A framework to efficiently smooth L_1 penalties for linear regression

Georg Hahn, Sharon M. Lutz, Nilanjana Laha, and Christoph Lange

Abstract

Penalized linear regression approaches that include an L_1 term have become an important tool in day-to-day statistical data analysis. One prominent example is the least absolute shrinkage and selection operator (Lasso), though the class of L_1 penalized regression operators also includes the fused and graphical Lasso, the elastic net, etc. Although the L_1 penalty makes their objective function convex, it is not differentiable everywhere, motivating the development of proximal gradient algorithms such as Fista, the current gold standard in the literature. In this work, we take a different approach based on smoothing. The methodological contribution of our article is threefold: (1) We introduce a unified framework to compute closed-form smooth surrogates of a whole class of L_1 penalized regression problems using Nesterov smoothing. The surrogates preserve the convexity of the original (unsmoothed) objective functions, are uniformly close to them, and have closed-form derivatives everywhere for efficient minimization via gradient descent; (2) We prove that the estimates obtained with the smooth surrogates can be made arbitrarily close to the ones of the original (unsmoothed) objective functions, and provide explicitly computable a priori error bounds on the accuracy of our estimates; (3) We propose an iterative algorithm to progressively smooth the L_1 penalty which increases accuracy and is virtually free of tuning parameters. The proposed methodology is applicable to a large class of L_1 penalized regression operators, including all the operators mentioned above. Using simulation studies, we compare our framework to current gold standards such as Fista, glmnet, gLasso, etc. Our simulation results suggest that our proposed smoothing framework provides estimates of equal or higher accuracy than the gold standards while keeping the aforementioned theoretical guarantees and having roughly the same asymptotic runtime scaling.

Keywords: Elastic net; Fista; Fused Lasso, Glmnet; Graphical Lasso; L_1 penalty; Lasso; Nesterov; Penalized linear regression; Smoothing.

1 Introduction

In this communication, we aim to efficiently solve L_1 penalized regression problems. Since first considered around 1800 by Adrien-Marie Legendre and Carl Friedrich Gauss, the least squares approach has traditionally been used to solve linear regression problems. Nevertheless, least squares estimates have two major drawbacks, a lack of robustness and a lack of sparsity in high dimensional settings, i.e. settings characterized by $p \gg n$, where $p \in \mathbb{N}$ denotes the number of covariates and $n \in \mathbb{N}$ is the sample size. Those two limitations have motivated the development of new estimation operators that are more suitable for such settings, e.g. the *least absolute shrinkage and selection operator* (Lasso) of [32], or the *least-angle regression* (LARS) of [13]. Many important extensions of the standard Lasso approach have been proposed, such as the *elastic net* of [36], the *fused Lasso* of [33], or the *graphical Lasso* of [19]. All Lasso approaches involve an L_1 penalty on the parameters which are being estimated to enforce sparseness.

This article focuses on penalized L_1 regression, where we model the relationship of a given design matrix $X \in \mathbb{R}^{n \times p}$ to an observed response $y \in \mathbb{R}^n$ with the help of some unknown parameters $\beta \in \mathbb{R}^p$. In the most general case, we consider any regression operator which can be written as

$$\Phi(\beta) = u(\beta) + v\left(|w_1(\beta)|, \ldots, |w_m(\beta)|\right), \tag{1}$$

where $u : \mathbb{R}^p \to \mathbb{R}$ implicitly depends on X and y, $v : \mathbb{R}^m \to \mathbb{R}$, and $w_j : \mathbb{R}^p \to \mathbb{R}$ for $m \in \mathbb{N}$ and $j \in \{1, \ldots, m\}$. The regression estimate is computed accordingly as $\arg\min_\beta \Phi(\beta)$.

Amongst others, all the previously mentioned estimation operators can be casted in the form of eq. (1). For instance, for a fixed design matrix $X \in \mathbb{R}^{n \times p}$ and response $y \in \mathbb{R}^n$, the objective function of the standard Lasso of [32] is obtained from eq. (1) by choosing $u(\beta) = \frac{1}{n} \|y - X\beta\|_2^2$, as well as $v(z) = \lambda \sum_{i=1}^p z_i$ for $z \in \mathbb{R}^p$, and $w_j(\beta) = \beta_j$ for $j \in \{1, \ldots, p\}$.

If the objective function in eq. (1) is convex, steepest descent (quasi-Newton) methods can be employed to compute $\arg\min_\beta \Phi(\beta)$. However, the non-differentiability of the L_1 penalty may cause a loss of accuracy in conventional gradient(-free) solvers. This will be touched upon in greater detail in our simulations.

We address this issue by providing a smooth surrogate Φ^μ of eq. (1), which depends on a

smoothing parameter $\mu > 0$. The smooth surrogate is derived by applying Nesterov smoothing [29] to the absolute values in eq. (1). Under the condition that u is differentiable and convex, v is differentiable and Lipschitz continuous, v preserves strict convexity, and w_j are differentiable for $j \in \{1, \ldots, p\}$, our framework provides the following guarantees:

1. The surrogate Φ^μ has explicit gradients everywhere.
2. The surrogate Φ^μ is strictly convex.
3. The surrogate Φ^μ is uniformly close to Φ, meaning that $\sup_{\beta \in \mathbb{R}^p} |\Phi(\beta) - \Phi^\mu(\beta)| \leq C_\mu = O(\mu)$ for a constant C_μ which does not depend on β.
4. We prove explicit bounds on $\| \arg\min_\beta \Phi(\beta) - \arg\min_\beta \Phi^\mu(\beta) \|_2$, that is on the distance between the minimizers of the unsmoothed objective function and its smooth surrogate.
5. We prove that $\| \arg\min_\beta \Phi(\beta) - \arg\min_\beta \Phi^\mu(\beta) \| \to 0$ in the supremum norm.

In particular, all these properties hold true for the aforementioned regression operators, thus making it possible to obtain a unified surrogate with the theoretical guarantees enumerated above for a class of common regression operators.

To summarize, the contribution of our article is threefold: (1) We introduce the closed-form surrogate Φ^μ; (2) We prove the theoretical properties of Φ^μ enumerated above; (3) Starting with a high degree of smoothness (i.e., a large value of μ), an iterative algorithm is proposed to progressively smooth the surrogate in order to facilitate minimization. The choice of the smoothing parameter does not have a major effect on the performance of our progressive smoothing algorithm, thus making it virtually free of tuning parameters.

We evaluate our proposed algorithms with respect to both accuracy and runtime in simulation studies, and compare them to the current gold standards in the literature. First, we use as a benchmark the implementation of the Fista algorithm of [3] that is included in the *fasta* R-package on CRAN [10]. Fista combines the basic iterations of the Iterative Shrinkage-Thresholding Algorithm of [11] with a Nesterov acceleration step. The algorithm can be regarded as a proximal gradient version of the method of [28]. Analogously to eq. (1), Fista requires the separate specification of the smooth and non-smooth parts of the objective function and their explicit gradients. For the

application of our smoothing algorithms to the elastic net, we benchmark against the *glmnet* algorithm of [15] that is implemented in the R-package *glmnet* [16]. The glmnet algorithm is a variant of Fista which performs a cyclic update of all coordinates, whereas Fista updates all coordinates per iteration. For the smoothed fused Lasso, we compare with the fused Lasso methodology in the R-package *genlasso* [1]. Finally, we benchmark our smoothed graphical Lasso against the R-package *glasso* [18].

Since the *least-angle regression* (LARS) of [13] or the group Lasso of [35] do not involve an L_1 penalty on the regression estimates, our smoothing framework does not apply to them. We provide a detailed literature review in Section A to highlight previous work, distinguish it from ours, and emphasize the contribution of our article.

This article is structured as follows. We benchmark our proposed methodology in a simulation study in Section 2 against state-of-the-art algorithms. The article concludes with a discussion in Section 3. Section 4 derives the smooth surrogate objective function. We also derive the aforementioned theoretical guarantees of our smoothing framework, and state precise conditions on the quantities involved in eq. (1) which ensure that the theoretical guarantees hold true. Moreover, we demonstrate how all the aforementioned L_1 penalized regression operators fit in our framework, and we refine our approach by proposing a iterative and virtually tuning-free smoothing procedure. A literature review (Section A), a detailed overview of Nesterov smoothing (Section B), all proofs (Section C), as well as further simulations (Section D) are provided in the appendix. The methodology of this article is implemented in the R-package *smoothedLasso*, available on CRAN [21].

Throughout the article, the ith entry of a vector v is denoted as v_i. The entry (i, j) of a matrix X is denoted as X_{ij}, while the ith row of X is denoted as $X_{i,\cdot}$ and the jth column is denoted as $X_{\cdot,j}$. The absolute value, the L_1 norm, as well as the Euclidean norm are written as $|\cdot|$, $\|\cdot\|_1$, and $\|\cdot\|_2$, respectively. The Kronecker delta function is written as δ_{ij}, defined as $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$. The size of a set S is denoted as $|S|$.

2 Results

In this section, we evaluate the performance of the regression estimates computed with our proposed smooth surrogate Φ^μ of eq. (11). We benchmark against the four approaches considered in Section 4.4: Those are the standard Lasso and the elastic net, as well as the fused Lasso and the graphical Lasso. The results for the two latter approaches are reported in Appendix D.

All experiments are performed on simulated data. We simulate a regression model of the type $y = X\beta$ by drawing the rows of the design matrix $X \in \mathbb{R}^{n\times p}$ from a multidimensional normal distribution with the following mean vector and covariance matrix. The entries of the p dimensional mean vector of the multidimensional normal distribution are drawn independently from a uniform distribution in $[0, 1]$. To ensure positive definiteness, the $p \times p$ dimensional covariance matrix of the multidimensional normal distribution is drawn from a Wishart distribution with sample size 1, p degrees of freedom, and scale matrix set to the $p \times p$ dimensional identity matrix.

After generating X, we draw the true parameters β independently from a standard normal distribution. To ensure sparseness, we set all but $nz \in \{0, \ldots, p\}$ out of the p entries to zero. We fix $nz = 0.2p$ in the entire simulation section. Finally, we calculate $y \in \mathbb{R}^n$ as $y = X\beta + \epsilon$ for some noise vector $\epsilon \in \mathbb{R}^n$. The entries of ϵ are generated independently from a Normal distribution with mean zero and some variance σ^2. The smaller we choose σ^2, the easier the recovery of β will be. In the entire simulation section we fix $\sigma^2 = 0.1$.

