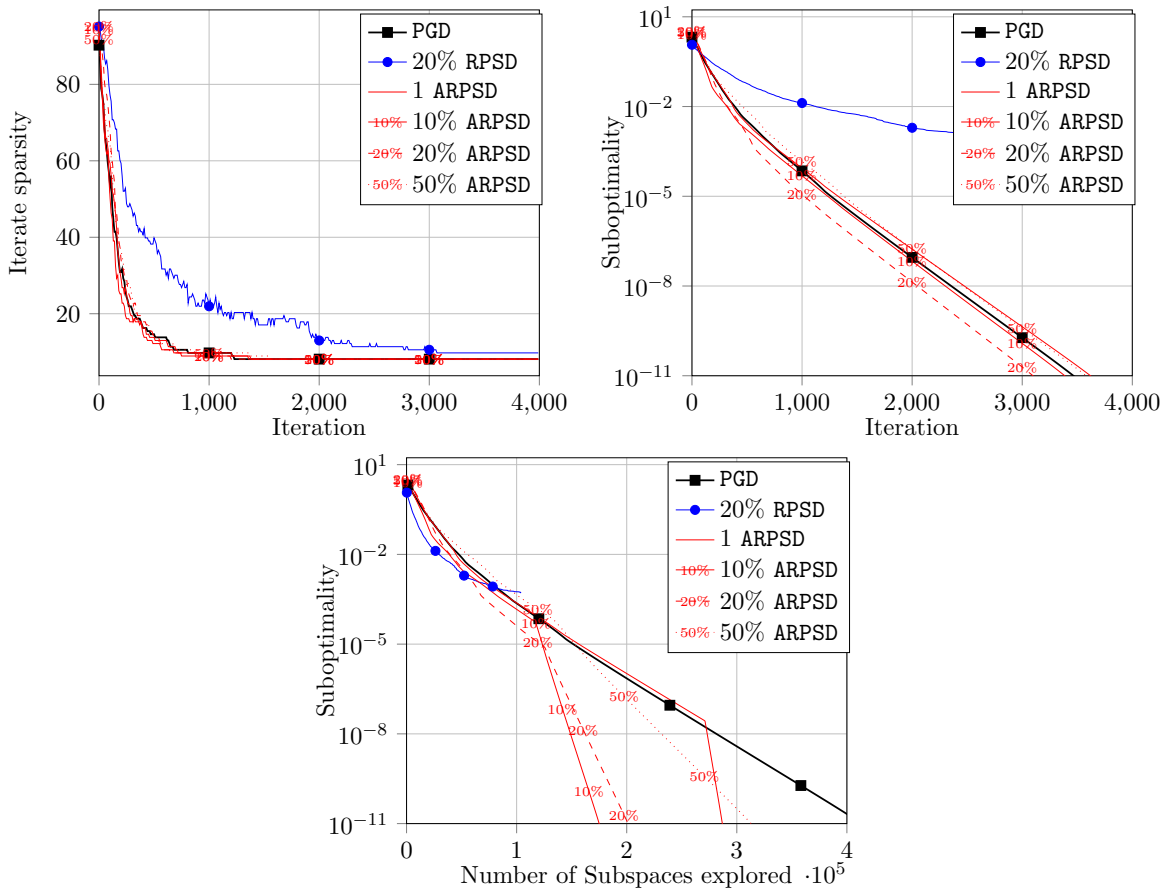

Figure 2-5. 1D-TV-regularized logistic regression (2.19c)

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.

