SKGLM: A FAST MODULAR SOLVER FOR NON-SMOOTH CONVEX AND NON-CONVEX OPTIMIZATION

Pierre-Antoine Bannier[1,2] & Mathurin Massias[3]

[1] *Ecole Polytechnique, Route de Saclay, Palaiseau, 91128, pierre-antoine.bannier@polytechnique.edu*
[2] *HEC Paris, 1 rue de la Libération, Jouy-en-Josas, 78350*
[3] *INRIA, 46 allée d'Italie, Lyon, 69007, mathurin.massias@inria.fr*

Abstract

With the mounting amount of data collected and processed every day, machine learning practicioners need automated ways to analyze and extract informative patterns out of this evergrowing mass of information. In the business and the scientific worlds, modeling complex phenomena from a small set of predictive variables is particularly sought after. For instance, companies try to identify the key predictors that push a customer to buy their products; in medicine, biostatisticians try to isolate a handful of genes responsible for the expression of a specific type of cancer. This intuition of underlying simplicity has long been discussed in science and is coined *the parcimony principle*. Simply put, for two statistical models of similar predictive power, the one with fewer predictive variables should be selected. Sparse generalized linear models (GLM) form an important class of parametric statistical estimators that estimate solutions having a handful of predictive variables, thus abiding by this parcimony principle. Their statistical properties have been extensively studied and are used everyday by machine learning practicioners to infer a relationship between a set of predictive variables and a target variable. The goal of this thesis is to create a fast and versatile algorithm to estimate the solutions of these highly-used models.

We present `skglm`, a state-of-the-art solver to efficiently estimate the solutions of sparse generalized linear models. This solver is released in a professional quality open-source library, integrated to the `scikit-learn` ecosystem. With this library, we fully bring to practicioners the power of a wide variety of sparse GLMs, through an off-the-shelf, lightning fast solver with a convenient API. Along this thesis, `skglm` has been presented in Bertrand, Klopfenstein, Bannier, Gidel, and Massias (2022).

We first study why sparsity creates non-smooth optimization problems, which require a new set of tools to be solved. We review the main properties of proximal operators and their ability to minimize a smooth approximation of a non-smooth function. Then we focus on Fenchel-Rockafellar duality theory and derive a few sparse GLM dual problems. All these tools are crucial to develop first-order methods in non-smooth optimization to solve composite 'smooth + nonsmooth' separable problems. We review proximal gradient descent and proximal coordinate descent. After having derived the convergence rates of these descent methods, we explain why a cyclic feature selection strategy in coordinate descent is usually the fastest. Using this strategy, we demonstrate that the larger step sizes used by coordinate descent makes it a faster method than proximal gradient descent. This theoretical result is validated by extensive experimentation and justifies why coordinate descent is used as the backbone solver in `skglm`.

However fast it may be, our experiments show that coordinate descent as is remains very slow for very large datasets typical in the real world. To tackle these very high-dimensional problems, coordinate descent must be equipped with additional techniques to reduce the complexity of the problem at hand. Screening rules are powerful tools to dramatically increase the convergence speed of a solver, but remains unavailable for non-convex penalties. Nonetheless, working sets can be used for convex and non-convex penalties by adapting the underlying scoring strategy of features. As demonstrated in our experiments, the working set strategy embarked in `skglm` plays a central role in the performance of the solver. The second component ubiquitous in modern solvers is acceleration. Inertial acceleration and non-linear extrapolation are analyzed. It is observed that although both acceleration techniques improve the convergence rate, non-linear extrapolation outperforms inertial acceleration on a wide variety of setups in practice. An instance of such extrapolation techniques is called Anderson acceleration and is integrated to `skglm`. Through ablation studies, we demonstrate the importance of the combination of a working set strategy and Anderson acceleration to achieve fast convergence speed.

Finally we investigate non-convex penalties and explain their superiority over convex penalties. When solving these optimization problems, no dual problem is available to offer a complementary perspective on the optimization problem under scrutiny. This forces us to adapt `skglm`'s working set algorithm by integrating new feature ranking strategies available in convex and non-convex regimes. We show through extensive experimentation that our methods is the fastest among existing non-convex solvers. We conclude this thesis by discussing the practical choices of implementation of `skglm`.

Acknowledgments

First of all, I would like to thank my master thesis director Mathurin Massias for his guidance and trust during this journey. Thanks for letting me be part of the `skglm` *team, I have learnt a lot from your precious feedbacks on my PRs as well as on your comments on this document.*

A big thank you to Quentin Bertrand and Quentin Klopfenstein. Working with you has been a pleasure of mine. Thanks for answering all my questions, taking the time to debug with me what will have become `skglm`*.*

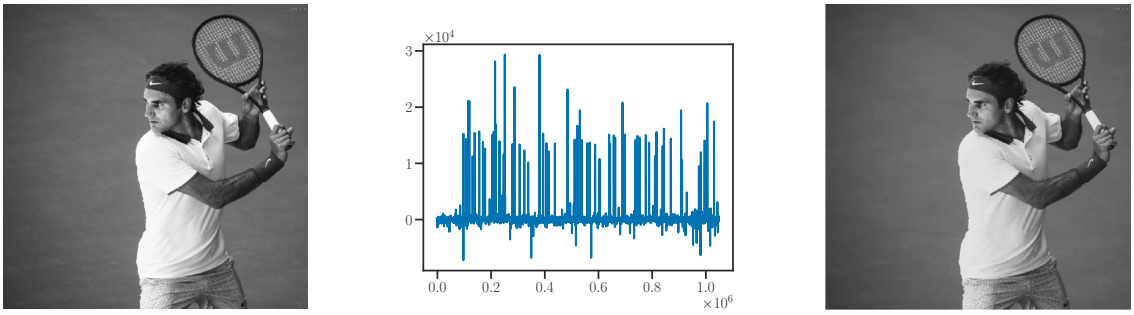
Finally, thanks to Badr Moufad for reviewing this document and contributing to `skglm`*. Your comments and your rigor are always very much appreciated.*

Contents

1 Introduction

As we have entered the big data era, petabytes of data are generated, collected and processed every day. This sheer amount of information is analyzed to extract informative patterns that are ubiquitous in every aspect of our lives: finance, healthcare, to name a few. Extracting useful information out of this large amount of data requires automated methods of data analysis and prediction, provided under the common denomination of "machine learning" (Bishop, 2007). Machine learning encompasses a wide variety of tasks requiring distinct assumptions and methodologies. In this manuscript, we focus on supervised learning, which consists in inferring the relationship between a set of predictive variables and one or multiple observed target variables.

A collection of observations (predictive variables and targets) constitutes a dataset. When dealing with very high-dimensional datasets, practicioners often resort to models reflecting a priori knowledge about the relationship between the predictive variables and the targets. An instance of such belief has been formulated in 1495 by William of Ockham: "*Plurality must never be posited without necessity*". The celebrated Ockham's razor (also coined *law of parcimony*) postulates that for two models having similar predictive power, one should retain the simpler of the two. In all likelihood, a significant amount of predictive variables in a high dimensional dataset can be discarded while still retaining a satisfactory predictive accuracy. For instance, not all of the 30,000 genes in the human genome are responsible for the expression of a specific cancer type: it is likely that only a handful of them in each chromosome are responsible for the formation of cancerous cells. Figure 1c shows that images can be reconstructed from few coefficients in their wavelet decomposition, while still retaining the visual quality of the original image. This sparsity assumption allows us to extract reproducible and informative patterns from high-dimensional datasets (Hastie et al., 2015), a crucial feat at the age of big data.



(a) Original (b) Wavelet coefficients (c) Reconstructed

Figure 1: **Wavelet decomposition and compression.** Signals can often be represented combining few atoms. The wavelet transform (Daubechies, 1992) of the image (Figure 1a) is computed. Relatively few wavelet coefficients (Figure 1b) explain most of the image energy. Compressing the image from its 5% largest wavelet coefficients yields Figure 1c.

Sparse generalized linear models (GLM) are a class of parametric statistical estimators enforcing sparsity. They are widely used to solve regression and classification tasks, in scientific and business applications such as in neuroscience (Strohmeier et al., 2016), genomics (Ghosh and Chinnaiyan, 2005; Rapaport et al., 2008) or computer vision (Mairal, 2010). Since their statistical properties have been extensively studied (Simon et al., 2013; Candes et al., 2008; Zou and Hastie, 2005) and they are implemented efficiently in numerous scientific libraries in `Python` with `scikit-learn` (Pedregosa et al., 2011), or in R with `glmnet` (Friedman et al., 2009) they are amongst the most standard and widely used statistical models. They are characterized by the sparsity of their solutions, that only have a small number of nonzero coefficients, and thus perform variable selection (Zou and Hastie, 2005).

Software implementations of these models rely on fast optimization algorithms (Fan et al., 2008; Friedman et al., 2009; Blondel and Pedregosa, 2016). When dealing with very large datasets, fast memory-efficient solvers are crucial to estimate a solution in reasonable time. They usually rely on two crucial components: a working set strategy and acceleration. In recent years, working set solvers applied to convex problems (Johnson and Guestrin, 2015; Massias et al., 2018; Ndiaye et al., 2020) have gained popularity to efficiently solve problems of evergrowing sizes. Another crucial component for convex solvers is acceleration, a set of techniques improving the convergence rate. In this manuscript, we distinguish two kinds of acceleration: inertial acceleration (Polyak, 1964; Nesterov, 1983) and extrapolation (Anderson, 1965). Acceleration is extensively used in deep learning (Kingma and Ba, 2014;

Ruder, 2016) and is now a standard practice for generalized linear models (Beck and Teboulle, 2009; Scieur et al., 2016; Mai and Johansson, 2020).

There exists various classes of sparsity-inducing penalties for GLMs. Convex penalties enjoy well-studied analytical properties (Bach et al., 2011), the most convenient of which is the local-to-global property, meaning that every local minimum is global. This is why, in the wake of the seminal work of Tibshirani (1996) on the Lasso, the ℓ_1 norm and its variations are the standard sparsity inducing penalties (Zou and Hastie, 2005; Roth and Fischer, 2008; Lee et al., 2006). Nonetheless, convex penalties are limited compared to their non-convex counterparts (Fan and Li, 2001; Zhang, 2010) that yield sparser solutions and mitigate the intrinsic bias of convex penalties (Breheny and Huang, 2011; Soubies et al., 2015). Currently, there is no implementation equivalent to `glmnet` (Friedman et al., 2009), `liblinear` (Fan et al., 2008) or `scikit-learn` (Pedregosa et al., 2011) for non-convex penalties. Therefore, practitioners face the following trade-off: using convex estimators implemented with fast optimization algorithms at the cost of biased and less sparse solutions, or resorting to non-convex estimators with no off-the-shelf fast implementation.

In this work, we propose a state-of-the-art solver based on working sets and Anderson acceleration to efficiently estimate the solutions of sparse generalized linear models. Through an extensive experimental validation, we demonstrate its versatility and its superiority over existing methods. This algorithm is released in a professional quality open-source library, integrated to the scikit-learn ecosystem, called `skglm` (Bertrand et al., 2022). With this library[1], we fully bring to practitioners the power of non-convex penalties for sparse GLMs, through an off-the-shelf, lightning fast solver with a convenient API.

Notation

For any integer d, $[d]$ denotes the set $\{1, \dots, d\}$. We denote by n the number of observations, by p the number of features, $X = [x_1 | \dots | x_p] \in \mathbb{R}^{n \times p}$ represents the design matrix, and $y \in \mathbb{R}^n$ is the observation vector. For $\mathcal{W} \subset [p]$, and $\beta \in \mathbb{R}^p$, $\beta_\mathcal{W}$ is restricted to the indices in $\mathcal{W}$, $X_\mathcal{W}$ is the matrix X restricted to the columns with indices in $\mathcal{W}$. The vector $\mathbf{1}_K$ (resp. $\mathbf{0}_K$) has K entries set to 1 (resp. 0). We denote by $\mathcal{S}_d$ the set of symmetric matrices in $\mathbb{R}^{d \times d}$, and by $\mathcal{S}_d^+$ its restriction to semi-definite positive matrices. On $\mathcal{S}_d$, we use the Löwner order defined as follows: $A \preceq B \iff B - A \in \mathcal{S}_d^+$. $\#$ denotes the cardinality of a set. The row-wise separable $\ell_{p,q}$ group-norm of a matrix A is written $\|A\|_{p,q} = (\sum_i (\sum_j A_{i,j}^p)^{q/p})^{1/q}$. The element-wise product between two vectors or matrices is denoted by $\odot$. For a set C, we denote Π_C the orthogonal projection onto C.

[1] available at https://github.com/scikit-learn-contrib/skglm.

2 Problem formulation

2.1 Reminder on convex analysis

This manuscript is devoted to continuous finite dimensional optimization problems. In this section, we provide a set of definitions and assumptions in order to analyze their properties. Then, a generic formulation of the problems under scrutiny is given.

Definition 1 (Domain)**.** Let f be an extended real-valued function $f : \mathbb{R}^d \to [-\infty, +\infty]$. The domain of f is the set

$$\mathrm{dom}(f) = \{x \in \mathbb{R}^d : f(x) < +\infty\} \ . \tag{1}$$

The extension to infinite values is useful to transform constrained problems into unconstrained ones.

Example 2 (Indicator function of a convex set)**.** Consider the following constrained minimization problem where C is a subset of $\mathbb{R}^d$:

$$\min_{x \in C} f(x) \ . \tag{2}$$

Problem 2 can be reformulated as an unconstrained problem using an indicator function:

$$\min_{x \in \mathbb{R}^d} f(x) + \iota_C(x) \ , \tag{3}$$

where ι_C is the indicator function of the set C, that is

$$\begin{cases} 0 & \text{if } x \in C \ , \\ +\infty & \text{otherwise} \ . \end{cases} \tag{4}$$

Definition 3 (Proper function)**.** A function $f : \mathbb{R}^d \to [-\infty, +\infty]$ is proper if its domain is non-empty and it does not take the value $-\infty$.

Definition 4 (Lower semicontinuous function)**.** A function $f : \mathbb{R}^d \to [-\infty, +\infty]$ is lower semicontinuous at $x_0 \in \mathbb{R}^d$ if

$$\liminf_{x \to x_0} f(x) \geq f(x_0) \ . \tag{5}$$

Proposition 5 (Characterization of lower semicontinuity, Beck (2017, Theorem 2.6))**.** *A function $f : \mathbb{R}^d \to [-\infty, +\infty]$ is lower semicontinuous on $\mathbb{E} \subseteq \mathbb{R}^d$ if and only if its epigraph on $\mathbb{E}$ is closed.*

Definition 6 (Convex function)**.** A function $f : \mathbb{R}^d \to [-\infty, +\infty]$ is convex on $\mathbb{E} \subseteq \mathbb{R}^d$ if

$$\forall x, y \in \mathbb{E}, \forall t \in [0, 1], \quad f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) \ . \tag{6}$$

Definition 7 (Coercive function)**.** A proper function $f : \mathbb{R}^d \to]-\infty, +\infty]$ is coercive if

$$\lim_{\|x\| \to \infty} f(x) = +\infty \ . \tag{7}$$

If f is proper, lower semicontinuous, convex and coercive, then the problem $\min_{x \in \mathbb{R}^d} f(x)$ admits at least one minimizer (Giusti, 2003).

2.2 Problem under scrutiny

Motivated by generalized linear models but not limited to it, we study problems of the form

$$\hat{\beta} \in \arg\min_{\beta \in \mathbb{R}^p} \Phi(\beta) \triangleq F(X\beta) + \Omega(\beta) \triangleq \sum_{i=1}^{n} f_i(x_i^\top \beta) + \sum_{j=1}^{p} \Omega_j(\beta_j) \ , \tag{8}$$

where all f_i are convex smooth functions and the functions Ω_j are proper lower semicontinuous functions not necessarily convex nor smooth. The penalty $\Omega = \sum_j \Omega_j$ is separable[2] and penalizes an overly complex solution $\hat{\beta}$. Maximum likelihood estimation of sparse generalized linear models yields problems of the form of Problem 8 (Hastie et al., 2015). However, it is worth noting that this form of problem is not restricted to generalized linear models; the regression with Huber loss (Huber, 1981) has a similar formulation but is not a generalized linear model. This problem formulation encompasses regression and classification models such as Lasso (Tibshirani, 1996), ElasticNet (Zou and Hastie, 2005), Group Lasso (Simon et al., 2013), sparse Logistic Regression (Lee et al., 2006) or the dual SVM with hinge loss (see Appendix A). All fall into the class of "smooth + non-smooth separable" problems.

[2]We handle the case of block-separable penalties later in the manuscript.

Example 8 (Lasso, Tibshirani (1996))**.** For instance, for a regression task, let $F(\cdot) = \frac{1}{2n} \|y - \cdot\|_2^2$ be the quadratic datafit and $\Omega_j(\cdot) = \lambda |\cdot|$ with $\lambda > 0$ a regularization hyperparameter. Plugging these two terms in Problem 8 yields the Lasso (Tibshirani, 1996)

$$\hat{\beta} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \sum_{j=1}^{p} |\beta_j| \quad . \tag{9}$$

Example 9 (Sparse logistic regression)**.** Likewise, for $y \in \{-1, 1\}^n$, let $F(\cdot) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(- \cdot y_i))$ and $\Omega_j(\cdot) = \lambda |\cdot|$, plugging them in Problem 8 yields the sparse logistic regression

$$\hat{\beta} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \beta^\top X_{i,:})) + \lambda \sum_{j=1}^{p} |\beta_j| \quad . \tag{10}$$

Finally, we add two regularity assumptions on the datafit term F.

Assumption 10 (Lipschitz gradients)**.** A convex differentiable function $F : \mathbb{R}^n \to \mathbb{R}$ has Lipschitz gradients if there exists $L \in [0, +\infty[$ such that

$$\forall x, y \in \mathbb{R}^n, \quad \|\nabla F(x) - \nabla F(y)\| \leq L \|x - y\| \quad . \tag{11}$$

A function is said to be smooth if it has Lipschitz gradients.

Assumption 11 (Strong convexity)**.** Let $F : \mathbb{R}^n \to]-\infty, +\infty]$ be a function. F is μ-strongly convex for a given $\mu > 0$ if $\mathrm{dom}(f)$ is convex and the following inequality holds for any $x, y \in \mathrm{dom}\, f$ and $t \in [0, 1]$:

$$F(tx + (1-t)y) \leq tF(x) + (1-t)F(y) - \frac{\mu}{2} t(1-t) \|x - y\|^2 \quad . \tag{12}$$

These two assumptions are central in the derivation of convergence rates. When F is differentiable or twice-differentiable, several useful properties can be deduced from Assumptions 10 and 11. These properties are extensively used in the proofs of lemmas and convergence bounds. All the proofs can be found in Beck (2017, Section 2). In the following, $\mathbb{E}$ is a subset of $\mathbb{R}^n$, we assume $F : \mathbb{E} \to \mathbb{R}$ is twice-differentiable.

Proposition 12. *The following propositions are equivalent:*

1. *F is L-smooth on $\mathbb{E}$*
2. $\forall x, y \in \mathbb{E}, \quad |F(y) - F(x) - \langle \nabla F(x), y - x \rangle| \leq \frac{L}{2} \|y - x\|^2$
3. $\forall x \in \mathbb{E}, \quad \nabla^2 F(x) \preceq LI$

Proposition 13. *The following propositions are equivalent:*

1. *F is μ-strongly convex on $\mathbb{E}$*
2. *$F - \frac{\mu}{2} \|\cdot\|^2$ is convex on $\mathbb{E}$*
3. $\forall x, y \in \mathbb{E}, \quad F(x) \leq F(y) + \langle \nabla f(x), x - y \rangle - \frac{\mu}{2} \|x - y\|^2$
4. $\forall x \in \mathbb{E}, \quad \nabla^2 F(x) \succeq \mu I$

As we have formalized the class of problems under scrutiny, we now introduce the relevant tools to minimize such objectives.

3 First order proximal methods for convex optimization

In this section, we introduce the optimization tools to estimate the parameters of convex sparse generalized linear models. We introduce the necessary mathematical tools needed to minimize specific non-smooth functions, then review algorithms to estimate the solutions of sparse generalized linear models.

3.1 Sparsity and non-smooth optimization

When the number of features is larger than the number of samples, there are infinitely many solutions that verify $X\beta + \epsilon = y$ with ϵ a random Gaussian vector. A classical way to select a relevant subset of solutions consists in using estimators based on the ℓ_0-norm. The ℓ_0-norm is defined by

$$\|\beta\|_0 = \#\{\beta_j : \beta_j \neq 0, \;\; j = 1, \dots, p\} \;\; . \tag{13}$$

When the solution is expected a priori to have at most $k \in \mathbb{N}^*$ non-zero coefficients, the problem reads

$$\hat{\beta} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \frac{1}{2}\|y - X\beta\|_2^2 \quad \text{s.t.} \quad \|\beta\|_0 \leq k \;\; . \tag{14}$$

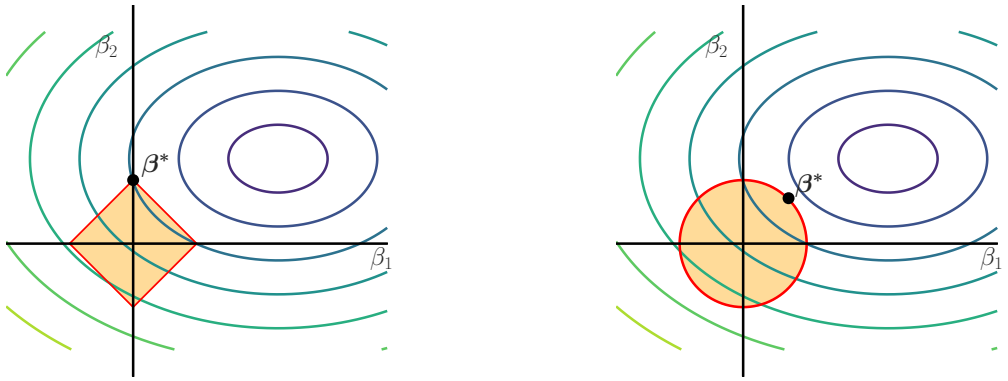
When no sparsity prior $k \in \mathbb{N}^*$ is known, Problem 14 is formulated in a penalized form

$$\hat{\beta} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_0 \;\; , \tag{15}$$

with $\lambda > 0$ a regularization hyperparameter. However, Problem 15 is a combinatorial non-convex NP-hard problem (Natarajan, 1995; Davis et al., 1997), such that one must look for solvable alternatives. Every ℓ_q-quasinorm[3] where $0 \leq q < 1$ induces sparsity while being non-convex. The ℓ_1-norm is the only one creating a convex optimization problem (Bach et al., 2011): it is the tightest convex relation of the ℓ_0-norm on the ℓ_∞ unit ball. Then the relaxed problem reads

$$\hat{\beta} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1 \;\; . \tag{16}$$

Problem 16 is called the Lasso (Tibshirani, 1996) in machine learning, but ℓ_1-regularization is also used in signal processing for signal denoising (Chen and Donoho, 1994). As shown in Figure 2, the constraint region for the ℓ_1-norm is a diamond, while it is a disk for the ℓ_2-norm. The solution of Problem 16 lies on the point β^* where the elliptical contours of the datafit term hit the constraint region of the penalty term. In higher-dimensional spaces, the diamond becomes a rhomboid and has many corners and flat edges, which makes it more likely for the coefficients to be zero (Bach et al., 2011; Iutzeler and Malick, 2020). This sparsity assumption results in the non-smoothness of the penalty. In order to obtain sparse solutions, the penalty term must be non-differentiable along the axes (Soubies, 2016, Remark 12.8).



(a) ℓ_1-norm. (b) ℓ_2-norm.

Figure 2: **Sparsity-inducing norms.** Contour plot of the unregularized datafit along with the constraint region in red for the ℓ_2-norm and ℓ_1-norm. The optimal solution is denoted as β^*.

