Optimization and Algorithms in Sparse Regression

Optimization and Algorithms in Sparse Regression

Screening Rules, Coordinate Descent, and Normalization

Johan Larsson

LUND
UNIVERSITY

Thesis for the degree of Doctor of Philosophy

Thesis Advisors
Jonas Wallin and Małgorzata Bogdan

Faculty Opponent
Professor Mário A. T. Figueiredo (Instituto Superior Técnico, Lisbon, Portugal)

To be presented, with the permission of the Lund University School of Economics and Business Administration of Lund University, for public criticism in the Clark Kent lecture hall (Kentsalen) at the Department of Statistics on Sunday, the 34th of December 2024 at 24:00.

Organization
LUND UNIVERSITY
Department of Statistics
Box 7080
SE–220 07 Lund
Sweden

Document name
DOCTORAL DISSERTATION

Date of disputation
2024-05-24

Sponsoring organization

Author(s)
Johan Larsson

Title and subtitle
Optimization and Algorithms in Sparse Regression: Screening Rules, Coordinate Descent, and Normalization

Abstract
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Key words
power, victory, awesomeness

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
English

ISSN and key title

ISBN
978-91-8104-076-0 (print)
978-91-8104-077-7 (pdf)

Recipient's notes

Number of pages
96

Price

Security classification

Optimization and Algorithms in Sparse Regression

Screening Rules, Coordinate Descent, and Normalization

Johan Larsson

LUND
UNIVERSITY

Cover illustration front: The elastic net path for a data set of diabetes patients.

Lund University School of Economics and Management
The Department of Statistics
Box 743, SE-220 07
Lund, Sweden

ISBN: 978-91-8104-076-0 (print)
ISBN: 978-91-8104-077-7 (electronic)

Printed in Sweden by Media-Tryck, Lund University, Lund 2024

There's a point when you go with what you've got.
Or you don't go.

—Joan Didion

Contents

Acknowledgements

Abstract

Popular Science Summary

List of Publications

This thesis is based on the following publications.

I **The Strong Screening Rule for SLOPE**
Johan Larsson, Małgorzata Bogdan, and Jonas Wallin (Dec. 6–12, 2020). In: *Advances in Neural Information Processing Systems 33*. 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Ed. by Hugo Larochelle et al. Vol. 33. Virtual: Curran Associates, Inc., pp. 14592–14603. ISBN: 978-1-71382-954-6

II **Look-Ahead Screening Rules for the Lasso**
Johan Larsson (Sept. 6, 2021). In: *22nd European Young Statisticians Meeting – Proceedings*. 22nd European Young Statisticians Meeting. Ed. by Andreas Makridis, Fotios S. Milienos, Panagiotis Papastamoulis, Christina Parpoula, and Athanasios Rakitzis. Athens, Greece: Panteion university of social and political sciences, pp. 61–65. ISBN: 978-960- 7943-23-1

III **The Hessian Screening Rule**
Johan Larsson and Jonas Wallin (Nov. 28–Dec. 9, 2022). In: *Advances in Neural Information Processing Systems 35*. 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Ed. by Sanmi Koyejo, Sidahmed Mohamed, Alekh Agarwal, Danielle Belgrave, Kyunghyun Cho, and Alice Oh. Vol. 35. New Orleans, USA: Curran Associates, Inc., pp. 15823–15835. ISBN: 978-1-71387-108-8

IV **Benchopt: Reproducible, Efficient and Collaborative Optimization Benchmarks**
Thomas Moreau, Mathurin Massias, Alexandre Gramfort, Pierre Ablin, Pierre-Antoine Bannier, Benjamin Charlier, Mathieu Dagréou, Tom Dupré la Tour, Ghislain Durif, Cassio F. Dantas, Quentin Klopfenstein, Johan Larsson, En Lai, Tanguy Lefort, Benoit Malézieux, Badr Moufad, Binh T. Nguyen, Alain

Rakotomamonjy, Zaccharie Ramzi, Joseph Salmon, and Samuel Vaiter (Nov. 28–Dec. 9, 2022). In: *Advances in Neural Information Processing Systems 35*. 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Ed. by Sanmi Koyejo, Sidahmed Mohamed, Alekh Agarwal, Danielle Belgrave, Kyunghyun Cho, and Alice Oh. Vol. 35. New Orleans, USA: Curran Associates, Inc., pp. 25404–25421. ISBN: 978-1-71387-108-8

v **Coordinate Descent for SLOPE**

Johan Larsson, Quentin Klopfenstein, Mathurin Massias, and Jonas Wallin (Apr. 25–27, 2023). In: *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics.* AISTATS 2023. Ed. by Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent. Vol. 206. Proceedings of Machine Learning Research. Valencia, Spain: PMLR, pp. 4802–4821

vi **Regularization and Scaling in Sparse Regression**

Introduction

As you know from teaching introductory statistics, 30 is infinity.