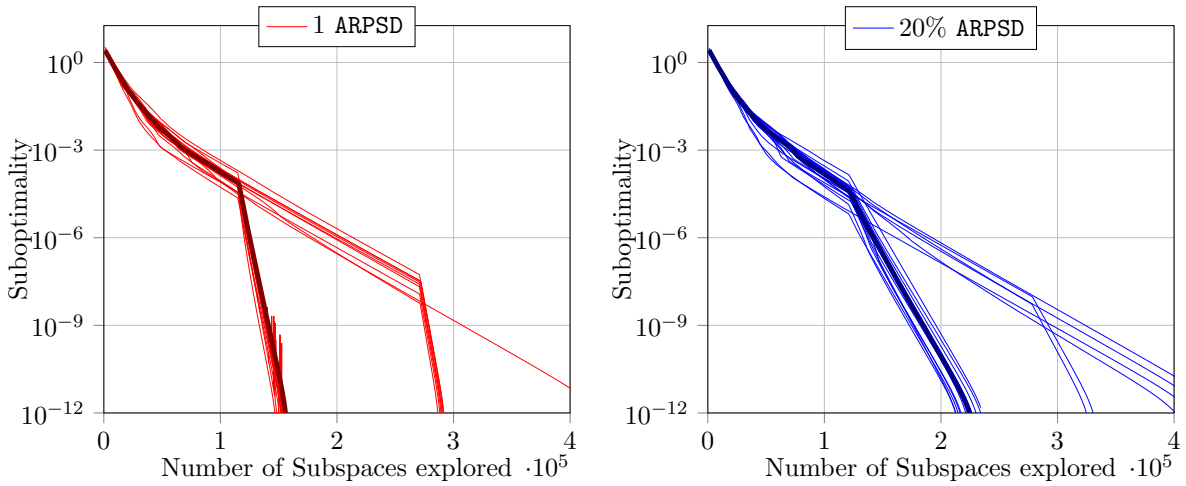

Figure 2-6. 20 runs of `ARPSD` and their median (in bold) on 1D-TV-regularized logistic regression (2.19c)

Chapter 3

Distributed learning

3.1 Introduction

Given the tremendous production of data and the ever-growing size of collections, there is a surge of interest in both Machine Learning and Optimization communities for the development of efficient and scalable distributed learning strategies. In such context, training observations are generally split over different computing nodes and learning is performed simultaneously where each node, also referred to as a *worker*, has its memory and processing unit. Note that this is different from shared-memory parallel computing, where each worker machine can potentially have access to all available memory [Val90, Kum02].

All distributed algorithms could be segregated according to the network structure. Some of them use networks with 1 central node called master that has a LAN connection with all the other nodes called workers [KMRR16, KMY+16, MIM18]. 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. It 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. It is easy to see, that from the theoretical point of view such setup is quite the same as the first one.

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. Another important part is in computing exact averages of values that are initially stored on the nodes [BPC+11a, OT06, OT09].

In our work, we focus on the centralized distributed setup without shared memory [MIM18] where all the data $\mathcal{D}$ is separated between M workers and all local subsets $\mathcal{D}_i$ are stored on each machine. Let us 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]. The study of distributed SGD in a star-shaped network [Yan13] raises the question of the communication-computation tradeoff. 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 those revert stochastic methods to the full-batch methods. Another way to look at this idea is local SGD methods that have been investigated in [Sti18, KMR19, KMR20, LSTS19, MKJ+17]. These methods perform well in practice, however the theoretical understanding is an open area for discovering.

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].

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, MIM18] provide a delay-independent analysis technique that allows integrating assumptions on the computing system with a constant stepsize.

3.2 Notations and Preliminaries

We place ourselves in a *totally asynchronous* as per the classification of Bertsekas and Tsitsiklis [**?**, Chap. 6.1], meaning that all workers are responsive but without bounded delays. In this section, we recall the `DAve-PG` algorithm, notations, and results from [MGTR19].

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

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

with $\alpha_i = |\mathcal{S}_i|/n$ being the proportion of observations locally stored in machine i, hence $\sum_{i=1}^{M} \alpha_i = 1$. $f_i(x) = \frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i} l_j(x)$ 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.

`DAve-PG` can solve Problem (P) by representing it in the form of a triplet (*workers weights*,*workers functions*,*global regularization*) i.e. $((\alpha_i),(f_i),r)$ and performing Algorithm 10.