[3] A quasinorm satistifies the norm axioms, except that the triangle inequality is replaced by $\|x + y\| \leq K(\|x\| + \|y\|)$ for some $K > 0$.

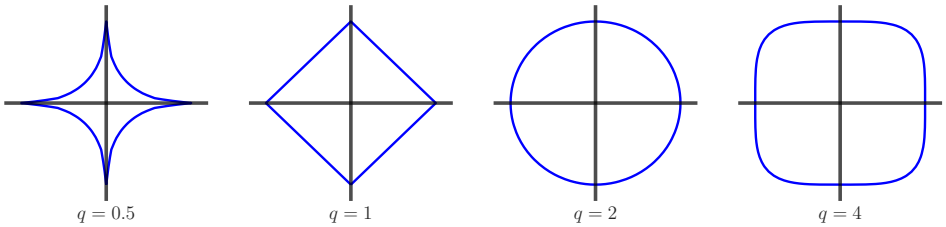


Figure 3: **Contour plots of ℓ_q penalties.** Unit ball of $\|\cdot\|_q$ for various values of q.

Proposition 14. *The ℓ_1-norm is non-differentiable along the axes.*

Proof. Let $(e_1, \dots, e_d)$ be the canonical basis of $\mathbb{R}^d$. Suppose there exists a derivative of $\|\cdot\|_1$ at e_i in the direction of e_j. Then there exists a differential operator D such that

$$\lim_{t \to 0} \frac{\|e_i + te_j\| - \|e_i\| - D(te_j)}{\|te_j\|} = \lim_{t \to 0} \frac{1 + |t| - 1 - tD(e_j)}{|t|} = 0 \ ,$$

or equivalently,

$$\lim_{t \to 0} \frac{-tD(e_j)}{|t|} = -1 \ .$$

This leads to a contradiction since it cannot hold for any value of $D(e_j)$. □

In Figure 3, the corners of the $\ell_{0.5}$ and ℓ_1 penalties describe the sparsity-inducing behavior of these penalties.

Proposition 14 prevents the use of standard gradient descent methods and requires the introduction of a new set of tools suited for non-smooth optimization.

3.1.1 Subgradients and the subdifferential

Definition 15 (Subgradient and subdifferential)**.** Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function and $x \in \mathbb{R}^d$. A subgradient of f at x is a vector $z \in \mathbb{R}^d$ that satisfies:

$$\forall x' \in \mathbb{R}^d, \quad f(x) + z^\top (x' - x) \leq f(x') \ . \tag{17}$$

The set of all subgradients of f at x is called the subdifferential $\partial f(x)$ of f at x (Moreau, 1965), and has a clear geometric interpretation: it is the set of slopes of all exact affine minorants at a point $x \in \mathbb{R}^d$ to the graph of the function f.

Theorem 16 (Subdifferential at a differentiable point, Beck (2017, Theorem 3.33))**.** *Let $f : \mathbb{R}^d \to]-\infty, +\infty]$ be a proper convex function, and let $x \in \operatorname{int}(\operatorname{dom}(f))$. If f is differentiable at x, then $\partial f(x) = \{\nabla f(x)\}$. Conversely, if f has a unique subgradient at x, then it is differentiable at x and $\partial f(x) = \{\nabla f(x)\}$.*

Theorem 17 (Fermat's rule, Bauschke and Combettes (2017, Theorem 16.3))**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. x^* is a minimizer of f if and only if*

$$0 \in \partial f(x^*) \ . \tag{18}$$

Proof. Let $x \in \mathbb{R}^d$. $x \in \arg\min f \iff \forall x' \in \mathbb{R}^d, \quad f(x) + \langle x' - x, 0 \rangle \leq f(x') \iff 0 \in \partial f(x)$. □

A useful concept tightly coupled with subgradients is dual norms.

Definition 18 (Dual norm)**.** The dual norm Ω^* of the norm Ω in $\mathbb{R}^d$ is defined for any vector $z \in \mathbb{R}^d$ by

$$\Omega^*(z) = \max_{x \in \mathbb{R}^d} z^\top x \ \text{ s.t. } \ \Omega(x) \leq 1 \ . \tag{19}$$

A well-known result in optimization is that the dual norm of the ℓ_q-norm, for $q \in [1, +\infty]$, is the $\ell_{q'}$-norm, with $q' \in [1, +\infty]$ such that $\frac{1}{q} + \frac{1}{q'} = 1$ (Bach et al., 2011, Section 1.4.2). In particular, the ℓ_2-norm is self-dual and the dual norm of the ℓ_1-norm is the ℓ_∞-norm.

Theorem 19 (Subdifferential of a norm, Bach et al. (2011, Proposition 1.2)). *Let Ω be a norm on $\mathbb{R}^d$, and $x \in \mathbb{R}^d$. The subdifferential of Ω at x is*

$$\partial\Omega(x) = \begin{cases} \{z \in \mathbb{R}^d : \ \Omega^*(z) \leq 1\} & \text{if } x = 0 \ , \\ \{z \in \mathbb{R}^d : \ \Omega^*(z) = 1 \text{ and } z^\top x = \Omega(x)\} & \text{otherwise} \, . \end{cases} \tag{20}$$

where Ω^* is the dual norm of Ω.

Example 20 (ℓ_1-norm). Applying Theorem 19 to the ℓ_1-norm in $\mathbb{R}^d$ yields

$$\partial \left\| \cdot \right\|_1 (0) = [-1, 1]^d \ . \tag{21}$$

When $d = 1$, the subdifferential at 0 reads $\partial \left| \cdot \right| (0) = [-1, 1]$ and contains every line with a slope in $[-1, 1]$ that are tangent to the graph of the norm at 0 as shown in Figure 4.

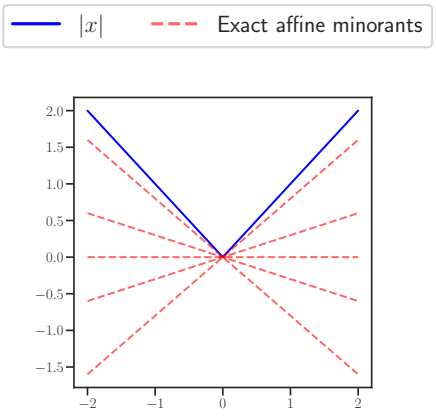


Figure 4: **Subgradient of ℓ_1-norm.** The subgradients of the absolute value at 0 are the slopes of the affine minorants of the function that are exact at 0.

3.1.2 Proximal operators

In addition to subgradients and subdifferentials, another fundamental tool for non-smooth optimization is proximal operators.

Definition 21 (Infimal convolution). Let f_1 and f_2 be functions from $\mathbb{R}^d$ to $[-\infty, +\infty]$. The infimal convolution of f_1 and f_2 at $x \in \mathbb{R}^d$ is

$$(f_1 \square f_2)(x) = \inf_{(u_1, u_2) \in \mathbb{R}^d \times \mathbb{R}^d} \{f_1(u_1) + f_2(u_2) : u_1 + u_2 = x\} = \inf_{u \in \mathbb{R}^d} \{f_1(u) + f_2(x - u)\} \ . \tag{22}$$

It can be used to construct a smooth approximation of any function f (Beck and Teboulle, 2012). Indeed, by computing the infimal convolution of a function f with the squared norm $\left\| \cdot \right\|^2$, one obtains a smoothed approximation of f. This specific smoothed approximation is called the Moreau envelope (Moreau, 1965) or Moreau-Yosida regularization.

Definition 22 (Moreau envelope). Let $f : \mathbb{R}^d \to [-\infty, +\infty]$ be a closed proper convex function, and $\lambda > 0$. The Moreau envelope of f with parameter λ reads

$$M_f^\lambda(v) = \inf_x \left(f(x) + \frac{1}{2\lambda} \|x - v\|_2^2 \right) \ . \tag{23}$$

The Moreau envelope gives a smooth approximation of f when f is not smooth, it is defined on $\mathbb{R}^d$ even when f is not and it is continuously differentiable everywhere. Besides, the set of minimizers of M_f^λ is the same as λf. Therefore, minimizing λf and M_f^λ is equivalent. Yet, the smoothness of M_f^λ makes this optimization problem easier to solve. The unique point that minimizes the Moreau envelope of λf is called the proximal operator of f.

Definition 23 (Proximal operator). Let $f : \mathbb{R}^d \to [-\infty, +\infty]$ be a closed proper convex function. The proximal operator $\mathbf{prox}_f : \mathbb{R}^d \to \mathbb{R}^d$ of f at level $\lambda > 0$ is defined by

$$\mathbf{prox}_{\lambda f}(v) = \arg\min_x \left(f(x) + \frac{1}{2\lambda} \|x - v\|_2^2 \right) \ . \tag{24}$$

Figure 5: **Moreau envelope and proximal operator of ℓ_1 norm.** The Moreau envelope of the ℓ_1-norm is the Huber loss. The proximal operator of the ℓ_1-norm is the soft-thresholding operator.

It is unique since f is proper and the argument in the $\arg\min$ is strongly convex. The proximal operator $\mathbf{prox}_{\lambda f}$ is a trade-off between being close to v and minimizing f. The proximal operator is usually parametrized by $\lambda > 0$ which can be interpreted as a relative weight between these two terms. Section 3.1.2 shows that the 1-dimensional Huber loss is a smooth approximation of the absolute value. We plot along this smooth approximation the proximal operator minimizing this smooth approximation. The connection between the proximal operator of λf and its Moreau envelope is clear:

$$M_f^\lambda(x) = f(\mathbf{prox}_{\lambda f}(x)) + \frac{1}{2\lambda}\|x - \mathbf{prox}_{\lambda f}(x)\|_2^2 \ . \tag{25}$$

Using Danskin's theorem (Danskin, 1967) on the Moreau envelope, it follows

$$\mathbf{prox}_{\lambda f}(x) = x - \lambda \nabla M_f^\lambda(x) \ , \tag{26}$$

which highlights that $\mathbf{prox}_{\lambda f}$ is a gradient step of size λ to minimize the Moreau envelope of f at level λ. The following theorem gives a natural link between proximal operators and fixed point theory.

Theorem 24 (Minimizer and fixed points, Bauschke and Combettes (2017, Corollary 6.40))**.** *Let $f : \mathbb{R}^d \to [-\infty, +\infty]$ be a closed proper convex function. The point $x^* \in \arg\min_{x \in \mathbb{R}^d} f(x)$ if and only if*

$$\mathbf{prox}_f(x^*) = x^* \ . \tag{27}$$

Many sparsity-inducing penalties admit a closed-form proximal operator, which makes them usable in optimization algorithms. As an example, we derive the well-know soft-thresholding operator (Daubechies et al., 2003) for the ℓ_1-norm.

Example 25 (Soft-thresholding)**.** Let $f(x) = \lambda\|x\|_1 = \sum_{i=1}^d \lambda|x_i|$, $x \in \mathbb{R}^d, \lambda > 0$. This penalty is separable. Let $f_i(x_i) = \lambda|x_i|$. The proximal operator of f_i reads

$$\begin{aligned}
\mathbf{prox}_{f_i}(x_i) &= \arg\min_u \left(\lambda|u| + \frac{1}{2}(u - x_i)^2 \right) \\
&= \arg\min_u \begin{cases} \lambda u + \frac{1}{2}(u - x_i)^2 & \text{if } u \geq 0 \ , \\ -\lambda u + \frac{1}{2}(u - x_i)^2 & \text{otherwise} \ . \end{cases}
\end{aligned}$$

In the first case, the minimum is attained in $\mathbb{R}_+$. Taking the derivative of $h_i(u) \triangleq \lambda u + \frac{1}{2}(u - x_i)^2$, it follows that at the minimum, $u = x_i - \lambda$. Therefore if $x_i \geq \lambda$, $\mathbf{prox}_{f_i}(x_i) = x_i - \lambda$. Using the same reasoning in the second case, it follows that $u = \lambda + x_i$. If $x_i \leq -\lambda$, $\mathbf{prox}_{f_i}(x_i) = \lambda + x_i$. There remains the case where $x_i \in [-\lambda, \lambda]$. We know that the ℓ_1-norm is convex and that its minimum exists, therefore if this minimum is not attained at a point of differentiability it must be attained at a point of non-differentiability, that is 0. Finally,

$$\mathbf{prox}_{f_i}(x_i) = \begin{cases} x_i - \lambda & \text{if } x_i \geq \lambda \ , \\ x_i + \lambda & \text{if } x_i \leq -\lambda \ , \\ 0 & \text{otherwise} \ , \end{cases} \tag{28}$$

which can be compactly written $\mathbf{prox}_{f_i}(x_i) = [|x_i| - \lambda]_+ \operatorname{sgn}(x_i)$. In a vectorial form, and applying proximal calculus rules, we obtain the soft-thresholding operator

$$\mathbf{prox}_f(x) \triangleq \mathcal{T}_\lambda(x) = [|x| - \lambda \mathbf{1}_d]_+ \odot \operatorname{sgn}(x) \ . \tag{29}$$

Table 1 summarizes the proximal operators used for some usual sparsity-inducing penalties.

Table 1: Proximal operators for convex sparse penalties.

Model	Penalty	$f(\beta)$	$\mathbf{prox}_f$
Lasso	ℓ_1	$\lambda\|\beta\|_1$	$[\|\beta\| - \lambda \mathbf{1}_p]_+ \odot \operatorname{sgn}(\beta)$
Group Lasso	$\ell_{2,1}$	$\lambda \sum_{g\in\mathcal{G}} \|\beta_g\|_2$	$\left(1 - \frac{\lambda}{\max(\|\beta_g\|_2, \lambda)}\right) \beta_g$
Multi-task Lasso	$L_{2,1}$	$\lambda\|\mathbf{B}\|_{2,1}$	$\left(1 - \frac{\lambda}{\max(\|\mathbf{B}_{:t}\|_2, \lambda)}\right) \mathbf{B}_{:t}$
Elastic Net	$\ell_1 + \ell_2$	$\lambda(\gamma\|\beta\|_1 + \frac{1-\gamma}{2}\|\beta\|_2^2)$	$[\|\beta\| - \gamma\lambda \mathbf{1}_p]_+ \odot \frac{\operatorname{sgn}(\beta)}{1+\lambda(1-\gamma)}$
Dual SVM	$\iota_{[0,C]}$	$\iota_{[0,C]}(\beta)$	$\Pi_{[0,C]}(\beta)$

3.1.3 Duality and conjugation

Duality theory offers complementary perspectives on convex optimization problems. There exists multiple kinds of duality theory including Lagrange (Boyd and Vandenberghe, 2004) and Fenchel-Rockafellar (Rockafellar, 1970). Although Lagrange duality is more general, Fenchel-Rockafellar duality is tailored for problems of the form of Problem 8.

Definition 26 (Fenchel conjugate)**.** For a function $f : \mathbb{R}^d \to [-\infty, +\infty]$, its Fenchel conjugate is the function $f^\star : \mathbb{R}^d \to [-\infty, +\infty]$

$$f^\star(y) = \sup_{x \in \operatorname{dom}(f)} \left(y^\top x - f(x)\right) \ . \tag{30}$$

The Fenchel conjugate is a tool that naturally arises in duality theory. The conjugate $f^\star$ is always a convex function as the pointwise supremum of convex (affine) functions.

Example 27 (Fenchel conjugate of a norm, Bach et al. (2011, Proposition 1.4))**.** Let Ω be a norm on $\mathbb{R}^p$. The Fenchel conjugate of Ω is the indicator function of the unit ball associated to its dual norm Ω^*.

$$\sup_{\beta\in\mathbb{R}^p} \left(z^\top \beta - \Omega(\beta)\right) = \iota_{\Omega^*}(\beta) = \begin{cases} 0 & \text{if } \Omega^*(z) \leq 1 \ , \\ +\infty & \text{otherwise .} \end{cases} \tag{31}$$

Theorem 28 (Fenchel-Young inequality, Beck (2017, Theorem 4.6))**.** *Let f be a proper convex function on $\mathbb{R}^p$. Let $\beta \in \mathbb{R}^p$ and $z \in \mathbb{R}^p$. Then,*

$$f(\beta) + f^\star(z) \geq \beta^\top z \tag{32}$$

with equality if and only if $z \in \partial f(\beta)$.

Theorem 29 (Fenchel-Rockafellar duality, Bauschke and Combettes (2017, Definition 15.19))**.** *Let $X \in \mathbb{R}^{n\times p}$ be a linear operator. Let $f : \mathbb{R}^n \to]-\infty, +\infty]$ and $g : \mathbb{R}^p \to [-\infty, +\infty]$ be convex functions. Consider the following primal problem*

$$p^* \triangleq \inf_{\beta\in\mathbb{R}^p} f(X\beta) + g(\beta) \ . \tag{33}$$

Then the dual problem of Problem 33 reads

$$d^* \triangleq \sup_{\theta\in\mathbb{R}^n} -f^\star(\theta) - g^\star(-X^\top\theta) \ , \tag{34}$$

where $f^\star$ (respectively $g^\star$) is the Fenchel conjugate of f (respectively g).

By Fenchel-Young inequality, it is clear that the primal is lower-bounded by the dual.

Proposition 30 (Weak duality, Bauschke and Combettes (2017, Proposition 15.18))**.** *The primal is always lower-bounded by the dual, that is*

$$p^* \geq d^* \ . \tag{35}$$

Strong duality. Based on Proposition 30, we define the duality gap $\mathcal{G}^* := p^* - d^* \geq 0$, holding at the optimum. For convex optimization problems, the duality gap can be null under a constraint qualification condition, a situation called strong duality Boyd and Vandenberghe (2004, Paragraph 5.3.2). For many sparse GLMs, strong duality holds making the lower bound tight. Practically speaking, strong duality implies that there are arbitrarily good certificates of suboptimality. This observation can be used heuristically to provide a stopping criterion to optimization algorithms. Given a tolerance $\epsilon > 0$, if at iteration k one can construct a dual feasible point $\theta^{(k)} \in \mathbb{R}^n$ such that the suboptimality gap $\mathcal{G}(\beta^{(k)}, \theta^{(k)}) \leq \epsilon$, then $\beta^{(k)} \in \mathbb{R}^p$ is guaranteed to be ϵ-suboptimal.

Example 31 (Lasso dual)**.** The primal problem of the Lasso reads

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1 \ , \tag{36}$$

and its dual problem is

$$\max_{\theta \in \Delta_X} \quad \frac{1}{2}\|y\|_2^2 - \frac{1}{2}\|\theta + y\|_2^2 \quad \text{s.t.} \quad \Delta_X \triangleq \{\theta \in \mathbb{R}^n : \ \|X^\top \theta\|_\infty \leq \lambda\} \ . \tag{37}$$

Proof. Let $f(u) \triangleq \frac{1}{2}\|y - u\|_2^2$ and $\psi(u) \triangleq \frac{1}{2}\|u\|_2^2$. We refer to Beck (2017, Section 4.3) for a comprehensive review of calculus rules associated to Fenchel conjugates. Using the fact that ψ is self-conjugate and that $f(u) = \psi(y - u)$, it follows that

$$f^*(u) = \frac{1}{2}\|u\|_2^2 + \langle u, y\rangle = -\frac{1}{2}\|y\|_2^2 + \frac{1}{2}\|u + y\|_2^2 \ . \tag{38}$$

Besides, $g(u) \triangleq \lambda\|u\|_1$. Using Example 27, it follows that

$$g^*(u) = \begin{cases} 0 & \text{if } \|u\|_\infty \leq \lambda \ , \\ \infty & \text{otherwise} \ . \end{cases} \tag{39}$$

Now, applying Theorem 29, and using a slight abuse of notation for the supremum, we obtain the desired result. □

Table 2: Primal and dual problems for sparse generalized linear models.

Model	**Primal**	**Dual**
Lasso	$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1$	$\max_{\theta \in \Delta_X} \frac{1}{2}\|y\|_2^2 - \frac{1}{2}\|\theta + y\|_2^2$ s.t. $\Delta_X = \{\theta \in \mathbb{R}^n : \|X^\top \theta\|_\infty \leq \lambda\}$
Group Lasso	$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \sum_{g \in \mathcal{G}} \|\beta_g\|_2$	$\max_{\theta \in \Delta_X} \frac{1}{2}\|y\|_2^2 - \frac{1}{2}\|\theta + y\|_2^2$ s.t. $\Delta_X = \{\theta \in \mathbb{R}^n : \max_{g \in \mathcal{G}} \|X_g^\top \theta\|_2 \leq \lambda\}$
Multi-task Lasso	$\min_{\mathbf{B} \in \mathbb{R}^{p \times T}} \frac{1}{2}\|\mathbf{Y} - X\mathbf{B}\|_\mathrm{F}^2 + \lambda\|\mathbf{B}\|_{2,1}$	$\max_{\mathbf{\Theta} \in \Delta_X} \frac{1}{2}\|\mathbf{Y}\|_\mathrm{F}^2 - \frac{1}{2}\|\mathbf{\Theta} + \mathbf{Y}\|_\mathrm{F}^2$ s.t. $\Delta_X = \{\mathbf{\Theta} \in \mathbb{R}^{n \times T} : \|X^\top \mathbf{\Theta}\|_{2,\infty} \leq \lambda\}$
Elastic Net	$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\left(\gamma\|\beta\|_1 + \frac{1-\gamma}{2}\|\beta\|_2^2\right)$	$\max_{\theta \in \Delta_X} -\frac{1}{2}\left(1 + \frac{1}{\lambda(1-\gamma)}\right)\|\theta\|_2^2 - \langle \theta, y\rangle$ s.t. $\Delta_X = \{\theta \in \mathbb{R}^n : \|X^\top \theta\|_\infty \leq \lambda\gamma\}$
Logistic Regression	$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^n \log(1 + \exp(-y_i X_{i,:}^\top \beta)) + \lambda\|\beta\|_1$	$\min_{\theta \in \Delta_X} \sum_{i=1}^n \theta_i \log(\theta_i) + (1 - \theta_i)\log(1 - \theta_i)$ s.t. $\begin{cases} \Delta_X = \{\theta \in \mathbb{R}^n : \|X^\top \theta\|_\infty \leq \lambda\} \\ \forall i \in [n], \ \ 0 \leq \theta_i \leq 1 \end{cases}$

3.2 First-order methods for non-smooth optimization

As we are now equipped with the right tools to tackle non-smooth optimization, we move on to first-order methods. We begin this section with some theoretical results on the descent lemma. Then, we present some of the most efficient first-order methods to solve sparse generalized linear models and derive their convergence rates.

3.2.1 Lipschitz gradients and descent lemma

Gradient descent is an iterative scheme used to solve minimization problems. Since the gradient gives the direction of the steepest increase in value, a natural approach consists in taking a step in the opposite direction of the gradient.

Definition 32 (Gradient descent)**.** Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function with L-Lipschitz gradients. The gradient descent method generates iterates $(x_k)_k$ defined in $\mathbb{R}^d$ by the relationship

$$x_{k+1} = x_k - \eta_k \nabla f(x_k) \ , \tag{40}$$

where $(\eta_k)_k$ is a sequence of step sizes.

We shall see that these step sizes are crucial for the convergence speed of first-order methods. In this subsection, we present the majorization-minimization framework and prove the convergence of gradient descent for L-smooth functions. Starting from Assumption 10, we derive a quadratic upper bound for a L-smooth function.

