Beyond L1: Faster and Better Sparse Models with skglm

Quentin Bertrand
Mila & UdeM, Canada
`quentin.bertrand@mila.quebec`

Quentin Klopfenstein
Luxembourg Centre for Systems Biomedicine
University of Luxembourg
Esch-sur-Alzette, Luxembourg

Pierre-Antoine Bannier
Independent Researcher

Gauthier Gidel
Mila & UdeM, Canada
Canada CIFAR AI Chair

Mathurin Massias
Univ. Lyon, Inria, CNRS, ENS de Lyon,
UCB Lyon 1, LIP UMR 5668, F-69342
Lyon, France

Abstract

We propose a new fast algorithm to estimate any sparse generalized linear model with convex or non-convex separable penalties. Our algorithm is able to solve problems with millions of samples and features in seconds, by relying on coordinate descent, working sets and Anderson acceleration. It handles previously unaddressed models, and is extensively shown to improve state-of-art algorithms. We release `skglm`, a flexible, `scikit-learn` compatible package, which easily handles customized datafits and penalties.

1 Introduction

Sparse generalized linear models play a central role in modern machine learning and signal processing. The Lasso (Tibshirani, 1996) and its derivatives (Zou and Hastie, 2005; Ng, 2004; Candes et al., 2008; Simon et al., 2013) have found numerous successful applications to large scale tasks in genomics (Ghosh and Chinnaiyan, 2005), vision (Mairal, 2010), or neurosciences (Strohmeier et al., 2016). This impact was made possible by two key factors: efficient algorithms and software implementations.

State-of-the-art algorithms for "smooth + non-smooth separable" problems predominantly rely on coordinate descent (CD, Tseng and S.Yun 2009; Nesterov 2012), which, when it can be applied, is more efficient than full gradient methods (Richtárik and Takáč, 2014, Sec. 6.1). Coordinate descent can even be improved with Nesterov-like acceleration, to obtain improved convergence rates (Lin et al., 2014; Fercoq and Richtárik, 2015). However, these better rates may fail to reflect in practical accelerations. On the contrary, Bertrand and Massias (2021) relied on Anderson acceleration (Anderson, 1965) to provide both better rates and practical acceleration for coordinate descent.

Even with efficient algorithms such as coordinate descent, the practical use of sparsity hits a computational barrier for problems with more than millions of features (Le Morvan and Vert, 2018). Multiple techniques have been proposed to make coordinate descent scale to huge problems. Notably, algorithms can be accelerated by reducing the number of variables to optimize over, using screening rules or working sets. Screening rules discard features from the problem in advance (El Ghaoui et al. 2010; Bonnefoy et al. 2015) or dynamically (Fercoq et al., 2015; Ndiaye et al., 2017). On the other side, working sets (Johnson and Guestrin, 2015; Massias et al., 2018) iteratively solve larger subproblems and progressively include variables identified as relevant.

For the Lasso and a few convex models, coordinate descent has been broadly disseminated to practitioners in off-the-shelf packages such as `glmnet` (Friedman et al., 2007) or `scikit-learn`

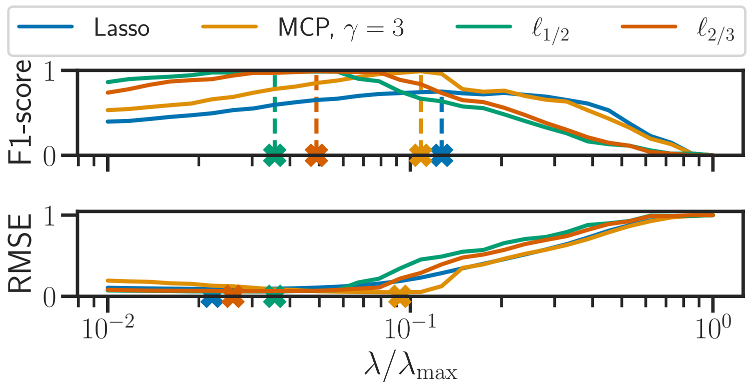


Figure 1: **Regularization paths computed with our algorithm.** Non-convex sparse penalties behave better than the L1 norm. Due to their lower bias, they achieve perfect support recovery, lower prediction error and their optimal regularization strength λ in estimation (top) and prediction (bottom) correspond.

(Pedregosa et al., 2011). More recently, `celer`, a state-of-the-art convex working set algorithm (Massias et al., 2020) allowed for successful applications of the Lasso in large scale problems in medicine (Reidenbach et al., 2021; Kim et al., 2021) or seismology (Muir and Zhan, 2021).

Yet the Lasso is limited: non-convex sparse models enjoy better theoretical and empirical properties (Breheny and Huang, 2011; Soubies et al., 2015). As illustrated in Figure 1, they yield sparser solutions than convex penalties and mitigate the intrinsic Lasso bias. Yet, they have not so often been applied to huge scale applications. This is mostly an algorithmic barrier: while coordinate descent can be applied to non-convex penalties (Breheny and Huang, 2011; Mazumder et al., 2011; Bolte et al., 2014), screening rules and working sets are heavily dependent on convexity or quadratic datafits (Rakotomamonjy et al., 2019, 2022).

In this work, we solve this issue by designing a **state-of-the-art generic algorithm** to solve a wide range of sparse generalized linear models. The contributions are the following:

- We propose a non-convex converging working set algorithm relying on Anderson accelerated coordinate descent. For a specific class of non-convex penalties, we show:
  (a) Convergence of the proposed working set algorithm (Proposition 5).
  (b) Support identification of coordinate descent (Proposition 10).
  (c) Local convergence rates for the Anderson extrapolation (Proposition 13).
- We provide an extensive experimental comparison and we show state-of-the-art improvements on a wide range of convex and non-convex problems. In addition we release an efficient and modular python implementation, with a `scikit-learn` API, for practitioners to apply non-convex penalties to large scale problems.

2 Framework and proposed algorithm

2.1 Problem setting

In this paper, we consider problems of the form:

$$\hat{\beta} \in \operatorname*{argmin}_{\beta \in \mathbb{R}^p} \Phi(\beta) \triangleq \underbrace{F(X\beta)}_{\triangleq f(\beta)} + \sum_{j=1}^{p} g_j(\beta_j) \ , \tag{1}$$

where F is smooth, and the functions g_j are proper and lower semicontinuous but not necessarily convex, whose proximal operator can be computed exactly. We write $g = \sum_j g_j$. Instances of Problem (1) include convex estimators: the Lasso, the elastic net, the sparse logistic regression, the dual of SVM with hinge loss. They also include non-convex penalties: $\ell_{0.5}$ and $\ell_{2/3}$ penalties (Foucart and Lai, 2009), the minimax concave penalty (MCP, Zhang 2010) or SCAD (Zhang, 2010), both with regression and classification losses. Formally, the assumptions are the following.