We implement the three methodologies we developed in this paper:

1. We carry out the minimization of the unsmoothed Φ of eq. (1) using R's function *optim*. Within the *optim* function, we select the quasi-Newton method *BFGS* for which we supply its explicit (though non-smooth) gradient

$$\left(\frac{\partial \Phi}{\partial \beta_i}\right)(\beta) = \left(\frac{\partial u}{\partial \beta_i}\right)(\beta) + \sum_{j=1}^{m} \left[\operatorname{sign}(w_j(\beta)) \cdot \left(\frac{\partial w_j}{\partial \beta_i}\right)(\beta)\right] \cdot (D_j v)(\theta),$$

   where the sign function is defined as $\operatorname{sign}(z) = 1$ for $z > 0$, $\operatorname{sign}(z) = -1$ for $z < 1$, and zero otherwise. This approach will be referred to as *unsmoothed Phi*.

2. We minimize the smooth surrogate Φ^μ of eq. (11) using its explicit gradient given in eq. (12) based on the entropy prox function of Section 4.2. The smoothing parameter of the smooth

surrogate Φ^μ was fixed at $\mu = 0.1$. This approach will be referred to as *smoothed Phi*.

3. We employ the progressive smoothing approach of Section 4.5. As suggested at the end of Section 4.5, we set the target smoothing parameter to $\mu_0 = 2^{-6}$ (roughly the square root of the machine precision) and employ $N = 9$ smoothing steps (thus implying an initial value of the smoothing parameter of $\mu = 2^{-6} \cdot 2^9 = 2^3$).

We benchmark our methods against the gold standards available in the literature for the four problems we consider:

1. As a general method, we use the Fista algorithm to minimize Φ as implemented in the *fasta* R-package [10], available on *The Comprehensive R Archive Network (CRAN)* [30]. The main function of the *fasta* package which implements the Fista algorithm, also called *fasta*, requires the separate specification of the smooth and non-smooth parts of the objective function including their explicit gradients. We follow Example 2 in the *vignette* of the *fasta* R-package in [10] and supply both as specified in eq. (1). Moreover, in contrast to our approach, a variety of tuning parameters need to be selected by the user, e.g. an initial starting value, an initial stepsize, parameters determining the lookback window for non-monotone line search and a shrinkage parameter for the stepsize. The initial stepsize for Fista is set to $\tau = 10$ as in Example 2 of [10]. The lookback window for non-monotone line search and the shrinkage parameter for the stepsize are left at their default values. We employ a uniformly random starting value for $\beta \in \mathbb{R}^p$ (each vector entry is drawn independently from $U[0, 1]$) as done for our own approaches.

2. For the elastic net comparison, we benchmark against the *glmnet* algorithm of [15], available in the R-package *glmnet* [16] on CRAN. The glmnet algorithm is a variant of Fista which performs a cyclic update of all coordinates, whereas Fista updates all coordinates per iteration. To solve an elastic net problem with glmnet, we set its parameter *alpha* to 0.5 as specified in the package vignette [16, page 24].

3. For the fused Lasso, we employ the R-package *genlasso* [1] on CRAN with default parameters. The fused Lasso is implemented in the function *fusedlasso*.

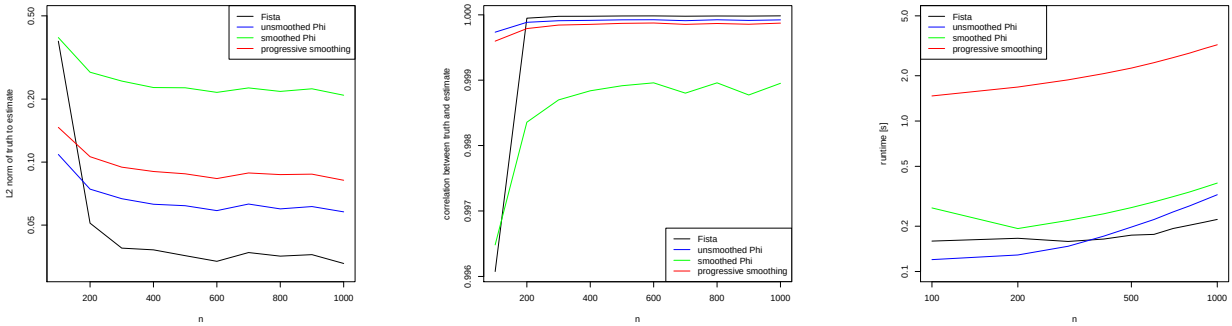

Figure 1: Standard Lasso: L_2 norm (left) and correlation (middle) between simulated truth and parameter estimates as a function of n while $p = 100$. Runtime in seconds (right) has a log scale on both axes. The comparison includes Fista, unsmoothed Φ, smoothed surrogate Φ^μ, and progressive smoothing.

4. For the graphical Lasso, we employ the R-package *glasso* [18] on CRAN with default parameters. The function to run the graphical Lasso is likewise called *glasso*.

Two algorithms are unconsidered in this article for the following reasons. Since the R-package *SIS* accompanying the work of [14] does itself rely on glmnet for computing regression estimates, we omit it in this section. The LARS algorithm of [13] is implemented in the R-package *lars* [23] on CRAN. As remarked in [15], LARS is slower than glmnet or Fista. Additionally, since the implementation of [23] always computes a full LASSO path, it is considerably slower than the aforementioned methods.

All results are averages over 100 repetitions. The regularization parameters of Section 4.4 were fixed at $\lambda = 1$, $\alpha = 0.5$, and $\gamma = 0.5$. We evaluate the accuracy of the aforementioned algorithms using two metrics. First, we report the L_2 norm of the true parameters β that we generate to the estimated ones returned by each method. Second, we report the standard vector correlation between true and estimated parameters. To assess computational efficiency and runtime scaling, we report log-log plots of the empirical runtimes we measure.

2.1 Standard Lasso

We start by applying our smoothing framework to the standard Lasso of [32]. To this end, we employ the choices of u, v, and w_j, $j \in \{1, \ldots, p\}$, for the standard Lasso given in Section 4.4. We generate the regression model as outlined in Section 2, and benchmark our three approaches

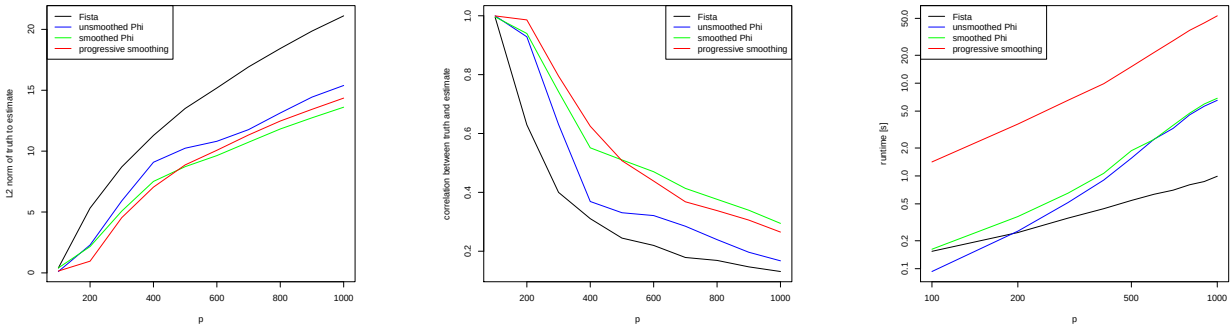

Figure 2: Standard Lasso: L_2 norm (left) and correlation (middle) between simulated truth and parameter estimates as a function of p while $n = 100$. Runtime in seconds (right) has a log scale on both axes. The comparison includes Fista, unsmoothed Φ, smoothed surrogate Φ^μ, and progressive smoothing.

(unsmoothed Φ, smoothed surrogate Φ^μ, and progressive smoothing) against the Fista algorithm. We are interested in accuracy and runtime of all algorithms as a function of n (while keeping $p = 100$ fixed) or p (while keeping $n = 100$ fixed).

Figure 1 shows results for the scaling in n while keeping $p = 100$ fixed. We observe that, as expected, estimation becomes easier as n becomes larger. Fista yields the best estimates when measuring the error in the L_2 norm, followed by the unsmoothed Lasso objective (unsmoothed Phi) and the progressive smoothing algorithm. Minimizing the smoothed Lasso objective (smoothed Phi) does not seem to have an advantage in this scenario. The correlation of the truth to the estimate seems equally high for almost all approaches. Moreover, we observe that all approaches have roughly the same runtime scaling in n. Since the progressive smoothing approach works by repeatedly minimizing the smoothed surrogate, its runtime is roughly a constant factor higher than the one for minimizing the smoothed surrogate Φ^μ.

The more important case is the scenario in which $p \gg n$. This scenario, depicted in Figure 2, shows a different picture. Fista does not seem to approximate the truth well when measuring the error with the L_2 norm, whereas minimizing the smoothed surrogate (smoothed Phi) and progressive smoothing yield much more accurate results. The picture is confirmed when measuring accuracy with the correlation between truth and estimate. Here, using Fista and minimizing the unsmoothed objective function results in estimates with a considerably lower correlation to the true parameter vector than using both our smoothing approaches (smoothed Phi or progressive smoothing). As can

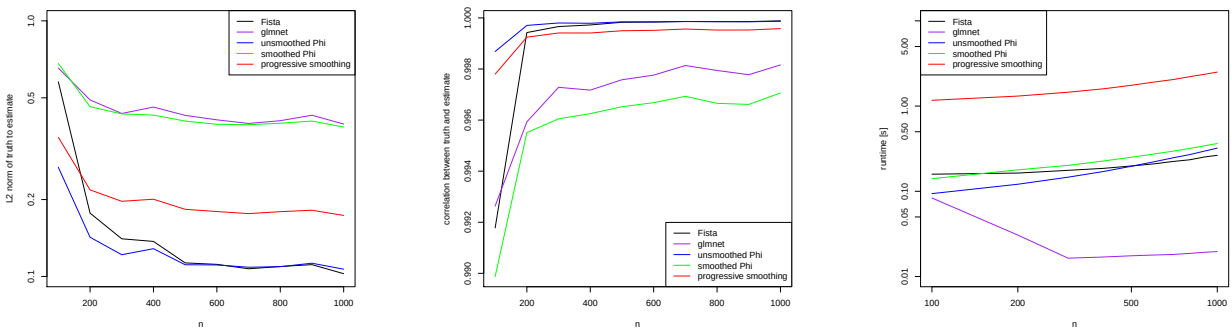

Figure 3: Elastic net: L_2 norm (left) and correlation (middle) between simulated truth and parameter estimates as a function of n while $p = 100$. Runtime in seconds (right) has a log scale on both axes. The comparison includes Fista, glmnet, unsmoothed Φ, smoothed surrogate Φ^μ, and progressive smoothing.

be seen from the left and middle panels in Figure 2, minimizing the unsmoothed Lasso objective Φ is more prone to instabilities due to its non-differentiability. This is to be expected and motivates our smoothing approach. Not surprisingly, while keeping the data size n fixed, the estimation becomes more challenging as p increases. The runtime scaling of our three methods is again similar, though Fista seems to have a slightly more favorable runtime scaling in p.