Theorem 33 (Descent lemma)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex L-smooth function with $L > 0$. A gradient descent iterate scheme with $\frac{1}{L}$ step sizes yields*

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2L} \|\nabla f(x_k)\|_2^2 \ . \tag{41}$$

Proof. For $x, y \in \mathbb{R}^d$, using the fundamental theorem of calculus,

$$f(y) = f(x) + \int_0^1 \langle \nabla f(x + t(y - x)), y - x \rangle \, \mathrm{d}t \ . \tag{42}$$

It follows that

$$\begin{aligned}
|f(y) - f(x) - \langle \nabla f(x), y - x \rangle| &= \left| \int_0^1 \langle \nabla f(x + t(y - x)) - \nabla f(x), y - x \rangle \, \mathrm{d}t \right| \\
&\leq \int_0^1 |\langle \nabla f(x + t(y - x)) - \nabla f(x), y - x \rangle| \, \mathrm{d}t \quad \text{by Jensen's inequality,} \\
&\leq \int_0^1 \|\nabla f(x + t(y - x)) - \nabla f(x)\|_2 \, \|y - x\|_2 \, \mathrm{d}t \quad \text{by Cauchy-Schwarz inequality,} \\
&\leq L \int_0^1 t \, \|x - y\|_2^2 \, \mathrm{d}t \quad \text{by } L\text{-smoothness,} \\
&\leq \frac{L}{2} \|x - y\|_2^2 \ .
\end{aligned}$$

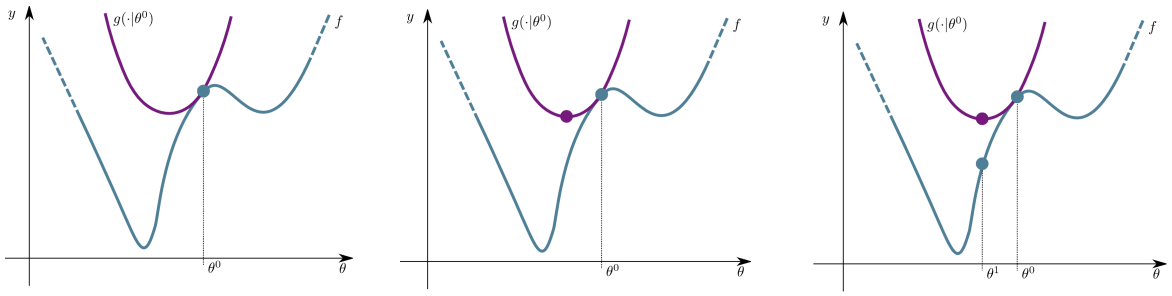
Since f is convex, it lies above its tangents and we obtain a quadratic upper bound for f

$$\forall x, y \in \mathbb{R}^d, \ \ f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|x - y\|_2^2 \ . \tag{43}$$

Note that this upper bound can be interpreted as a quadratic approximation of f. It is very close to a Taylor expansion except that $\nabla^2 f(x)$ is replaced by an isotropic matrix LI_n. Equation (43) upper-bounds f as shown in Figure 6. Using Equation (43), we can now plug the gradient descent update rule $x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$ with $y = x_{k+1}$ and $x = x_k$

$$\begin{aligned}
f(x_{k+1}) &\leq f(x_k) + \langle \nabla f(x_k), -\frac{1}{L} \nabla f(x_k) \rangle + \frac{L}{2} \left\| \frac{1}{L} \nabla f(x_k) \right\|_2^2 \\
&\leq f(x_k) - \frac{1}{2L} \|\nabla f(x_k)\|_2^2 \ .
\end{aligned}$$

□



(a) Step 1. (b) Step 2. (c) Step 3.

Figure 6: **Majorization-minimization framework.** Majorize with a quadratic upper bound, minimize this bound and perform the update. Courtesy from Salmon.

We obtain a theoretical guarantee of minimization of the objective function for gradient descent. This lemma implies that every time a gradient descent update is performed, the objective function f decreases by at least $\frac{1}{2L}\|\nabla f(x_k)\|_2^2$.

Majorization-minimization (MM) is a useful framework to visualize how gradient descent converges to an optimum (Figure 6). MM consists in iteratively minimizing at step k a surrogate majorizing function $g(\cdot|x_k)$ that satisfies

$$\begin{cases} \forall x \in \mathbb{R}^d, \;\; f(x) \leq g(x|x_k) & \text{: domination / upper bound ,} \\ f(x_k) = g(x_k|x_k) & \text{: tangency / tightness at } x_k \text{ .} \end{cases} \tag{44}$$

For functions with L-Lipschitz gradients, the upper bound is given by Equation (43). Taking the gradient of $g(x|x_k) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{L}{2}\|x_k - x\|_2^2$, finding a critical point of g and reorganizing yields the update rule

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k) \;\; . \tag{45}$$

This update rule makes it clear why the Lipschitz constant is ubiquitous when minimizing a L-smooth function: it is the optimal step size to minimize in one step the quadratic upper bound of f.

3.2.2 Proximal gradient descent

The previous section introduces the majorization-minimization framework in the context of smooth functions. For sparsity-inducing penalties which create non-smooth optimization problems, one needs to come up with a variant of gradient descent to successfully minimize a function. Going back to a problem of the form of Problem 8, F is assumed convex, L-smooth and Ω is convex but non-smooth. Since F is L-smooth, using the quadratic upper bound Equation (43) for Problem 8 yields a possible choice:

$$\forall y \in \mathbb{R}^d, \;\; g(y|\beta_k) = F(\beta_k) + \langle \nabla F(\beta_k), y - \beta_k \rangle + \frac{L}{2}\|y - \beta_k\|_2^2 + \Omega(y) \;\; . \tag{46}$$

Using Equation (45), the minimizer of the quadratic upper bound of F is a proximal gradient step of size $\frac{1}{L}$:

$$\begin{aligned} \beta_{k+1} &= \arg\min_{y \in \mathbb{R}^d} \left(F(\beta_k) + \langle \nabla F(\beta_k), y - \beta_k \rangle + \frac{L}{2}\|y - \beta_k\|_2^2 + \Omega(y) \right) \\ &= \arg\min_{y \in \mathbb{R}^d} \left(\Omega(y) + \frac{L}{2} \left\| y - \left(\beta_k - \frac{1}{L}\nabla f(\beta_k) \right) \right\|_2^2 \right) \\ &= \text{prox}_{\frac{\Omega}{L}} \left(\beta_k - \frac{1}{L}\nabla f(\beta_k) \right) \;\; . \end{aligned}$$

For most well-known problems, the proximal operator has a closed-form which usually makes it cheap to evaluate. For instance, combining the soft thresholding operator Equation (29) with proximal gradient descent yields the celebrated iterative soft-thresholding algorithm (ISTA, Daubechies et al. (2003)) used to minimize a ℓ_1-regularized problem.

A central question for descent algorithms solving sparse GLM optimization problems is the identification of the support in finite time. Remember that sparse models generate solutions living in lower-dimensional spaces than the original feature space: a sparse logistic regression model or a Lasso model generates solutions having few non-zero coefficients.

Definition 34 (Generalized support, Nutini et al. (2019))**.** For a penalty function $g_j, j \in [p]$ and a vector $\beta \in \mathbb{R}^p$, its generalized support $\mathcal{S}_\beta \subseteq [p]$ is the set of indices $j \in [p]$ such that g_j is differentiable at β_j

$$\mathcal{S}_\beta = \{j \in [p] : \partial g_j(\beta_j) \text{ is a singleton}\} \ . \tag{47}$$

For the ℓ_1-norm, the support corresponds to the set of non-zero coefficients. For the dual SVM with Hinge loss (see Appendix A), it corresponds to the set of indices such that $\beta_j \in]0, C[$. The model identification question consists in asking if there exists a number of steps $K > 0$ at which $\mathcal{S}_{\beta^{(K)}} \subseteq \mathcal{S}_{\hat{\beta}}$. This property is crucial to ensure that true null coefficients eventually vanish after a finite number of steps.

Convergence rate. We give a simple proof of the convergence rate of (proximal) gradient descent. Note that proximal gradient descent and gradient descent have the same convergence rate, the main difference lying in the additional overhead of evaluating the proximal operator. These proofs are established using ordinary differential equation (ODE) theory. The connection between ODEs and optimization has been established by taking infinitesimal step sizes such that the trajectory taken by gradient descent iterates converges to a curve modeled by an ODE. Lyapunov functions are important functions in the theory of ODEs to prove the stability of an equilibrium of an ODE. They are used by d'Aspremont et al. (2021) or Bansal and Gupta (2017) to derive the convergence rates of gradient descent in various settings.

Definition 35 (Lyapunov potential)**.** Let f be a convex function over $\mathbb{R}^d$ and $\beta^\star \in \arg\min_\beta f(\beta)$. The Lyapunov potential (or energy) associated to f given $a_k \geq 0$ is the sequence $(\phi_k)_k$ defined by

$$\phi_k \triangleq a_k(f(\beta_k) - f(\beta^*)) + \frac{L}{2}\|\beta_k - \beta^*\|_2^2 \ . \tag{48}$$

Theorem 36 (Potential inequality, d'Aspremont et al. (2021, Theorem 4.2))**.** *Let f be an L-smooth convex function, $\beta^* \in \arg\min_\beta f(\beta)$, and $k \in \mathbb{N}$. Using the sequence of descent iterates $(\beta_k)_k$, it holds that $\phi_{k+1} \leq \phi_k$, with $\beta_{k+1} = \beta_k - \frac{1}{L}\nabla f(\beta_k)$ and $a_{k+1} = 1 + a_k$.*

Using Theorem 36 with $a_k = k$, it follows that

$$N(f(\beta_N) - f(\beta^*)) \leq \phi_N \leq \phi_{N-1} \leq \cdots \leq \phi_0 = \frac{L}{2}\|\beta_0 - \beta^*\|_2^2 \ , \tag{49}$$

which yields

$$f(\beta_N) - f(\beta^*) \leq \frac{L\|\beta_0 - \beta^*\|_2^2}{2N} \ . \tag{50}$$

This implies that proximal gradient descent has a convergence rate $O(1/N)$, with N the number of iterations, or equivalently that at most $O(1/\epsilon)$ iterations are needed to obtain an ϵ-suboptimal solution.

Now, we add Assumption 11 on the objective function: f is μ-strongly convex ($\mu > 0$). Again, potential functions prove to be very useful tools to derive this convergence rate. This time, for $a_k \geq 0$, let $\phi_k \triangleq a_k(f(\beta_k) - f(\beta^*)) + \frac{L+\mu a_k}{2}\|\beta_k - \beta^*\|_2^2$.

Theorem 37 (Potential inequality μ-convex, d'Aspremont et al. (2021, Theorem 4.10))**.** *Let f be an L-smooth μ-strongly convex function, $\beta^* \in \arg\min_\beta f(\beta)$, and $k \in \mathbb{N}$. For a sequence of descent iterates $(\beta_k)_k$, it holds that $\phi_{k+1} \leq \phi_k$, with $\beta_{k+1} = \beta_k - \frac{1}{L}\nabla f(\beta_k)$ and $a_{k+1} = \frac{1+a_k}{1-\frac{\mu}{L}}$.*

Following the same argument as previously, we obtain the following convergence rate for a fixed step size $\frac{1}{L}$,

$$f(\beta_N) - f(\beta^*) \leq \frac{\mu\|\beta_0 - \beta^*\|_2^2}{2((1-\mu/L)^{-N} - 1)} \ , \tag{51}$$

which implies that proximal gradient descent has a convergence rate $O(\log 1/\epsilon)$ for an ϵ-suboptimal solution. Adding strong convexity essentially shows that the convergence is much faster because the function cannot get too flat, since it is lower bounded by parabolas.

3.2.3 Coordinate descent

We end the review of optimization algorithms by studying coordinate descent (Luo and Tseng, 1992). The central idea in coordinate descent is to solve coordinate-wise simple subproblems iteratively until convergence. Denoting $\Phi : \mathbb{R}^p \to \mathbb{R}$ a convex objective function of the form of Problem 8, coordinate descent successively minimize approximately one-dimensional functions $\Phi_{|\beta_j} : \mathbb{R} \to \mathbb{R}$, updating one coordinate at a time while keeping others unchanged. These subproblems are usually easier and cheaper to solve. Coordinate descent methods have been extensively studied in the literature and extended to non-smooth separable problems (Richtárik and Takáăź, 2014; Fercoq and Richtárik, 2013), such as Lasso (Tibshirani, 1996) or ElasticNet (Zou and Hastie, 2005). In the context of non-smooth optimization, a coordinate-wise proximal gradient descent is performed for every coordinate at every iteration. We give below details about proximal gradient descent and proximal coordinate descent.

Algorithm 1 PROXIMAL GRADIENT DESCENT

input : $n_{\text{iter}} \in \mathbb{N}, \beta^{(0)} \in \mathbb{R}^p, L \in \mathbb{R}^*$

for $k = 0, \dots, n_{\text{iter}}$ **do**

  $\beta^{(k+1)} = \text{prox}_{\frac{g}{L}}\left(\beta^{(k)} - \frac{1}{L}\nabla f(\beta^{(k)})\right)$

return $\beta^{(n_{\text{iter}}+1)}$

Algorithm 2 PROXIMAL COORDINATE DESCENT

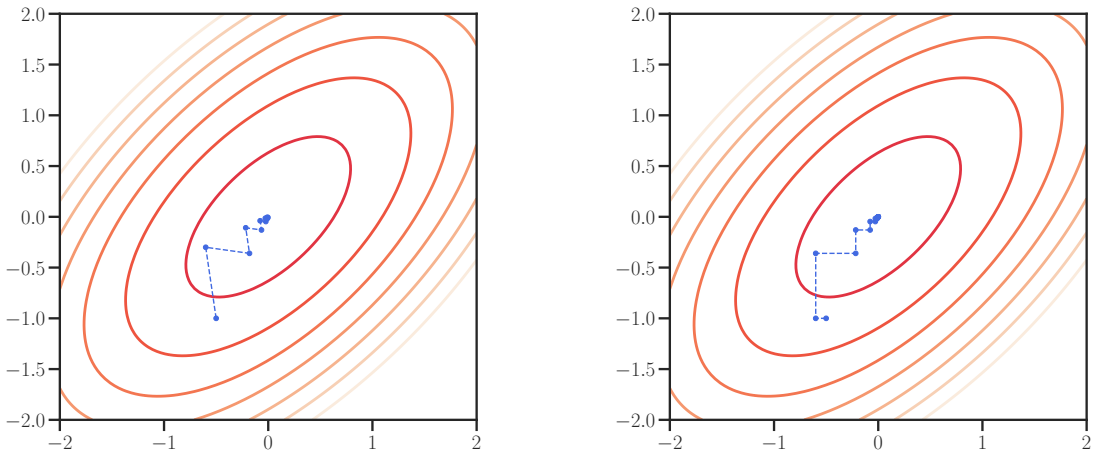
input : $n_{\text{iter}} \in \mathbb{N}, \beta^{(0)} \in \mathbb{R}^p, L \in \mathbb{R}^p$

for $k = 0, \dots, n_{\text{iter}}$ **do**

  for $j = 1, \dots, p$ **do**

    $\beta_j^{(k+1)} = \text{prox}_{\frac{g_j}{L_j}}\left(\beta_j^{(k)} - \frac{1}{L_j}\nabla_j f(\beta^{(k)})\right)$

return $\beta^{(n_{\text{iter}}+1)}$



(a) Gradient descent.

(b) Coordinate descent.

Figure 7: **Contour plots of objectives and descent iterates.** Gradient descent takes steps orthogonal to the level sets of the objective (Figure 7a), while coordinate descent takes steps in the directions of the axes (Figure 7b).

The performance of coordinate descent is in part dictated by the coordinate selection strategy. Features to be updated can be selected in different ways, which leads to the question of the most optimal strategy.

Cyclic. The most trivial strategy consists in cycling through the coordinates. At the k-th iteration, the strategy consists in

$$\text{Pick} \quad j_k = (k \pmod p) + 1 \ . \tag{52}$$

It has been explored by Tseng and Yun (2009) and theoretically analyzed by Beck et al. (2015). It is the strategy used in practice by `skglm`.

Random. Another strategy consists in picking an index using a random distribution (Nesterov, 2012). The coordinates are then drawn according to a pre-defined distribution: a standard choice is to uniformly sample among the p coordinates.

$$\text{Pick} \quad j_k = j \quad \text{with probability} \quad \frac{1}{p}, \quad \forall j \in [p] \ . \tag{53}$$

Various experiments (Shi et al., 2016) show that the random strategy consistently underperforms cycling through the coordinates. Nonetheless, in the worst case, Sun and Ye (2016) prove that a random strategy is better than cycling through the entire feature set.

Greedy. Indices can be chosen according to a heuristic (Nutini et al., 2015): the index with the largest decrease in the objective (Maximum Block Improvement, Chen et al. (2012)), with the largest gradient magnitude (Gauss-Southwell strategy, Southwell (1941)). This strategy differs from the two other as it dynamically identifies a precise coordinate to optimize. A comprehensive presentation of these strategies has been made by Massias (2017, Section 3.2). For a quadratic datafit, updates can be performed using a Gram matrix $X^\top X$, making the cost of updating coordinates cheap. In this setting, speed-ups over the naïve cyclic approach have been observed for Lasso-type problems. However, for non-quadratic datafits, the Greedy approach is computationally very expensive, making it slow.

Even though index selection strategy can reduce the number of epochs, a poor implementation of coordinate descent can lead to suboptimal results. Indeed, for a quadratic datafit, a coordinate update can essentially be updated in two fashions: updating a Gram matrix and updating the residuals. Remember a single coordinate descent update rule for a quadratic datafit, a step size $L_j = \|X_{:j}\|^2$ and a proper lower-semicontinuous penalty g reads

$$\beta_{j+1} = \mathbf{prox}_{\frac{g_j}{L_j}}(\beta_j - \frac{1}{L_j} X_{:j}^\top (X\beta - y)) \ . \tag{54}$$

Residual update. The usual way of updating coordinates consists in jointly updating the residuals. The update reads

$$\begin{cases} r & \leftarrow r^{(k-1)} + X_{:,j}\beta_j^{(k-1)} \qquad \text{if } \beta_j^{(k-1)} \neq 0 \\ \beta_j^{(k)} & \leftarrow \mathbf{prox}_{\frac{g_j}{L_j}}(\frac{1}{L_j} X_{:,j}^\top r) \qquad . \\ r^{(k)} & \leftarrow r - X_{:,j}\beta_j^{(k)} \qquad \text{if } \beta_j^{(k)} \neq 0 \end{cases} \tag{55}$$

This update is efficient when $p \gg n$.

Gradient update. The gradient update consists in observing that the gradient of a quadratic datafit relies on the Gram matrix $G = X^\top X$. This matrix lives in $\mathbb{R}^{p \times p}$, and can provide significant speed-ups if pre-computed. Indeed, for moderate-sized datasets (typically $p \leq 10\,000$), the Gram matrix fits in memory and can be pre-computed in reasonable time. This initial cost of pre-computation is then amortized in the faster coordinate updates. Instead of storing the residuals $r \in \mathbb{R}^n$, the gradient $\nabla f \in \mathbb{R}^p$ is maintained. Then the update reads

$$\begin{cases} \delta\beta_j^{(k)} & \leftarrow \mathbf{prox}_{\frac{g_j}{L_j}}(\beta_j^{(k-1)} - \frac{1}{L_j}\nabla_j^{(k-1)} f) - \beta_j^{(k-1)} \\ \beta_j^{(k)} & \leftarrow \beta_j^{(k-1)} + \delta\beta_j \qquad \text{if } \delta\beta_j \neq 0 \ . \\ \nabla^{(k)} f & \leftarrow \nabla^{(k-1)} f + G_j \delta\beta \qquad \text{if } \delta\beta_j \neq 0 \end{cases} \tag{56}$$

The main cost associated to this strategy is the update of the p-dimensional vector ∇f. It is particularly interesting when $n \gg p$: it is usually cheaper to store the Gram matrix than a vector of residuals and the single coordinate update are faster. `skglm` embarks such a strategy for quadratic datafits when $p \leq 10\,000$ and $n \gg p$. A detailed time and space analyses can be found in Massias (2017, Table 1).

3.3 Comparison of non-accelerated first-order methods

3.3.1 Experiments

We compare the performance of coordinate descent and proximal gradient descent on multiple datasets and estimators: the Lasso, the sparse logistic regression and the ElasticNet. The benchmark procedure is explained in details in Appendix C and was performed using the benchmarking tool `Benchopt` (Moreau et al., 2022). We use datasets from `libsvm` (Chang and Lin, 2011), presented in Table 3.

Parametrization. In the sparse GLM literature, the regularization hyperparameter $\lambda \geq 0$ is usually parametrized as a fraction of $\lambda_{\max}$, the smallest regularization value for which $\hat{\beta} = 0$.

Table 3: Characteristics of the datasets.

Datasets	#samples n	#features p	density
leukemia	38	7129	1
rcv1	20 242	19 959	3.6×10^{-3}
finance	16 087	4 272 227	1.4×10^{-3}
news20	19 996	1 355 191	3.4×10^{-4}

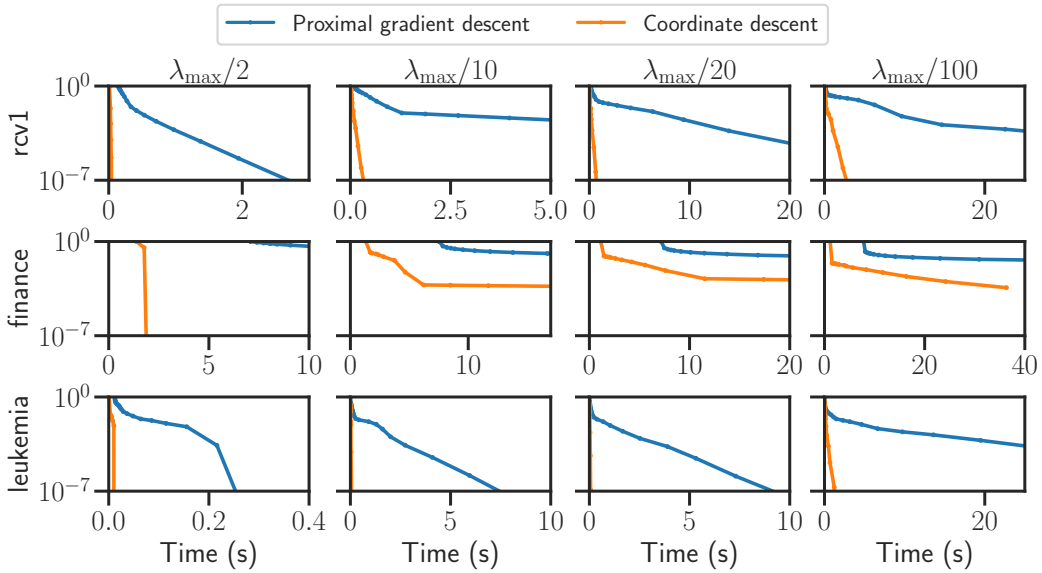


Figure 8: **Lasso.** Duality gap as a function of time, on 3 different datasets: *leukemia*, *finance* and *rcv1*.

Lasso. Figures 8 to 10 make it clear that coordinate descent offers far better performance than proximal gradient descent. In low-regularization regime, proximal gradient descent does not converge in reasonable time. We give an explanation of the superior performance of coordinate descent over proximal gradient descent in Section 3.3.2. Note that on high-dimensional datasets like *finance*, for $\lambda \leq \frac{\lambda_{\max}}{10}$, coordinate descent struggles to converge. This fact calls for acceleration techniques to ensure a faster convergence in high-dimensional settings. Such techniques are investigated in the next section.