—Andrew Gelman

1 Background

With modern advances in science and technology, statistical models and the data on which they are fit are becoming increasingly complex. Data sets are expanding in size, often both in terms of the number of variables (features) as well as the number of observations. In some fields, this growth in complexity has been paralleled with more effective methods with which to collect observations, as in, for instance, crowd science and social media data. But in other areas, collecting data still amounts to a costly endeavor. In bioinformatics, for example, ethical concerns and rising requirements on the quality of data have only served to *raise* the costs of data collection. And as a result, the data collected in these fields is becoming *wider*: the ratio between the number of variables (features) and the number of observations is increasing (Table 1).

The growth in the number of observations is, relatively speaking, a luxury problem, since it, at least as far as the model is concerned, provides only benefit. But an expansion in the number of features (wider data) is a more delicate issue. The problem is that if all the features that we have collected are important then we are out of luck as far as understanding it with a statistical model goes. Instead, we have to more or less hope that there is a *sparse* representation of our data that, with some acceptable loss of information, allows us to understand the problem that we are studying.

We can call this hope the *sparsity assumption*, which can be motivated through the *bet-on-sparsity principle*: assume that the underlying signal is sparse (Figure 1) and use a sparse method to model it. If the assumption is correct, then our method has a chance of doing well. But if the assumption is incorrect, then our method will not work—but

Table 1: Tall and wide data. Each row is an observation, for instance the measurement on a person in a study, and each column (feature) represents all the measurements on a variable for all the observations.

(a) Tall data

Feature 1	Feature 2	Feature 3
0	0.32	1
1	1	-1
⋮	⋮	⋮

(b) Wide data

Feature 1	Feature 2	Feature 3	⋯
0	0.32	1	⋯
1	1	-1	⋯

no other method would (Hastie, Robert Tibshirani, and Friedman 2009).

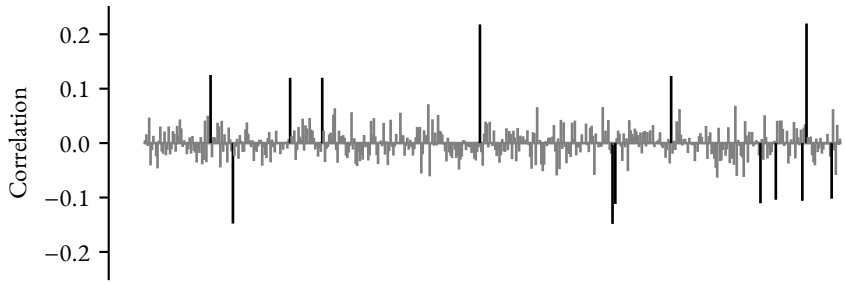


Figure 1: A relatively sparse signal. The plot shows standard Pearson correlations between the response vector $\boldsymbol{y}$ and each feature in the `madelon` data set (Guyon et al. 2004). Correlations above 0.1 have been colored in black, the rest in gray.

The success of neural networks and other complex models that model high-dimensional data well yet do not enforce sparsity does raise questions as to the validity of this principle. But in our setting, which, loosely speaking, is *explainable* methods for regression, it still bears relevance.

Technically speaking, we are interested in data sets that are made up of a $n \times p$ matrix of features $\boldsymbol{X}$ and a response vector of length n, $\boldsymbol{y}$:

$$\boldsymbol{X} = \begin{bmatrix} 1.5 & 0.3 & \cdots & x_{1,p} \\ -0.9 & 0.1 & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix}, \qquad \boldsymbol{y} = \begin{bmatrix} 0.2 \\ -0.9 \\ \vdots \\ y_n \end{bmatrix},$$

where we have inserted some arbitrary values for the sake of illustration. The data presented in Table 1 corresponds to $\boldsymbol{X}$ here (each row is an observation, each column a feature).

In the simplest case, we assume that $\boldsymbol{y}$ is a linear combination of the features in $\boldsymbol{X}$ plus some noise ($\boldsymbol{\epsilon}$), for instance measurement noise, which we write mathematically as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \beta_0 + \boldsymbol{\epsilon},$$

where $\boldsymbol{\beta}$ is a vector of coefficients and β_0 the *intercept*. In this representation of the data, the coefficients $\boldsymbol{\beta}$ are the parameters that we are interested in estimating and represent the effect each feature has on the response vector $\boldsymbol{y}$. Assuming that this is in fact the true relationship between $\boldsymbol{\beta}$, $\boldsymbol{X}$, and $\boldsymbol{y}$, a natural choice of model to fit this data with is *linear regression*[1].