2.2 Elastic net

We repeat the experiments of Section 2.1 for the elastic net of [36] using the R-package *glmnet* [15]. For this, we employ the choices of u, v, and w_j, $j \in \{1, \ldots, p\}$, for the elastic net given in Section 4.4. We generate models $y = X\beta$ as described in Section 2, and evaluate Fista, glmnet, unsmoothed Φ, smoothed surrogate Φ^μ, and progressive smoothing with respect to their accuracy and scaling behavior in n and p.

Figure 3 shows scaling results for n while keeping $p = 100$ fixed. We observe that glmnet and the simple smoothing approach (smoothed Phi) do not seem to perform well in this experiment, while Fista as well as the minimization of the unsmoothed objective Φ and progressive smoothing perform notably better. When looking at the correlation between truth and estimate, all methods with the exception of glmnet and the smoothed surrogate Φ^μ (smoothed Phi) again perform similarly. The runtime of all methods does not seem very dependent on n.

Analogously, Figure 4 presents scaling results for the more interesting case that $p \gg n$ while

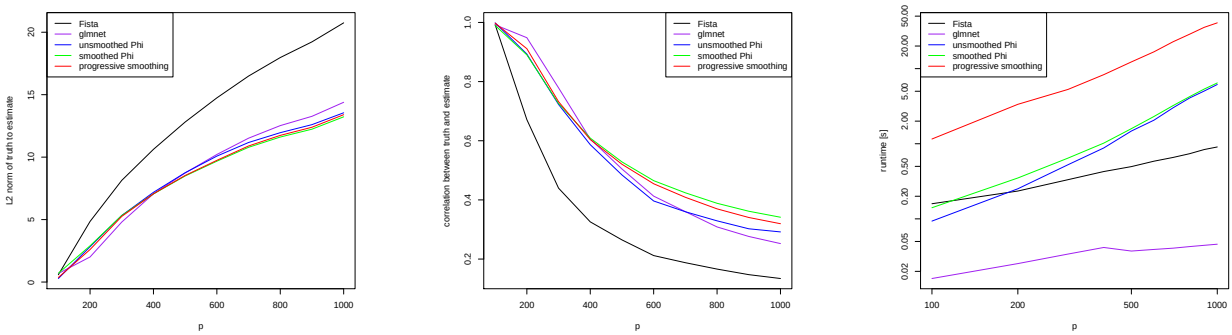

Figure 4: Elastic net: L_2 norm (left) and correlation (middle) between simulated truth and parameter estimates as a function of p while $n = 100$. Runtime in seconds (right) has a log scale on both axes. The comparison includes Fista, glmnet, unsmoothed Φ, smoothed surrogate Φ^μ, and progressive smoothing.

keeping $n = 100$ fixed. These results confirm the observations made for the case of the standard Lasso in Figure 2. As expected, the estimation of the parameters becomes more challenging for all methods as p increases while n stays fixed.

Fista seems to yield estimates of poorest quality when assessing accuracy with the L_2 distance between truth and estimate, while the other approaches perform more favorably. Regarding the correlation between truth and estimate, Fista again performs suboptimally, while glmnet and the minimization of the unsmoothed Φ (unsmoothed Phi) perform better. Using our smoothed surrogate (smoothed Phi) and the progressive smoothing algorithm perform best in that they retain the largest correlation between their estimates to the true parameters. As expected, the runtime scaling in p of Fista and glmnet is roughly similar, and the asymptotic runtimes of both seem to be lower than for our methods. Minimizing the unsmoothed objective function and the smooth surrogate seems to have an almost identical asymptotic runtime, while progressive smoothing is a constant factor slower as expected.

Further experiments for the fused and graphical Lasso can be found in Section D and broadly confirm the results of this section.

2.3 Verification of theoretical bounds

Finally, we verify the theoretical bounds derived in Proposition 5 for the case of the standard Lasso. In this experiment, we therefore again employ the choices of u, v, and w_j, $j \in \{1, \ldots, p\}$, for the

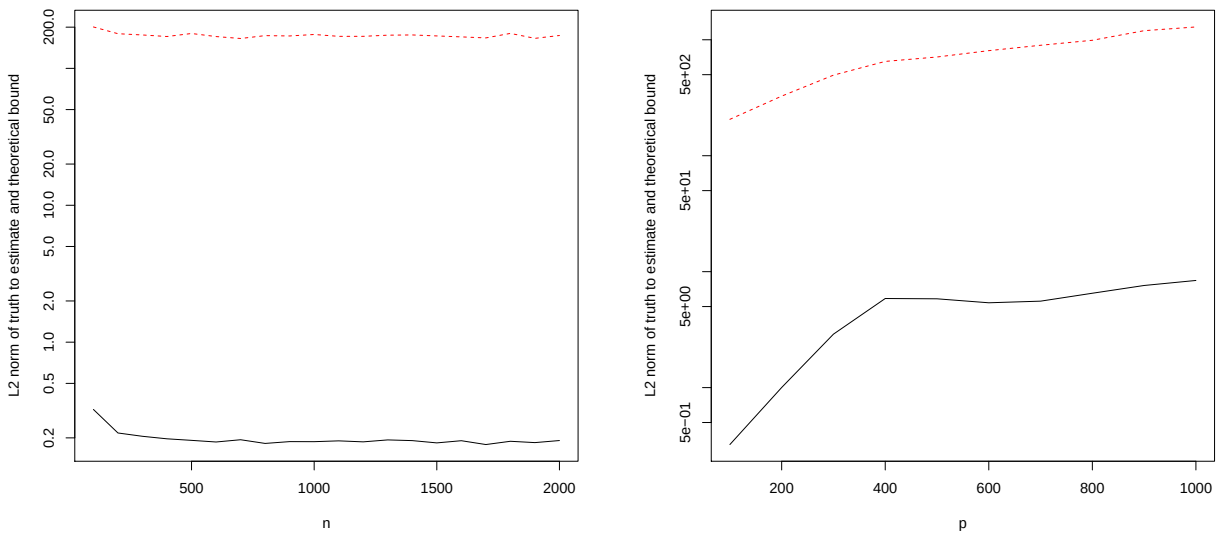

Figure 5: True L_2 norm between minimizers of unsmoothed and smoothed Lasso (left hand side of eq. (13)) in black, and theoretical upper bound (right hand side of eq. (13)) in red. Scaling in n while $p = 100$ is fixed (left) and scaling in p while $n = 100$ is fixed (right). Log scale on the y-axis.

standard Lasso given in Section 4.4.

First, we compute regression estimates with the unsmoothed and smoothed Lasso objective functions as done in Section 2.1. We employ a regularization parameter of $\lambda = 1$, and a smoothing parameter of $\mu = 1$. The generation of the model $y = X\beta$ was done as described in Section 2.

After performing the minimization we record $\|x_1 - x_2\|_2$, the left hand side of eq. (13). This quantity is assumed to be unknown as we aim to avoid the computation of the minimizer x_2 of the unsmoothed objective and solely focus on the minimizer x_1 of the smoothed surrogate instead.

Moreover, we compute the right hand side of eq. (13), precisely $\sup_{y_1 \in S_\delta(x_1)} \|\nabla F_1(y_1)\|_2^{-1} \cdot C_\delta (\delta L_\delta + 2\epsilon) + C_\delta \delta$. In the case of the standard Lasso considered here, F_1 is the differentiable and strictly convex function Φ^μ, that is the smooth surrogate of the standard Lasso (see the discussion at the end of Section 4.3). The parameter δ is chosen by the user: for more stable results, we compute the bound of eq. (13) on the grid $\delta \in \{0.1, 0.2, \ldots, 1\}$ and take the minimal value. The value of ϵ is given by eq. (8). Taking the supremum over the parameter y_1 on the sphere $S_\delta(x_1)$ is performed in a Monte Carlo fashion by computing the bound of eq. (13) for random points generated uniformly on $S_\delta(x_1)$ and taking the maximum. Last, C_δ can be computed explicitly using a scalar product between two known quantities as shown in the proof of Proposition 5, and

L_δ is taken to be the local Lipschitz constant of Φ^μ at its minimizer x_1. This makes the right hand side of eq. (13) explicitly computable without knowledge of the minimizer x_2 of the unsmoothed (original) Lasso objective.

Results for the left and right hand sides of eq. (13) are shown in Figure 5, once for the scaling in n while $p = 100$ is kept fixed, and once for the scaling in p while $n = 100$ is kept fixed. As expected, the estimation becomes easier as n increases for a fixed p, and more challenging as p increases for a fixed n.

Overall, we observe that eq. (13) indeed provides a valid, explicit upper bound on the error between the unsmoothed and smoothed minimizers for each fixed choice of μ. In the case of the scaling in n, the bound is very conservative, but in the more challenging scenario of the scaling in p, the bound seems to accurately follow the error we make by using the smoothed surrogate. Note that by Propositions 3 and 5, this error will eventually go to zero as the smoothing parameter $\mu \to 0$.

3 Discussion

In this paper, we propose a smoothing framework for a general class of L_1 penalized regression operators that includes most common approaches, e.g. the Lasso, elastic net, fused and graphical Lasso, etc. The framework is based on a closed-form smooth surrogate and its closed-form gradient. Since the aforementioned regression operators are convex but non-smooth, due to the non-differentiability of their L_1 norm, minimizing the smooth surrogates instead allows for a fast and efficient computation of regression estimates.

Most importantly, we prove a series of theoretical results which apply to the entire class of regression operators belonging to our framework. Under regularity conditions, we show that the smooth surrogate is both strictly convex and uniformly close (in supremum norm) to the original (unsmoothed) objective function. This has the important implication that the regression estimates obtained by minimizing the smooth surrogate can be made arbitrarily close to the ones of the original (unsmoothed) objective function. Additionally, we provide explicitly computable error bounds on the accuracy of the estimates obtained with the smooth surrogate. Since we can efficiently carry out the minimization of the smooth surrogate, our approach yields easily computable regression

estimates that are both guaranteed to be close to the estimates of the original objective function, and allow for a priori error estimation. Furthermore, we develop a progressive smoothing algorithm for iterative smoothing. This approach is virtually free of tuning parameters.

Our simulation studies support our theoretical conclusions that minimizing the proposed smooth surrogate yields estimates of equal or better quality than many of the gold standard approaches available in the literature (the Fista, glmnet, gLasso algorithms, etc.). At the same time, our approach has roughly the same runtime scaling as the established approaches we include in our comparisons while keeping all aforementioned theoretical guarantees.