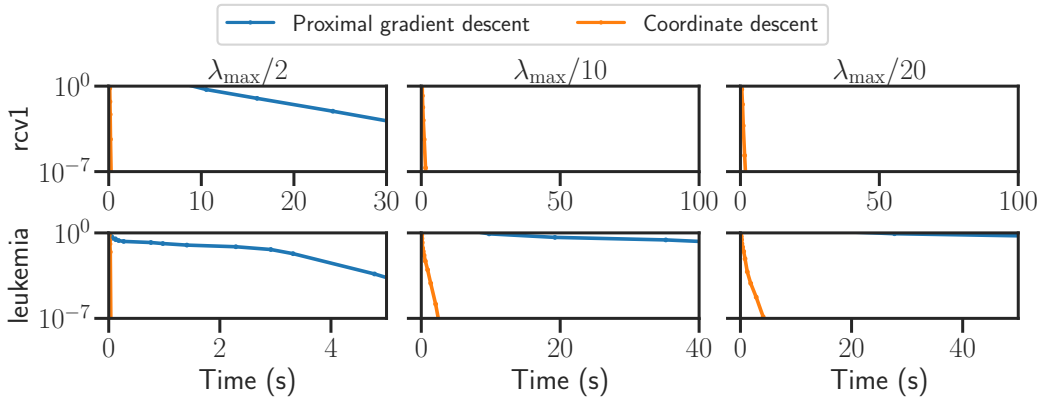


Figure 9: ℓ_1**-regularized logistic regression, suboptimality.** Suboptimality gap as a function of time, on 2 different datasets: *leukemia* and *rcv1*.

Sparse logistic regression. The sparse logistic regression corresponds to Problem 10.

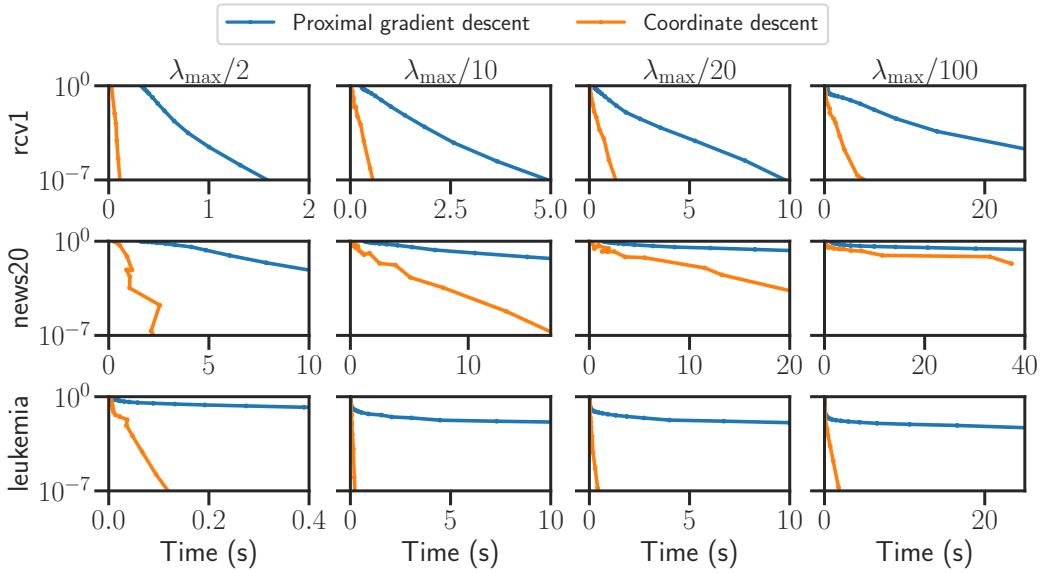


Figure 10: **ElasticNet, suboptimality.** Suboptimality gap as a function of time, on 3 different datasets: *rcv1*, *news20* and *leukemia*. $\gamma = 0.7$.

ElasticNet. The ElasticNet is an estimator with a quadratic datafit and a $\ell_1 + \ell_2$-regularization. For a design matrix $X \in \mathbb{R}^{n \times p}$ and a target vector $y \in \mathbb{R}^n$, it is defined as

$$\arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \left(\gamma \|\beta\|_1 + \frac{(1-\gamma)}{2} \|\beta\|_2^2 \right) \quad , \tag{57}$$

where $\gamma \in [0, 1]$ controls the ratio of ℓ_1-regularization compared to ℓ_2-regularization.

3.3.2 Why is coordinate descent faster than proximal gradient descent?

Proposition 38. *Let $f : \mathbb{R}^p \to \mathbb{R}$ be a quadratic datafit and $g : \mathbb{R}^p \to \mathbb{R}$ be a proper lower semicontinuous function. The step sizes $\{\frac{1}{L_k} : k = 1, \dots, p\}$ associated to the coordinate descent scheme to optimize the problem $f + g$ are larger than the step size $\frac{1}{L}$ of the proximal gradient descent scheme associated to the same objective.*

Proof. Let $X \in \mathbb{R}^{n \times p}$ be a design matrix. $\|X_{:j}\| = \|Xe_j\| \leq \sup_{\|u\| \leq 1} \|Xu\| = \|X\|_2$. Therefore, $\frac{1}{\|X_{:,j}\|_2^2} \geq \frac{1}{\|X\|_2^2}$, which shows that a CD step is larger than the generic proximal gradient descent step. □

Intuitively, one could say that coordinate descent has access to more information than proximal gradient descent. While proximal gradient descent has access to a step size common to all coordinates, coordinate descent can use step sizes proportional to the steepness of the curve in every direction. To emphasize the criticality of larger step sizes, we carry out the following experiment. We run multiple coordinate descent algorithms, each time choosing a step size between $\frac{1}{L}$ and $\frac{1}{L_j}$. Figure 11 makes it clear that coordinate descent run with gradient descent steps is as slow as gradient descent.

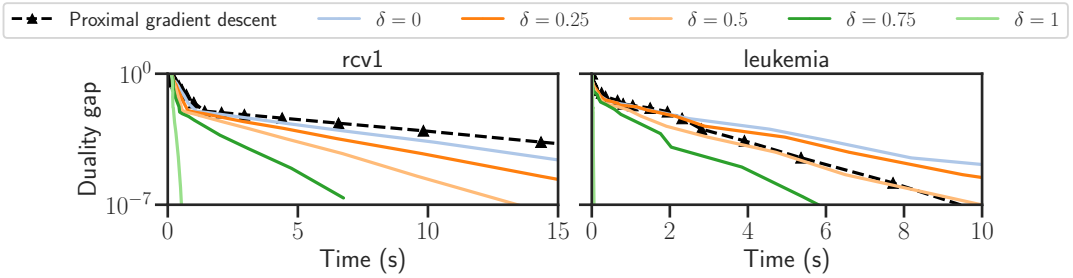
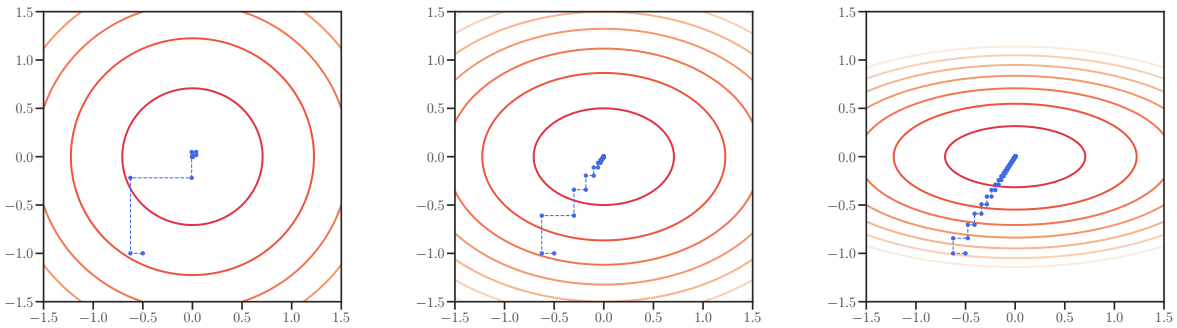


Figure 11: **Impact of step sizes on CD convergence speed.** Duality gap as a function of time. PGD compared to CD with step sizes $\gamma_j = \frac{\delta}{L_j} + \frac{1-\delta}{L}$, for multiple values of δ. The experiment is carried out with an ordinary least-square estimator.

3.3.3 Condition numbers and convergence

As seen previously, the step size has a crucial impact on the convergence speed, and should be carefully adapted if possible to ensure the fastest convergence possible. In this subsection, we focus on the quadratic datafit and give intuition to the reader on the critical aspect of a well-conditioned problem.



(a) $\kappa = 1$ (b) $\kappa = 2$ (c) $\kappa = 5$

Figure 12: **Impact of the condition number on coordinate descent convergence.**

Proposition 39. *Considering a sequence of iterates $(\beta^k)_k$ of a (proximal) gradient descent schema, the update rule reads*

$$\beta^{k+1} \leftarrow \underbrace{\left(I_p - \frac{1}{L} X^\top X\right)}_{\triangleq T_{\mathrm{GD}}} \beta^k - \frac{1}{L} X^\top y \ . \tag{58}$$

Proposition 40. *For a quadratic datafit, the iteration matrix T_{GD} has a spectral radius*

$$\rho(T_{\mathrm{GD}}) < 1 \ . \tag{59}$$

Proof. By assumption, the eigenvalues of the Hessian $\nabla_f^2 = X^\top X$ are bounded above by $L > 0$ and below by $\mu > 0$. These eigenvalues are all positive since the quadratic objective is μ-strongly convex. Therefore the spectral radius $\rho(T_{\mathrm{GD}}) = 1 - \frac{\mu}{L} < 1$ since $\frac{\mu}{L} > 0$. □

Therefore, the sequence $\left(\beta^k\right)_{k \geq 0}$ converges to a fixed point, which is a minimizer of Problem 8 (Polyak, 1987, Theorem 1, Section 2.1.2). The inverse of the condition number $\kappa = \frac{L}{\mu}$ controls the convergence speed of $\left(\beta^{(k)}\right)_{k \geq 0}$. The closer to 1, the more well-conditioned the objective function, as suggested by Figure 12.

Similar results have been proven for coordinate descent by Bertrand (2021, Lemma 1.7 and 1.8).

4 Accelerating first-order methods for faster convergence

4.1 Acceleration techniques

It is now clear why the backbone solver in `skglm` is coordinate descent. The experiments of Section 3.3.1 hint that "vanilla" coordinate descent is not sufficient to ensure high performance in large dimensional settings. Acceleration techniques have to be integrated to `skglm` in order to achieve state-of-the-art performance. Building on the seminal work of Nesterov (1983), numerous acceleration techniques have been developed over the years, provably refining convergence rates as well as providing practical speed ups. We shall differentiate two distinct kinds of acceleration: inertial acceleration à la Nesterov and non-linear extrapolation. In this section, we identify the differences and answer practical questions regarding the implementations of these acceleration techniques.
We first study acceleration and prove an improvement in the convergence rate of CD. Then we present screening rules (or *backward* techniques) where features are progressively removed from the problem, then working set techniques (or *forward* techniques) where more and more features are added to the problem. Finally, we perform an ablation study to demonstrate the effect of each presented technique on the convergence speed of solvers addressing convex problems.

4.1.1 Inertial acceleration

Following the seminal work by Polyak (1964) introducing the heavy ball method (Algorithm 3), Nesterov (1983) presented an accelerated gradient descent method achieving a $\mathcal{O}(1/N^2)$ convergence rate. This work was further adapted by Beck and Teboulle (2009) who presented FISTA, an accelerated optimization algorithm for non-smooth composite problems. As for accelerated coordinate descent, inertial accelerated versions of coordinate descent (Nesterov, 2012; Lin et al., 2014; Fercoq and Richtárik, 2013) achieve $\mathcal{O}(1/N^2)$ rates.

Nesterov (1983) replaced the fixed inertia hyperparameter from the heavy ball method by a sequence of iterates.

Algorithm 3 HEAVY BALL METHOD, (Polyak, 1964)

input : $n_{\text{iter}} \in \mathbb{N}, \alpha \in [0,1], \gamma > 0$
init : $x^{(0)} = y^{(0)} = 0_{\mathbb{R}^p}$
for $k = 0, \dots, n_{iter}$ **do**
  $y^{(k)} = x^{(k)} + \alpha(x^{(k)} - x^{(k-1)})$ `// Inertia`
  $x^{(k+1)} = y^{(k)} - \gamma \nabla F(x^{(k)})$
return $x^{(n_{iter}+1)}$

Algorithm 4 Nesterov (1983)

input : $n_{\text{iter}} \in \mathbb{N}, \gamma > 0$
init : $x^{(0)} = y^{(0)} = 0_{\mathbb{R}^p}, t^{(1)} = 1$
for $k = 0, \dots, n_{iter}$ **do**
  $t^{(k+1)} = \frac{1+\sqrt{1+4(t^{(k)})^2}}{2}$
  $y^{(k)} = x^{(k)} + \frac{t^{(k)}-1}{t^{(k+1)}}(x^{(k)} - x^{(k-1)})$
  $x^{(k+1)} = y^{(k)} - \gamma \nabla F(y^{(k)})$
return $x^{(n_{iter}+1)}$

Example 41 (FISTA)**.** In Section 3.2.2, we studied ISTA, a proximal descent algorithm solving a ℓ_1-regularized optimization problem. Nesterov acceleration has been adapted to ISTA yielding FISTA (Beck and Teboulle, 2009).

Algorithm 5 FISTA (Beck and Teboulle, 2009)

input : $n_{\text{iter}} \in \mathbb{N}, L \in \mathbb{R}^*$
init : $\beta^{(1)} = z^{(1)} = 0_{\mathbb{R}^p}, \gamma^{(1)} = 1$
for $k = 1, \dots, n_{iter}$ **do**
  $\beta^{(k+1)} = \text{prox}_{\frac{g}{L}}\left(z^{(k)} - \frac{1}{L}\nabla f(z^{(k)})\right)$
  $\gamma^{(k+1)} = \frac{1+\sqrt{1+4(\gamma^{(k)})^2}}{2}$
  $z^{(k+1)} = \beta^{(k+1)} + \frac{\gamma^{(k)}-1}{\gamma^{(k+1)}}(\beta^{(k+1)} - \beta^{(k)})$
return $\beta^{(n_{iter}+1)}$

Convergence rate. To motivate the choice of Nesterov accelerated gradient descent, we rely on potential functions (Bansal and Gupta, 2017). d'Aspremont et al. (2021) starts from the following update rules

$$\begin{aligned} y_k &= x_k + \tau_k(z_k - x_k) \\ x_{k+1} &= y_k - \alpha_k \nabla f(y_k) \\ z_{k+1} &= z_k - \gamma_k \nabla f(y_k) \ . \end{aligned} \tag{60}$$

Like in the proofs of convergence rate for gradient descent using potential functions, the main idea is to make a_{k+1} as large as possible as a function of a_k. Indeed, the convergence rate of an algorithm is the inverse of the growth rate of the sequence $(a_k)_k$. Therefore, the triplet $(\tau_k, \alpha_k, \gamma_k)$ must be chosen in order to maximize this growth rate. In practice, the original Nesterov acceleration (Nesterov, 1983) implies that $a_{k+1} = a_k + \frac{1}{2}(1 + \sqrt{4a_k + 1})$, yielding $\tau_k = 1 - a_k/a_{k+1}$, $\alpha_k = 1/L$ and $\gamma_k = (a_{k+1} - a_k)/L$.

Theorem 42 (Potential inequality, accelerated. d'Aspremont et al. (2021, Theorem 4.8))**.** *Let f be an L-smooth convex function, $x^* \in \arg\min_x f(x)$, and $k \in \mathbb{N}$. For $a_k \geq 0$, let $\phi_k \triangleq a_k(f(x_k) - f(x^*)) + \frac{L}{2}\|z_k - x^*\|_2^2$. The iterates of Algorithm 4 satisfy $\phi_{k+1} \leq \phi_k$, with $a_{k+1} = a_k + \frac{1+\sqrt{4a_k+1}}{2}$.*

Using Theorem 42, a recursive argument gives

$$A_N(f(x_N) - f(x^*)) \leq \phi_N \leq \cdots \leq \phi_0 = \frac{L}{2}\|x_0 - x^*\|_2^2, \tag{61}$$

which yields

$$f(x_N) - f(x^*) \leq \frac{L\|x_0 - x^*\|_2^2}{2A_N} \leq \frac{2L\|x_0 - x^*\|_2^2}{N^2}. \tag{62}$$

The last inequality is given by:

$$A_N = A_{N-1} + \frac{1 + \sqrt{4A_{N-1} + 1}}{2} \geq A_{N-1} + \frac{1}{2} + \sqrt{A_{N-1}} \geq \left(\sqrt{A_{N-1}} + \frac{1}{2}\right)^2 \geq \frac{N^2}{4} \ . \tag{63}$$

Finally, let's add the μ-convexity assumption with $\mu > 0$ on the L-smooth function f.

Theorem 43 (Potential inequality with μ-convexity, accelerated. d'Aspremont et al. (2021, Theorem 4.12))**.** *Let f be an L-smooth μ-strongly convex function, $x^* = \arg\min_x f(x)$, and $k \in \mathbb{N}$. For $a_k \geq 0$, let the potential function be $\phi_k \triangleq a_k(f(x_k) - f(x^*)) + \frac{L+\mu a_k}{2}\|z_k - x^*\|_2^2$. For a sequence of iterates $(x_k)_k, (z_k)_k \in \mathbb{R}^d$, the iterates of Algorithm 4 satisfy $\phi_{k+1} \leq \phi_k$, with $a_{k+1} = \frac{2a_k + 1 + \sqrt{4a_k + 4qa_k^2 + 1}}{2(1-q)}$ and $q = \frac{\mu}{L}$. The accelerated descent method gives the following convergence rate*

$$f(x_N) - f(x^*) \leq \min\left(\frac{2}{N^2}, \left(1 - \sqrt{\frac{\mu}{L}}\right)^N\right) L\|x_0 - x^*\|_2^2 \ . \tag{64}$$

4.1.2 Non-linear extrapolation

As seen in the previous subsection, Nesterov acceleration enjoys improved convergence rate. However, practical gains in speed are hard to obtain for coordinate descent (Bertrand and Massias, 2021, Appendix A.1). Therefore, it was not implemented in `skglm`. We focus instead on a form of non-linear extrapolation (Wynn, 1962; Smith et al., 1987) called Anderson acceleration (Anderson (1965), AA), that is a central component of `skglm` performance. This technique has been adapted to non-smooth composite problems (Zhang et al., 2018; Mai and Johansson, 2020; Poon and Liang, 2020). It enjoys accelerated rates for quadratic datafits (Varga and Golub, 1961), but weaker results are found for non-quadratic programming (Scieur et al., 2016). This can be explained easily. Anderson extrapolation is designed to accelerate the convergence of sequences based on fixed point linear iterations.

Definition 44. For a sequence of iterates $(x_k)_k$ in $\mathbb{R}^d$, a linear fixed point iteration reads

$$x_{k+1} = Tx_k + b \ , \tag{65}$$

with $T \in \mathbb{R}^{d \times d}$ the iteration matrix and $b \in \mathbb{R}^d$.

When (proximal) gradient descent is performed on them, quadratic functions have a linear gradient which gives linear descent iterations (see T^{GD} in Equation (58)). Non-quadratic objectives might not have linear descent iterations because the gradient is non-linear. If the objective is smooth, local approximations have to be made with quadratic functions. More recently, AA has been adapted by Bertrand and Massias (2021) for coordinate descent, showing significant speed-ups in the quadratic and non-quadratic cases.

AA accelerates the convergence of fixed-point iterations. Theoretically, it guarantees a faster convergence rate than its non-accelerated counterparts, as well as practical speed ups. These observed speed-ups are explained by the fact that AA is memory-efficient, line search-free and does not add substantial overhead. Besides, it does not require extensive hyperparameter fine-tuning.

Let $T : \mathbb{R}^d \to \mathbb{R}^d$ be a linear mapping. We consider the following fixed-point problem:

$$\text{Find } x \in \mathbb{R}^d \text{ such that } x = T(x) \ . \tag{66}$$

AA leverages past information on the iterate structure to efficiently solve Equation (66). Given a sequence of iterates $(x_n)_n$, AA builds x_{n+1} by finding the point which violates the least the fixed point equality in the subspace spanned by the K latest iterates. More formally, if we let the extrapolated point $x_{n+1} = \sum_{k=1}^K c_k^{(n)} x_{n-k}$ be an affine combination of the past iterates, AA seeks to find $c^{(n)} \in \mathbb{R}^K$ such that

$$\begin{aligned} c^{(n)} &= \underset{c^\top \mathbf{1} = 1}{\arg\min} \|T(x_{n+1}) - x_{n+1}\|_2 \ , \\ &= \underset{c^\top \mathbf{1} = 1}{\arg\min} \left\| \sum_{k=1}^K c_k (T(x_{n-k}) - x_{n-k}) \right\|_2 \ . \end{aligned} \tag{67}$$

By letting $U_n = [T(x_n) - x_n, \ldots, T(x_{n-K}) - x_{n-K}] \in \mathbb{R}^{d \times K}$ be the residual matrix at the n-th iteration, Equation (67) can be compactly written

$$c^{(n)} = \underset{c^\top \mathbf{1} = 1}{\arg\min} \|U_n c\|_2 \ . \tag{68}$$

The problem Equation (68) has a closed-form solution which reads

$$c^{(n)} = \frac{(U_n^\top U_n)^{-1}\mathbf{1}}{\mathbf{1}^\top (U_n^\top U_n)^{-1}\mathbf{1}} \ . \tag{69}$$

Theorem 24 gives the connection between fixed point theory and proximal gradient descent. Finding a fixed point of proximal gradient descent is equivalent to minimizing a non-smooth optimization problem. Mai and Johansson (2020) shows how to adapt Anderson acceleration to proximal gradient descent.

Now, we focus exclusively on extrapolated coordinate descent. Extrapolated coordinate descent can be performed in an *offline* or *online* fashion. *Offline* extrapolation consists in extrapolating points with more and more iterates: K consistently grows. On the contrary, the *online* variant extrapolates a point from a fixed-sized sequence of iterates. A practical trick in the implementation of *online* extrapolation consists in cyclically extrapolating iterates: after one extrapolation, K new iterates are computed before extrapolating again (see Algorithm 6).

Algorithm 6 ONLINE ACCELERATED COORDINATE DESCENT (Bertrand and Massias, 2021)

input : $n_{\text{iter}} \in \mathbb{N}, \beta^{(0)} \in \mathbb{R}^p, L \in \mathbb{R}^p$
1 **for** $k = 1, \dots, n_{iter}$ **do**
2 $\beta = \beta^{(k-1)}$
3 **for** $j = 1, \dots, p$ **do**
4 $\tilde{\beta}_j = \beta_j$
5 $\beta_j = \text{prox}_{\frac{\lambda g_j}{L_j}}(\beta_j - X_{:j}^\top \nabla f(X\beta)/L_j)$
6 $X\beta += (\beta_j - \tilde{\beta}_j) X_{:j}$
7 $\beta^{(k)} = \beta$ // $\mathcal{O}(np)$
8 **if** $k = 0 \pmod K$ // $\mathcal{O}(K^3 + pK^2)$
9 **then**
10 $U = [\beta^{(k-K+1)} - \beta^{(k-K)}, \dots, \beta^{(k)} - \beta^{(k-1)}]$
11 $c = (U^\top U)^{-1}\mathbf{1}_K / \mathbf{1}_K^\top (U^\top U)^{-1}\mathbf{1}_K \in \mathbb{R}^K$
12 $\beta_e = \sum_{i=1}^K c_i \beta^{(k-K+i)}$
13 **if** $f(X\beta_e) + \lambda g(\beta_e) \leq f(X\beta^{(k)}) + \lambda g(\beta^{(k)})$ **then**
14 $\beta^{(k)} = \beta_e$ // guaranteed convergence
15 **return** $\beta^{(n_{iter})}$

Anderson extrapolation does not theoretically guarantee global convergence of the iterates for non-quadratic datafits. Therefore to ensure Algorithm 6 remains a descent method, a primal decrease check has to be performed before assigning the extrapoled point to the coefficient vector. `skglm` implements Algorithm 6 as is. Note that in `skglm` the residual update at Line 6 can be replaced by a gradient update as explained in Section 3.2.3 when $n \gg p$ and p is of moderate size.

Convergence rate. The convergence rate varies depending on the structure of the iteration matrix T. When T is symmetric (like for gradient descent with a quadratic datafit, see Equation (58)), Scieur (2019) proves the following convergence rate.