Assumption 1. *$f : \mathbb{R}^p \to \mathbb{R}$ is convex and differentiable and for all $j \in [p]$, the restriction of $\nabla_j f$ to the j-th coordinate is L_j-Lipschitz: for all $(x, h) \in \mathbb{R}^p \times \mathbb{R}$, $|\nabla_j f(x + he_j) - \nabla_j f(x)| \leq L_j |h|$.*

Assumption 2. *For any $j \in [p]$, $g_j : \mathbb{R} \to \mathbb{R}$ is proper, closed, and lower bounded.*

Following Attouch and Bolte 2009; Bolte et al. 2014 we focus on finding a critical point of Φ.

Definition 3. *Using the Fréchet subdifferential (Kruger, 2003), a critical point $x \in \mathbb{R}^p$ is a point which satisfies $-\nabla f(x) \in \partial g(x)$.*

Assumptions 1 and 2 are usual, and, under boundedness of the iterates, ensure convergence of forward-backward and coordinate descent algorithms to a critical point (Attouch et al. 2013, Thm 5.1, Bolte et al. 2014, Thm. 3.1). In addition, our work focuses on the case where g_j's present non-differentiability points, leading to the following extended notion of sparsity.

Definition 4 (Generalized support)**.** *The* generalized support *of $\beta \in \mathbb{R}^p$ is the set of indices $j \in [p]$ such that g_j is differentiable at β_j:* $\operatorname{gsupp}(\beta) = \{j \in [p] : \partial g_j(\beta_j) \text{ is a singleton}\}$.

Penalties such as ℓ_1, ℓ_q $(0 < q < 1)$, MCP or SCAD are only not differentiable at 0, and this corresponds to the usual notion of sparsity. But Definition 4 goes beyond sparsity and extends to estimators such as SVM, where $g_j = \iota_{[0,C]}$ and the generalized support is the complement of the support vectors' set $\{j \in [p] : \beta_j = 0 \text{ or } \beta_j = C\}$. The generalized support of a critical point is usually of cardinality much smaller than p, and its knowledge makes the problem easier and faster to solve. Our working set algorithm exploits this structure in order to converge faster.

2.2 Proposed algorithm

The proposed algorithm exploits two main ideas:

- A working set strategy, able to handle a large class of convex and non-convex penalties (Algorithm 1).
- An Anderson accelerated coordinate descent for non-convex problems (Algorithm 2). The building blocks of Algorithm 2, coordinate descent (`CD`, Algorithm 3) and Anderson extrapolation (`Anderson`, Algorithm 4), can be found in Appendix A.

To avoid wasting computation on features outside the generalized support, working set algorithms iteratively select a subset of coordinates deemed important (the *working set*), and solve Problem (1) restricted to them. The key question is thus the notion of *important* features. Stemming from Definition 3, we rank features by their violation of the optimality condition: $\operatorname{score}_j^\partial = \operatorname{dist}(-\nabla_j f(\beta), \partial g_j(\beta))$. For example, the MCP Fréchet subdifferential at 0 is $\partial g_j(0) = [-\lambda, \lambda]$, and the proposed score reads

$$\operatorname{score}_j^\partial = \begin{cases} \max\{0, |\nabla_j f(\beta)| - \lambda\} & \text{if} \quad \beta_j = 0 \ , \\ |\nabla_j f(\beta) + \nabla g_j(\beta_j)| & \text{otherwise} \ . \end{cases} \tag{2}$$

To control the working set growth, we use $\operatorname{score}_j^\partial$ to rank the features. Then, with $n_k = \max(n_{k-1}, 2\,|\operatorname{gsupp}(\beta^{(t)})|)$ we take the n_k largest of them in the working set, while retaining features currently in the working set. This growth quickly rises to the unknown size of the generalized support while avoiding overshooting, as backed up by recent theory in Ndiaye and Takeuchi (2021).

Proposition 5. *Let $\mathcal{W}_t$ be the t-th working set. Suppose that Algorithm 2 converges toward a critical point, and for all $t \geq 0$, $\mathcal{W}_t \subset \mathcal{W}_{t+1}$, then the iterates of Algorithm 1 converge towards a critical point of Problem (1).*

Algorithm 1 `skglm` (proposed)

```
input: $X, \beta \in \mathbb{R}^p, n_{\text{out}} \in \mathbb{N}$,
       $n_{\text{in}} \in \mathbb{N}, \text{ws\_size} \in \mathbb{N}, \epsilon > 0$
1 for $t = 1, \dots, n_{\text{out}}$ do
2     score $= \left(\text{dist}\left(-\nabla_j f(\beta), \partial g_j(\beta_j)\right)\right)_{j \in [p]}$
3     ws_size $= \max(\text{ws\_size}, 2 \times |\text{gsupp}(\beta)|)$
      // ws_size features with largest scores
4     ws $= \text{arg\_topK}(\text{score}, \text{K} = \text{ws\_size})$
5     if $\max_{j \in [p]} \text{dist}\left(-\nabla_j f(\beta), \partial g_j(\beta_j)\right) \leq \epsilon$
        then stop
6     else // accelerated CD on working set
7     $\beta \leftarrow$ inner_solver$(X, \beta, \text{ws}, n_{\text{in}}, \epsilon)$
8 return $\beta$
```

Algorithm 2 `inner_solver`

```
input: $X, \beta^{(0)} \in \mathbb{R}^p, \text{ws} \subset [p], n_{\text{in}}, \epsilon, M = 5$
1 for $k = 1, \dots, n_{\text{in}}$ do
2     $\beta^{(k)} \leftarrow$ CD$(X, \beta^{(k-1)}, X\beta, \text{ws})$ // Algo. (3)
3     if $k \bmod M = 0$ then
          // Algo. (4), $\mathcal{O}(M^2|\text{ws}| + M^3)$
4         $\beta^{\text{extr}}_{\text{ws}} \leftarrow$ Anderson$(\beta^{(k-M)}_{\text{ws}}, \dots, \beta^{(M)}_{\text{ws}})$
          // test objective $\mathcal{O}(n|\text{ws}|)$
5         if $\Phi(\beta^{\text{extr}}_{\text{ws}}) < \Phi(\beta^{(k)}_{\text{ws}})$ then
6             $\beta^{(k)}_{\text{ws}} \leftarrow \beta^{\text{extr}}_{\text{ws}}; X\beta \leftarrow X_{\text{ws}}\beta^{\text{extr}}_{\text{ws}}$
7     if $\max_{j \in \text{ws}} \text{dist}\left(-\nabla_j f(\beta), \partial g_j(\beta_j)\right) \leq \epsilon$
        then stop
8 return $\beta^{(k)}$
```

Proof of Proposition 5 can be found in Appendix B.1. The second key ingredient to our algorithm is to use state-of-the-art Anderson accelerated coordinate descent for non-convex problems. In Section 2.3 we show that coordinate descent yields finite time support identification for a large class of non-convex problems (Proposition 10), which leads to acceleration (Proposition 13). As experiments demonstrate in Section 3, this rate allows our algorithm to surpass state-of-the-art solvers.