4 Methods

This section states the precise conditions and results of our smoothing framework for regression operators with an objective function of the form of eq. (1). The results of this section will be used in Section 4.5 to develop an adaptive procedure which iteratively smoothes the objective function and yields very stable regression estimates.

The smoothing of eq. (1) will be achieved with the help of Nesterov smoothing, which is briefly summarized in Section 4.1. In Section 4.2 we smooth the L_1 contribution in eq. (1), leading to the smooth surrogate objective function. Section 4.3 states the theoretical guarantees provided for the smooth surrogate, and the precise conditions required for the guarantees to hold true. Section 4.4 gives examples of how common regression operators fit into our framework. Finally, Section 4.5 refines our approach by proposing a iterative and virtually tuning-free smoothing procedure.

4.1 Summary of Nesterov smoothing

In [29], the author considers the smoothing of a piecewise affine and convex function $f : \mathbb{R}^q \to \mathbb{R}$, where $q \in \mathbb{N}$. Since f is piecewise affine, it can be represented as $f(z) = \max_{i=1,\dots,k} \left(A[z,1]^\top\right)_i$, where $k \in \mathbb{N}$ is the number of linear pieces (components). Here, the rows of the matrix $A \in \mathbb{R}^{k\times(q+1)}$ contain the coefficients of each linear piece, with column $q+1$ containing all constant coefficients. Moreover, $[z,1] \in \mathbb{R}^{q+1}$ denotes the vector obtained by concatenating $z \in \mathbb{R}^q$ and the scalar 1.

In [29], an approximation f^μ of f is derived which is both smooth and uniformly close to f. The approximation f^μ depends on a so-called proximity (or prox) function, see Section B.1, which is

parameterized by a smoothing parameter $\mu > 0$ controlling the degree of smoothness. We consider two choices of the prox function, presented in Section B.2. The so-called entropy prox function yields a closed-form expression of the smooth approximation f_e^μ given by

$$f_e^\mu(z) = \mu \log \left(\frac{1}{k} \sum_{i=1}^{k} e^{\frac{\left(A[z,1]^\top\right)_i}{\mu}} \right), \tag{2}$$

which satisfies the uniform approximation bound

$$\sup_{z \in \mathbb{R}^q} |f(z) - f_e^\mu(z)| \leq \mu \log(k). \tag{3}$$

Another choice, introduced in Section B.2, is the squared error prox function, given by $\rho_s(\tau) = \frac{1}{2} \sum_{i=1}^{k} \left(\tau_i - \frac{1}{k}\right)^2$ for $\tau \in \mathbb{R}^k$. Let the vector $c(z) = (c_1(z), \ldots, c_k(z)) \in \mathbb{R}^k$ be defined component-wise by $c_i(z) = 1/\mu \cdot \left(A[z,1]^\top\right)_i - 1/k$ for $i \in \{1, \ldots, k\}$. Let $\hat{c}(z) \in \mathbb{R}^k$ be the Michelot projection [26] of $c(z)$ onto the k-dimensional unit simplex Q_k (see Section B.2.2). In [29] it is shown that

$$f_s^\mu(z) = \langle \hat{c}(z), A[z,1]^\top \rangle - \mu \rho_s(\hat{c}(z)) \tag{4}$$

is a smooth approximation of f satisfying the uniform bound

$$\sup_{z \in \mathbb{R}^q} |f(z) - f_s^\mu(z)| \leq \mu \left(1 - \frac{1}{k}\right). \tag{5}$$

Further details on the above results can be found in Section B.

4.2 The smooth surrogate

We employ the results of Section 4.1 to smooth the absolute value in eq. (1). In the following subsections, we always consider $k = 2$, $z \in \mathbb{R}$, and the specific choice of the matrix $A \in \mathbb{R}^{2\times 2}$ given by

$$A = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}. \tag{6}$$

The specific choice of A allows us to express the (one dimensional) absolute value as a piecewise affine and convex function $f(z) = \max\{-z, z\} = \max_{i=1,2} \left(A[z,1]^\top\right)_i$.

Simplifying eq. (2) with the specific choice of A of eq. (6) results in the entropy prox approximation of the absolute value and its explicit derivative, given by

$$\begin{aligned} f_e^\mu(z) &= \mu \log\left(\frac{1}{2}e^{-z/\mu} + \frac{1}{2}e^{z/\mu}\right), \\ \frac{\partial}{\partial z} f_e^\mu(z) &= \frac{-e^{-z/\mu} + e^{z/\mu}}{e^{-z/\mu} + e^{z/\mu}} =: g_e^\mu(z). \end{aligned} \tag{7}$$

The approximation bound of eq. (3) simplifies to

$$\sup_{z\in\mathbb{R}} |f(z) - f_e^\mu(z)| \leq \mu \log(2) =: C_e^\mu. \tag{8}$$

In a similar fashion, the squared error prox smoothing of eq. (4) with A as in eq. (6) leads to the following approximation of the absolute value and its explicit derivative:

$$\begin{aligned} f_s^\mu(z) &= \langle \hat{c}(z), [-z, z]\rangle - \mu\rho_s(\hat{c}(z)), \\ \frac{\partial}{\partial z} f_s^\mu(z) &= \langle \hat{c}(z), [-1, 1]\rangle =: g_s^\mu(z), \end{aligned} \tag{9}$$

where $\hat{c}(z) \in \mathbb{R}^2$ denotes the Michelot projection of $c(z) = 1/\mu \cdot [-z, z] - 1/2$ onto the two-dimensional unit simplex. A derivation of the derivative of f_s^μ can be found in [20, Lemma 4]. The approximation bound of eq. (5) simplifies to

$$\sup_{z\in\mathbb{R}} |f(z) - f_s^\mu(z)| \leq \frac{1}{2}\mu =: C_s^\mu. \tag{10}$$

With the smooth approximation of the absolute value in place, we define the surrogate of Φ as

$$\Phi^\mu(\beta) = u(\beta) + v\left(f^\mu(w_1(\beta)), \ldots, f^\mu(w_m(\beta))\right), \tag{11}$$

where in the remainder of Section 4, f^μ always denotes either the entropy prox smoothed absolute value f_e^μ, or the squared error prox smoothed absolute value f_s^μ.

4.3 Theoretical guarantees

Depending on the conditions imposed upon u, v, and w_j for $j \in \{1, \ldots, m\}$ in eq. (1), the surrogate Φ^μ of eq. (11) has a variety of properties.

Condition 1. *The functions u, v and w_j of eq. (1) are differentiable everywhere for all $j \in \{1, \ldots, m\}$.*

Proposition 1. *Under Condition 1, the surrogate Φ^μ is differentiable everywhere with explicit gradient*

$$\left(\frac{\partial \Phi^\mu}{\partial \beta_i}\right)(\beta) = \left(\frac{\partial u}{\partial \beta_i}\right)(\beta) + \sum_{j=1}^{m} \left[g^\mu(w_j(\beta)) \cdot \left(\frac{\partial w_j}{\partial \beta_i}\right)(\beta) \right] \cdot (D_j v)(\theta), \tag{12}$$

where g^μ is the derivative of f^μ as given in eq. (7) and eq. (9), $D_j v$ is the derivative of v with respect to its jth argument, and $\theta = (f^\mu(w_1(\beta)), \ldots, f^\mu(w_m(\beta))) \in \mathbb{R}^m$.

The proof of Proposition 1 follows from a direct calculation. Subject to the following commonly satisfied condition, the surrogate Φ^μ is strictly convex.

Condition 2. *In eq. (1), the function u is convex and v preserves strict convexity of its input arguments.*

Proposition 2. *Under Condition 2, Φ^μ is strictly convex.*

The proof of Proposition 2 follows from the fact that according to Proposition 6 in Section C, f_e^μ and f_s^μ are strictly convex. Thus if u is convex and v preserves strict convexity, Φ^μ is itself strictly convex as the sum of a convex function and a strictly convex function. Alternatively, if v only preserves convexity, Φ^μ will still be convex (given u is convex). However, the results of Propositions 4 and 5 below do not hold true any more as they require strict convexity.

Next, we consider an approximation bound of Φ^μ to Φ. Under the following condition, the bounds on f_e^μ in eq. (8) and f_s^μ in eq. (10) translate to a guaranteed error bound on the surrogate Φ^μ.

Condition 3. *The function v of eq. (1) is Lipschitz continuous with Lipschitz constant L_v.*

Condition 3 can easily be verified. Given Condition 1 is already satisfied, and v is thus differentiable, then a bounded derivative (in L_1 norm) of v will make v Lipschitz continuous with $L_v := \sup_{z\in\mathbb{R}^m} \|\nabla v(z)\|_1 < \infty$.

Condition 3 is used in the following proposition to establish error bounds for Φ^μ with respect to Φ.

Proposition 3. *Let C^μ denote either C_e^μ of eq.* (8) *or C_s^μ of eq.* (10) *depending on whether f_e^μ or f_s^μ are employed in Φ^μ. Under Condition 3,*

$$\sup_{\beta\in\mathbb{R}^p} |\Phi^\mu(\beta) - \Phi(\beta)| \leq mL_vC^\mu = O(\mu).$$

In Proposition 3, the dimension m and the Lipschitz constant L_v are fixed for a particular estimation problem, thus allowing us to make the approximation error arbitrarily small as the smoothing parameter $\mu \to 0$.

Most importantly, we are not only interested in the error bound between the objective function Φ and its surrogate Φ^μ, but moreover in how the minimizer $\arg\min_{\beta\in\mathbb{R}^p} \Phi^\mu(\beta)$ compares to the one obtained had we computed $\arg\min_{\beta\in\mathbb{R}^p} \Phi(\beta)$.

Following [31, Lemma 2.9], the following proposition shows that continuity, strict convexity and a vanishing approximation error of the surrogate implies that the global minimizers of Φ and Φ^μ converge to each other in the supremum norm metric.

Proposition 4. *Let $F_1 : \mathbb{R}^s \to \mathbb{R}$ be continuous and strictly convex for $s \in \mathbb{N}$. Then $x_1 = \arg\min_{x\in\mathbb{R}^s} F_1(x)$ is continuous at F_1 with respect to the supremum norm.*

Under Conditions 1–3, Φ^μ satisfies the requirements of Proposition 4. Using the result of Proposition 3, $\sup_{\beta\in\mathbb{R}^p} |\Phi^\mu(\beta) - \Phi(\beta)| \to 0$ for $\mu \to 0$ implies that the minimizers of Φ^μ and Φ converge to each other in the supremum norm. This result is stronger than the one of [3, Theorem 4.4], who prove that the FISTA method finds a minimizer which is of similar quality than the true minimizer.