In the presence of noise, however, there generally exists no β that will fit the data perfectly, and we must therefore accept that the linear regression model is only an approximation. The natural follow-up question is then: what *is* a good approximation? To answer this question, we need to define some measure of error. The most common measure, at least as far as linear regression models go, is the sum of squared errors between the predicted response vector

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$$

and the true response vector $\boldsymbol{y}$, that is,

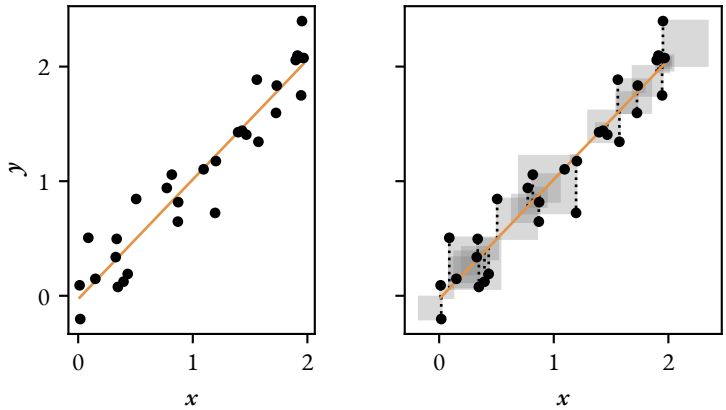
$$\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

The smaller this measure—the better the fit, which means that finding a vector $\boldsymbol{\beta}$ that minimizes this error can be posed as the following optimization problem:

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2. \tag{1}$$

We let $\boldsymbol{\beta}^*$ to be the solution to this problem—the optimum, and will use $\hat{\boldsymbol{\beta}}$ to refer to the estimate that we obtain from some algorithm. Generally, we will assume that the algorithm has converged to the optimum so that $\boldsymbol{\beta}^* = \hat{\boldsymbol{\beta}}$, but in actuality we almost always have some amount of *suboptimality*, that is, typically $|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}| > 0$. The factor of $1/2$ in Problem (1) is included for convenience, for reasons that will become clear later on.

[1] In general, we also need additional assumptions on the noise term $\boldsymbol{\epsilon}$.



(a) The slope of the orange line is β. The point where the line intersects the y-axis is the intercept β_0.

(b) The dotted lines are the residuals and the grey squares are the squared errors. The orange line is that which minimizes the total surface of all of these squares summed together.

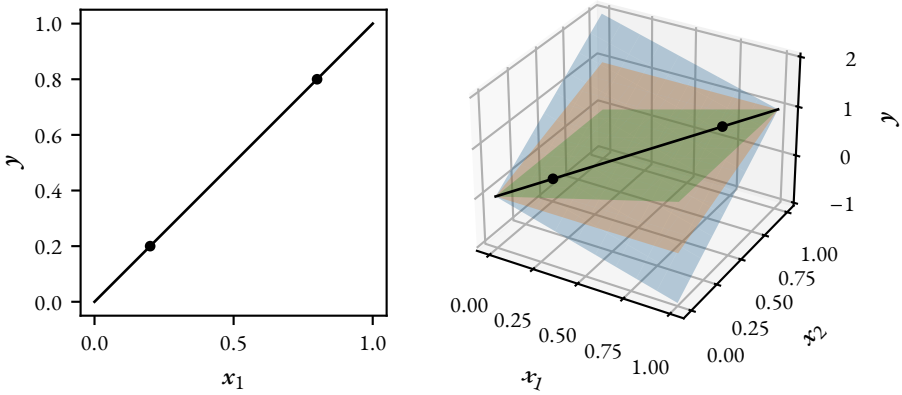
Figure 2: Simple ordinary least-squares linear regression for a one-feature problem

Solving Problem (1) is equivalent to fitting the ordinary least-squares (OLS) regression model, which, for a simple case of a single feature, we have illustrated in Figure 2. Picking a different measure of error would lead to a different method (and often a different linear regression line), but in this thesis we will focus on the method of least squares.

If we have many more observations than features ($n \gg p$), then our linear regression model might stand a decent chance of recovering the true model (coefficients). But if the tables were turned and the features instead outnumbered the observations ($p \gg n$), the model would in fact break down.

The problem is that our linear regression model will be able to fit out particular data set perfectly but (often) not generalize well to new data, even if it comes from the same underlying data-generating mechanism. This is called *overfitting* and it happens because we have more parameters (regression coefficients) than unknowns (observations). This means that there are now many different regression models that will fit the data equally well and that the solution, therefore, is not unique. In Figure 3, we illustrate what overfitting looks like for the linear regression model in one and two parameters.

In principle, the problem of overfitting in linear regression is the same as that in polynomial regression: as we increase the number of parameters (degree of the poly-



(a) With one feature, the simple ordinary least-squares regression line fits the data perfectly.

(b) With two features, an infinite number of regression planes fits the data perfectly.