2.3 Anderson accelerated coordinate descent analysis for α-semi-convex penalties

We now turn to our main technical contributions: we show that Algorithm 2 achieves finite time support identification (Proposition 10) of the generalized support (Definition 4) for specific class of non-smooth non-convex penalties (Assumption 6), which includes the MCP (Proposition 7). Based on Proposition 10, we are able to derive convergence rates for Anderson acceleration (Proposition 13).

We study our inner solver (Algorithm 2); for convenience we still refer to β and X for their counterparts restricted to the working set. The following assumptions are required.

Assumption 6 (α-semi-convex)**.** *For all $j \in [p]$ g_j/L_j is α-semi-convex,* i.e., *$g_j/L_j + \alpha\|\cdot\|^2/2$ is convex, with $\alpha < 1$.*

Note that in statistics, the admissible value range of hyperparameters for MCP and SCAD are datafit-dependent, (see Breheny and Huang 2011, Sec. 2.1, normalized columns and $\gamma > 1 = 1/\|X_{:j}\| = 1/L_j$ or Soubies et al. 2015, Eq. 4.2) and yields α-semi-convexity for MCP and SCAD[1].

Proposition 7 (α-semi-convexity of MCP)**.** *Let* $\text{MCP}_{\lambda,\gamma}(x) \triangleq \begin{cases} \lambda|x| - \frac{x^2}{2\gamma} \ , & \text{if } |x| \leq \gamma\lambda \ , \\ \frac{1}{2}\gamma\lambda^2 \ , & \text{if } |x| > \gamma\lambda \ . \end{cases}$
If $\gamma > 1/L_j$, then $\text{MCP}_{\lambda,\gamma}/L_j$ is α-semi-convex with $\alpha = \frac{1}{2}(1 + \frac{1}{\gamma L_j})$ (i.e., *Assumption 6 holds).*

Note that Assumption 6 does not hold for the ℓ_q-penalties ($0 < q < 1$), for which we propose an alternative in Appendix C.

Assumption 8 (Existence)**.** *Problem (1) admits at least one critical point.*

In Proposition 10, convergence of Algorithm 2 toward a critical point $\hat{\beta}$ is assumed, and the following assumption is made on this critical point.

Assumption 9 (Non degeneracy)**.** *The considered critical point $\hat{\beta} \in \mathbb{R}^p$ is non-degenerated: for all $j \notin \text{gsupp}(\hat{\beta})$,*

$$-\nabla f_j(\hat{\beta}) \in \text{interior}(\partial g_j(\hat{\beta}_j)). \tag{3}$$

Assumption 9 is a generalization of qualification constraints (Hare and Lewis, 2007, Sec. 1), and is usual in the machine learning literature (Zhao and Yu, 2006; Bach, 2008; Vaiter et al., 2015). For

[1]However MCP and SCAD are not α-semiconvex for all hyperparameter values.

the ℓ_1-norm, if the entries of the design matrix X are drawn from an i.i.d normal distribution, then Assumption 9 holds with high probability (Candes and Tao, 2005; Rudelson and Vershynin, 2008).

Equipped with the previous assumptions we show that coordinate descent achieves model identification for this class of non-convex problems.

Proposition 10 (Model identification of CD)**.** *Suppose*

1. *Assumptions 1, 2, 6 and 8 hold.*
2. *The sequence $(\beta^{(k)})_{k\geq 0}$ generated by coordinate descent (Algorithm 2 without extrapolation) converges toward a critical point $\hat{\beta}$.*
3. *Assumption 9 holds for $\hat{\beta}$.*

Then, Algorithm 2 (without extrapolation) identifies the model in finitely many iterations: there exists $K > 0$ such that for all $k \geq K$, $\beta^{(k)}_{\mathcal{S}^c} = \hat{\beta}_{\mathcal{S}^c}$.

In other words, for k large enough, $\beta^{(k)}$ shares the generalized support of $\hat{\beta}$. The identification property was proved for a proximal gradient descent algorithm in the non-convex case (Liang et al., 2016) under the assumption that the non-smooth function g is partly smooth (Lewis, 2002). For ourselves, Proposition 10 not rely on the partly smooth assumption to ensure identification property. Authors are not aware of previous identification results for coordinate descent in the non-convex case.

In addition, if f and g are locally regular on the generalized support at the considered critical point, our algorithm enjoys local acceleration when combined with Anderson extrapolation (Proposition 13).

Assumption 11 (Locally $\mathcal{C}^3$)**.** *For all $j \in \mathcal{S} \triangleq \operatorname{gsupp}(\hat{\beta})$, g_j is locally $\mathcal{C}^3$ around $\hat{\beta}_j$, and f is locally $\mathcal{C}^3$ around $\hat{\beta}$.*

Assumption 11 on the function f is mild and holds for usual machine learning datafitting terms. Assumption 11 on the functions g_j, $j \in \mathcal{S}$, is stronger: for instance, for the MCP, it implies $\hat{\beta}_j \neq \gamma\lambda$ for all $j \in \mathcal{S}$. However this assumption is standard in the literature, see Liang et al. 2016, Sec. 3.3

Assumption 12. *(Local strong convexity) The Hessian of f at the considered critical point $\hat{\beta} \in \mathbb{R}^p$, restricted to its generalized support $\mathcal{S}$, is positive definite,* i.e., $\nabla^2_{\mathcal{S},\mathcal{S}} f(\hat{\beta}) + \nabla^2_{\mathcal{S},\mathcal{S}} g(\hat{\beta}) \succ 0$.

Assumption 12 requires local strong convexity restricted to the generalized support $\mathcal{S}$, which is standard in the MCP / SCAD literature (Breheny and Huang 2011, Section 4.1) and is usual to derive local linear rates of convergence (Liang et al., 2016, Section 3.3). For instance, for the Lasso, if the entries of the design matrix X are drawn from a continuous distribution, then Assumption 12 holds with probability one (Tibshirani, 2013, Lemma 4).