Although Proposition 4 shows convergence, it does not give an explicit error bound on the distance between the two minimizers. This is done in the next result.

Proposition 5. *Let $s \in \mathbb{N}$ and $\epsilon > 0$. Let $F_1 : \mathbb{R}^s \to \mathbb{R}$ be differentiable and strictly convex. Let $F_2 : \mathbb{R}^s \to \mathbb{R}$ be such that $\sup_{x\in\mathbb{R}^s} |F_1(x) - F_2(x)| \leq \epsilon$. Let $x_i = \arg\min_{x\in\mathbb{R}^s} F_i(x)$ be the two minimizers for $i \in \{1, 2\}$. Then for any $\delta > 0$, there exist two constants $C_\delta > 0$ and $L_\delta > 0$ independent of x_2 such that*

$$\|x_1 - x_2\|_2 \leq \sup_{y_1\in S_\delta(x_1)} \|\nabla F_1(y_1)\|_2^{-1} \cdot C_\delta \left(\delta L_\delta + 2\epsilon\right) + C_\delta \delta, \tag{13}$$

where $S_\delta(x) = \{y : \|x - y\|_2 = \delta\} \subset \mathbb{R}^s$ denotes the sphere of radius δ around $x \in \mathbb{R}^s$ with respect to the Euclidean norm.

Note that Proposition 5 does not generalize to non strictly convex functions. Under Conditions 1–3, applying Proposition 5 with F_1 taken to be the differentiable and strictly convex Φ^μ and F_2 taken to be Φ immediately gives an explicit bound on the distance between the two minimizers $\arg\min_{\beta\in\mathbb{R}^p} \Phi^\mu(\beta)$ and $\arg\min_{\beta\in\mathbb{R}^p} \Phi(\beta)$. This bound can be computed using the minimizer of Φ^μ only. We demonstrate the quality of the bound in the simulations of Section 2.3.

4.4 Applications to popular L1 penalized regression operators

Let $X \in \mathbb{R}^{n\times p}$, $y \in \mathbb{R}^n$, and $\beta \in \mathbb{R}^p$ as introduced in Section 1. Additionally, let $\lambda, \alpha, \gamma \geq 0$. A variety of popular regression operators can be written in the form of eq. (1) and satisfy the conditions of Section 4.3.

1. The least absolute shrinkage and selection operator (Lasso) of [32] is defined as

$$\arg\min_{\beta\in\mathbb{R}^p} \frac{1}{n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 .$$

   It can be expressed in the form of eq. (1) with $u(\beta) = \frac{1}{n} \|y - X\beta\|_2^2$, as well as $v(z) = \lambda \sum_{i=1}^p z_i$ for $z \in \mathbb{R}^p$, and $w_j(\beta) = \beta_j$ for $j \in \{1, \ldots, p\}$. Here, $\partial u/\partial\beta_i = -\frac{2}{n}\langle y - X\beta, X_{\cdot,i}\rangle$, $\partial w_j/\partial\beta_i = \delta_{ij}$, and $D_j v = \lambda$ for $i, j \in \{1, \ldots, p\}$.

2. The elastic net of [36] augments the Lasso with an L_2 penalty on the parameters β, resulting in

$$\arg\min_{\beta\in\mathbb{R}^p} \frac{1}{2n} \|y - X\beta\|_2^2 + \alpha \|\beta\|_1 + \frac{1}{2}(1 - \alpha) \|\beta\|_2^2 ,$$

where the parameterization is taken from [17], which is also used in the specification of the R-package *glmnet* [16]. It can be expressed in the form of eq. (1) with $u(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \frac{1}{2}(1-\alpha) \|\beta\|_2^2$, as well as $v(z) = \alpha \sum_{i=1}^p z_i$ for $z \in \mathbb{R}^p$ and w_j as in the previous case of the standard Lasso, where $j \in \{1, \ldots, p\}$. The derivative of u is $\partial u / \partial \beta_i = -\frac{1}{n} \langle y - X\beta, X_{\cdot,i} \rangle + (1-\alpha)\beta_i$.

3. The fused Lasso of [33] is defined as

$$\arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 + \lambda\gamma \sum_{(i,j)\in E}^{p} |\beta_i - \beta_j| ,$$

where E is the edge set of some underlying graph, see [1]. Let $E = \{e_1, \ldots, e_{|E|})$ in some arbitrary ordering, and $e_i = (e_i^1, e_i^2)$ for $i \in \{1, \ldots, |E|\}$. The fused Lasso can be casted in the form of eq. (1) with $u(\beta)$ defined similarly to the standard Lasso, as well as $v(z) = \lambda \sum_{i=1}^p z_i + \lambda\gamma \sum_{i=p+1}^m z_i$ for $m = p + |E|$ and $z \in \mathbb{R}^m$. Accordingly, $w_j(\beta) = \beta_j$ for $j \in \{1, \ldots, p\}$ and $w_j(\beta) = \beta_{e^1_{j-p}} - \beta_{e^2_{j-p}}$ for $j \in \{p+1, \ldots, m\}$. Here, $D_j v = \lambda$ for $j \in \{1, \ldots, p\}$ and $D_j v = \lambda\gamma$ for $j \in \{p+1, \ldots, m\}$. Moreover, $\partial w_j / \partial \beta_i = \delta_{ij}$ for $j \in \{1, \ldots, p\}$ and $\partial w_j / \partial \beta_i = \delta_{e^1_{j-p},i} - \delta_{e^2_{j-p},i}$ for $j \in \{p+1, \ldots, m\}$.

4. The graphical Lasso of [19, 18] considers observations $X_1, \ldots, X_n \sim N(0, \Sigma)$ from a multivariate Gaussian distribution and estimates the precision matrix $\Theta = \Sigma^{-1} \in \mathbb{R}^{p \times p}$ as

$$\arg\min_{\Theta \succ 0} tr(S\Theta) - \log\det(\Theta) + \lambda \sum_{i \neq j} |\Theta_{ij}| ,$$

where S is the sample covariance matrix, tr denotes the trace of a matrix, and the arg min is taken over all positive definite matrices Θ. Define $\beta = (\Theta_{1,1}, \ldots, \Theta_{1,p}, \Theta_{2,1}, \ldots) \in \mathbb{R}^{p^2}$ as the vector containing all parameters of Θ stacked together by row. As before, we define the differentiable part of eq. (1) to include the trace and log terms, thus $u(\Theta) = tr(S\Theta) - \log\det(\Theta)$ with derivative $\nabla u = S^\top - \Theta^{-1}$, and $v : \mathbb{R}^m \to \mathbb{R}$ and $w_i : \mathbb{R}^{p^2} \to \mathbb{R}$ with $m = p^2$ as in the case of the standard Lasso.

In all above cases, the functions u, v, and w_i are differentiable for $i \in \{1, \ldots, m\}$, thus satisfying Condition 1. Their derivatives are given explicitly for convenience.

Algorithm 1: Progressive smoothing

input: Φ^μ, $\mu_0 > 0$, $N \in \mathbb{N}$;
Set $\hat{\beta}_{N+1} \in \mathbb{R}^p$ randomly;
for $i = N, \ldots, 0$ **do**
  $\mu \leftarrow 2^i \mu_0$;
  $\hat{\beta}_i \leftarrow \arg\min_{\beta \in \mathbb{R}^p} \Phi^\mu(\beta)$ with starting value $\hat{\beta}_{i+1}$;
end
return $\hat{\beta}_0$;

Moreover, all functions u above, including the one of the graphical Lasso, are convex as they can be written as the linear combinations of (Euclidean) norms. Importantly, the functions v are always linear combinations of their inputs with positive coefficients and thus preserve strict convexity. Therefore, Condition 2 is satisfied.

Finally, the functions v are all differentiable and have a bounded gradient (in L_1 norm), thus making them Lipschitz continuous as required in Condition 3. For instance, $v(z) = \lambda \sum_{i=1}^p z_i$ for $z \in \mathbb{R}^p$ in case of the standard Lasso, thus we obtain $\nabla v = \lambda[1, \ldots, 1]$ and $L_v := \sup_{z \in \mathbb{R}^p} \|\nabla v(z)\|_1 = \lambda p$ using the fact that λ is non-negative.

4.5 Progressive smoothing algorithm

This section proposes an adaptive smoothing technique for the surrogate Φ^μ. Simulations in Section 2 show that the progressive smoothing algorithm yields stable estimators for linear regression.

Instead of solving the smoothed problem $\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^p} \Phi^\mu(\beta)$ directly for some $\mu > 0$, we employ a progressive smoothing procedure along the following rationale: We start with a large value of the smoothing parameter μ to facilitate the minimization. After computing $\hat{\beta}$, we decrease the smoothing parameter and repeat the minimization using the previously found minimizer as the new starting value. This approach is based on the heuristic idea that as μ decreases and the smoothed surrogate Φ^μ approaches Φ, the found minimizers in each iteration remain close to each other and converge to the minimizer of Φ.

Algorithm 1 formalizes our approach. The input of the algorithm is the function Φ^μ, a target smoothing parameter $\mu_0 > 0$, and a number of smoothing steps $N \in \mathbb{N}$. After initializing a random starting value $\hat{\beta}_{N+1} \in \mathbb{R}^p$ for the first minimization, we gradually decrease the degree of smoothness according to $\mu = 2^i \mu_0$ from $i = N$ to the target level μ_0 at $i = 0$. In each iteration i, we compute a

new estimate $\hat{\beta}_i$ using the current smoothing level μ and the previous estimate $\hat{\beta}_{i+1}$ as the starting value. The output of the algorithm is $\hat{\beta}_0$, the parameter estimate corresponding to the target smoothing degree μ_0.

Importantly, the advantage of Algorithm 1 consists in the fact that the precise specification of the smoothing parameter does not play a major role. It suffices to start with any sufficiently large value (that is, $2^N \mu_0 \gg 1$) and to end the iteration with any sufficiently small value μ_0, for instance of the order of the machine precision or of the square root of the machine precision. This effectively makes Algorithm 1 free of tuning parameters.

Funding

The initial methodology work for this paper was funded by Cure Alzheimer's Fund; Funding for this research was provided through the National Heart, Lung, and Blood Institute [U01HL089856, U01HL089897, P01HL120839, P01HL132825, 2U01HG008685] and National Human Genome Research Institute [R01HG008976].

A Literature review

Since the seminal publication of the Lasso [32], numerous approaches have focused on (smoothing) approaches to facilitate the minimization of the Lasso objective function, or other related objective functions. Nevertheless, the following publications differ from our work in that they do not consider a unified framework to smooth L_1 regularized regression operators, and if they provide a tailored solution to a particular regression operator, the bounds provided differ from the ones we establish on the accuracy of the unsmoothed objective and surrogate. Most importantly, none of the publications we are aware of quantifies the distance between the minimizer of the unsmoothed objective and the minimizer of the surrogate a priori (as we do with our explicitly computable bound), and no progressive smoothing procedure yielding stable regression estimates is derived.