Theorem 45 (Convergence rate for T symmetric, Scieur (2019))**.** *Let the iteration matrix T be symmetric semi-definite positive, with spectral radius $\rho(T) < 1$. Let $x^* \in \mathbb{R}^p$ be the limit of the sequence $(x^{(k)})_k$. Let $\zeta = (1 - \sqrt{1-\rho})/(1 + \sqrt{1-\rho})$. Then the iterates of online Anderson acceleration satisfy, with $B = (Id - T)^2$:*

$$\|x_{e\text{-}on}^{(k)} - x^*\|_B \leq \left(\frac{2\zeta^{K-1}}{1 + \zeta^{2(K-1)}}\right)^{k/K} \|x^{(0)} - x^*\|_B \ . \tag{70}$$

Weaker results for T non-symmetric have been explicited by Bollapragada et al. (2018) or for T pseudo-symmetric in the context of coordinate descent by Bertrand and Massias (2021).

4.2 Leveraging the sparse structure of the solutions leads to significant speed ups

4.2.1 Screening rules

Coordinate descent is the preferred first-order method to solve problems of the form of Problem 8 not only for its larger step sizes but also for the tight control it offers on the variables that are optimized. However, as demon-

strated in Figure 18, it is not fast enough for large-sized problems like `finance`. An easy way to speed up solvers consists in restraining the size of the problems solved: we could ignore zero coefficients if we identify them. Screening procedures are methods used to reduce the dimensionality of the problem at hand (Fan and Lv, 2008). While working sets prioritize features likely to be included in the support of the solution, screening techniques discard irrelevant features. The *safe rules* introduced by El Ghaoui et al. (2011) refer to a set of rules that discard features guaranteed to be zero at the optimum. Wang et al. (2014) adapted these set of rules in a sequential fashion, leveraging the computation done for a previous regularization level to efficiently compute the rules at the current level. Bonnefoy et al. (2015) elaborated *dynamic safe rules*, a method to screen variables before and along the iterations of a descent method. Eventually, the state of the art of screening rules called *gap safe rules* was introduced by Ndiaye et al. (2017), which relies on duality gap computations.

We consider a problem of the form of Problem 8 with Ω a sparsity-inducing norm on $\mathbb{R}^p$, thus convex. We suppose all $f_i : \mathbb{R} \to \mathbb{R}$ are convex and differentiable functions with L-lipschitz gradients. The dual problem is obtained using Theorem 29

$$\hat{\theta} \in \arg\min_{\theta \in \Delta_X} \mathcal{D}(\theta) \triangleq -\sum_{i=1}^{n} f_i^*(-\theta_i), \qquad (\mathcal{D})$$
$$\text{s.t. } \Delta_X = \{\theta \in \mathbb{R}^n : \Omega^*(X^\top \theta) \leq \lambda\}.$$

Screening rules are derived from the optimality conditions defined by the convex set Δ_X. A key proposition for screening introduced by El Ghaoui et al. (2011) is the following.

Proposition 46 (Safe screening rules, El Ghaoui et al. (2011))**.**

$$\forall j \in [p], \quad \Omega_j^*(X^\top \hat{\theta}) < \lambda \Rightarrow \hat{\beta}_j = 0 \ . \tag{71}$$

Proposition 46 allows to identify the equicorrelation set $\mathcal{E}_\lambda = \{j \in [p] : \Omega_j^*(X^\top \hat{\theta}) = \lambda\}$. This equicorrelation set contains the non-zero coefficients in the solution of Problem 8. However, $\hat{\theta}$ is unknown during the optimization procedure which makes Proposition 46 uninformative. To circumvent this issue, Fercoq et al. (2015) proposes to elaborate a *safe region* $\mathcal{R} \subset \mathbb{R}^n$ that contains $\hat{\theta}$. The region is considered safe if it guarantees not to wrongly discard features. Ideally, we want to find a region such that for a large number of $j \in [p]$, $\sup_{\theta \in \mathcal{R}} \Omega_j^*(X_{:j}^\top \theta) < \lambda$, hence for many j's, $\hat{\beta}_j = 0$. To see practical speed gain, computing a screening rule must be efficient, that is the computation of $\sup_{\theta \in \mathcal{R}} \Omega^*(X^\top \theta)$ must be cheap.

Definition 47 (Safe active set, Fercoq et al. (2015))**.** A region $\mathcal{R} \subset \mathbb{R}^n$ creates a safe active set $\mathcal{A}^{(\lambda)}(\mathcal{R})$

$$\mathcal{A}^{(\lambda)}(\mathcal{R}) = \{j \in [p] : \sup_{\theta \in \mathcal{R}} \ \Omega_j^*(X^\top \theta) \geq \lambda\} \ . \tag{72}$$

In a sequential fashion, for nested regions $\mathcal{R}_t \subset \mathcal{R}_{t+1}$, we naturally have $\mathcal{A}^{(\lambda)}(\mathcal{R}_t) \subset \mathcal{A}^{(\lambda)}(\mathcal{R}_{t+1})$.

Various safe regions have been considered like balls (El Ghaoui et al., 2011; Ndiaye et al., 2017) or domes (Fercoq et al., 2015). The simple choice consists in choosing a ball $\mathcal{R} \triangleq \mathcal{B}(\theta, r)$, with $\theta \in \mathbb{R}^n$ and $r > 0$.

Proposition 48 (Safe sphere test, Ndiaye et al. (2017), Equation 8)**.** *The safe sphere test reads*

$$\textit{If } \ \Omega^*(X^\top \hat{\theta}) + r\Omega^*(X_{:,j}) < \lambda, \ \textit{ then } \ \hat{\beta}_j = 0. \tag{73}$$

The center of the ball must be a dual feasible point in Δ_X. A simple heuristic to ensure a point is in the dual feasible set consists in rescaling any point $z \in \mathbb{R}^n$ by the maximum of $\Omega^*(X^\top z)$ or λ. During the iterations of an optimization algorithm, one has access to $\beta^{(t)} \in \mathbb{R}^p$ and can create a dual feasible point by rescaling the residual term $\theta^{(t)} = \nabla F(X\beta^{(t)}) \in \mathbb{R}^n$. As for the radius, Theorem 49 shows the gap-based quantity that should be used to build a safe region for any $(\beta, \theta) \in \mathbb{R}^p \times \Delta_X$.

Theorem 49 (Gap safe sphere, Ndiaye et al. (2017), Thm. 6)**.** *Assuming that F has L-Lipschitz gradient, the distance between the dual feasible solution $\hat{\theta} \in \mathbb{R}^n$ and any point $\theta \in \mathbb{R}^n$ is bounded as follows*

$$\forall (\beta, \theta) \in \mathbb{R}^p \times \Delta_X, \ \ \|\hat{\theta} - \theta\|_2 \leq \sqrt{2L \operatorname{Gap}_\lambda(\beta, \theta)} \ . \tag{74}$$

Proof. Since $\forall i \in [n]$, f_i have L-Lipschitz gradients, f_i^* are $\frac{1}{L}$-strongly convex (Hiriart-Urruty and Lemaréchal, 1993, Theorem 4.2.2, p.83). Therefore, the dual function $\mathcal{D}$ is $\frac{1}{L}$-strongly concave:

$$\forall (\theta_1, \theta_2) \in \mathbb{R}^n \times \mathbb{R}^n, \ \ \mathcal{D}(\theta_2) \leq \mathcal{D}(\theta_1) + \langle \nabla \mathcal{D}(\theta_1), \theta_2 - \theta_1 \rangle - \frac{1}{2L}\|\theta_1 - \theta_2\|_2^2 \ . \tag{75}$$

Now, plugging $\theta_1 = \hat{\theta}$ and $\theta_2 = \theta \in \Delta_X$, it follows

$$\mathcal{D}(\theta) \leq \mathcal{D}(\hat{\theta}) + \langle \nabla \mathcal{D}(\hat{\theta}), \theta - \hat{\theta} \rangle - \frac{1}{2L} \|\hat{\theta} - \theta\|_2^2 \ . \tag{76}$$

Since $\hat{\theta}$ maximizes $\mathcal{D}$ on Δ_X, $\langle \nabla \mathcal{D}(\hat{\theta}), \theta - \hat{\theta} \rangle \leq 0$, then

$$\mathcal{D}(\theta) \leq \mathcal{D}(\hat{\theta}) - \frac{1}{2L} \|\hat{\theta} - \theta\|_2^2 \ . \tag{77}$$

By weak duality, $\forall \beta \in \mathbb{R}^p, \mathcal{D}(\hat{\theta}) \leq \mathcal{P}(\beta)$, hence

$$\forall (\beta, \theta) \in \mathbb{R}^p \times \Delta_X, \ \ \mathcal{D}(\theta) \leq \mathcal{P}(\beta) - \frac{1}{2L} \|\hat{\theta} - \theta\|_2^2 \ , \tag{78}$$

and by reorganizing

$$\forall (\beta, \theta) \in \mathbb{R}^p \times \Delta_X, \ \ \|\hat{\theta} - \theta\|_2 \leq \sqrt{2L \operatorname{Gap}_\lambda(\beta, \theta)} \ . \tag{79}$$

□

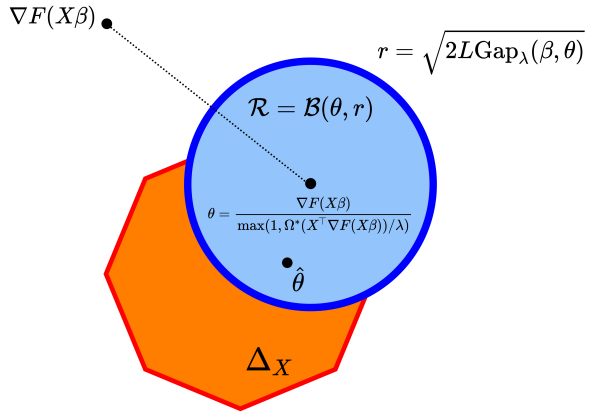


Figure 13: **Safe sphere region.** The safe sphere region $\mathcal{R} \subset \mathbb{R}^n$ is a ball of center $\theta = \nabla F(X\beta) / \max(1, X^\top \nabla F(X\beta)/\lambda)$ and radius $r = \sqrt{2L \operatorname{Gap}_\lambda(\beta, \theta)}$. The dual feasible solution $\hat{\theta}$ is in the safe region $\mathcal{R}$. The center θ is obtained by rescaling $\nabla F(X\beta)$ to make it live in Δ_X.

Example 50 (Lasso)**.** Adapting Proposition 48 to the ℓ_1-norm, the quantity to monitor is

$$d_j = \left| X_{:j}^\top \hat{\theta} \right| + \|X_{:,j}\|_2 \sqrt{2L \operatorname{Gap}_\lambda(\beta, \theta)} \ . \tag{80}$$

For all $j \in [p]$, if $d_j < 1$, $\hat{\beta}_j = 0$.

Screening rules provide great speedups, in particular when applied sequentially to evaluate a regularization path. However, in order for the screening rules to be informative, one needs to wait for the gap to be small. The initial screening made with large gaps create uninformative regions which waste computations in the beginning. Furthermore, as we shall see in Section 5, screening rules are unavailable in a non-convex regime. They are not used in `skglm`.

4.2.2 Working set

An alternative to avoid wasting early computations on useless features is to solve a series of growing subproblems. This idea has been adapted to SVM (Joachims, 1999; Fan et al., 2008) and is suited for sparse generalized linear models (Johnson and Guestrin, 2015). Working-set algorithms are algorithms addressing optimization problems by solving a series of smaller subproblems, which are cheaper to solve than the entire problem at once. More formally, let $\mathcal{W}$ be the working set, that is the set of variables involved in the optimization of these subproblems. Working set consists in solving subproblems of the form

$$\min_{\beta \in \mathbb{R}^p} f(X\beta) + \lambda \Omega(\beta) \quad \text{s.t.} \quad \beta_{\mathcal{W}^C} = 0 \ , \tag{81}$$

that is Problem 8 has to be solved with the additional constraint that every feature that is not in the working set should have a corresponding zero coefficient. This idea is ubiquitous in the literature (Roth and Fischer, 2008; Kowalski et al., 2011; Tibshirani et al., 2012; Massias, 2017) and has been adapted in numerous packages. The

R `GLMNET` package, `Blitz` (Johnson and Guestrin, 2015) or more recently `Celer` (Massias et al., 2018) and `skglm` (Bertrand et al., 2022) all use working set strategies.

Inner and outer solvers. We call *inner-loop problem* Problem 81. This problem is solved in `skglm` using proximal coordinate descent as introduced in Algorithm 2 and Anderson acceleration (see Section 4.1.2). We refer to *outer-loop problem* the problem which consists in adding features to the working set. Note that since solutions to successive inner-loop problems are typically close to each other, working set solvers are efficient when used with warm start. Warm start consists in solving the current subproblem by starting the descent method from the solution of the previous solved subproblem. Since the solutions are expected to be close to each other, this initialization allows to start the descent closer to the optimum.

Which strategy should be used to grow the working set? The first outer-loop problem initializes a working set with a pre-defined fixed size. If the approximate optimality condition is not satisfied (*e.g.* the duality gap is too high) for the current working set $\mathcal{W}$, some variables in $\mathcal{W}^C$ must be added to the working set. Working sets are usually grown using an arithmetic (*i.e.* adding p_{new} features at every outer-loop iteration) or geometric growth (*i.e.* multiplying by $\alpha > 1$ the size of the working set). In practice, the strategy we adopted in `skglm` is to double the size of the working set at every outer-loop iteration, an arithmetic growth being too slow (Massias et al., 2018, Appendix A.2).

Which features should be added to the working set? Choosing the right strategy to rank features is crucial in order to ensure high performance with working sets. At every outer iteration, a record of the highest-ranked features should be maintained, in order to add to the working set these high-ranked inactive variables. For now, we focus on ranking strategies based on violation to first-order optimality conditions and include the ones that violate them the most. Massias et al. (2018) proposed a ranking strategy based on gap-safe screening rules (see Section 4.2.1) for the Lasso, which can be extended to all convex sparse generalized linear models. Using the gap-safe rules from Example 50, features are ranked based on the following score

$$\forall j \in [p], \quad d_j(\theta) \triangleq \frac{1 - \left|X_{:j}^\top \theta\right|}{\|X_{:j}\|_2} \ . \tag{82}$$

Algorithm 7 uses this strategy to build working sets in the outer loop. Nonetheless, it is not the strategy used by `skglm`. This ranking strategy is not available in a non-convex setting due to the non-existence of a dual problem. Other ranking strategies are preferred, they are presented in details in Section 5.2.1 and Section 5.2.2.

Algorithm 7 COORDINATE DESCENT WITH WORKING SET

input : $X \in \mathbb{R}^{n \times p}, \beta \in \mathbb{R}^p, n_{\text{in}} \in \mathbb{N}^*, n_{\text{out}} \in \mathbb{N}^*, \text{ws\_size} \in \mathbb{N}^*, \epsilon > 0$

for $t = 1, \dots, n_{out}$ **do**
- $\theta^{(t)} = \nabla F(X\beta^{(t)}) / \max(1, \Omega^*(X^\top \nabla F(X\beta^{(t)}))/\lambda)$ `// Make` $\theta^{(t)}$ `dual feasible`
- $\text{score} = \left(d_j(\theta^{(t)})\right)_{j \in [p]}$ `// Rank features`
- $\text{ws\_size} = \max(\text{ws\_size}, 2|\mathcal{S}_{\beta^{(t)}}|)$ `// Double ws_size every outer-loop iteration`
- $\mathcal{W} = \text{arg\_topK}(\text{score}, K = \text{ws\_size})$ `// Take ws_size features`
- **if** $\max_{j \in [p]}(d_j(\theta^{(t)})) \leq \epsilon$ **then**
  - **break**
- **else**
  - $\beta^{(t+1)} \leftarrow$ Algorithm 2 $(X, y, \beta^{(t)}, n_{\text{in}}, \mathcal{W})$ `// Warm start from` $\beta^{(t)}$

return $\beta^{(n_{out})}$

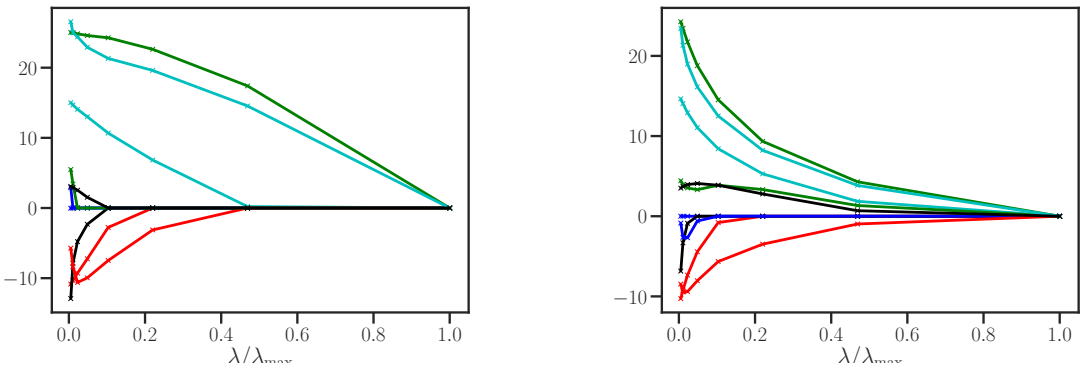
4.2.3 Homotopy methods

Homotopy methods are used in optimization to efficiently compute sparse generalized linear model solutions by evaluating a regularization path. The regularization path is the function $\lambda \mapsto \beta^{(\lambda)}$ for a given $\lambda > 0$. These methods have been extensively studied in the field of signal processing (Malioutov et al., 2005; Z. Wen et al., 2010; Sun and Yu, 2020) and sparse dictionary learning (Hale et al., 2008). In this subsection, we explain how homotopy methods work and give an example with ElasticNet (Zou and Hastie, 2005). A similar example for the Lasso (Tibshirani, 1996) can be found in Bach et al. 2011.

Homotopy methods are built on the observation that regularization paths are piecewise affine functions for quadratic datafits. The paths are fully characterized by the slope coefficients of the affine pieces and its *breakpoints* (the point

where the slopes change). This regularization path has been carefully described in Osborne et al. 2000 for the Lasso (Tibshirani, 1996) and exploited to develop the LARS algorithm (Efron et al., 2004).

Main intuition. Homotopy methods are intimately connected to working sets. For non-zero indices of $\beta \in \mathbb{R}^p$ (*i.e.* for features in the support), a convex sparse penalty is a locally linear function of β. Therefore, for a finite number of λ, one can solve local problems using Algorithm 2 and piece together local solutions to get solutions along the whole regularization path. This is this locally-linear property which gives the piecewise shape of the regularization path. This algorithm proves to be computationally-efficient in that it has the same time complexity as solving Problem 8 with quadratic programming for a single λ. There remains some central questions: how to "detect" breakpoints, how to "detect" when a feature enters or leaves the active set. Homotopy methods rely on closed-form expression of the optimality conditions to answer these questions. We present below a homotopy algorithm to compute the full regularization path of the ElasticNet.



(a) Lasso. (b) Elastic Net.

Figure 14: **Regularization paths for Lasso and ElasticNet.** Coefficient values with respect to the regularization level $\lambda/\lambda_{\max}$. Each curve represents a coefficient in Lasso (Figure 14a) and ElasticNet (Figure 14b) regressions. As the regularization level decreases, coefficients take larger and larger values. The breakpoints occur when one coefficient becomes non-zero.

Example 51 (ElasticNet)**.** The ElasticNet model reads

$$\Phi(\beta) \triangleq \underset{\beta \in \mathbb{R}^p}{\arg\min} \underbrace{\frac{1}{2} \|y - X\beta\|_2^2}_{\triangleq f(\beta)} + \lambda \underbrace{\left(\gamma \|\beta\|_1 + \frac{1-\gamma}{2} \|\beta\|_2^2 \right)}_{\triangleq g(\beta)} \quad . \tag{83}$$

At the optimum, applying Theorem 17 yields $-\frac{1}{\lambda} \nabla f(\beta) \in \partial g(\beta)$. Then, using proximal calculus,

$$\nabla f(\beta) = X^\top (y - X\beta) \quad \text{and} \quad \partial g(\beta) = \begin{cases} [-\gamma, \gamma] & \text{for } \beta_j = 0 \ , \\ \gamma \operatorname{sgn}(\beta_j) + (1-\gamma)\beta_j & \text{for } \beta_j \neq 0 \ . \end{cases} \tag{84}$$

Hence the optimality conditions of ElasticNet reads

$$\begin{cases} \left| X_{:j}^\top (y - X\beta) \right| \leq \lambda\gamma & \text{for } \beta_j = 0 \ , \\ X_{:j}^\top (y - X\beta) = \lambda\gamma \operatorname{sgn}(\beta_j) + \lambda(1-\gamma)\beta_j & \text{for } \beta_j \neq 0 \ . \end{cases} \tag{85}$$

Let $t \triangleq \operatorname{sgn}(\beta) \in \mathbb{R}^p$. For a given level of regularization $\lambda > 0$, we can re-write the second optimality condition in vectorial form:

$$X_{\mathcal{S}_\beta}^\top (y - X_{\mathcal{S}_\beta} \beta_{\mathcal{S}_\beta}) = \lambda\gamma t_{\mathcal{S}_\beta} + \lambda(1-\gamma)\beta_{\mathcal{S}_\beta} \quad . \tag{86}$$

Re-organizing this expression yields a closed-form solution of $\beta_{\mathcal{S}_\beta}^{(\lambda)}$ for some $\lambda > 0$:

$$\beta_{\mathcal{S}_\beta}^{(\lambda)} = \left(X_{\mathcal{S}_\beta}^\top X_{\mathcal{S}_\beta} + \lambda(1-\gamma) I \right)^{-1} \left(X_{\mathcal{S}_\beta}^\top y - \lambda\gamma t_{\mathcal{S}_\beta} \right) \quad , \tag{87}$$

and by definition $\beta_{\mathcal{S}_\beta^C}^{(\lambda)} = 0$. Besides, notice that $\left(X_{\mathcal{S}_\beta}^\top X_{\mathcal{S}_\beta} + \lambda(1-\gamma) I \right)$ is a positive definite matrix, hence its inverse exists. Equation (87) indicates that as long as $t_{\mathcal{S}_\beta}$ and $\mathcal{S}_\beta$ remain unchanged, $\lambda \mapsto \beta_{\mathcal{S}_\beta}^{(\lambda)}$ is affine. Now,

given an active set and the sign of the coefficients in the active set, we are able to compute analytically $\beta^{(\beta)}_{\mathcal{S}_\beta}$ over a portion of the regularization path. Note that we make the implicit assumption that $\lambda \mapsto \beta^{(\lambda)}_{\mathcal{S}_\beta}$ is continuous. This is a well-know result for the Lasso (Osborne et al., 2000) and ElasticNet based on two assumptions. First, $X^\top_{\mathcal{S}_\beta} X_{\mathcal{S}_\beta}$ must always be invertible. Second, updating $\mathcal{S}_\beta$ along the path should only consist in adding or removing one feature at a time. There remains to detect the breakpoints of the regularization path. Using Equation (85), one needs to monitor when an inactive variable verifies the second optimality condition; and conversely, one needs to monitor when an active variable verifies the first optimality condition. In the former case, the variable must be included in the active set, while it must be removed in the latter case.