Proposition 13. *Consider a critical point $\hat{\beta}$ and suppose*

1. *Assumptions 1, 2 and 8 hold.*
2. *The functions f and g_j, $j \in [p]$ are piecewise quadratic (which is the case for the MCP regression).*
3. *The sequence $(\beta^{(k)})_{k\geq 0}$ generated by Anderson accelerated coordinate descent with updates from 1 to p and p to 1 (Algorithm 2 with extrapolation) converges to a critical point $\hat{\beta}$.*
4. *Assumptions 9, 11 and 12 hold for $\hat{\beta}$.*

Then there exists $K \in \mathbb{N}$, and a $\mathcal{C}^1$ function $\psi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ such that, for all $k \in \mathbb{N}, k \geq K$:

$$\beta^{(k)}_j = \hat{\beta}_j \text{ , for all } j \in \mathcal{S}^c , \tag{4}$$

Let $T \triangleq \mathcal{J}\psi(\hat{\beta})$, $H \triangleq \nabla^2_{\mathcal{S},\mathcal{S}} f(\hat{\beta}) + \nabla^2_{\mathcal{S},\mathcal{S}} g(\hat{\beta})$, $\zeta \triangleq (1 - \sqrt{1-\rho(T)})/(1 + \sqrt{1-\rho(T)})$ and $B \triangleq (T - \mathrm{Id})^\top (T - \mathrm{Id})$. Then $\rho(T) < 1$ and the iterates of Anderson extrapolation enjoy local accelerated convergence rate:

$$\|\beta^{(k-K)}_{\mathcal{S}} - \hat{\beta}_{\mathcal{S}}\|_B \leq \left(\sqrt{\kappa(H)} \frac{2\zeta^{M-1}}{1+\zeta^{2(M-1)}}\right)^{(k-K)/M} \|\beta^{(K)}_{\mathcal{S}} - \hat{\beta}_{\mathcal{S}}\|_B \ . \tag{5}$$

The proof can be found in Appendix B.5.

Table 1: Most popular packages for sparse generalized linear models.

Name	Acceleration	Huge scale	Nncvx	Modular
`glmnet` (Friedman et al., 2010)	✗	✗	✗	✗ (Fortran)
`scikit-learn` (Pedregosa et al., 2011)	✗	✗	✗	✗ (Cython)
`lightning` (Blondel and Pedregosa, 2016)	✗	✗	✗	✓ (Cython)
`celer` (Massias et al., 2018)	✓	✓	✗	✗ (Cython)
`picasso` (Ge et al., 2019)	✗	✗	✓	✗ (`C++`)
`pyGLMnet` (Jas et al., 2020)	✗	✗✗	✗	✓ (Python)
`fireworks` (Rakotomamonjy et al., 2022)	✗	✓	✓	N.A. (Python)
`skglm` (ours)	✓	✓	✓✓	✓ (Python)

Related work. Most Anderson acceleration convergence results are shown for quadratic objectives for specific algorithms: gradient descent (Golub and Varga, 1961; Anderson, 1965), ADMM (Poon and Liang, 2019), coordinate descent (Bertrand et al., 2020). Outside of the quadratic case, convergence results are usually significantly weaker (Scieur et al., 2016; Sidi, 2017; Brezinski et al., 2018; Mai and Johansson, 2019; Ouyang et al., 2020). Regarding the smooth non-convex case, Wei et al. (2021) proposed a stochastic Anderson acceleration and proved convergence towards a critical point. Proposition 13 generalizes Scieur et al. (2020, Prop 2.1) and Bertrand and Massias (2021, Prop. 4) to the proximal convex and α-semi-convex cases. To our knowledge this is one of the first quantitative results for Anderson acceleration in a non-convex setting.

2.4 Comparison with existing work

In this section we compare our contribution to existing algorithms and implementations, which are summarized in Table 1. *Huge scale* refers to the fact that the algorithm can run on problems with millions of variables. *Non-convex* tells if the algorithm handles non-convex penalties. *Modular* indicates that it is easy to add a new model, through a different datafitting term or penalty.

The packages `glmnet` (Friedman et al., 2010), `scikit-learn` (Pedregosa et al., 2011) and `lightning` (Blondel and Pedregosa, 2016) implement coordinate descent (cyclic or random). They rely on compiled code such as Fortran or Cython, making it very difficult to implement new models[2] or faster algorithms like working set[3]. They do not handle non-convex penalties.

More recent algorithms such as `blitz` (Johnson and Guestrin, 2015), `celer` (Massias et al., 2018), `picasso` (Ge et al., 2019) or `fireworks` (Rakotomamonjy et al., 2022) use working set strategies. `celer` and `blitz` are state-of-the-art algorithms for the Lasso, but their score to prioritize features relies on duality. `fireworks` extends `blitz` to some non-convex penalties (writing as difference of convex functions), with $\text{score}_j^{\text{fireworks}} = \text{dist}(-\nabla_j f(\beta), \partial g_j(0))$. Yet this rule does not consider the subdifferential of g at the current point, but at 0, which is a coarse information. Finally, `fireworks`, building upon the seminal non convex working set solver of Boisbunon et al. (2014), does not provide accelerated convergence rates and does not come with a public implementation. `picasso` (Ge et al., 2019) lacks modularity (penalties are hardcoded), and the solver is not suited for huge scale (it does not support large sparse matrices). Deng and Lan (2019) proposed an algorithm based on inertially accelerated coordinate descent, which fails to provide practical speedups according to Bertrand and Massias (2021).

Contrary to these algorithms, ours is generic and relies only on the knowledge of ∇f and prox_g. For any new penalty, this information can be written in a few lines of Python code, compiled with numba (Lam et al., 2015) for speed efficiency. We therefore improve state-of-the-art algorithms in the convex case, and generalize to virtually any datafit and penalty, even nonconvex.

3 Experiments

Our package relying on numpy and numba (Lam et al., 2015; Harris et al., 2020) is attached in the supplementary material. An open source, fully tested and documented version of the code can be

[2]https://github.com/scikit-learn/scikit-learn/pull/10745 (4 years old)
[3] https://github.com/scikit-learn/scikit-learn/pull/7853 (5 years old)