Figure 3: A linear regression problem with two observations. The example is of course artifical, but demonstrating overfitting in more dimensions than this would require more than the faculties of our limited human minds are capable of.

nomial in this case), the model eventually becomes too flexible and overfits (Figure 4). Note that, in this example, overfitting occurs even when he number of parameters is smaller than the numbe of observations. This is the case with linear regression too, for which overfitting may occur in cases when n is larger than p.

This problem of over-parameterization is one of the main motivations for the use of *regularization*.

2 Regularization

A solution to the problem of over-parameterization is to *regularize* the model, by shrinking or altogether removing some of its parameters. In other words, we set a kind of budget on β, allowing only some of its elements to be non-zero or limiting its size. We call the indices of the selected features the model's *support*. If features one and three are selected, then the support if $\{1, 3\}$. If all of the features are in the model, then the support if $\{1, 2, \ldots, p\}$ (as in the case of OLS regression). Finally, if no features are selected, then the support is the empty set $\emptyset$.

The simplest type of regularization is called *best-subset selection*, in which we simply limit the number of coefficients we allow to be non-zero to a constant k, which is

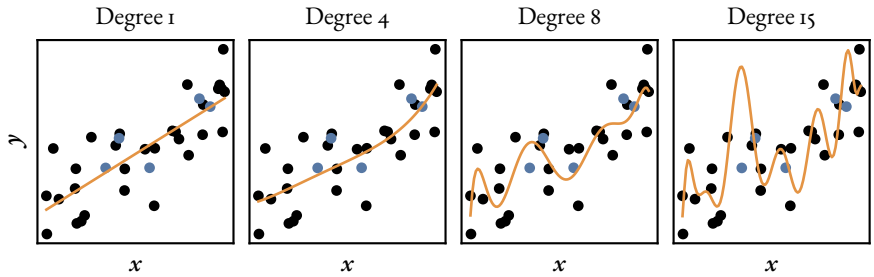


Figure 4: Polynomial regression for a one-feature problem. The data is generated from the simple linear model $= 2x_i + \varepsilon_i$, where $\varepsilon \sim \text{Normal}(0, 0.5^2)$ identically and independently. The line fits the data increasingly well as the degree of the polynomial increases, but when new data arrives (the blue points), we see that the model generalizes poorly.

equivalent to selecting k of the features in $\boldsymbol{X}$. We examine all $\binom{p}{k}$ possible combinations of features and then pick the combination that fits the our data best. Best subset selection can formally be posed as the following optimization problem:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2, \\ &\text{subject to} && \|\boldsymbol{\beta}\|_0 \leq k, \end{aligned}$$

where $\|\cdot\|_0$ is the ℓ_0 norm[2] is the number of non-zero elements in a vector.

In other words, if $p = 3$ and $k = 2$, for instance, the following models would satisfy our constraints:

$$\boldsymbol{\beta} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \boldsymbol{\beta} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \qquad \text{and} \qquad \boldsymbol{\beta} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

But the following model would not:

$$\boldsymbol{\beta} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

The primary problem with this method is that it is computationally infeasible when p is large since the number of possible models grows exponentially with p. With

[2] Technically speaking, the ℓ_0 norm is not actually a true norm.

$p = 100$ and $k = k$, for instance, there are

$$\binom{100}{5} = 75\,287\,520$$

possible models to consider. And this problem is further exacerbated by the fact that we typically has to consider a variety of values for k.

In an interesting turn of events, however, Bertsimas, A. King, and Mazumder (2016) has shown that the best subset selection problem can actually be written as a mixed-integer optimization problem. This enables the use of modern optimization software, which can then handle best subset problems of dimensions that previously were thought unattainable. Yet, although this result has offered a considerable improvement in run-time performance for the algorithm, it is still the case that it struggles in high dimensions. Solving a problem with $n = 500$ and $p = 100$ for $k \in \{1, 2, \ldots, 50\}$, for instance, still comes down to a hefty 76.8 hours of computation (Hastie, Robert Tibshirani, and Ryan Tibshirani 2020). In contrast, the time taken to fit the model that we will consider next, the lasso, amounts to 0.014 seconds for the equivalent problem.