Algorithm 8 HOMOTOPY FOR ELASTICNET

input : $X \in \mathbb{R}^{n \times p}, \gamma \in [0, 1], \eta \in]0, 1[, \lambda_{\min} > 0,$
init : $\lambda_0 = \|X^\top y\|_\infty, \mathcal{W} = \emptyset, \mathcal{B} = \emptyset, \beta = \mathbf{0}_p, K = \lfloor \ln(\lambda_0/\lambda_{\min}) / \ln(1/\eta) \rfloor,$

for $k = 0, \dots, K-1$ **do**
  $\lambda^{(k+1)} = \eta \lambda^{(k)}$
  $\beta^{(k)} = \left(X^\top_{\mathcal{W}} X_{\mathcal{W}} + \lambda^{(k)}(1-\gamma) I\right)^{-1} \left(X^\top_{\mathcal{W}} y - \lambda^{(k)} \gamma t_{\mathcal{W}}\right)$
  for $j \in \mathcal{W}$ **do**
    if $\beta^{(k)}_j = 0$ **then**
      $\mathcal{W}$.remove(j) `// Remove inactive variables from the working set`
  for $j \in \mathcal{W}^C$ **do**
    if $\left|X_j^\top (y - X\beta^{(k)})\right| = \gamma \lambda^{(k)}$ **then**
      $\mathcal{W}$.add(j) `// Add active variable to the working set`
  $\mathcal{B}$.add($\beta^{(k)}$)
return $\mathcal{B}$

Time complexity. The time complexity of Algorithm 8 is dominated by the inversion of $\left(X^\top_{\mathcal{W}} X_{\mathcal{W}} + \lambda^{(k)}(1-\gamma) I\right)$. The initial inversion of this matrix costs $\mathcal{O}(p^3)$ flops using a Cholesky factorization. Nonetheless, assuming only one feature is added or removed at a time, at each breakpoint the rank of the matrix is changed by one unit (either by the addition or the removal of one feature in the working set). Rank-one updates cost an additional $\mathcal{O}(p^2)$ flops at every iteration.

4.3 Experiments

Since we have studied several techniques to accelerate descent algorithms, we carry out experiments to prove the superiority of these techniques. We focus exclusively on proximal gradient descent and coordinate descent, since they are among the fastest algorithms to solve sparse generalized linear models. All the experiments are performed using `Benchopt` (Moreau et al. (2022), Appendix C).

4.3.1 Inertial acceleration vs. extrapolation

We perform a benchmark on the efficiency of different types of acceleration. We compare proximal gradient descent with Nesterov acceleration (FISTA, Beck and Teboulle (2009)), vanilla proximal gradient descent (ISTA) and coordinate descent with Anderson extrapolation. Figure 15 shows that on various setups Anderson extrapolation is orders of magnitude faster than FISTA. This result does not come as a surprise and has already been extensively investigated in Bertrand and Massias (2021).

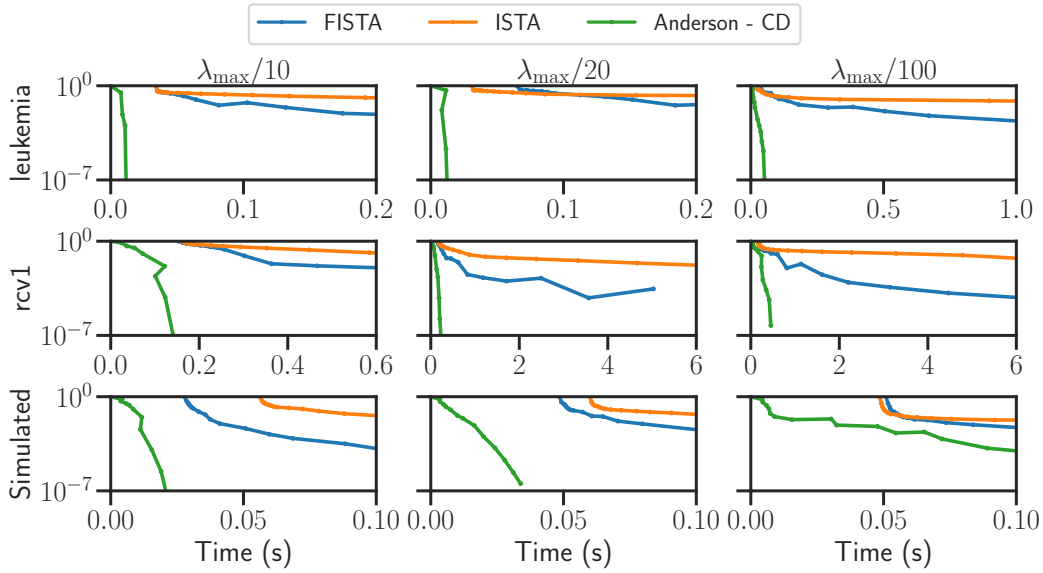


Figure 15: **Comparison inertia vs extrapolation.** Duality gap as a function of time for the Lasso, on 3 different datasets: *leukemia*, *rcv1* and a simulated dataset (200 samples and 1000 features).

4.3.2 Ablation studies

We use the Lasso to carry out ablation studies on the relevance of acceleration techniques studied in the previous subsection. Details of the datasets used to carry out the experiments are given in Section 3.3.1.

Impact of working set. As shown in Figure 16, the working set plays a crucial role for high-dimensional datasets like `finance` and `news20`. In this experiment, we compare a coordinate descent solver with a working set strategy, and one without a working set strategy. With more than a million features, solvers without a working set cycle on the whole feature set. On the contrary, working set solvers are able to progressively identify the support by quickly solving small inner subproblems used to warm-start the next inner solver. The larger the dimension of the feature set, the more gain in speed working sets provide.

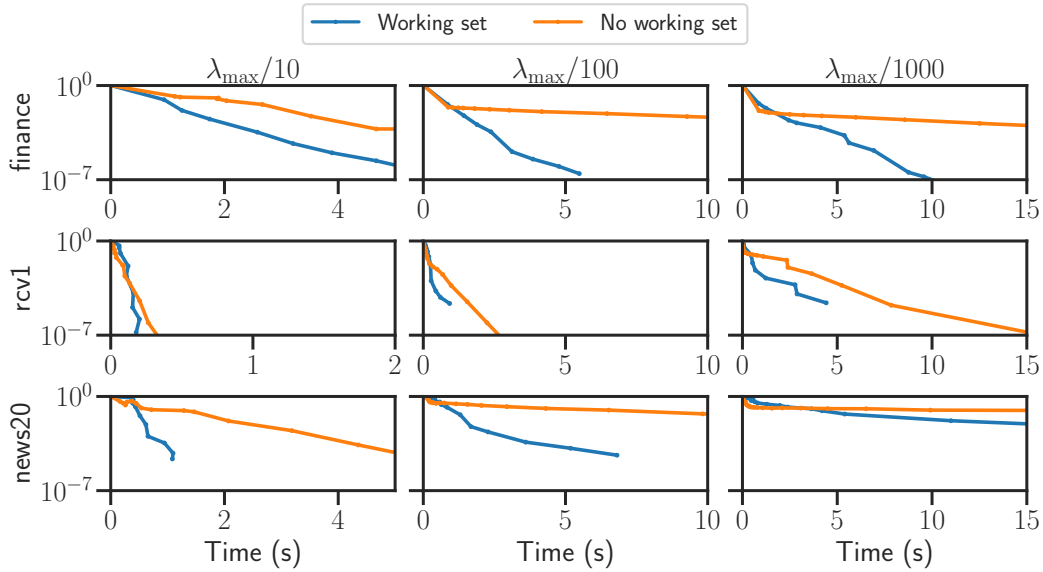


Figure 16: **Ablation of working set.** Duality gap for the Lasso as a function of time for different regularization levels, on 3 different datasets: *finance*, *rcv1* and *news20*.

Impact of extrapolation. Figure 17 compares two solvers using working sets: one with extrapolation and one without. It is worth noting that extrapolation has an impact for high level of regularization and large datasets. It is particularly visible for the `finance` dataset where extrapolation finds better primal points as soon as the support

is identified. The combination of a working set strategy with Anderson acceleration is particularly relevant: the working set allows for a quick identification of the support and extrapolation makes the solver quickly converge once the support is identified.

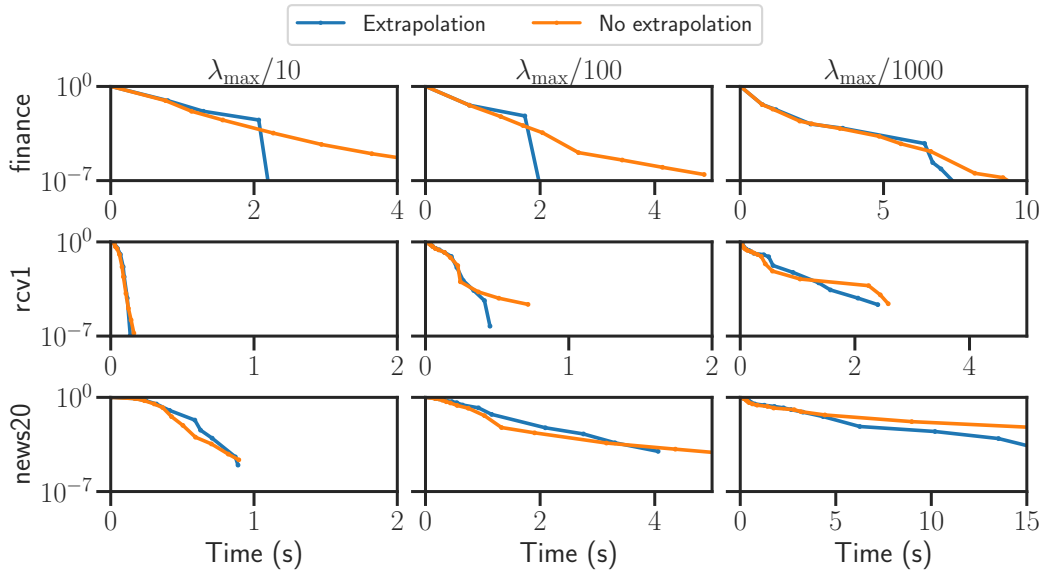


Figure 17: **Ablation of extrapolation.** Duality gap of the Lasso as a function of time for different regularization levels, on 3 different datasets: *finance*, *rcv1* and *news20*.

4.3.3 Comparing off-the-shelf solvers for convex penalties

We focus on the Lasso (Problem 9) and sparse Logistic Regression (Problem 10) problems and benchmark several `Python` solvers. All these solvers solve Problem 9 and Problem 10 by coordinate descent using different techniques. Table 4 gives a summary of the heuristics and meta-algorithms used to accelerate the optimization procedure. Results are presented in Figure 18 and Figure 19.

For very large-dimensional datasets like `finance` (4 272 227 features), the working set is crucial to avoid cycling over the whole feature set at every coordinate descent epoch. `scikit-learn` (a solver without a working set strategy) over-optimizes the first sub-problems: before finding the support, it is unnecessary to optimize over a sub-optimality gap below 10^{-5}.

We highlight that low-level optimization to the algorithm of `Blitz` has been made since the publication of the paper (Johnson and Guestrin, 2015). `Blitz` now uses screening rules to select the features to be included in priority to the working set, a strategy similar to `Celer`. We notice that both solvers offer the best performance on `finance` and `leukemia`.

Table 4: Convex solvers for sparse generalized linear models.

Name	Acceleration	Working set	Screening rules	Language
`glmnet` (Friedman et al., 2009)	✗	✓	✗	Fortran
`scikit-learn` (Pedregosa et al., 2011)	✗	✗	✗	Cython
`Lightning` (Blondel and Pedregosa, 2016)	✗	✓	✗	Cython
`celer` (Massias et al., 2018)	✓	✓	✓	Cython
`Blitz` (Johnson and Guestrin, 2015)	✗	✓	✓	C++
`skglm` (Bertrand et al., 2022)	✓	✓	✗	Python

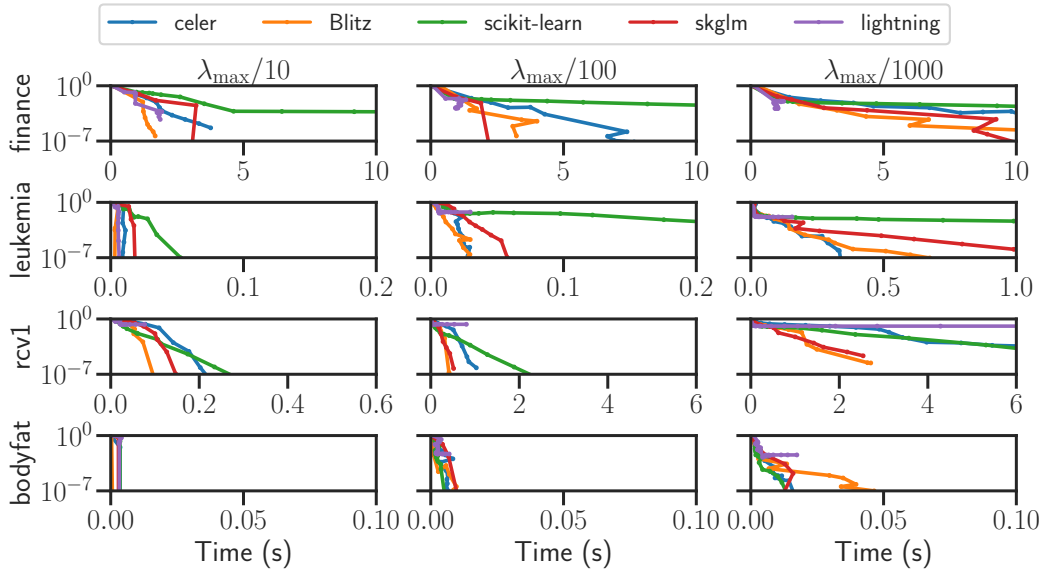


Figure 18: **Lasso.** Duality gap as a function of time, on 4 different datasets: *finance*, *leukemia*, *rcv1* and *bodyfat*.

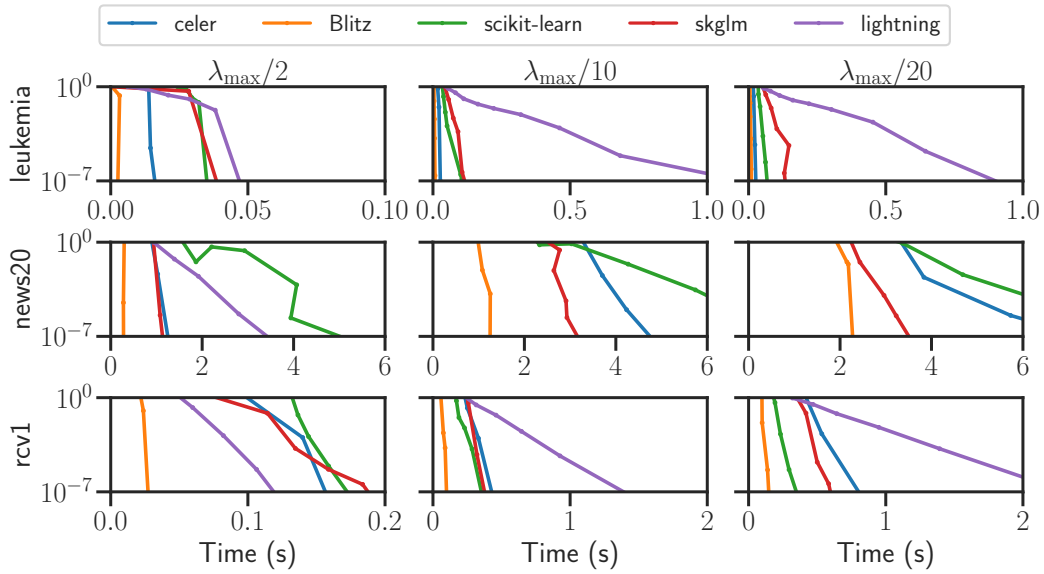


Figure 19: **Sparse logistic regression.** Duality gap as a function of time, on 3 different datasets: *leukemia*, *news20* and *rcv1*.

4.3.4 Benchmarking regularization paths

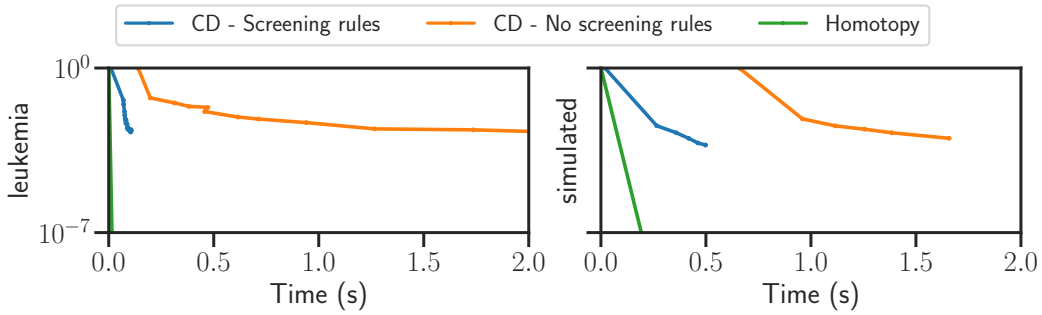


Figure 20: **Lasso path.** Duality gap as a function of time, on 2 different datasets: *leukemia* and a $(1000, 1000)$ simulated dataset.

Figure 20 compares multiple techniques to evaluate the regularization path of a Lasso estimator. The regularization path is the function $\lambda \mapsto \beta^{(\lambda)}$ (see Section 4.2.3 for more details). The homotopy method is clearly faster, as it leverages the linear piecewise structure of the regularization path. On the opposite, the other two solvers require solving 100 optimization problems. Even though, some computational tricks like warm start are applied to speed up the algorithm, solving 100 optimization problems remain very expensive. It is worth noting that sequential screening rules (Bonnefoy et al., 2015; Ndiaye et al., 2017) are particularly efficient to evaluate the path.

5 `skglm`, a versatile solver for convex and non-convex optimization

5.1 Non-convex optimization

5.1.1 Sparser and less biased solutions with non-convex penalties

Altough convex penalties are widely used, they create estimators biased towards 0. This shrinkage of the coefficients is particularly significant for large coefficients. Ideally, estimators should shrink small coefficients to 0 while keeping large coefficients untouched. Such penalties exist and are non-convex: they yield sparser solutions than

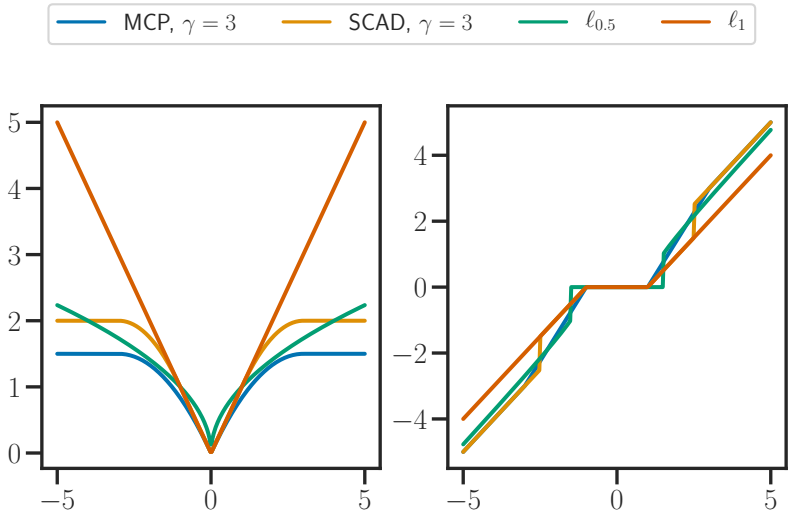


Figure 21: **Non-convex penalties and their proximal operators.** The $\ell_{0.5}$-quasinorm, MCP (Zhang, 2010) and SCAD (Fan and Li, 2001) are amongst the most widely-used non-convex penalties (left). Note that they are plotted with the convex ℓ_1-penalty.

convex estimators and mitigate the intrinsic Lasso bias. Figure 21 shows the different one-dimensional penalty and compares it to the ℓ_0-norm. The Smoothly Clipped Absolute Deviation (SCAD, Fan and Li (2001)) and the Minimax Concave Penalty (MCP, Zhang (2010)) are two instances of non-convex penalties. They enjoy the *oracle* property, meaning that "they perform as well as if the analyst had known in advance which coefficients were zero

and which were nonzero" (Breheny and Huang, 2011). We are particularly interested in them since they have closed form proximal operators that can be efficiently evaluated. Note that there exist others concave penalties like the $\ell_{0.5}$-quasinorm.

Example 52 (Minimax concave penalty, Zhang (2010))**.** MCP is parametrized by a curvature coefficient $\gamma > 0$ that controls the concavity of the penalty. MCP is defined on $\mathbb{R}^+$:

$$p_{\lambda,\gamma}(x) = \begin{cases} \lambda x - \frac{x^2}{2\gamma} & \text{if } x \leq \gamma\lambda \ , \\ \frac{1}{2}\gamma\lambda^2 & \text{otherwise.} \end{cases} \tag{88}$$

To better understand how MCP works, it is worth studying its first derivative.

$$p'_{\lambda,\gamma}(x) = \begin{cases} \lambda - \frac{x}{\gamma} & \text{if } x \leq \gamma\lambda \ , \\ 0 & \text{otherwise.} \end{cases} \tag{89}$$

MCP starts by applying a penalization rate $\lambda > 0$ similar to the ℓ_1-norm but continuously relaxes this penalization. Once $x > \gamma\lambda$, the penalization rate drops to 0 and no shrinkage is applied on the coefficients, thus mitigating the bias towards 0.

Example 53 (Smoothly Clipped Absolute Deviation, Fan and Li (2001))**.** Like MCP, SCAD is parametrized by a curvature coefficient $\gamma > 2$. It is defined on $\mathbb{R}^+$.

$$p_{\lambda,\gamma}(x) = \begin{cases} \lambda x & \text{if } x \leq \lambda \ , \\ \frac{\lambda\gamma x - 0.5(x^2+\lambda^2)}{\gamma-1} & \text{if } \lambda < x \leq \gamma\lambda \ , \\ \frac{\lambda^2(\gamma^2-1)}{2(\gamma-1)} & \text{otherwise.} \end{cases} \tag{90}$$

A similar reasoning can be made for the penalization rate of SCAD.

$$p'_{\lambda,\gamma}(x) = \begin{cases} \lambda & \text{if } x \leq \lambda, \\ \frac{\gamma\lambda - x}{\gamma-1} & \text{if } \lambda < x \leq \gamma\lambda, \\ 0 & \text{if } x > \gamma\lambda \ . \end{cases} \tag{91}$$

Example 54 ($\ell_{0.5}$-penalty)**.** As stated in Section 3.1, it is a well-known fact that ℓ_p^p-penalties with $p \in [0, 1]$ yield sparse solutions. The non-convex penalty $\ell_{0.5}$ yields sparser solutions than the ℓ_1-norm.

$$\|x\|_{0.5} = \sum_{j=1}^{p} \sqrt{|x_j|} \ . \tag{92}$$

Figure 22 demonstrates the ability of non-convex penalty to recover the correct support and zero-out true null coefficients. The root mean squared error is the lowest, while the F1-score is the highest with non-convex penalties.

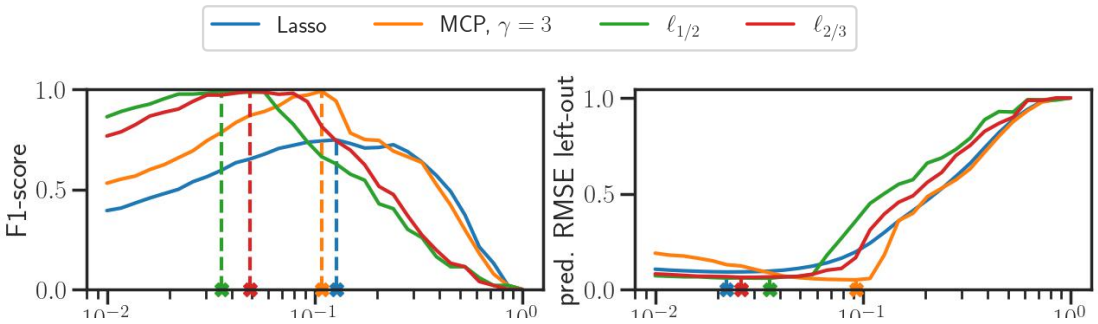


Figure 22: **Comparison of support recovery between convex and non-convex penalties.** The non-convex penalties are able to achieve higher F1-score than the ℓ_1-norm (Lasso). Non-convex penalties yield sparser solutions and are less biased: the RMSE between the true coefficients and the reconstructed coefficients is lower. The experiment was carried out on a simulated 1000×1000-dimensional matrix. The true coefficients are known since it is a simulated setup.