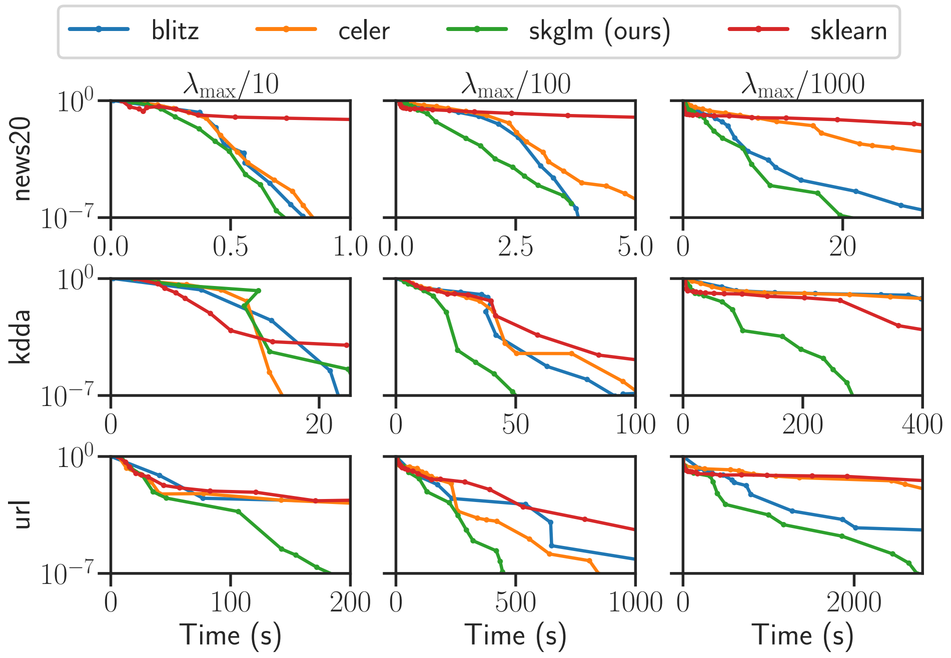


Figure 2: **Lasso, duality gap.** Normalized duality gap as a function of time for the Lasso on multiple datasets, for multiple values of λ.

found at https://github.com/scikit-learn-contrib/skglm. We use datasets from `libsvm`[4] (Fan et al. 2008, see table 2).

We compare multiple algorithms to solve popular Machine Learning and inverse problems: Lasso, Elastic net, multitask sparse regression, MCP regression. The compared algorithms are the following:

- `scikit-learn` (Pedregosa et al., 2011), which implements coordinate descent in Cython,
- `celer` (Massias et al., 2020), which combines working sets, screening rules, coordinate descent, and Anderson acceleration in the dual, in Cython,
- `blitz` (Johnson and Guestrin, 2015), which combines working sets with prox-Newton iterations (Lee et al., 2012) in C++,
- coordinate descent (CD, Tseng and S.Yun 2009),
- `skglm` (Algorithm 1, ours), using $M = 5$ iterates for the Anderson extrapolation.

Other solvers. Experiment per experiment, there exist niche solvers (such as aggressive Gap Safe Rules, Ndiaye et al. 2020). Since our goal is a *general purpose* algorithm able to deal with many models, we do not include them in the comparison. In addition, we focus on solving a single instance of Problem (1), rather than a regularization path (*i.e.,* a sequence of problems for multiple regularization strengths). As `glmnet` is designed to compute regularization paths, we could not include it in the comparison. The reader can refer to Johnson and Guestrin (2015, Fig. 4) or Figure 8 in Appendix E for comparisons on single optimization problems with `glmnet`; `glmnet` and additional algorithms are discussed in Appendix E.

How to do a fair comparison between solvers? To plot the convergence curves, we use the `benchopt`[5] benchmarking package (Moreau et al., 2022). In order to automate and reproduce optimization benchmarks it treats solvers as black boxes. It launches them several times with increasing maximum number of iterations, and stores the resulting objective values and times to reach it. As each point on a solver curve is obtained in a different run, the curves are not monotonic, and there may be several points corresponding to the same time. This merely reflects the variability in solvers running time across runs; we refer to Figure 10 in Appendix E.6 for the inevitability of this phenomenon with black box solvers.

[4] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[5] https://github.com/benchopt/benchopt

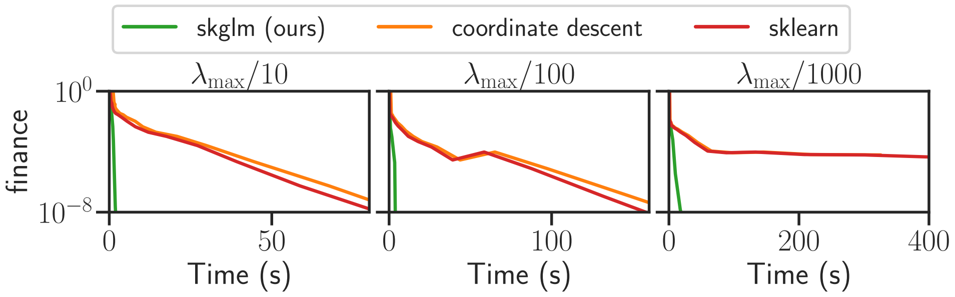
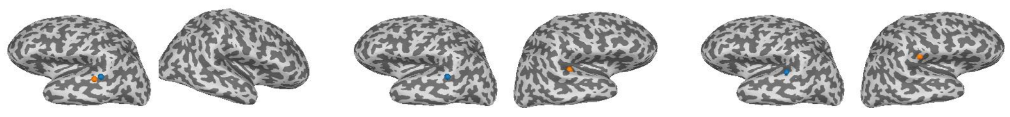


Figure 3: **Elastic net, duality gap.** Normalized duality gap as a function of time for the elastic net for multiple values of λ, $\rho = 0.5$.



(a) $\ell_{2,1}$. (b) $\ell_{2,0.5}$. (c) Block MCP.

Figure 4: **Real data, brain source locations recovered by convex and non-convex penalties after a right auditory stimulation.** 4(a) shows that a convex penalty fails at identifying one source in each hemisphere, while 4(b) and 4(c) demonstrates the capability of non-convex penalties to recover the correct solution.

3.1 Convex problems

Lasso. In Figure 2 we compare solvers for the Lasso: ($f = \frac{1}{2n}\|y - X\cdot\|^2$, $g_j = \lambda|\cdot|$). We parametrize λ as a fraction of $\lambda_{\max} = \|X^\top y\|_\infty / n$, smallest regularization strength for which $\hat{\beta} = 0$. For large scale datasets (*rcv1*, *news20*), `skglm` yields performances better or similar to the state-of-the-art algorithms `blitz` and `celer`. For huge scale datasets (*kdda* and *url*), `skglm` yields significant speedups over them. The improvement over the popular `scikit-learn` can be of two orders of magnitude. Thus, *while dealing with many more models, our algorithm still yields state-of-the-art speed for basic ones*.

Elastic net. Our approach easily generalizes to other problems, such as the elastic net ($f = \frac{1}{2n}\|y - X\cdot\|^2$, $g_j = \lambda(\rho|\cdot| + \frac{1-\rho}{2}(\cdot)^2)$). Figure 3 shows the duality gap as a function of time for `skglm` (ours), `sklearn`, and our numba implementation of coordinate descent. The proposed algorithm is orders of magnitude faster than `scikit-learn` and vanilla coordinate descent, in particular for large datasets and low regularization parameter values (*finance*, $\lambda_{\max}/1000$). Note that `blitz` does not implement a solver for the elastic net. Many Lasso solvers would easily handle the elastic net, but relying on Cython/C++ code makes the implementation time-consuming. By contrast, it takes 40 lines of code to define an $\ell_1 + \ell_2$-squared penalty with our implementation. An additional experiment on the dual of SVM with hinge loss is in Appendix E.4.