2.1 The Lasso

One solution to the complexity problem of best-subset selection is to relax the constraint to one that makes the problem easier to solve, yet still retains the sparsity-enforcing property of the ℓ_0 norm. A natural candidate for this is the ℓ_1 norm, which leads to the following problem,

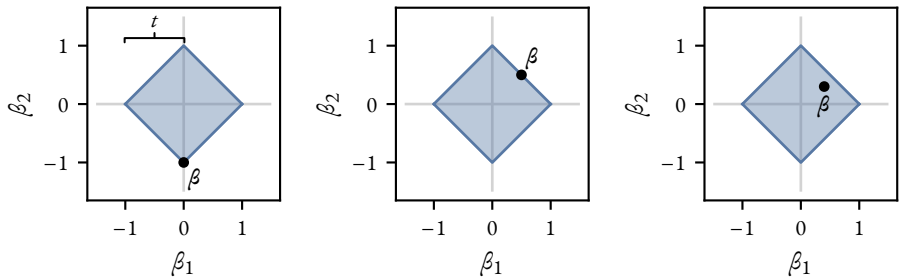
$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2, \\ &\text{subject to} && \|\boldsymbol{\beta}\|_1 \leq t, \end{aligned} \tag{2}$$

where all we did was replace the ℓ_0 norm with the ℓ_1 norm,

$$\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j|$$

As a consequence, we have replaced the integer-valued k with a real-valued (but positive) t. This problem is known as ℓ_1-regularized regression or, more commonly, as the *lasso*[3]. The lasso was introduced to the statistics community by Robert Tibshirani (1996) but actually stems from much earlier research done in the field of signal processing by

[3]The lasso is sometimes written as the acronym LASSO for *least absolute shrinkage and selection operator*, but we will stick with the lower-case version here, which the authors themselves use in their recent work.



(a) A sparse solution, $\boldsymbol{\beta} = [0, -1]^\intercal$.

(b) A dense solution, $\boldsymbol{\beta} = [0.5, 0.5]^\intercal$.

(c) A solution in which the constraint is inactive, $\boldsymbol{\beta} = [0.4, 0.3]^\intercal$. t is too small to have any effect on the solution.

Figure 5: The ℓ_1 norm ball in $\mathbb{R}^2$ with some possible solutions indicated by $\boldsymbol{\beta}$. The ℓ_0 ball (for best-subset selection) and $k = 1$ would be lines of infinite length. along both of the axes.

Santosa and Symes (1986). Donoho and Johnstone (1994, 1995) subsequently introduced the concept of the *basis pursuit* problem, which is closely related to the lasso, and developed much of the theoretical framework for the lasso.

If you have encountered the lasso previously, it is likely that you have seen it formulated as the following unconstrained optimization problem:

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1.$$

In this section, however, we will prefer the constrained formulation of the problem as it is given in Problem (2), which we think is intuitive and easier to understand. The two formulations are equivalent, however, and will lead to exactly the same solution for a suitable choice of t and λ. Later, in Section 3.2, we will make this connection clear.

We saw previously that the ℓ_0 constraint in best-subset selection puts a budget on the number of features allowed in the model. The ℓ_1 norm, in contrast, instead puts a budget on the *size* of the coefficients. This enforces not only sparsity but also shrinkage in the solution. In Figure 5, we have visualized how this constraint affects the solution of the least-squares objective.

There is an extensive body of work on the lasso and it has spawned a number of offshoots, such as the fused lasso (Robert Tibshirani, M. Saunders, et al. 2005), group lasso (Yuan and Lin 2005), adaptive lasso (Zou 2006), graphical lasso (Friedman, Hastie,

and Robert Tibshirani 2008), and square-root lasso (Belloni, Chernozhukov, and Wang 2011). In this thesis, however, we focus on the standard lasso.

Note, also, that the lasso is not limited to just regularized *linear* regression but can in fact be used for the entire family of generalized linear models, such as logistic, Poisson, multinomial, and multivariate regression, as well as survival models such as Cox regression. The use of the ℓ_1-norm penalty has also found its way into many other areas of statistics as well as the fields of signal processing and machine learning, including matrix factorization, clustering, and deep learning.

An interesting property of the lasso is that it is possible (and computationally feasible) to exactly solve the lasso problem for all possible values of $t \in [0, \infty)$. This is called the *lasso path* (Figure 6). It begins at $t = 0$, for which the constraint region is a point, forcing all of the coefficients to be exactly zero. Then as t increases, the constraint region grows, allowing the coefficients to enter the model and grow. The reason for why it is possible to solve for the full path is that the solution vector β, as a function of t, is linear and continuous between the values of t for which features enter or leave the model.

One problem with the lasso, however, is that it does not deal with the correlated features as intuition (at least that of the author) would suggest. If two features are correlated highly enough, for instance, the lasso will select one of them and drop the other (set its coefficient to zero).[4] This is not necessarily a problem for the prediction $\hat{y}$, but it means that the estimated coefficients no longer provide trustworthy estimates of variable (feature) importance. This effect is the result of the behavior of the ℓ_1 norm, which penalizes the *size* of the coefficients. If two features provide the same, or nearly the same, information about the response, then the optimization problem can attain a lower value by setting one of the coefficients to zero.

This is a problem that the *elastic net*—the topic of the next section—is designed to overcome.