5.1.2 Proximal operators in the non-convex case

These sparser estimators come at a heavy price. First, non-convex estimators no longer enjoy the local-to-global property. It is very likely that an optimization algorithm converges to a local minimum. Second, the algorithm now depends on the initialization of the coefficient vector: for the same algorithm initialized at different points, they may converge to different optima. Third, several tools used in convex optimization are no longer available.

As per Definition 23, the proximal operator is not necessarily defined for non-convex functions. Since proximal operators remain crucial in non-smooth optimization, we present conditions under which proximal mappings are available for a larger class of functions than convex functions.

Definition 55 (Prox-regular functions, (Poliquin and Rockafellar, 1996, Definition 1.1))**.** A function $f : \mathbb{R}^d \to]-\infty, +\infty]$ is prox-regular at x_0 for a subgradient $g_0 \in \partial f(x_0)$ if f is locally lower semi-continuous at x_0 and there exists $r > 0$ and $\epsilon > 0$ such that

$$f(x') \geq f(x) + g^\top (x' - x) - \frac{r}{2} \|x' - x\|_2^2 \quad , \tag{93}$$

whenever $\|x' - x_0\| < \epsilon$, $\|x - x_0\| < \epsilon$ with $x' \neq x$ and $|f(x) - f(x_0)| < \epsilon$ while $\|g - g_0\| < \epsilon$. The function f is said to be prox-regular if it is prox-regular at every $x_0 \in \mathbb{R}^d$.

Definition 56 (Prox-bounded functions, Hare and Sagastizábal (2009))**.** A function $f : \mathbb{R}^d \to]-\infty, +\infty]$ is prox-bounded if there exists $r \geq 0$ such that $M_f^r(x) > -\infty$ for some $x \in \mathbb{R}^d$. The infimum of all such r is called the threshold of prox-boundedness of f.

Prox-boundedness merely states that there exists some level $r \geq 0$ for which the Moreau envelope of f is defined in at least one point.

Hare and Sagastizábal (2009) show that if a function f is prox-regular and prox-bounded at a point $x_0 \in \mathbb{R}^d$ for the subgradient $g_0 \in \mathbb{R}^d$, then in a neighborhood of $x_0 + \frac{1}{r} g_0 \in \mathbb{R}^d$, the proximal mapping $\mathbf{prox}_{rf}$ exists, is single-valued and Lipschitz continuous in some neighborhood of $x_0 + \frac{1}{r} g_0 \in \mathbb{R}^d$. We refer to Luu et al. (2017) for a proof on the prox-regularity and prox-boundedness of SCAD (Fan and Li, 2001). We end this subsection by giving a recapitulative table of non-convex penalties and their associated proximal operators.

Table 5: Proximal operators for non-convex sparse penalties

Penalty	$f(\beta)$	$\mathbf{prox}_f$
$\ell_{0.5}$	$\sum_{j=1}^p \sqrt{\beta_j}$	$\begin{cases} 0 & \text{if } \lvert\beta_j\rvert < \frac{3}{2}\lambda^{\frac{2}{3}} \ , \\ \frac{2}{3}\beta_j(1 + \cos(\frac{2}{3}\arccos(-\frac{3^{\frac{3}{2}}\lambda}{4}\lvert\beta_j\rvert^{-\frac{3}{2}}))) & \text{otherwise.} \end{cases}$
MCP	$\sum_{j=1}^p \mathrm{MCP}(\lvert\beta_j\rvert)$	$\begin{cases} 0 & \text{if } \lvert\beta_j\rvert < \lambda \ , \\ \beta_j & \text{if } \lvert\beta_j\rvert > \lambda\gamma \ , \\ \mathrm{sgn}(\beta_j)\frac{\lvert\beta_j\rvert - \lambda}{1 - 1/\gamma} & \text{otherwise.} \end{cases}$

5.1.3 A partial fix: iteratively reweighting convex norms?

The adaptive Lasso (Zou, 2006) has been proposed as a way to fit convex models sparser than the original Lasso (Tibshirani, 1996), by introducing a weight vector $w \in \mathbb{R}^p$ in front of every coefficient that is penalized

$$\beta^* \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \sum_{j=1}^p w_j |\beta_j| \ . \tag{94}$$

Algorithm 9 ITERATIVE REWEIGHTED L1 MINIMIZATION

input : $w \in \mathbb{R}^p, n_{\text{iter}} \in \mathbb{N}, \epsilon \in \mathbb{R}^+, \lambda \in \mathbb{R}^+$
init : $w = \mathbf{1}_p$
for $t = 1, \dots, n_{iter}$ **do**
  Solve the weighted ℓ_1 minimization problem with Algorithm 7: $\beta^{(t)} \leftarrow \arg\min \|y - X\beta\|_2^2 + \lambda \|w \odot \beta\|_1$
  Update the weights: $\forall i \in [p], \quad w_i^{(t+1)} \leftarrow \frac{1}{|\beta_i^{(t)}| + \epsilon}$
return β

Large weights are used to encourage zero entries in the coefficient vector, while small weights would discourage them. As a rule of thumb, the weights should relate inversely to the true coefficient magnitude. However, the true coefficient vector $\beta^* \in \mathbb{R}^p$ is unknown. How can a valid set of weights be obtained without knowing $\beta^* \in \mathbb{R}^p$ in the first hand?

Candes et al. (2008) propose a majorization-minimization (Sun et al. (2017), MM) algorithm to iteratively reweight the coefficients of the Lasso estimator. A MM algorithm iteratively minimizes a surrogate function that majorizes the objective function as shown in Figure 6. Consider the following constrained problem

$$\min_{\beta \in \mathbb{R}^p} \quad \frac{1}{2n} \|y - X\beta\|_2^2 + \sum_{i=1}^{p} \log(|\beta_i| + \epsilon) \quad . \tag{95}$$

Problem 95 is equivalent to the following

$$\min_{\beta, u \in \mathbb{R}^p} \quad \frac{1}{2n} \|y - X\beta\|_2^2 + \sum_{i=1}^{p} \log(u_i + \epsilon) \quad , \quad |\beta_i| \leq u_i, \quad i \in [p] \quad . \tag{96}$$

Problem 96 is easier to solve since we get rid of the absolute value in the minimized term as it is now placed as an additional constraint. To apply a MM algorithm, we need to majorize the log-sum function. Yet, the log-sum function is concave and therefore below its tangents. Thus by taking a first-order Taylor expansion, we obtain a linearization of the log-sum function in a neighborhood of $u \in \mathbb{R}^p$. More formally, let $g(u) = \sum_{i=1}^{p} \log(u_i + \epsilon)$. The first-order Taylor expansion in a neighborhood of $u^{(t)} \in \mathbb{R}^p$ yields

$$\begin{aligned} u^{(t+1)} &= \arg\min_{u \in \mathbb{R}^p} g(u^{(t)}) + \nabla g(u^{(t)})^\top (u - u^{(t)}) \\ &= \arg\min_{u \in \mathbb{R}^p} \sum_{i=1}^{p} \log(u_i^{(t)} + \epsilon) + (u - u^{(t)}) \sum_{i=1}^{p} \frac{1}{u_i^{(t)} + \epsilon} \quad . \end{aligned} \tag{97}$$

By removing the terms that do not depend on u, it follows that

$$u^{(t+1)} = \arg\min_{u \in \mathbb{R}^p} \sum_{i=1}^{p} \frac{u_i}{u_i^{(t)} + \epsilon} \quad . \tag{98}$$

And by equivalence,

$$\beta^{(t+1)} = \arg\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{p} \frac{|\beta_i|}{|\beta_i^{(t)}| + \epsilon} \quad . \tag{99}$$

By letting $\forall i \in [p], w_i^{(t)} = \frac{1}{|\beta_i^{(t)}| + \epsilon}$, it follows:

$$\beta^{(t+1)} = \arg\min_{\beta \in \mathbb{R}^p} \quad \frac{1}{2n} \|y - X\beta\|_2^2 + \sum_{i=1}^{p} w_i^{(t)} |\beta_i| \quad . \tag{100}$$

This gives an algorithm that iteratively reweight the ℓ_1-norm. Note that this algorithm minimizes a concave objective (log-sum is concave), by iteratively solving convex subproblems. Hence, due to the concavity of the objective function, we are not guaranteed to converge to a global minimum.

Why choosing the log-sum function? The log-sum penalty function has the potential to be more sparsity-encouraging than the ℓ_1 norm. More precisely, the smaller $\epsilon > 0$ the closer the log-sum function is from the

ℓ_0 norm. However, $\epsilon > 0$ cannot be set arbitrarily small since as ϵ gets closer from 0, the problem Problem 95 becomes increasingly concave and the iterative reweighted ℓ_1 algorithm is more likely to be stuck at a local optimum. In practice, the coefficient estimation is robust to the choice of ϵ.

Algorithm 9 works well in practice. In neuroscience, it is used to solve the M/EEG inverse problem by iteratively solving $\ell_{2,1}$-reweighted subproblems (Strohmeier et al., 2016; Bannier et al., 2021). Nonetheless, it still requires to solve multiple convex surrogate problems, an intractable effort in large-dimensional settings. Direct optimization algorithms for non-convex problems have to be elaborated.

5.2 Feature-ranking strategies for non-convex penalties

Since we choose to optimize non-convex objectives, we are removed some informative and very useful tools to optimize a function. In particular, working set strategies can no longer be built using aggressive screening rules as explained in Section 4.2.2. We shall study which scores can be derived in a non-convex regime to rank features that should be included in a working set.

5.2.1 Distance of the gradient to the subdifferential

Following Theorem 17, at the optimum the negative gradient of the datafit term belongs to the subdifferential of the penalty:

$$-\nabla f(\hat{\beta}) \in \partial g(\hat{\beta}) \ . \tag{101}$$

A natural criterion can be derived from this rule. Considering $\nabla f(\beta) \in \mathbb{R}^p$ and $\partial g(\beta) \subseteq \mathbb{R}^p$, one can compute the distance between the gradient and the subdifferential. More precisely, since we are interested in ranking features, we can compute the coordinate-wise gradient of f and evaluate the distance between the gradient and the subdiffferential at $\beta_j \in \mathbb{R}$. Therefore, for $j \in [p]$, features can be ranked based on the following score evaluated at $\beta \in \mathbb{R}^p$

$$\text{score}_j = \text{dist}(-\nabla_j f(\beta), \partial g_j(\beta)) \ . \tag{102}$$

Equation (102) consists in ranking features based on their violation of the optimality condition. A similar idea is used when using the KKT: features are ranked based on their violation of the Karush-Kuhn-Tucker (KKT) conditions. This new ranking strategy yields Algorithm 10.

Algorithm 10 `skglm` (Bertrand et al., 2022)

input : $X \in \mathbb{R}^{n \times p}, \beta \in \mathbb{R}^p, n_{\text{in}} \in \mathbb{N}^*, n_{\text{out}} \in \mathbb{N}^*, \text{ws\_size} \in \mathbb{N}^*, \epsilon > 0$

for $k = 1, \dots, n_{out}$ **do**
- $\text{score} = (\text{dist}(-\nabla_j f(\beta), \partial g_j(\beta)))_{j \in [p]}$
- $\text{ws\_size} = \max(\text{ws\_size}, 2|\mathcal{S}_\beta|)$ `// Double ws_size every outer-loop iteration`
- $\mathcal{W} = \text{arg\_topK}(\text{score}, K = \text{ws\_size})$ `// Take ws_size features with largest distance to the subdifferential`
- **if** $\max_{j \in [p]} \text{dist}(-\nabla_j f(\beta), \partial g_j(\beta)) \leq \epsilon$ **then**
  - **break**
- **else**
  - $\beta \leftarrow$ Algorithm 2 $(X, y, \beta, n_{\text{in}}, \mathcal{W})$ `// Warm start from` β

return β

This scoring strategy is a crucial part of `skglm` as it allows to setup a working set strategy for convex and non-convex penalties. Table 6 and Table 7 in Appendix D summarize the distance to the subdifferential score for the most used convex and non-convex penalties. We adopt a geometric growth of the working set which enables Algorithm 10 to quickly converge towards the generalized support while avoiding overshooting, as explained in Ndiaye and Takeuchi 2021. A proof of the convergence of Algorithm 10 is given in Bertrand et al. (2022, Proposition 5).

5.2.2 Violation of the fixed point iterate

For some penalties, the distance between the gradient and the subdifferential is uninformative. Indeed, if the subdifferential is $\mathbb{R}$, then the distance to the gradient restricted to the j-th coordinate is always null. Another ranking strategy needs to be elaborated. In the non-convex case, one needs to rely on more general notions of subdifferentiability. In particular, the Fréchet subdifferential is defined and used in the non-convex case.

Definition 57 (Fréchet subdifferential, Kruger (2003, Definition 1.1))**.** Let $f : \mathbb{R}^d \to]-\infty, +\infty]$ be a function finite at $x \in \mathbb{R}^d$. The Fréchet-subdifferential of f at x is the set

$$\hat{\partial} f(x) = \left\{ v \in \mathbb{R}^d : \liminf_{x' \to x} \frac{f(x') - f(x) - \langle v, x' - x \rangle}{\|x' - x\|} \geq 0 \right\} . \tag{103}$$

Proposition 58 (Kruger (2003, Proposition 1.2))**.** *If a function $f : \mathbb{R}^d \to]-\infty, +\infty]$ is convex, then for any $x \in \mathbb{R}^d$, $\partial f(x) = \hat{\partial} f(x)$.*

Proposition 59 (Subdifferential of $\ell_{0.5}$)**.** *The Fréchet-subdifferential of the $\ell_{0.5}$-penalty at 0 is $\mathbb{R}$.*

Proof. Take $g_j(\cdot) = \sqrt{\cdot}$. On one hand, by definition, $\hat{\partial} g_j(0) \subseteq \mathbb{R}$. On the other hand, let $x \in \mathbb{R}$. It follows that $\liminf_{u \to 0} \frac{\sqrt{u} - xu}{|u|} = +\infty$. Therefore $\hat{\partial} g_j(0) = \mathbb{R}$. □

Since for all $\beta \in \mathbb{R}^p$, $\nabla_j f(\beta)$ is a real scalar, the distance to the subdifferential is always 0, making the score uninformative.

The idea to create another scoring strategy remains the same: deriving an optimality condition and evaluating the violation of this condition for every feature. Using Theorem 24, $\hat{\beta} \in \mathbb{R}^p$ is a critical point of the objective if

$$\hat{\beta} = \mathbf{prox}_{g_j / L_j}(\hat{\beta} - \frac{1}{L_j} \nabla_j f(\hat{\beta})) \ . \tag{104}$$

Therefore, the proposed criterion consists in evaluating the violation of the fixed point iterate

$$\text{score}_j = \left| \hat{\beta}_j - \mathbf{prox}_{g_j / L_j} \left(\hat{\beta}_j - \frac{1}{L_j} \nabla_j f(\hat{\beta}) \right) \right| \ . \tag{105}$$

This criterion can be seen as a restriction of the scoring strategy proposed in Section 5.2.1. Indeed, every fixed point of the proximal operator is a critical point while the converse may not be true. Besides, it is powerful as it only relies on $\nabla_j f(\hat{\beta})$ and $\mathbf{prox}_{g_j / L_j}$, two quantities that are known for the vast majority of problems of the form Problem 8.

5.3 `skglm` in details

We have reviewed all the components needed to present in depth `skglm`. The algorithm revolves around two central components: a working set strategy and Anderson acceleration. We present how the algorithm has been implemented and discuss some design choices to ensure the framework is modular. `skglm` combines two acceleration techniques to obtain significant speed-ups.

1. **Working set**: as explained in Section 4.2.2, a working set strategy avoids spending useless computation on features out of the generalized support. The working set is grown geometrically to quickly identify the generalized support using the scoring strategy presented in Section 5.2.1. When this strategy is uninformative, we rely on the fixed point violation strategy presented in Section 5.2.2.

2. **Anderson acceleration**: experiments performed in Section 3.3.1 have demonstrated the practical speed ups obtained by Anderson acceleration (Section 4.1.2) for quadratic and non-quadratic datafits. Anderson acceleration is applied with ease to non-convex problems and enjoys faster convergence.

The library has been written in full `Python` using the just-in-time compiler `Numba` (Lam et al., 2015). The comparison performed in Appendix B shows that `Numba` and `Cython` have similar runtimes. To maintain a readable and flexible codebase, `Numba` has been chosen to accelerate the runtime of our library `skglm`.

The library has been designed in order to be very modular. Datafits and penalties are defined separately in `Python` classes. Instantiated objects are then fed to a solver `Python` function which outputs the problem solution. This design enables us to quickly support new penalties and datafits: a new penalty is typically written in under 40 lines of code as demonstrated in Listing 1 and Listing 2 (see Appendix E). This highly modular design is also very convenient to implement new solvers: Gram-based and residual-based coordinate descent solvers for instance (see Section 3.2.3).

`skglm` exhibits several advantages:

1. **Speed**: `skglm` achieves state-of-the-art performance for non-convex penalties.

2. **Modularity**: the API design makes it really easy to add one datafit or one penalty.

3. **Flexiblity**: it is very easy to try various combinations of models (*e.g.* logistic datafits with a SCAD penalty).

This package has been released in the `scikit-learn` ecosystem and abides by the highest standards (Buitinck et al., 2013) of machine learning library API.

References

D. G. Anderson. Iterative procedures for nonlinear integral equations. *J. ACM*, 12(4):547–560, oct 1965. ISSN 0004-5411. doi: 10.1145/321296.321305.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. In Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors, *Optimization for Machine Learning*, Neural information processing series, pages 19–49. MIT Press, 2011.

P.-A. Bannier, Q. Bertrand, J. Salmon, and A. Gramfort. Electromagnetic neural source imaging under sparsity constraints with sure-based hyperparameter tuning, 2021.

N. Bansal and A. Gupta. Potential-function proofs for first-order methods, 2017.

H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2017.

A. Beck. *First-order methods in optimization / Amir Beck, Tel-Aviv University, Tel-Aviv, Israel.* MOS-SIAM series on optimization. Society for Industrial and Applied Mathematics, Mathematical Optimization Society, 2017. ISBN 9781611974980.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm with application to wavelet-based image deblurring. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 693–696, 2009. doi: 10.1109/ICASSP.2009.4959678.

A. Beck and M. Teboulle. Smoothing and first order methods: A unified framework. *SIAM Journal on Optimization*, 22(2):557–580, 2012. doi: 10.1137/100818327.

A. Beck, E. Pauwels, and S. Sabach. The Cyclic Block Conditional Gradient Method for Convex Optimization Problems. *SIAM Journal on Optimization*, (4):2024–2049, 2015. doi: 10.1137/15M1008397.

S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. Seljebotn Sverre, and K. Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

Q. Bertrand. *Hyperparameter selection for high dimensional sparse learning : application to neuroimaging*. PhD thesis, Université Paris-Saclay, 2021.

Q. Bertrand and M. Massias. Anderson acceleration of coordinate descent. 2021.

Q. Bertrand, Q. Klopfenstein, P.-A. Bannier, G. Gidel, and M. Massias. Beyond l1: Faster and better sparse models with skglm. 2022. doi: 10.48550/ARXIV.2204.07826.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.

M. Blondel and F. Pedregosa. Lightning: large-scale linear classification, regression and ranking in python, 2016.

R. Bollapragada, D. Scieur, and A. d'Aspremont. Nonlinear acceleration of momentum and primal-dual algorithms, 2018.

A. Bonnefoy, V. Emiya, L. Ralaivola, and R. Gribonval. Dynamic Screening: Accelerating First-Order Algorithms for the Lasso and Group-Lasso. *IEEE Transactions on Signal Processing*, 63(19):5121–5132, October 2015. ISSN 1053-587X, 1941-0476. doi: 10.1109/TSP.2015.2447503. arXiv:1412.4080 [cs, stat].

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

P. Breheny and J. Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The Annals of Applied Statistics*, 5(1):232 – 253, 2011. doi: 10.1214/10-AOAS388.

L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Vanderplas, A. Joly, B. Holt, and G. Varoquaux. Api design for machine learning software: experiences from the scikit-learn project. 2013. doi: 10.48550/ARXIV.1309.0238.

E. J. Candes, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier analysis and applications*, 14(5):877–905, 2008.

C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2 (3), may 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199.

B. Chen, S. He, Z. Li, and S. Zhang. Maximum block improvement and polynomial optimization. *SIAM Journal on Optimization*, 22(1):87–107, 2012. doi: 10.1137/110834524.

S. Chen and D. Donoho. Basis pursuit. *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, 1:41–44 vol.1, 1994. doi: 10.1109/ACSSC.1994.471413.

J. M. Danskin. *The theory of Max-Min and its application to weapons allocation problems*. 1967.

A. d'Aspremont, D. Scieur, and A. Taylor. Acceleration Methods. *Foundations and Trends in Optimization*, 5 (1-2):1–245, 2021. ISSN 2167-3888, 2167-3918. doi: 10.1561/2400000036. arXiv:2101.09545 [cs, math].

I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, USA, 1992. ISBN 0898712742.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, 2003.

G. Davis, S. Mallat, and M. Avellaneda. Greedy adaptive approximation. *Constr Approx*, 13:57–98, 03 1997. doi: 10.1007/BF02678430.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2), apr 2004. doi: 10.1214/009053604000000067.

L. El Ghaoui, V. Viallon, and T. Rabbani. Safe Feature Elimination for the LASSO and Sparse Supervised Learning Problems. (arXiv:1009.4219), May 2011. doi: 10.48550/arXiv.1009.4219. arXiv:1009.4219 [cs, math] type: article.

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001. doi: 10.1198/016214501753382273.

J. Fan and J. Lv. Sure independence screening for ultra-high dimensional feature space. *J Roy Stat Soc*, B 70, 01 2008.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, jun 2008. ISSN 1532-4435.

O. Fercoq and P. Richtárik. Accelerated, parallel and proximal coordinate descent. *SIAM Journal on Optimization*, 2013. doi: 10.48550/ARXIV.1312.5799.

O. Fercoq, A. Gramfort, and J. Salmon. Mind the duality gap: safer rules for the Lasso. (arXiv:1505.03410), December 2015. doi: 10.48550/arXiv.1505.03410. arXiv:1505.03410 [cs, math, stat] type: article.

J. Friedman, T. Hastie, and R. Tibshirani. glmnet: Lasso and elastic-net regularized generalized linear models. *R package version*, 1(4), 2009.

D. Ghosh and A. M. Chinnaiyan. Classification and selection of biomarkers in genomic data using lasso. *Journal of Biomedicine and Biotechnology*, 2005(2):147, 2005.

E. Giusti. *Direct Methods in the Calculus of Variations*. World Scientific, 2003. ISBN 978-981-238-043-2.

E. Hale, W. Yin, and Y. Zhang. Fixed-Point Continuation for l1-Minimization: Methodology and Convergence. *SIAM Journal on Optimization*, 19, October 2008. doi: 10.1137/070698920.

W. Hare and C. Sagastizábal. Computing proximal points of nonconvex functions. *Mathematical Programming*, 116(1):221–258, January 2009. ISSN 1436-4646. doi: 10.1007/s10107-007-0124-6.

T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman and Hall/CRC, 2015. ISBN 1498712169.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms. II*, volume 306. Springer-Verlag, 1993.

P. J. Huber. *Robust Statistics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1981. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2_594.

F. Iutzeler and J. Malick. Nonsmoothness in Machine Learning: specific structure, proximal identification, and applications. *Set-Valued and Variational Analysis*, 28(4):661–678, December 2020. doi: 10.1007/s11228-020-00561-1.