3.2 Non-convex problems

In this subsection we propose a comparison on two non convex problems.

MCP regression. MCP regression is Problem (1) with $f = \frac{1}{2n}\|y - X\cdot\|^2$, $g_j = \mathrm{MCP}_{\lambda,\gamma}$ for $\gamma > 1$. As usual for this problem, we scale the columns of X to have norm $\sqrt{n}$. On Figure 5, we compare our algorithm to `picasso` on a dense dataset ($n = 1000, p = 5000$); as this package does not support large sparse design matrices, for the *rcv1* dataset we use an iterative reweighted L1 algorithm (Candes et al., 2008). Since the derivative of the MCP vanishes for values bigger than $\lambda\gamma$, this approach requires solving weighted Lassos with some 0 weights. Up to our knowledge, our

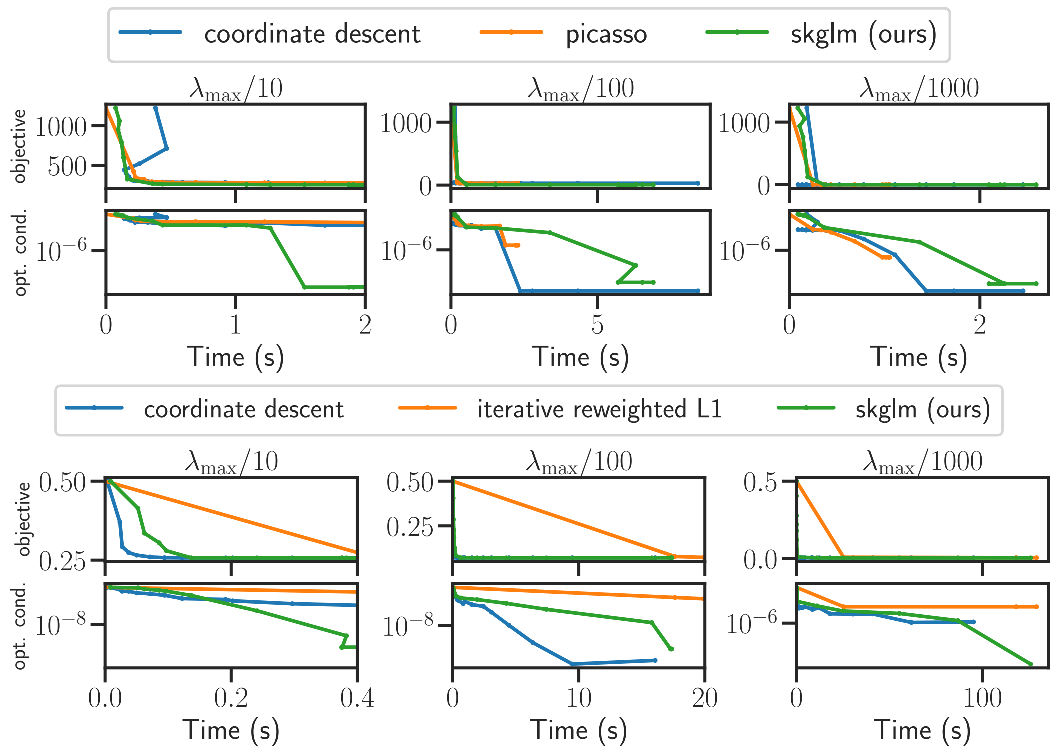


Figure 5: **MCP, objective value and violation of first order condition.** Objective value and violation of optimality condition of the iterates, $\mathrm{dist}(-\nabla f(\beta^{(k)}), \partial g(\beta^{(k)}))$, as a function of time for the MCP for multiple values of λ ($\gamma = 3$) on a simulated dense dataset (top) and the rcv1 dataset (normalized columns).

algorithm is the only efficient one with such a property. Our algorithm handles problems of large size, converges to a critical point, and, due to its progressive inclusion of features, is able to reach a sparser critical point than it competitors.

Application to neuroscience To demonstrate the usefulness of our algorithm for practitioners, we apply it to the magneto-/electroencephalographic (M/EEG) inverse problem. It consists in reconstructing the spatial cortical current density at the origin of M/EEG measurements made at the surface of the scalp. Non-convex penalties (Strohmeier et al., 2015) exhibit several advantages over convex ones (Gramfort et al., 2013): they yield sparser physiologically-plausible solutions and mitigate the ℓ_1 amplitude bias. Here the setting is multitask: $Y \in \mathbb{R}^{n \times T}$ and thus we use block penalties (details in Appendix D). We use real data from the `mne` software (Gramfort et al., 2014); the experiment is a right auditory stimulation, with two expected neural sources to recover in each auditory cortex. In Figure 4, while the $\ell_{2,1}$ penalty fails at localizing one source in each hemisphere, the non-convex penalties recover the correct locations. This emphasizes on the critical need for fast solvers for non-convex sparse penalties as well as our algorithm's ability to handle the latter. In this work we focused on optimization-based estimators to solve the inverse problem, note that one could have resort to other techniques, such as Bayesian techniques (Ghosh and Doshi-Velez, 2017; Fang et al., 2020).

Ablation study. To evaluate the influence of the two components of Algorithm 1, an ablation study (Figure 6) is performed. Four algorithms are compared: with/without working sets and with/without Anderson acceleration. Figure 6 represents the duality gap of the Lasso as a function of time for multiple datasets and values of the regularization parameters λ (parametrized as a fraction of $\lambda_{\max}$). First, Figure 6 shows that working sets always bring significant speedups. Then, when combined with working set, Anderson acceleration bring significant speed-ups, especially for hard problems with low regularization parameters. An interesting observation is that on large scale datasets (*news20* and