In [14], the authors consider smoothing approaches for the Lasso L_1 penalty, the SCAD (Smoothly Clipped Absolute Deviation) penalty, and hard thresholding penalties. However, their smoothing approaches are not based on the framework of [29]. Instead, the authors employ a quadratic approximation at the singularity of the penalties to achieve a smoothing effect, and they propose a

one-step shooting algorithm for minimization. However, their main focus is on root-n consistency results of the resulting estimators and asymptotic normality results for the SCAD penalty, results which the authors state do not all apply to the Lasso.

Some smoothing approaches [6, 7, 2] build upon the first-order accelerated gradient descent algorithm of [29]. Those variants of Nesterov's algorithm are iterative methods which are unrelated to our adaptive smoothing procedure. A detailed overview of several variants of the first-order accelerated gradient descent algorithm can be found in [5].

The publication of [4] can be regarded as an extension of the work of [29]. The authors consider a more general smoothing framework which, as a special case, includes the same smoothing we establish for the absolute value in the L_1 penalty of the Lasso. However, in contrast to our work, the authors do not consider a smooth surrogate of a general form that applies to a wide range of regression operators, and the theoretical guarantees on the surrogate that we derive are unconsidered.

In [22], the authors likewise smooth the absolute value in the L_1 penalty of the Lasso using Nesterov's technique, and they state the same bound on the difference between the unsmoothed and smoothed Lasso objective functions taken from Nesterov's results. However, their results focus on the Lasso only, the general surrogate we provide is unconsidered, and no explicit a priori bounds on the accuracy of the obtained minimizers are given. Importantly, the work of [22] deviates from our work in that the authors enforce that the smoothed Lasso penalty passes through zero, leading the focus of their article to be on another smoothed Lasso approach which is based on the error function of the normal distribution.

Further work available in the literature employs Nesterov's smoothing techniques for a variety of specialized Lasso objective functions. For instance, the publication of [8] considers the group Lasso and employs Nesterov's formalism to smooth the Lasso penalty using the squared error proximity function which we also consider. Nevertheless, the authors focus on adapting Nesterov's first-order accelerated gradient descent algorithm in order to compute the Lasso regression estimate, whereas we focus on deriving theoretical bounds on the smooth surrogate and our progressive smoothing algorithm. In our work, we do not propose a modification of Nesterov's first-order accelerated gradient descent algorithm. The group Lasso is also considered in [9], who separate out the simple nonsmooth L_1 penalty from the more complex structure-inducing penalties, and only smooth the

latter. This leaves the L_1 norm on the parameters unchanged, thus still enforcing individual feature level sparsity. In contrast to the work of [9], we always smooth all L_1 penalties which are present in the objective function.

The joint Lasso is considered in [12], who state an iterative minimization procedure which smoothes the Lasso penalty using Nesterov's techniques. The authors state closed form derivatives for minimization (as we do as well in the form of the more general derivatives of the smooth surrogate), but no other theoretical results are given.

One variant of the original Lasso which has recently gained attention is the concomitant Lasso. The concomitant Lasso augments the original Lasso with a term $\sigma/2$ for which a second regularization parameter σ is introduced [27]. The parameter σ is meant to be decreased to zero. Smoothing the concomitant Lasso has the advantage that Nesterov's techniques do not need to be applied to the L_1 penalty. Instead, the smooth concomitant Lasso has a closed form expression which is different from the smoothed Lasso approaches we consider [27], and results in the literature [24] are only named in analogy to the smoothing terminology introduced in [29]. Since the concomitant Lasso does not contain an L_1 penalty, our results to not apply.

B Nesterov smoothing

This section follows [29, Sections 2 and 4]. It introduces the basic formalism of Nesterov smoothing in Section B.1 and concretizes the approach in Section B.2.

B.1 Description of Nesterov smoothing

We are given a piecewise affine and convex function $f : \mathbb{R}^q \to \mathbb{R}$ which we aim to smooth, where $q \in \mathbb{N}$. We assume that f is composed of $k \in \mathbb{N}$ linear pieces (components). The function f can be expressed as

$$f(z) = \max_{i=1,\dots,k} \left(A[z,1]^\top \right)_i , \tag{14}$$

where $A \in \mathbb{R}^{k\times(q+1)}$ is a matrix whose rows contain the linear coefficients for each of the k pieces (with the constant coefficients being in column $q+1$), $z \in \mathbb{R}^q$, and $[z,1] \in \mathbb{R}^{q+1}$ denotes the vector

obtained by concatenating z and the scalar 1 (see Section 4).

Let $\|\cdot\|_k$ be a norm on $\mathbb{R}^k$ and $\langle\cdot,\cdot\rangle$ be the Euclidean inner product. Define the unit simplex $Q_k \subseteq \mathbb{R}^k$ as

$$Q_k = \left\{ w = (w_1, \ldots, w_k) \in \mathbb{R}^k : \sum_{i=1}^{k} w_i = 1, \text{ and } w_i \geq 0 \text{ for all } i = 1, \ldots, k \right\}.$$

To introduce the smoothing procedure, a proximity function (or prox function) is defined first on Q_k, see [29]. A prox function ρ is any nonnegative, continuously differentiable, and strongly convex function (with parameter $\sigma > 0$ and with respect to the norm $\|\cdot\|_k$). The latter means that ρ satisfies

$$\rho(s) \geq \rho(t) + \langle \nabla\rho(t), s - t\rangle + \frac{\sigma}{2}\|s - t\|_k^2$$

for all $s, t \in Q_k$.

For any $\mu > 0$, consider the function

$$f^{\mu}(z) = \max_{w \in Q_k} \left\{ \langle A[z, 1]^{\top}, w\rangle - \mu\rho(w) \right\}. \tag{15}$$

According to [29, Theorem 1], the function f^{μ} defined in eq. (15) is convex and everywhere differentiable in z for any $\mu > 0$. The function f^{μ} depends only on the parameter μ controlling the degree of smoothness. For $\mu = 0$, we recover the original unsmoothed function since $f^0(z) = \max_{w \in Q_k} \left\{ \langle A[z, 1]^{\top}, w\rangle \right\} = f(z)$. The gradient $z \mapsto \frac{\partial}{\partial z} f^{\mu}(z)$ is Lipschitz continuous with a Lipschitz constant that is proportional to μ^{-1}. A closed form expression of both the gradient and the Lipschitz constant are given in [29, Theorem 1].

Importantly, the function f^{μ} is a uniform smooth approximation of $f = f^0$ since

$$f^0(z) - \mu \sup_{w \in Q_k} \rho(w) \leq f^{\mu}(z) \leq f^0(z) \tag{16}$$

for all $z \in \mathbb{R}^q$, meaning that the approximation error is uniformly upper bounded by

$$\sup_{z \in \mathbb{R}^q} |f(z) - f^{\mu}(z)| \leq \mu \sup_{w \in Q_k} \rho(w) = O(\mu). \tag{17}$$

Indeed, eq. (16) holds true since for all $z \in \mathbb{R}^q$,

$$f^\mu(z) \geq \sup_{w \in Q_k} \langle A[z,1]^\top, w \rangle - \mu \sup_{w \in Q_k} \rho(w) = f^0(z) - \mu \sup_{w \in Q_k} \rho(w),$$
$$f^\mu(z) = \sup_{w \in Q_k} \left\{ \langle A[z,1]^\top, w \rangle - \mu \rho(w) \right\} \leq \sup_{w \in Q_k} \langle A[z,1]^\top, w \rangle = f^0(z),$$

where it was used that both the function ρ and the parameter μ are nonnegative.

B.2 Two choices for the proximity function

We consider two choices of the prox function ρ.

B.2.1 Entropy prox function

The entropy prox function $\rho_e : \mathbb{R}^k \to \mathbb{R}$ is given by

$$\rho_e(w) = \sum_{i=1}^{k} w_i \log(w_i) + \log(k)$$

for $w \in \mathbb{R}^k$.

Setting the norm $\|\cdot\|_k$ as the L_1 norm in $\mathbb{R}^k$, [29, Lemma 3] shows that ρ_e is strongly convex with respect to the L_1 norm and satisfies $\sup_{w \in Q_k} \rho_e(w) = \log(k)$. Using eq. (17), we obtain the uniform bound

$$\sup_{z \in \mathbb{R}^q} |f(z) - f_e^\mu(z)| \leq \mu \log(k)$$

for the entropy smoothed function f_e^μ obtained by using ρ_e in eq. (15). Interestingly, smoothing with the entropy prox function admits a closed-form expression of f_e^μ given by

$$f_e^\mu(z) = \max_{w \in Q_k} \left\{ \sum_{i=1}^{k} w_i \left(A[z,1]^\top \right)_i - \mu \left(\sum_{i=1}^{k} w_i \log(w_i) + \log(k) \right) \right\} = \mu \log \left(\frac{1}{k} \sum_{i=1}^{k} e^{\frac{\left(A[z,1]^\top\right)_i}{\mu}} \right),$$

see [29, Lemma 4].

B.2.2 Squared error prox function

The squared error prox function is given by

$$\rho_s(w) = \frac{1}{2}\sum_{i=1}^{k}\left(w_i - \frac{1}{k}\right)^2.$$

In [25] it is shown that the optimization in eq. (15) with squared error prox function is equivalent to the convex program

$$f_s^\mu(z) = \min_{w\in Q_k}\left(\frac{1}{k}\sum_{i=1}^{k} w_i^2 - \sum_{i=1}^{k} w_i c_i(z)\right), \tag{18}$$

where $c_i(z) = 1/\mu \cdot \left(A[z,1]^\top\right)_i - 1/k$ depends on A and μ and is defined for any $i \in \{1,\dots,k\}$. The problem in eq. (18) is equivalent to finding the Euclidean projection of the vector $c(z) = (c_1(z),\dots,c_k(z)) \in \mathbb{R}^k$ onto the k-dimensional unit simplex Q_k. This projection can be carried out efficiently using the algorithm of [26], for which a computationally more efficient version was proposed in [34] that we use in our implementations. Denoting the Euclidean projection of the vector $c(z)$ onto Q_k as vector $\hat{c}(z)$, the squared error prox approximation of f can be written as

$$f_s^\mu(z) = \langle \hat{c}(z), A[z,1]^\top\rangle - \mu\rho_s(\hat{c}(z)).$$

As $\sup_{w\in Q_k}\rho_s(w) = 1 - \frac{1}{k}$ [29, Section 4.1], we obtain the uniform bound

$$\sup_{z\in\mathbb{R}^q}|f(z) - f_s^\mu(z)| \le \mu\left(1 - \frac{1}{k}\right)$$

for the squared error smoothing approach.