T. Joachims. Making large scale svm learning practical. *Advances in Kernel Methods: Upport Vector Machines*, 10 1999. doi: 10.17877/DE290R-5098.

T. Johnson and C. Guestrin. Blitz: A principled meta-algorithm for scaling sparse optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1171–1179, Lille, France, 07–09 Jul 2015. PMLR.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2014. doi: 10.48550/ARXIV.1412.6980.

M. Kowalski, P. Weiss, A. Gramfort, and S. Anthoine. Accelerating ISTA with an active set strategy. page 7, December 2011.

A. Kruger. On fréchet subdifferentials. *Journal of Mathematical Sciences*, 116:3325–3358, 07 2003. doi: 10.1023/A:1023673105317.

S. K. Lam, A. Pitrou, and S. Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

S. Lee, H. Lee, P. Abbeel, and A. Ng. Efficient l1 regularized logistic regression. volume 21, 01 2006.

Q. Lin, Z. Lu, and L. Xiao. An Accelerated Proximal Coordinate Gradient Method and its Application to Regularized Empirical Risk Minimization. (arXiv:1407.1296), July 2014. doi: 10.48550/arXiv.1407.1296. arXiv:1407.1296 [math] type: article.

Z.-Q. T. Luo and P. Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72:7–35, 1992.

T. D. Luu, J. Fadili, and C. Chesneau. Sampling from non-smooth distribution through Langevin diffusion. working paper or preprint, March 2017.

V. V. Mai and M. Johansson. Anderson Acceleration of Proximal Gradient Methods. (arXiv:1910.08590), June 2020. doi: 10.48550/arXiv.1910.08590. arXiv:1910.08590 [cs, math] type: article.

J. Mairal. *Sparse coding for machine learning, image processing and computer vision*. PhD thesis, École normale supérieure de Cachan, 2010.

D. Malioutov, M. Cetin, and A.S. Willsky. Homotopy continuation for sparse signal representation. 5:v/733–v/736 Vol. 5, April 2005. doi: 10.1109/ICASSP.2005.1416408.

M. Massias. From safe screening rules to working sets for faster lasso-type solvers. *10th NIPS Workshop on Optimization for Machine Learning*, 12 2017.

M. Massias, A. Gramfort, and J. Salmon. Celer: a fast solver for the lasso with dual extrapolation. 2018.

J. J. Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société Mathématique de France*, 93:273–299, 1965. doi: 10.24033/bsmf.1625.

T. Moreau, M. Massias, A. Gramfort, P. Ablin, P.-A. Bannier, B. Charlier, M. Dagréou, T. Dupré la Tour, G. Durif, C. F. Dantas, Q. Klopfenstein, J. Larsson, E. Lai, T. Lefort, B. Malézieux, B. Moufad, B. T. Nguyen, A. Rakotomamonjy, Z. Ramzi, J. Salmon, and S. Vaiter. Benchopt: Reproducible, efficient and collaborative optimization benchmarks, 2022.

B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995. doi: 10.1137/S0097539792240406.

E. Ndiaye and I. Takeuchi. Continuation path with linear convergence rate, 2021.

E. Ndiaye, O. Fercoq, A. Gramfort, and J. Salmon. Gap Safe screening rules for sparsity enforcing penalties. (arXiv:1611.05780), December 2017. doi: 10.48550/arXiv.1611.05780. arXiv:1611.05780 [cs, math, stat] type: article.

E. Ndiaye, O. Fercoq, and J. Salmon. Screening rules and its complexity for active set identification. *Journal of Convex Analysis*, 2020.

Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012. doi: 10.1137/100802001.

Y. E. Nesterov. A method for solving the convex programming problem with quadratic convergence rate. *Dokl. Akad. Nauk SSSR*, 269:543–547, 1983.

J. Nutini, M. Schmidt, I. H. Laradji, M. Friedlander, and H. Koepke. Coordinate descent converges faster with the gauss-southwell rule than random selection. *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, page 1632–1641, 2015.

J. Nutini, M. Schmidt, and W. Hare. "active-set complexity" of proximal gradient: How long does it take to find the sparsity pattern? *Optimization Letters*, 13, 06 2019. doi: 10.1007/s11590-018-1325-z.

M. R. Osborne, B. Presnell, and B. A. Turlach. On the LASSO and Its Dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000. ISSN 1061-8600. doi: 10.2307/1390657. Publisher: [American Statistical Association, Taylor & Francis, Ltd., Institute of Mathematical Statistics, Interface Foundation of America].

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830, 2011.

R. A. Poliquin and R. T. Rockafellar. Prox-regular functions in variational analysis. *Transactions of the American Mathematical Society*, 348:1805–1838, 1996.

B. Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4:1–17, 12 1964. doi: 10.1016/0041-5553(64)90137-5.

B. Polyak. *Introduction to Optimization*. 07 1987.

C. Poon and J. Liang. Geometry of First-Order Methods and Adaptive Acceleration. (arXiv:2003.03910), September 2020. doi: 10.48550/arXiv.2003.03910. arXiv:2003.03910 [math] type: article.

F. Rapaport, E. Barillot, and J.-P. Vert. Classification of arraycgh data using fused svm. In *ISMB*, pages 375–382, 2008.

P. Richtárik and M. Takáăź. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Math. Program.*, 144(1–2):1–38, apr 2014. ISSN 0025-5610. doi: 10.1007/s10107-012-0614-z.

R. T. Rockafellar. *Convex analysis*. Princeton Mathematical Series. Princeton University Press, Princeton, N. J., 1970.

V. Roth and B. Fischer. The group-lasso for generalized linear models: Uniqueness of solutions and efficient algorithms. page 848–855, 2008. doi: 10.1145/1390156.1390263.

S. Ruder. An overview of gradient descent optimization algorithms. 2016. doi: 10.48550/ARXIV.1609.04747.

J. Salmon. Majorization minimization. URL http://josephsalmon.eu/enseignement/UW/STAT593/MajorizationMinimization.pdf.

D. Scieur. Generalized framework for nonlinear acceleration, 2019.

D. Scieur, A. d'Aspremont, and F. Bach. Regularized nonlinear acceleration, 2016.

H.-J. M. Shi, S. Tu, Y. Xu, and W. Yin. A primer on coordinate descent algorithms, 2016.

N. Simon, J. Friedman, T. J. Hastie, and R. Tibshirani. A sparse-group lasso. *J. Comput. Graph. Statist.*, 22(2):231–245, 2013.

D. Smith, W. Ford, and A. Sidi. Extrapolation methods for vector sequences. *SIAM Review*, 29:199–233, 06 1987. doi: 10.1137/1029042.

E. Soubies. *Sur Quelques Problèmes de Reconstruction en Imagerie MA-TIRF et en Optimisation Parcimonieuse par Relaxation Continue Exacte de Critères Pénalisés en Norme-l0.* PhD thesis, Université Nice Sophia-Antipolis, 2016.

E. Soubies, L. Blanc-Féraud, and G. Aubert. A Continuous Exact l0 penalty (CEL0) for least squares regularized problem. *SIAM Journal on Imaging Sciences*, 8(3):pp. 1607–1639 (33 pages), July 2015. doi: 10.1137/151003714.

R.V. Southwell. *Relaxation Methods in Engineering Science: A Treatise on Approximate Computation*. Oxford engineering science series. Oxford University Press, 1941.

D. Strohmeier, Y. Bekhti, J. Haueisen, and A. Gramfort. The iterative reweighted mixed-norm estimate for spatio-temporal MEG/EEG source reconstruction. *IEEE transactions on medical imaging*, 35(10):2218–2228, 2016.

R. Sun and Y. Ye. Worst-case complexity of cyclic coordinate descent: $o(n^2)$ gap with randomized version, 2016.

Y. Sun, P. Babu, and D. P. Palomar. Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, 2017. doi: 10.1109/TSP.2016.2601299.

Z. Sun and Y. Yu. A Homotopy Coordinate Descent Optimization Method for l_0-Norm Regularized Least Square Problem. (arXiv:2011.06841), November 2020. doi: 10.48550/arXiv.2011.06841. arXiv:2011.06841 [cs] type: article.

R. Tibshirani. Regression shrinkage and selection via the lasso. 58(1):267–288, 1996.

R. Tibshirani, J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, and R. J. Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 74(2):245–266, 2012. ISSN 13697412, 14679868.

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

R.S. Varga and G.H. Golub. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order richardson iterative methods. part i. *Numerische Mathematik*, 3:147–156, 1961.

J. Wang, J. Zhou, J. Liu, P. Wonka, and J. Ye. A safe screening rule for sparse logistic regression. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

P. Wynn. Acceleration techniques for iterated vector and matrix problems. *Mathematics of Computation*, 16(79): 301–322, 1962.

Zaiwen Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang. A Fast Algorithm for Sparse Reconstruction Based on Shrinkage, Subspace Optimization, and Continuation. *SIAM Journal on Scientific Computing*, 32(4):1832–1857, January 2010. ISSN 1064-8275. doi: 10.1137/090747695. Publisher: Society for Industrial and Applied Mathematics.

C.-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2), apr 2010. doi: 10.1214/09-aos729.

J. Zhang, B. O'Donoghue, and S. Boyd. Globally Convergent Type-I Anderson Acceleration for Non-Smooth Fixed-Point Iterations. (arXiv:1808.03971), August 2018. doi: 10.48550/arXiv.1808.03971. arXiv:1808.03971 [math] type: article.

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476): 1418–1429, 2006. doi: 10.1198/016214506000000735.

H. Zou and T. J. Hastie. Regularization and variable selection via the elastic net. 67(2):301–320, 2005.

A Deriving the dual SVM with Hinge loss

With $\forall i \in [n], x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}$, the primal formulation of the support vector machine (with slack variables) reads

$$\begin{aligned} \min_{\beta \in \mathbb{R}^p} \; & C \sum_{i=1}^{n} \zeta_i + \frac{1}{2} \|\beta\|^2 \\ \text{s.t.} \; & y_i(\beta^\top x_i) \geq 1 - \zeta_i, \;\; \forall i \in [n], \\ & \zeta_i \geq 0, \;\; \forall i \in [n]. \end{aligned} \tag{106}$$

where $\zeta_i, i \in [n]$ are slack variables. Without slack variables, the (hard-margin) SVM assumes that classes are perfectly linearly separable and that the conditional class distributions do not overlap. In practice, this assumption is rarely satisfied and requires the introduction of slack variables, yielding the soft-margin SVM. Data points are allowed to be misclassified on the "wrong side" of the decision boundary, with a penalty that increases proportionally with the distance of the misclassified point to the boundary. The hyperparameter $C > 0$ is a trade-off between the slack variable penalty and the margin. The associated Lagrangian is

$$\mathcal{L}(\beta, \zeta, \alpha, \mu) = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} \zeta_i - \sum_{i=1}^{n} \alpha_i (y_i(\beta^\top x_i) - 1 + \zeta_i) - \sum_{i=1}^{n} \mu_i \zeta_i \;\; . \tag{107}$$

with $\forall i \in [n], \;\; \alpha_i, \mu_i \geq 0$ the Lagrange multipliers. Differentiating the Lagrangian with respect to the primal variables β and ζ yields

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \beta} &= 0 \Rightarrow \beta = \sum_{i=1}^{n} \alpha_i y_i x_i \;\; , \\ \frac{\partial \mathcal{L}}{\partial \zeta_i} &= 0 \Rightarrow \alpha_i = C - \mu_i \;\; . \end{aligned} \tag{108}$$

Eliminating β under these conditions in $\mathcal{L}$ yields the dual problem of SVM

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^n} \; & -\frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k y_i y_k x_k^\top x_i + \sum_{i=1}^{n} \alpha_i \\ \text{s.t.} \; & \forall i \in [n], \;\; 0 \leq \alpha_i \leq C \;\; . \end{aligned} \tag{109}$$

In vectorial form, with the Gram matrix $G \in \mathbb{R}^{n \times n}$ such that $G_{i,k} = y_i y_k x_k^\top x_i$, the problem reads

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{2} \alpha^\top G \alpha - \sum_{i=1}^{n} \alpha_i \quad \text{s.t.} \quad \forall i \in [n], \;\; 0 \leq \alpha_i \leq C \;\; . \tag{110}$$

The box constraints of the dual problem can be expressed with an indicator function

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{2} \alpha^\top G \alpha - \sum_{i=1}^{n} \alpha_i + \iota_{[0,C]}(\alpha_i) \;\; . \tag{111}$$

Problem 111 falls into the framework of Problem 8, with the penalty function $g = \iota_{[0,C]}$.

B Comparing runtimes: `Numba` vs. `Cython`

To implement fast solvers, writing algorithms in low-latency languages is crucial. A trivial choice consists in choosing a low-level language like C or C++ to ensure fast algebraic operations. However, maintaining and extending a codebase written exclusively in these low-latency languages remains a challenge. To circumvent this issue, other alternatives like `Cython` (Behnel et al., 2011) or `Numba` (Lam et al., 2015) have emerged.

`Numba` is a compiler for `Python` scripts. Simply put, it enables to speed up critical computational-intensive parts of a `Python` script by adding a one-line function decorator. This allows for easier code maintenance and comprehension for `Python` users. On the other hand, `Cython` acts as a bridge between `C` and `Python`, enabling near-`C` performance at compile time, at the expense of writing the codebase in a heavier `C`-like syntax. `Cython` and `Numba` having similar performance at compile time, we opted for `Numba` for ease-of-use when designing `skglm` (Bertrand et al., 2022).

C Fair comparison between solvers with `Benchopt`

The `Benchopt` library (Moreau et al., 2022) is a benchmarking tool for optimization solvers in machine learning. Solvers are treated as black boxes, fed with an input dataset $\mathcal{D}$ and minimizing some objective f. The solver is run successively for 1 iteration, then 2 (starting again from 0), then 3, etc. This allows to generate a convergence curve, indicative of the performance of a solver. Directly measuring the time per iteration, or the total time taken by an algorithm would not be a reproducible strategy since solvers are heavily reliant on the underlying hardware specifications of the machine they are running on.

Plots generated by `Benchopt` should be analyzed by taking into account the steepness of the curve (the steeper the better), as well as the number of iterations taken by the solver to converge.

D Distance to the subdifferential for convex and non-convex penalties

Table 6: Distance to the subdifferential for single-task convex and non-convex penalties

Penalty	$\mathrm{dist}(-\nabla_j f(\beta), \partial g_j(\beta))$
ℓ_1	$\begin{cases} \max(0, \lvert\nabla_j f(\beta)\rvert - \lambda) & \text{if } \beta_j = 0 \ , \\ \lvert -\nabla_j f(\beta) - \lambda \mathrm{sgn}(\beta_j)\rvert & \text{otherwise .} \end{cases}$
$\ell_1 + \ell_2$	$\begin{cases} \max(0, \lvert\nabla_j f(\beta)\rvert - \lambda\gamma) & \text{if } \beta_j = 0 \ , \\ \lvert -\nabla_j f(\beta) - \lambda(\gamma \mathrm{sgn}(\beta_j) + (1-\gamma)\beta_j)\rvert & \text{otherwise .} \end{cases}$
$\ell_{0.5}$	$\begin{cases} 0 & \text{if } \beta_j = 0 \ , \\ \left\lvert -\nabla_j f(\beta) - \lambda \frac{\mathrm{sgn}(\beta_j)}{2\sqrt{\lvert\beta_j\rvert}} \right\rvert & \text{otherwise .} \end{cases}$
$\ell_{2/3}$	$\begin{cases} 0 & \text{if } \beta_j = 0 \ , \\ \left\lvert -\nabla_j f(\beta) - \lambda \frac{2\mathrm{sgn}(\beta_j)}{3\lvert\beta_j\rvert^{1/3}} \right\rvert & \text{otherwise .} \end{cases}$
MCP	$\begin{cases} \max(0, \lvert\nabla_j f(\beta)\rvert - \lambda) & \text{if } \beta_j = 0 \ , \\ \left\lvert \nabla_j f(\beta) + \lambda \mathrm{sgn}(\beta_j) - \frac{\beta_j}{\gamma} \right\rvert & \text{if } \lvert\beta_j\rvert < \lambda\gamma \ , \\ \lvert\nabla_j f(\beta)\rvert & \text{otherwise .} \end{cases}$
SCAD	$\begin{cases} \max(0, \lvert\nabla_j f(\beta)\rvert - \lambda) & \text{if } \beta_j = 0 \ , \\ \lvert\nabla_j f(\beta) + \lambda \mathrm{sgn}(\beta_j)\rvert & \text{if } \lvert\beta_j\rvert < \lambda \ , \\ \left\lvert \nabla_j f(\beta) + \frac{\lambda\gamma \mathrm{sgn}(\beta_j) - \beta_j}{\gamma - 1} \right\rvert & \text{if } \lambda \leq \lvert\beta_j\rvert \leq \lambda\gamma \ , \\ \lvert\nabla_j f(\beta)\rvert & \text{otherwise.} \end{cases}$
$\iota_{[0,C]}$	$\begin{cases} \max(0, -\nabla_j f(\beta)) & \text{if } \beta_j = 0 \ , \\ \max(0, \nabla_j f(\beta)) & \text{if } \beta_j = \lambda \ , \\ \lvert\nabla_j f(\beta)\rvert & \text{otherwise .} \end{cases}$

Table 7: Distance to the subdifferential for multi-task convex and non-convex penalties

Penalty	$\operatorname{dist}(-\nabla_j f(\mathbf{B}), \partial g_j(\mathbf{B}))$
$\ell_{2,1}$	$\begin{cases} \max(0, \lVert\nabla_j f(\mathbf{B})\rVert - \lambda) & \text{if } \mathbf{B}_{j,:} = \mathbf{0} \\ \left\lVert \nabla_j f(\mathbf{B}) + \lambda \frac{\mathbf{B}_{j,:}}{\lVert \mathbf{B}_{j,:} \rVert} \right\rVert & \text{otherwise.} \end{cases}$
$\ell_{2,0.5}$	$\begin{cases} 0 & \text{if } \mathbf{B}_{j,:} = \mathbf{0} \\ \left\lVert \nabla_j f(\mathbf{B}) + \lambda \frac{\mathbf{B}_{j,:}}{(2\lVert \mathbf{B}_{j,:} \rVert)^{3/2}} \right\rVert & \text{otherwise.} \end{cases}$
Block MCP	$\begin{cases} \max(0, \lVert\nabla_j f(\mathbf{B})\rVert - \lambda) & \text{if } \mathbf{B}_{j,:} = \mathbf{0} \\ \left\lVert \nabla_j f(\mathbf{B}) + \lambda \frac{\mathbf{B}_{j,:}}{\lVert \mathbf{B}_{j,:} \rVert} - \frac{\mathbf{B}_{j,:}}{\gamma} \right\rVert & \text{if } \lVert\mathbf{B}_{j,:}\rVert \leq \lambda\gamma \\ \lVert\nabla_j f(\mathbf{B})\rVert & \text{otherwise.} \end{cases}$
Block SCAD	$\begin{cases} \max(0, \lVert\nabla_j f(\mathbf{B})\rVert - \lambda) & \text{if } \mathbf{B}_{j,:} = \mathbf{0} \\ \left\lVert \nabla_j f(\mathbf{B}) + \lambda \frac{\mathbf{B}_{j,:}}{\lVert \mathbf{B}_{j,:} \rVert} \right\rVert & \text{if } \lVert\mathbf{B}_{j,:}\rVert \leq \lambda \\ \left\lVert \nabla_j f(\mathbf{B}) + \frac{\lambda\gamma - \lVert \mathbf{B}_{j,:} \rVert}{\lVert \mathbf{B}_{j,:} \rVert (\gamma - 1)} \mathbf{B}_{j,:} \right\rVert & \text{if } \lambda \leq \lVert\mathbf{B}_{j,:}\rVert \leq \gamma\lambda \\ \lVert\nabla_j f(\mathbf{B})\rVert & \text{otherwise.} \end{cases}$

E Example code in `skglm`

```python
class MCPenalty(BasePenalty):
    def __init__(self, alpha, gamma):
        self.alpha = alpha
        self.gamma = gamma

    def value(self, w):
        """Compute the value of MCP."""
        s0 = np.abs(w) < self.gamma * self.alpha
        value = np.full_like(w, self.gamma * self.alpha ** 2 / 2.)
        value[s0] = (self.alpha * np.abs(w[s0]) - w[s0]**2
                     / (2 * self.gamma))
        return np.sum(value)

    def prox_1d(self, value, stepsize, j):
        """Compute the proximal operator of MCP."""
        tau = self.alpha * stepsize
        g = self.gamma / stepsize
        if np.abs(value) <= tau:
            return 0.
        if np.abs(value) > g * tau:
            return value
        return np.sign(value) * (np.abs(value) - tau) / (1. - 1./g)

    def subdiff_distance(self, w, grad, ws):
        """Compute distance of -grad to the subdifferential at w."""
        subdiff_dist = np.zeros_like(grad)
        for idx, j in enumerate(ws):
            if w[j] == 0:
                subdiff_dist[idx] = max(
                    0, np.abs(grad[idx]) - self.alpha)
            elif np.abs(w[j]) < self.alpha * self.gamma:
                subdiff_dist[idx] = np.abs(
                    grad[idx] + self.alpha * np.sign(w[j])
                    - w[j] / self.gamma)
            else:
                # distance of grad to 0
                subdiff_dist[idx] = np.abs(grad[idx])
        return subdiff_dist

    def is_penalized(self, n_features):
        """Return a binary mask with the penalized features."""
        return np.ones(n_features, bool_)

    def generalized_support(self, w):
        """Return a mask with non-zero coefficients."""
        return w != 0
```

Listing 1: **MCP implementation in `skglm` within a `Python` class**. A penalty in `skglm` must implement only 5 methods to be fully usable.

```python
class Quadratic(BaseDatafit):
    def __init__(self):
        pass

    def initialize(self, X, y):
        self.Xty = X.T @ y
        n_features = X.shape[1]
        self.lipschitz = np.zeros(n_features, dtype=X.dtype)
        for j in range(n_features):
            self.lipschitz[j] = (X[:, j] ** 2).sum() / len(y)

    def initialize_sparse(
            self, X_data, X_indptr, X_indices, y):
        n_features = len(X_indptr) - 1
        self.Xty = np.zeros(n_features, dtype=X_data.dtype)
        self.lipschitz = np.zeros(n_features, dtype=X_data.dtype)
        for j in range(n_features):
            nrm2 = 0.
            xty = 0
            for idx in range(X_indptr[j], X_indptr[j + 1]):
                nrm2 += X_data[idx] ** 2
                xty += X_data[idx] * y[X_indices[idx]]

            self.lipschitz[j] = nrm2 / len(y)
            self.Xty[j] = xty

    def value(self, y, w, Xw):
        return np.sum((y - Xw) ** 2) / (2 * len(Xw))

    def gradient_scalar(self, X, y, w, Xw, j):
        return (X[:, j] @ Xw - self.Xty[j]) / len(Xw)

    def gradient_scalar_sparse(
        self, X_data, X_indptr, X_indices, y, Xw, j):
        XjTXw = 0.
        for i in range(X_indptr[j], X_indptr[j+1]):
            XjTXw += X_data[i] * Xw[X_indices[i]]
        return (XjTXw - self.Xty[j]) / len(Xw)

    def full_grad_sparse(
            self, X_data, X_indptr, X_indices, y, Xw):
        n_features = X_indptr.shape[0] - 1
        n_samples = y.shape[0]
        grad = np.zeros(n_features, dtype=Xw.dtype)
        for j in range(n_features):
            XjTXw = 0.
            for i in range(X_indptr[j], X_indptr[j + 1]):
                XjTXw += X_data[i] * Xw[X_indices[i]]
            grad[j] = (XjTXw - self.Xty[j]) / n_samples
        return grad
```

Listing 2: **Quadratic datafit implementation in `skglm`**. The Quadratic datafit is concisely implemented in just 50 lines of code and 6 methods.