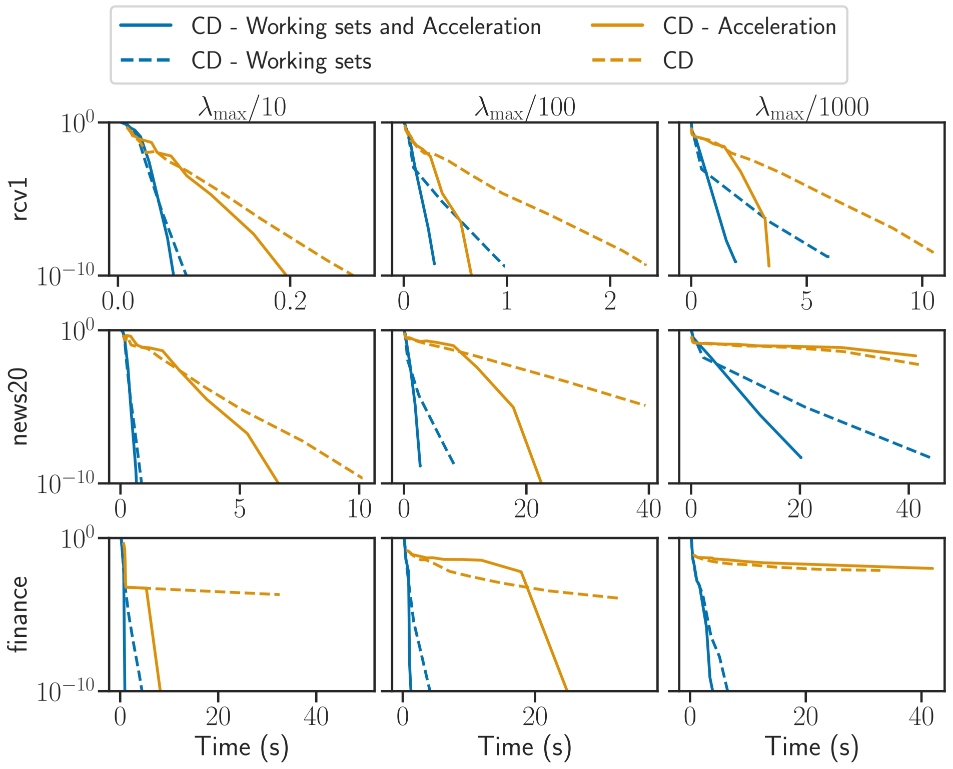


Figure 6: **Lasso, duality gap.** Normalized duality gap as a function of time for the Lasso.

finance) and for low regularization parameters ($\lambda_{\max}/100$ and $\lambda_{\max}/1000$) Anderson acceleration *without* working set does not bring acceleration. This highlights the importance of combining Anderson acceleration with working sets.

Conclusion and broader impact. In this paper, we have proposed an accelerated versatile algorithm for a specific class of non-smooth non-convex problems. Based on working sets, coordinate descent and Anderson acceleration, we have improved state of the art on convex problems, and handled previously out-of-reach problems. Thorough experiments demonstrated the speed and interest of our approach. A limitation of this work is the considered function class (α-semi-convex), which can be seen as restrictive. One possible extension would be weakly convex functions (Davis and Drusvyatskiy, 2019, Sec. 1). We deeply believe that the high quality code provided will benefit to practitioners, and ease the use of non-convex penalties for real world problems, from neuroimaging to genomics. We proposed an optimization algorithm and do not see potential negative societal impacts.

Acknowledgements

The experiments were ran on the CBP cluster of ENS de Lyon (Quemener and Corvellec, 2013). QB would like to thank Samsung Electronics Co., Ldt. for funding this research. GG is supported by an IVADO grant.

References

D. G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12(4): 547–560, 1965.

H. Attouch and J. Bolte. On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. *Mathematical Programming*, 116(1):5–16, 2009.

H. Attouch, J. Bolte, and B. F. Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods. *Mathematical Programming*, 137(1):91–129, 2013.

F. Bach. Consistency of the group Lasso and multiple kernel learning. *J. Mach. Learn. Res.*, 9: 1179–1225, 2008.

Q. Bertrand and M. Massias. Anderson acceleration of coordinate descent. In *AISTATS*, 2021.

Q. Bertrand, Q. Klopfenstein, M. Blondel, S. Vaiter, A. Gramfort, and J. Salmon. Implicit differentiation of lasso-type models for hyperparameter optimization. *ICML*, 2020.

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

A. Boisbunon, R. Flamary, and A. Rakotomamonjy. Active set strategy for high-dimensional non-convex sparse optimization problems. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1517–1521. IEEE, 2014.

J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1):459–494, 2014.

A. Bonnefoy, V. Emiya, L. Ralaivola, and R. Gribonval. Dynamic screening: accelerating first-order algorithms for the Lasso and Group-Lasso. *IEEE Trans. Signal Process.*, 63(19):20, 2015.

S. Boyd, N. Parikh, and E. Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

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, 2011.

C. Brezinski, M. Redivo-Zaglia, and Y. Saad. Shanks sequence transformations and anderson acceleration. *SIAM Review*, 60(3):646–669, 2018.

E. J. Candes and T. Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

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.

D. Davis and D. Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

Q. Deng and C. Lan. Efficiency of coordinate descent methods for structured nonconvex optimization. *arXiv preprint arXiv:1909.00918*, 2019.

L. El Ghaoui, V. Viallon, and T. Rabbani. Safe feature elimination for the lasso and sparse supervised learning problems. *arXiv preprint arXiv:1009.4219*, 2010.

R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

S. Fang, S. Zhe, K.-C. Lee, K. Zhang, and J. Neville. Online bayesian sparse learning with spike and slab priors. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 142–151. IEEE, 2020.

O. Fercoq and P. Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.

O. Fercoq, A. Gramfort, and J. Salmon. Mind the duality gap: safer rules for the lasso. In *ICML*, pages 333–342. PMLR, 2015.

S. Foucart and M.-J. Lai. Sparsest solutions of underdetermined linear systems via ℓ_q-minimization for $0 < q \leq 1$. *Applied and Computational Harmonic Analysis*, 26(3):395–407, 2009.

J. Friedman, T. J. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Ann. Appl. Stat.*, 1(2):302–332, 2007.

J. Friedman, T. J. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *J. Stat. Softw.*, 33(1):1, 2010.

J. Ge, X. Li, H. Jiang, H. Liu, T. Zhang, M. Wang, and T. Zhao. Picasso: A sparse learning library for high dimensional data analysis in r and python. *The Journal of Machine Learning Research*, 20 (1):1692–1696, 2019.

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.

S. Ghosh and F. Doshi-Velez. Model selection in bayesian neural networks via horseshoe priors. *arXiv preprint arXiv:1705.10388*, 2017.

G. H. Golub and R. S. Varga. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order richardson iterative methods. *Numerische Mathematik*, 3(1):147–156, 1961.

A. Gramfort, D. Strohmeier, J. Haueisen, M. S. Hämäläinen, and M. Kowalski. Time-frequency mixed-norm estimates: Sparse M/EEG imaging with non-stationary source activations. *NeuroImage*, 70: 410–422, 2013.

A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, and M. S. Hämäläinen. MNE software for processing MEG and EEG data. *NeuroImage*, 86:446 – 460, 2014.

W. L. Hare and A. S. Lewis. Identifying active manifolds. *Algorithmic Operations Research*, 2(2): 75–75, 2007.

C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al. Array programming with numpy. *arXiv preprint arXiv:2006.10256*, 2020.

M. Jas, T. Achakulvisut, A. Idrizović, D. Acuna, M. Antalek, V. Marques, T. Odland, R. Garg, M. Agrawal, Y. Umegaki, P. Foley, H. Fernandes, D. Harris, B. Li, O. Pieters, S. Otterson, G. De Toni, C. Rodgers, E. Dyer, M. Hamalainen, K. Kording, and P. Ramkumar. Pyglmnet: Python implementation of elastic-net regularized generalized linear models. *Journal of Open Source Software*, 5(47):1959, 2020.