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 following notation.

Algorithm 10 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

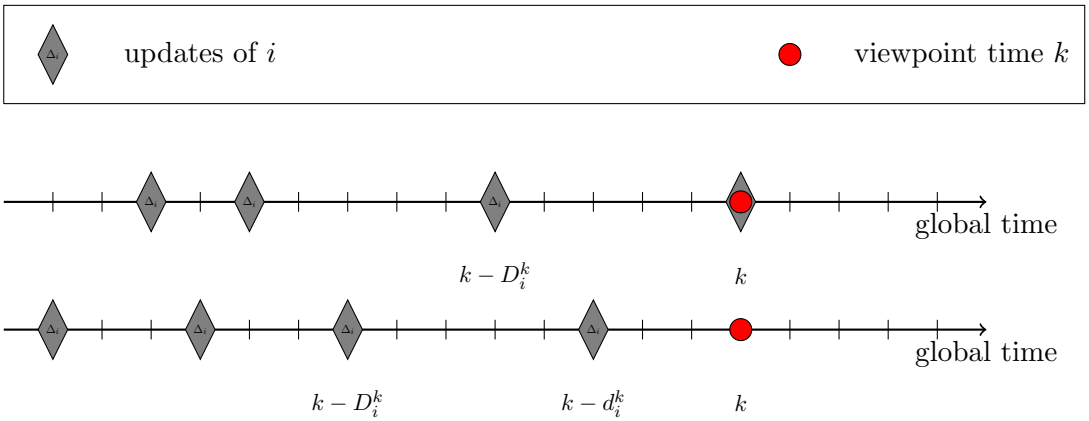

Figure 3-1. Notations of delays at iteration k.

- *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 3-1).

- *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

$$\bar{x}^k = \sum_{i=1}^{M} \pi_i x_i^k = \sum_{i=1}^{M} \pi_i x^{k - d_i^k}. \tag{3.1}$$

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

$$k_{m+1} = \min \left\{ k : k - D_i^k \geq k_m \text{ for all } i \right\}. \tag{3.2}$$

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 (3.1) 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.

As in [MGTR19], all convergence results will involve this notion of epochs.

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 achieved easily by *exchanging* quadratic between functions. This setup allows us to feature a single stepsize in our algorithm which simplifies the presentation to focus on the contributions.

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 paper as we will consider the case when $\mu = 0$ in the upcoming sections.

Theorem 3.1 (Th. 3.2 [MGTR19]). *Let the functions (f_i) be μ-strongly convex ($\mu > 0$) and L-smooth. Let g be convex lsc. 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 $k \in [k_m, k_{m+1})$

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

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, for ℓ_0 and ℓ_1 norms, it corresponds respectively to the hard and soft thresholding operations which set the smallest coordinates to zero, thus sparsifying the output. This kind of proximal sparsification was succesfully used in the case of synchronous distributed learning; see e.g. [WKSZ17, SFJJ16]. However worker-to-master communications do not need to be sparse that 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 [MIM18, 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 below).

We first propose an asynchronous distributed algorithm featuring a sparsification of upward communications (slave-to-master). Its sparsification mechanism consists in randomly and independly zeroing 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 learning rate that depend neither on communication delays nor on the number of machines.

But the theoretical analysis prevent us from using different probabilities in the case, when the problem is ill-conditioned. Getting around this we propose two different solutions. First, we analyze the practical interest of such algorithms and show empirically that algorithm converges often and the "expected" amount of communications between machines is smaller than without sparsification. The second approach is in iterative proximal reconditioning of our problem. The same idea is used in Catalyst acceleration scheme [LMH15], where the inner problem is well conditioned with any possible condition number.

Moreover, in the case of ℓ_1-regularized problems, we show that the generated master iterates identify some sparsity pattern in finite time with probability one, resulting in sparse downward communications (master-to-slave). Thus, all communications are eventually sparse.