2.2 The Elastic Net

The elastic net is a combination of the lasso and ridge regression,[5] which can be written as the following optimization problem:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|y - X\beta\|_2^2, \\ &\text{subject to} && \alpha\|\beta\|_1 + (1-\alpha)\|\beta\|_2^2 \leq t. \end{aligned} \tag{3}$$

[4] In the case where the features are perfectly correlated, it may in fact be the optimization algorithm that decides which of them is picked.

[5] Ridge regression is also known as Tikhonov regression; and in deep learning, ℓ_1-regularization is typically called *weight decay*.

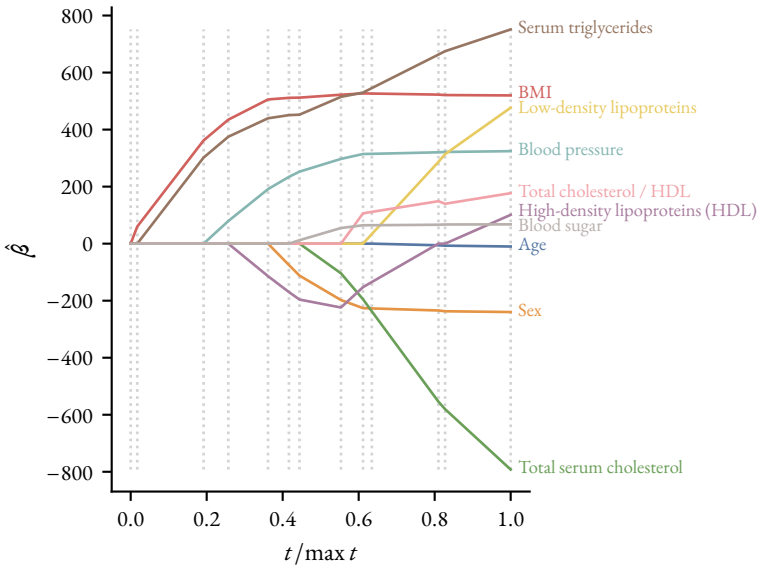
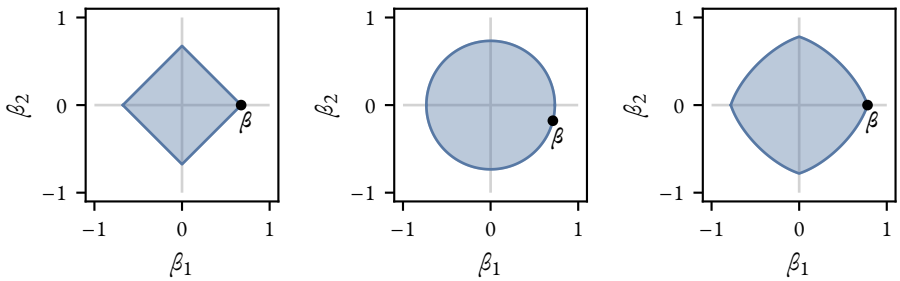


Figure 6: The lasso path for the `diabetes` data set (Efron et al. 2004), which consists of $n = 442$ observations and $p = 10$ features. The path shows the coefficients as a function of the parameter t, which controls the size of the constraint region. The path is piecewise linear with kinks occurring only when features enter or exit the model. At $t = 0$, the model is completely sparse and the support is $\emptyset$—the empty set. At $t/\max t = 1$, the model is the ordinary least-squares model and the support is the full set of predictors, $\{1, 2, \ldots, p\}$.

The difference compared to the lasso is that we have transformed our constraint into a linear combination of the ℓ_1 and squared ℓ_2 norms. The parameter α controls the balance between these constraints. By setting $\alpha = 1$, we turn the problem into the lasso (Figure 7a). And at $\alpha = 0$, we instead have ridge regression (Figure 7b). Any value $\alpha \in (0, 1)$ yields a combination of the two (Figure 7c).



(a) When $\alpha = 1$, the constraint is the lasso (ℓ_1-norm) ball. And in this case the solution is sparse.

(b) When $\alpha = 0$, the constraint is the ridge (ℓ_2-norm) ball. Here, the solution is not sparse.

(c) When $\alpha \in (0, 1)$, the constraint region is a combination of the ℓ_1 and ℓ_2 balls. In this case we have $\alpha = 1/2$. Once again, the solution is sparse.

Figure 7: The constraint regions for the elastic net for different values of α

The elastic net was first proposed by Zou and Hastie (2005). In addition to dealing with the problems encountered in using the lasso for highly correlated features, the elastic net also yields improved predictive performance in many situations. The latter fact is perhaps not so surprising given that it is a combination of methods that essentially assume different structure in the data. The lasso works well when the true signal is sparse, while ridge regression handles the situation where there are weak signals better. And since the elastic net contains both these models as special cases, in addition to any combination thereof, it naturally extends to a greater variety of problems.