T. B. Johnson and C. Guestrin. Blitz: A principled meta-algorithm for scaling sparse optimization. In *ICML*, volume 37, pages 1171–1179, 2015.

Y. J. Kim, N. Brackbill, E. Batty, J. Lee, C. Mitelut, W. Tong, EJ Chichilnisky, and L. Paninski. Nonlinear decoding of natural images from large-scale primate retinal ganglion recordings. *Neural Computation*, 33(7):1719–1750, 2021.

Q. Klopfenstein, Q. Bertrand, A. Gramfort, J. Salmon, and S. Vaiter. Model identification and local linear convergence of coordinate descent. *arXiv preprint arXiv:2010.11825*, 2020.

A. Y. Kruger. On Fréchet subdifferentials. *Journal of Mathematical Sciences*, 116(3):3325–3358, 2003.

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.

M. Le Morvan and J.-P. Vert. WHInter: A working set algorithm for high-dimensional sparse second order interaction models. In *ICML*, pages 3635–3644. PMLR, 2018.

J. D. Lee, Y. Sun, and M. Saunders. Proximal Newton-type methods for convex optimization. *Advances in Neural Information Processing Systems*, 25:827–835, 2012.

A. S. Lewis. Active sets, nonsmoothness, and sensitivity. *SIAM Journal on Optimization*, 13(3): 702–725, 2002.

J. Liang, J. Fadili, and G. Peyré. A multi-step inertial forward-backward splitting method for non-convex optimization. *Advances in Neural Information Processing Systems*, 29:4035–4043, 2016.

Q. Lin, Z. Lu, and L. Xiao. An accelerated proximal coordinate gradient method. In *NeurIPS*, pages 3059–3067. 2014.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

V. V. Mai and M. Johansson. Anderson acceleration of proximal gradient methods. In *ICML*. 2019.

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

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

M. Massias, S. Vaiter, A. Gramfort, and J. Salmon. Dual extrapolation for sparse generalized linear models. *J. Mach. Learn. Res.*, 2020.

R. Mazumder, J. H. Friedman, and T. Hastie. Sparsenet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495):1125–1138, 2011.

T. Moreau, M. Massias, A. Gramfort, P. Ablin, B. Charlier, P.-A. Bannier, M. Dagréou, T. Dupré la Tour, G. Durif, C. F. Dantas, Q. Klopfenstein, et al. Benchopt: Reproducible, efficient and collaborative optimization benchmarks. *arXiv preprint arXiv:2206.13424*, 2022.

J. B. Muir and Z. Zhan. Seismic wavefield reconstruction using a pre-conditioned wavelet–curvelet compressive sensing approach. *Geophysical Journal International*, 227(1):303–315, 2021.

E. Ndiaye and I. Takeuchi. Continuation path with linear convergence rate. *arXiv preprint arXiv:2112.05104*, 2021.

E. Ndiaye, O. Fercoq, A. Gramfort, and J. Salmon. Gap safe screening rules for sparsity enforcing penalties. *J. Mach. Learn. Res.*, 18(128):1–33, 2017.

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.

A. Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *ICML*, page 78, 2004.

J. Nutini. *Greed is good: greedy optimization methods for large-scale structured problems*. PhD thesis, University of British Columbia, 2018.

J. Nutini, M. W. Schmidt, I. H. Laradji, M. P. Friedlander, and H. A. Koepke. Coordinate descent converges faster with the Gauss-Southwell rule than random selection. In *ICML*, pages 1632–1641, 2015.

W. Ouyang, Y. Peng, Y. Yao, J. Zhang, and B. Deng. Anderson acceleration for nonconvex ADMM based on Douglas-Rachford splitting. In *Computer Graphics Forum*, volume 39, pages 221–239. Wiley Online Library, 2020.

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.

C. Poon and J. Liang. Trajectory of alternating direction method of multipliers and adaptive acceleration. In *NeurIPS*, pages 7357–7365, 2019.

E. Quemener and M. Corvellec. Sidus—the solution for extreme deduplication of an operating system. *Linux Journal*, 2013(235):3, 2013.

A. Rakotomamonjy, G. Gasso, and J. Salmon. Screening rules for lasso with non-convex sparse regularizers. In *ICML*, pages 5341–5350, 2019.

A. Rakotomamonjy, R. Flamary, G. Gasso, and J. Salmon. Provably convergent working set algorithm for non-convex regularized regression. In *AISTATS*, 2022.

D. A. Reidenbach, A. Lal, L. Slim, O. Mosafi, and J. Israeli. Gepsi: A python library to simulate gwas phenotype data. *bioRxiv*, 2021.

P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

M. Rudelson and R. Vershynin. On sparse reconstruction from fourier and gaussian measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 61(8):1025–1045, 2008.

D. Scieur, A. d'Aspremont, and F. Bach. Regularized nonlinear acceleration. In *Advances In Neural Information Processing Systems*, pages 712–720, 2016.

D. Scieur, A. d'Aspremont, and F. Bach. Regularized nonlinear acceleration. *Mathematical Programming*, 179(1):47–83, 2020.

A. Sidi. *Vector extrapolation methods with applications*. SIAM, 2017.

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

E. Soubies, L. Blanc-Féraud, and G. Aubert. A continuous exact ℓ_0 penalty (cel0) for least squares regularized problem. *SIAM Journal on Imaging Sciences*, 8(3):1607–1639, 2015.

D. Strohmeier, A. Gramfort, and J. Haueisen. MEG/EEG source imaging with a non-convex penalty in the time-frequency domain. In *Pattern Recognition in Neuroimaging, 2015 International Workshop on*, 2015.

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. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 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.

R. J. Tibshirani. The lasso problem and uniqueness. *Electronic Journal of statistics*, 7:1456–1490, 2013.

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

S. Vaiter, G. Peyré, and J. Fadili. Low complexity regularization of linear inverse problems. In *Sampling Theory, a Renaissance*, pages 103–153. Springer, 2015.

F. Wei, C. Bao, and Y. Liu. Stochastic Anderson mixing for nonconvex stochastic optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

F. Wen, L. Chu, P. Liu, and R. Qiu. A survey on nonconvex regularization-based sparse and low-rank recovery in signal processing, statistics, and machine learning. *IEEE Access*, 6:69883–69906, 2018.

C.-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942, 2010.

P. Zhao and B. Yu. On model selection consistency of lasso. *J. Mach. Learn. Res.*, 7:2541–2563, 2006.

H. Zou and T. J. Hastie. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67(2):301–320, 2005.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See limitations paragraph
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Propositions 10, 13 and 14.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] In particular we acknowledge the python ecosystem.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]