C Proofs

Proposition 6. *The entropy prox smoothed function f_e^μ of eq. (7) and the squared error prox smoothed f_s^μ of eq. (9) are strictly convex functions.*

Proof of Proposition 6. The second derivative of f_e^μ is given by

$$\frac{\partial^2}{\partial z^2} f_e^\mu(z) = \frac{4e^{2x/\mu}}{\mu\left(e^{2x/\mu}+1\right)^2}$$

and hence always positive, thus making f_e^μ strictly convex. Similar arguments show that f_s^μ is strictly convex. □

Proof of Proposition 3. Using the fact that according to Condition 3, v is Lipschitz continuous with Lipschitz constant L_v, we have

$$\begin{aligned}
\sup_{\beta\in\mathbb{R}^p} \left|\Phi^\mu(\beta) - \Phi(\beta)\right| &= \sup_{\beta\in\mathbb{R}^p} \Big| v\left(|w_1(\beta)|, \ldots, |w_m(\beta)|\right) - v\left(f^\mu(w_1(\beta)), \ldots, f^\mu(w_m(\beta))\right) \Big| \\
&\leq \sup_{\beta\in\mathbb{R}^p} L_v \cdot \Big\| \left(|w_1(\beta)|, \ldots, |w_m(\beta)|\right) - \left(f^\mu(w_1(\beta)), \ldots, f^\mu(w_m(\beta))\right) \Big\|_1 \\
&\leq \sup_{\beta\in\mathbb{R}^p} L_v \cdot \sum_{i=1}^m \Big| |w_i(\beta)| - f^\mu(w_i(\beta)) \Big| \\
&\leq m L_v C^\mu,
\end{aligned}$$

thus establishing the claimed bound. □

Proof of Proposition 4. Since F_1 is continuous and strictly convex, it lays in the Skorohod topology $\mathcal{D}_K$ as defined in [31, Definition 2.2]. According to [31, Lemma 2.9], the argmax functional is continuous at F_1 with respect to the supremum norm metric. □

Proof of Proposition 5. Let $y_1 \in S_\delta(x_1)$, $y_1 \neq x_1$, be such that x_1, x_2 and y_1 are collinear.

Since F_1 is differentiable, we know that ∇F_1 exists. Since F_1 is strictly convex, the minimum x_1 is unique and $\nabla F_1(y_1) \neq 0$ as $y_1 \neq x_1$. Since F_1 is convex, the tangent at every point stays below the function. Thus considering the tangent at y_1, we have for all z that

$$F_1(y_1) + \nabla F_1(y_1)^\top (z - y_1) \leq F_1(z),$$

and thus we can bound $F_1 - \epsilon$ from below as

$$F_1(y_1) + \nabla F_1(y_1)^\top (z - y_1) - \epsilon \leq F_1(z) - \epsilon. \tag{19}$$

Observe that since $\sup_{x\in\mathbb{R}^s} |F_1(x) - F_2(x)| \leq \epsilon$, and since F_1 is strictly convex and F_2 is convex, the minimum x_2 of F_2 must satisfy $F_2(x_2) \leq F_1(x_1) + \epsilon$. Moreover, since at x_2 we have $F_2(x_2) \in [F_1(x_2) - \epsilon, F_1(x_2) + \epsilon]$, since the tangent of eq. (19) stays below $F_1(x_2) - \epsilon$, and since x_1, x_2 and y_1 are collinear, the solution z_0 satisfying

$$F_1(y_1) + \nabla F_1(y_1)^\top (z_0 - y_1) - \epsilon = F_1(x_1) + \epsilon \tag{20}$$

has the property that x_2 cannot be further away from x_1 than z_0, thus $\|x_1 - x_2\|_2 \leq \|x_1 - z_0\|_2$. Rewriting $z_0 - y_1$ in eq. (20) as $z_0 - x_1 + x_1 - y_1$ and rearranging terms yields

$$\nabla F_1(y_1)^\top (z_0 - x_1) = F_1(x_1) - F_1(y_1) + 2\epsilon - \nabla F_1(y_1)^\top (x_1 - y_1).$$

Rewriting $\nabla F_1(y_1)^\top (z_0 - x_1)$ as $\|\nabla F_1(y_1)\|_2 \cdot \|z_0 - x_1\|_2 \cdot \cos(\theta)$ for some $\theta \in [0, \pi/2)$ and applying the L_2 norm on both sides yields, after applying the triangle inequality on the right hand side,

$$\|\nabla F_1(y_1)\|_2 \cdot \|z_0 - x_1\|_2 \cdot |\cos(\theta)| \leq \|F_1(x_1) - F_1(y_1)\|_2 + 2\epsilon + \|\nabla F_1(y_1)\|_2 \cdot \|x_1 - y_1\|_2,$$

which after rearranging yields

$$\|z_0 - x_1\|_2 \leq \frac{\|F_1(x_1) - F_1(y_1)\|_2 + 2\epsilon + \|\nabla F_1(y_1)\|_2 \cdot \|x_1 - y_1\|_2}{|\cos(\theta)| \cdot \|\nabla F_1(y_1)\|_2}.$$

We write $|\cos(\theta)|^{-1} = C_{y_1}$ and note that θ is determined by y_1 (and z) but independent of x_2. Since x_1 is fixed, and F_1 is differentiable, it is also locally Lipschitz in a ball around x_1 (note that the Lipschitz constant is independent of x_2). Thus there exists $L_{y_1} > 0$ such that $\|F_1(x_1) - F_1(y_1)\|_2 \leq L_{y_1}\|x_1 - y_1\|_2$. This is true for any $y_1 \in S_\delta(x_1)$, thus by taking the supremum over $y_1 \in S_\delta(x_1)$ we arrive at two constants $C_\delta > 0$ and $L_\delta > 0$ (independent of y_1) which satisfy

$$\|x_1 - x_2\|_2 \leq \|z_0 - x_1\|_2 \leq C_\delta \left[\|\nabla F_1(y_1)\|_2^{-1}(\delta L_\delta + 2\epsilon) + \delta\right],$$

where it was used that $\|x_1 - y_1\|_2 = \delta$ since $y_1 \in S_\delta(x_1)$, and that $\|x_1 - x_2\|_2 \leq \|x_1 - z_0\|_2$.

The bound $\|x_1 - x_2\|_2 \leq \|z_0 - x_1\|_2$ is conditional on y_1 and x_1, x_2 being collinear. Since x_2 is

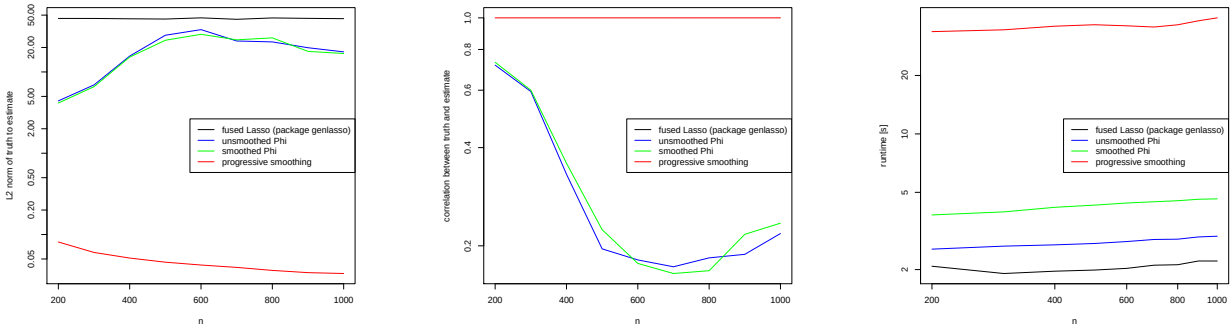

Figure 6: Fused Lasso: L_2 norm (left) and correlation (middle) between simulated truth and parameter estimates as a function of n while $p = 100$. Runtime in seconds (right) has a log scale on both axes. The comparison includes fusedlasso (R-package genlasso), unsmoothed Φ, smoothed surrogate Φ^μ, and progressive smoothing.

unknown, we take the supremum over all $y_1 \in S_\delta(x_1)$, resulting in

$$\|x_1 - x_2\|_2 \leq \sup_{y_1 \in S_\delta(x_1)} \|\nabla F_1(y_1)\|_2^{-1} \cdot C_\delta(\delta L_\delta + 2\epsilon) + C_\delta \delta,$$

thus establishing the claimed upper bound. □

D Additional simulation studies

This section presents two additional simulations for the fused Lasso of [33] (provided in the *fusedlasso* function of the R-package *genlasso*, see [1]), and the graphical Lasso of [19] (provided in the *glasso* function of the R-package *glasso*, see [18]). The simulation setup is the one used in Section 2.

D.1 Fused Lasso

For the fused Lasso, a dependency structure among the entries of the parameter vector $\beta \in \mathbb{R}^p$ has to be generated. This dependency structure is given by the adjacency matrix E, see Section 4.4. We generate E at random with edge probability 0.5 in each repetition. In this experiment, we employ the choices of u, v, and w_j, $j \in \{1, \ldots, p\}$, for the fused Lasso given in Section 4.4.

Figure 6 shows scaling results in n while keeping $p = 100$ fixed. We observe that the function fusedlasso of the R-package genlasso seems to have troubles finding good estimates, while optimizing the objective function of the fused Lasso, given in Section 4.4, with our progressive smoothing

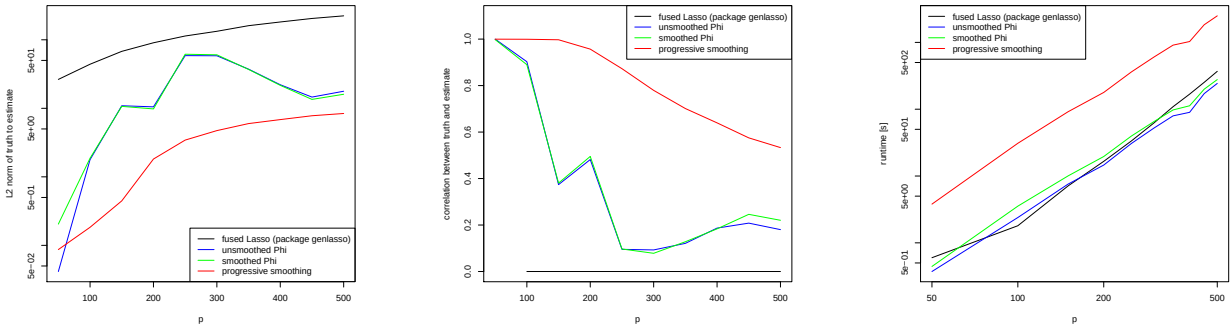

Figure 7: Fused Lasso: L_2 norm (left) and correlation (middle) between simulated truth and parameter estimates as a function of p while $n = 100$. Runtime in seconds (right) has a log scale on both axes. The comparison includes fusedlasso (R-package genlasso), unsmoothed Φ, smoothed surrogate Φ^μ, and progressive smoothing.

algorithm works much better and yields stable regression estimates having a low L_2 norm and a high correlation with the truth. The *fusedlasso* function and minimizing the unsmoothed objective yield fastest results, followed by our smoothing approach. Progressive smoothing is a constant factor slower as expected. Overall, there only seems to be a weak runtime dependence on n.