2.3 SLOPE

Another way of dealing with the problem of correlated features is to use *Sorted L-One Penalized Estimation* (SLOPE) (Bogdan, Berg, Sabatti, et al. 2015; Bogdan, Berg, Su, et al. 2013; Zeng and Figueiredo 2014). SLOPE is a generalization of both the lasso and *the octagonal shrinkage and clustering algorithm for regression* (OSCAR) (Bondell and

Reich 2008). It is represented by the following optimization objective:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2, \\ &\text{subject to} && \sum_{j=1}^{p} \lambda_j |\beta|_{(j)} \leq t, \end{aligned} \tag{4}$$

where $\boldsymbol{\lambda}$ is a non-increasing and non-negative sequence of penalization weights and where we define the subscript operator (j) such that

$$|\beta|_{(1)} \geq |\beta|_{(2)} \geq \cdots \geq |\beta|_{(p)}.$$

The left-hand side of the constraint in SLOPE is, perhaps somewhat surprisingly, actually a norm: the *sorted* ℓ_1 *norm*.

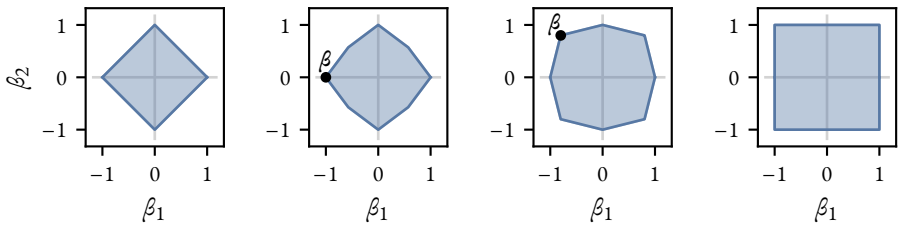
The perhaps most salient feature of SLOPE is that it clusters coefficients (Figueiredo and Nowak 2014; Schneider and P. Tardivel 2022). This is a consequence of the sorted ℓ_1 norm and the choice of $\boldsymbol{\lambda}$—larger differences between adjacent elements increase the propensity for clustering. This property makes SLOPE well-adapted to the case when features are highly correlated, which is a situation that the lasso struggles with. In cases where the lasso might set one of the correlated features to zero, SLOPE will often instead set them to exactly the same value. This is a property that is not shared by the elastic net, which handles correlation (although not quite as delicately), but does not cluster coefficients.

But as we mentioned previously, SLOPE is actually a generalization of lasso and thus contains it as a special case (Figure 8a), which is attained by setting all of the elements of the penalization weight vector $\boldsymbol{\lambda}$ to the same value. On the opposite end, setting only the first element to a non-zero value and the remaining ones to zero yields the infinity norm (Figure 8d).

SLOPE also has other appealing properties, such as the ability to (under certain assumptions on the design) control the false discovery rate[6] (Bogdan, Berg, Sabatti, et al. 2015) and recover sparsity and ordering patterns in the solution (Bogdan, Dupuis, et al. 2022). Another key feature is that the problem is convex, which has implications that we will come to appreciate in Section 3. This latter fact also puts SLOPE apart from other competing penalization methods such as the minimax concave penalty (MCP) (Zhang 2010) and smoothly clipped absolute deviation (SCAD) (Fan and R. Li 2001).

The constraints in the lasso and elastic net are parameterized by one and two parameters, respectively: t in the case of the lasso, and (t, α) in the case of the elastic

[6] By false discovery rate, we mean the fraction of coefficients incorrectly identified as non-zero (false discoveries) as a proportion of the total number of non-zero coefficients (discoveries).



(a) When $\lambda = [1, 1]^\intercal$, the constraint region is the lasso ball, $\|\beta\|_1$.

(b) Here we have $\lambda = [1, 3/4]^\intercal$, but no clustering occurs.

(c) With $\lambda = [1, 1/4]^\intercal$, the solution is now clustered.

(d) When $\lambda = [1, 0]^\intercal$, the constraint is the infinity norm, $\|\beta\|_\infty$.

Figure 8: SLOPE balls (the sorted ℓ_1 norm) for various choices of the penalization weight vector λ. It is the kinks at the boundaries, occurring when $|\beta_1| = |\beta_2|$, that induce clustering. The larger the difference between adjacent values in λ, the stronger the clustering effect becomes.

net. SLOPE, in contrast, is parameterized not only by t but also by the λ vector, which has p elements—one for each feature. Finding an optimal setting for all of those $p + 1$ parameters is a non-trivial task when p is large. As a result, we typically need to re-parameterize the problem. The most common form for this parameterization is the Benjamini–Hochberg sequence, which sets the penalization weights to

$$\lambda_i = \Phi^{-1}(1 - q_i), \qquad q_i = \frac{qi}{2p},$$

where Φ^{-1} is the quantile function of the standard normal distribution and $q \in [0, 1]$. It is this choice that gives SLOPE its false discovery rate control property (Bogdan, Berg, Sabatti, et al. 2015). Note that if we use a linearly decreasing sequence instead, then we would recover OSCAR. And if we use a constant sequence, we recover the lasso.