Finally, we furthermore leverage on this identification to improve our sparsification technique by preferably sampling the entries in the support of the master model; this approach can be seen as an automatic dimension reduction procedure, resulting in better performance in terms of quantity of information exchanged that is proven to be better than non-sparsified algorithm in terms of communications with specific probability for coordinates outside the support.

Outline. This chapter is organized as follows.

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.

[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.

[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.

[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.

[CMBJ16] Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. In *International Conference on Learning Representations Workshop Track*, 2016.

[Con13] Laurent Condat. A direct algorithm for 1-d total variation denoising. *IEEE Signal Processing Letters*, 20(11):1054–1057, 2013.

[CP08] Patrick L Combettes and Jean-Christophe Pesquet. Proximal thresholding algorithm for minimization over orthonormal bases. *SIAM Journal on Optimization*, 18(4):1351–1376, 2008.

[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.

[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.

[FGMP18] Jalal Fadili, Guillaume Garrigos, Jérome Malick, and Gabriel Peyré. Model consistency for learning with mirror-stratifiable regularizers. *arXiv preprint arXiv:1803.08381*, 2018.

[FGS15] Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Mind the duality gap: safer rules for the lasso. *arXiv preprint arXiv:1505.03410*, 2015.

[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.

[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.

[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.

[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.

[KHR19] 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. *arXiv preprint arXiv:1901.08689*, 2019.

[KMR19] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. First analysis of local gd on heterogeneous data. *arXiv preprint arXiv:1909.04715*, 2019.

[KMR20] A Khaled, K Mishchenko, and P 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: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] AN KOLMOGOROV. Grundbegriffe der wahrscheinlichkeitreichnung. *Ergebnisse der Mathematik*, 1933.

[KSJ19] Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*, 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.

[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.

[LPLJ17] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. Asaga: Asynchronous parallel saga. In *Artificial Intelligence and Statistics*, pages 46–54, 2017.

[LS97] Claude Lemaréchal and Claudia Sagastizábal. Practical aspects of the moreau–yosida regularization: Theoretical preliminaries. *SIAM Journal on Optimization*, 7(2):367–385, 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.

[Mai15] Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

[MGTR19] Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

[MIM18] Konstantin Mishchenko, Franck Iutzeler, and Jérôme Malick. A distributed flexible delay-tolerant proximal gradient algorithm. *arXiv preprint arXiv:1806.09429*, 2018.

[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 Konečnỳ, 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 Richtarik, and Martin Takac. 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.

[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] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[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. *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.

[OT06] Alex Olshevsky and John N Tsitsiklis. Convergence rates in distributed consensus and averaging. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 3387–3392. IEEE, 2006.

[OT09] Alex Olshevsky and John N Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM Journal on Control and Optimization*, 48(1):33–55, 2009.

[PB+14] Neal Parikh, Stephen Boyd, et al. 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. *arXiv preprint arXiv:1703.02518*, 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-Bibiane Schönlieb. Local convergence properties of saga/prox-svrg and acceleration. *arXiv preprint arXiv:1802.02554*, 2018.

[Pol63] Boris T Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963.

[Pol69a] Boris T 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.

[RT16] Peter Richtárik and Martin Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, 10(6):1233–1243, 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.

[SHY17] Tao Sun, Robert Hannah, and Wotao Yin. Asynchronous coordinate descent under more realistic assumptions. In *Advances in Neural Information Processing Systems*, 2017.

[SJNS19] 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.

[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.

[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.

[Sti18] Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

[Sto74] William F Stout. Almost sure convergence. 1974.

[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.

[Teb18] Marc Teboulle. A simplified view of first order methods for optimization. *Mathematical Programming*, 170(1):67–96, 2018.

[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.

[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.

[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.

[YLY11] Lei Yuan, Jun Liu, and Jieping Ye. Efficient methods for overlapping group lasso. In *Advances in neural information processing systems*, pages 352–360, 2011.

[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.