Figure 7 shows similar results for the scaling in p. Here, the *fusedlasso* function again yields estimates with the largest deviation from the generated truth in the L_2 norm, and also the lowest correlation with the true parameters. Progressive smoothing yields the most accurate and stable results with respect to both L_2 norm and correlation, while the other two approaches (unsmoothed objective and smooth surrogate, denoted as unsmoothed Phi and smoothed Phi) exhibit a more unstable behavior. The runtime scaling of all methods is roughly similar in p.

D.2 Graphical Lasso

For the graphical Lasso, we generate a sample covariance matrix S from a Wishart distribution with sample size 1, $p \in \{5, \ldots, 30\}$ degrees of freedom, and scale matrix set to the $p \times p$ dimensional identity matrix, see Sections 4.4 and 2. To define Φ and its smooth surrogate Φ^μ, we employ the functions u, v, and w_j, $j \in \{1, \ldots, m\}$ with $m = p(p+1)/2$, for the graphical Lasso given in Section 4.4.

We then compute estimates for the unsmoothed graphical Lasso objective function using the R-package *glasso*. Additionally, we minimize the unsmoothed Φ for the graphical Lasso as well as our

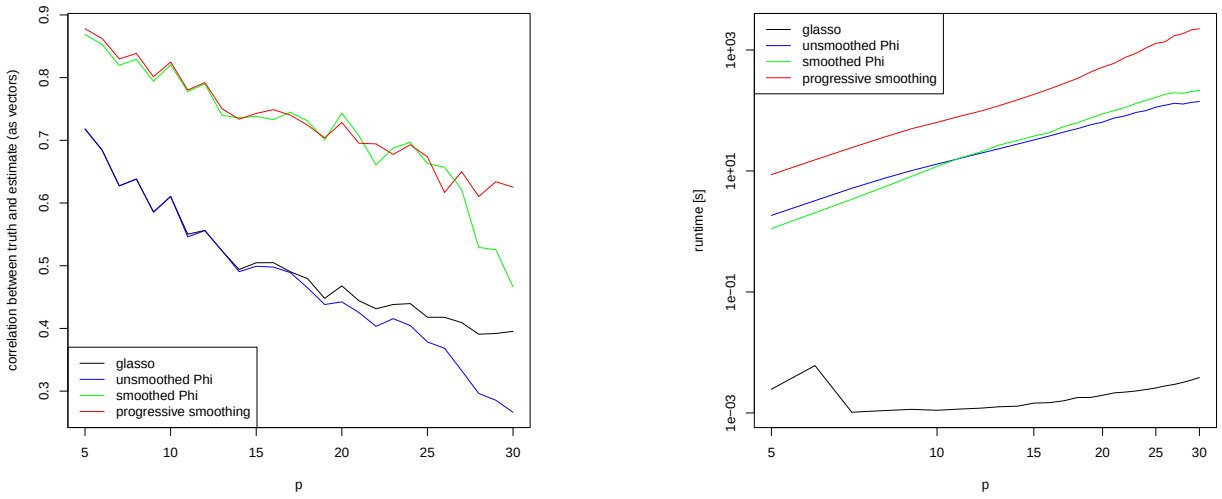

Figure 8: Graphical Lasso: correlation (left) between simulated truth and parameter estimates as a function of n. Runtime in seconds (right) has a log scale on both axes. The comparison includes glasso (R-package glasso), unsmoothed Φ, smoothed surrogate Φ^μ, and progressive smoothing.

smooth surrogate Φ^μ using the BFGS algorithm in R's function *optim*, and employ our progressive smoothing algorithm. To optimize over all positive definite matrices Θ in the objective function of the graphical Lasso (see Section 4.4), we parametrize Θ with its Cholesky decomposition. To be precise, we write $\Theta = CC^\top$, where C is a lower triangular matrix. This ensures that Θ will always be positive definite. As Θ is fully parametrized via C, and C is lower diagonal, we actually only optimize over $p(p+1)/2$ parameters.

We report the correlation between the true and estimated precision matrices (for this the matrices are unfolded as vectors), as well as the runtime.

Results are given in Figure 8. The results show that estimates of the precision matrix found with the glasso R-package and by minimizing the unsmoothed objective function have a lower correlation with the generated truth than the estimates obtained with our two smoothing approaches. The implementation of the graphical Lasso in the R-package glasso seems highly optimized, resulting in fast runtimes. Our smoothing approaches are not as highly optimized and thus slower than glasso, though all three exhibit roughly a similar runtime scaling.

References

[1] Taylor B. Arnold and Ryan J. Tibshirani. genlasso: Path Algorithm for Generalized Lasso Problems, 2020. R-package version 1.5: `https://cran.r-project.org/package=genlasso`.

[2] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data. *Journal of Machine Learning Research*, 9:485–516, 2008.

[3] Amir Beck and Marc Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM J Imaging Sciences*, 2(1):183–202, 2009.

[4] Amir Beck and Marc Teboulle. Smoothing And First Order Methods: A Unified Framework. *Siam J Optim*, 22(2):557–580, 2012.

[5] Stephen R. Becker and Emmanuel J. Candès. Templates for convex cone problems with applications to sparse signal recovery. *Math Prog Comp*, 3:165–218, 2011.

[6] A. Belloni, V. Chernozhukov, and L. Wang. Square-root lasso: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98(4):791–806, 2011.

[7] Xi Chen, Seyoung Kim, Qihang Lin, Jaime G. Carbonell, and Eric P. Xing. Graph-Structured Multi-task Regression and an Efficient Optimization Method for General Fused Lasso. *arXiv:1005.3579*, pages 1–21, 2010.

[8] Xi Chen, Qihang Lin, Seyoung Kim, Jaime G. Carbonell, and Eric P. Xing. An efficient proximal gradient method for general structured sparse learning. *Journal of Machine Learning Research*, 11, 2010.

[9] Xi Chen, Qihang Lin, Seyoung Kim, Jaime G. Carbonell, and Eric P. Xing. Smoothing proximal gradient method for general structured sparse regression. *Ann Appl Stat*, 6(2):719–752, 2012.

[10] E. Chi, T. Goldstein, C. Studer, and R. Baraniuk. fasta: Fast Adaptive Shrinkage/Thresholding Algorithm, 2018. R-package version 0.1.0: `https://cran.r-project.org/package=fasta`.

[11] I. Daubechies, M. Defrise, and C.D. Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math.*, 57(11):1413–1457, 2004.

[12] Frank Dondelinger and Sach Mukherjee. The joint lasso: high-dimensional regression for group structured data. *Biostatistics*, 21:219–235, 2020.

[13] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann Stat*, 32(2):407–499, 2004.

[14] Jianqing Fan and Runze Li. Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties. *J Am Stat Assoc*, 96(456):1348–1360, 2001.

[15] J. Friedman, T. Hastie, and R. Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[16] J. Friedman, T. Hastie, R. Tibshirani, B. Narasimhan, K. Tay, N. Simon, and J. Qian. glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models, 2020. R-package version 4.0: `https://cran.r-project.org/package=glmnet`.

[17] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization Paths for Generalized Linear Modelsvia Coordinate Descent. *J Stat Softw*, 33:1–22, 2010.

[18] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. glasso: Graphical Lasso: Estimation of Gaussian Graphical Models, 2019. R-package version 1.11: `https://cran.r-project.org/package=glasso`.

[19] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[20] Georg Hahn, Moulinath Banerjee, and Bodhisattva Sen. Parameter Estimation and Inference in a Continuous Piecewise Linear Regression Model. http://www.cantab.net/users/ghahn/preprints/PhaseRegMultiDim.pdf, 2017.

[21] Georg Hahn, Sharon M. Lutz, Nilanjana Laha, and Christoph Lange. smoothedLasso: Smoothed LASSO Regression via Nesterov Smoothing, 2020. R-package version 1.4: `https://cran.r-project.org/package=smoothedLasso`.

[22] Hamed Haselimashhadi and Veronica Vinciotti. A Differentiable Alternative to the Lasso Penalty. *arXiv:1609.04985*, pages 1–12, 2016.

[23] T. Hastie and B. Efron. lars: Least Angle Regression, Lasso and Forward Stagewise, 2013. R-package version 1.2: `https://cran.r-project.org/package=lars`.

[24] Mathurin Massias, Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Generalized Concomitant Multi-Task Lasso for Sparse Multimodal Regression. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain*, volume 84. PMLR, 2018.

[25] R. Mazumder, A. Choudhury, G. Iyengar, and B. Sen. A Computational Framework for Multivariate Convex Regression and Its Variants. *J Am Stat Assoc*, 114(525):318–331, 2019.

[26] C. Michelot. A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$. *J Optimiz Theory App*, 50(1):195–200, 1986.

[27] Eugene Ndiaye, Olivier Fercoq, Alexandre Gramfort, Vincent Leclère, and Joseph Salmon. Efficient Smoothed Concomitant Lasso Estimation for High Dimensional Regression. In *7th International Conference on New Computational Methods for Inverse Problems*, 2017.

[28] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Dokl Akad Nauk SSSR*, 269(3):543–547, 1983.

[29] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program. Ser. A*, 103:127–152, 2005.

[30] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Stat Comp, Vienna, Austria, 2014.

[31] Emilio Seijo and Bodhisattva Sen. A continuous mapping theorem for the smallest argmax functional. *Electron J Stat*, 5:421–439, 2011.

[32] R. Tibshirani. Regression Shrinkage and Selection Via the Lasso. *J Roy Stat Soc B Met*, 58(1):267–288, 1996.

[33] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and Smoothness via the Fused Lasso. *J Roy Stat Soc B Met*, 67(1):91–108, 2005.

[34] W. Wang and M. Carreira-Perpiñán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv:1309.1541*, pages 1–5, 2013.

[35] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *J Roy Stat Soc B Met*, 68(1):49–67, 2005.

[36] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *J Roy Stat Soc B Met*, 67(2):301–320, 2005.