This reparameterization reduces the number of parameters to just two: (t, q)—the same number as the elastic net. And it means that it is tractable to find optimal settings for the parameters using the methods that we will introduce in the next section.

2.4 Hyperparameter Optimization

An issue that we have so far largely ignored is how to pick good values for t in the case of the lasso, (t, α) in the case of the elastic net, and (t, q) in the case of SLOPE. We call these *hyperparameters* of the problems. Under strong assumptions on our

data, in particular the error term ϵ, it is possible to derive optimal settings of some of these hyperparameters. The problem, however, is that we typically do not know the distribution of the error term. And in high dimensions, estimating it is not easy either.

The alternative, which is the dominating procedure in practice, is to resort to hyperparameter optimization, in which we treat the problem of finding good hyperparameters as an upper-level optimization problem (on top of the optimization problem of finding β).[7] This procedure is often also called *model validation*. This is most commonly done via a grid search in which a grid is constructed across the hyperparameter space. In the lasso case, for instance, it is common to construct a linearly spaced sequence of t values.[8]

In the simplest case, typically called *hold-out validation*, the data set is then split into a training set and a validation set. The lasso is fit for the full t sequence on the training data; in other words, we fit a full lasso path, and an error is then computed on the validation data. At the end, we pick the t value with the best score on the validation data. Often, there is also a separate test data set that is held-out before hyperparameter optimization; at the end, this test set is used to obtain a final unbiased goodness-of-fit measure for the model selected during hyper-optimization.

A more common, although slightly more involved, method is to use k-fold cross-validation (Figure 9), which is similar to hold-out validation, except the data is iteratively split into k folds; the method is run for as many iterations. In the kth iteration, the kth fold is held out as a validation set on which an error is computed after the model has been fit on the remaining $k - 1$ folds. After the last iteration, a cross-validation error is computed by averaging the validation error over all of the validation folds. Typical choices of k are 5 and 10. If $k = n$, then the method is called *leave-one-out cross-validation*. In a sense, cross-validation can be seen as a variance-reduction technique for hold-out validation. The downside, however, is that this comes at a price of increased bias since the validation sets are also used during training. There is also *repeated* k-fold cross validation, which simply repeats the cross-validation procedure for some number of times, each time with different splits.

For the lasso, it is typical to construct a grid of 100 t values, which means that we have to, for instance, fit $100k$ lasso models if we use cross-validation. This can easily become computationally expensive, especially if either or both of n and p are large. In the case of the elastic net and SLOPE, the problem is complicated further since we then have α or q to optimize over as well. As a result, it is vital that there are efficient methods for fitting these models, which is the topic of the following two sections. In Section 3, we discuss the various optimization methods that we can use to solve the lasso, the

[7] In this sense, hyperparameter optimization can be viewed as a bilevel optimization problem.

[8] If we were to use the unconstrained version of the lasso in which the model is parameterized by λ, the sequence is typically *geometrically* spaced instead.

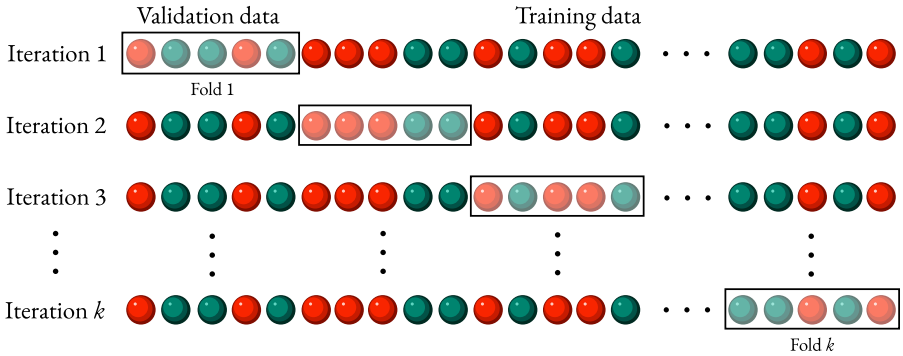


Figure 9: An illustration of k-fold cross-validation. The data is split into k subsets (folds). In the kth iteration, the model is fit to the training data consisting of $k - 1$ folds and then applied to the held-out validation data in fold k, on which an error is computed. A final cross-validation error is then computed by averaging the error across all of the iterations. This figure is an edited version of an illustration by Gufosowa (via Wikimedia Commons, licensed under CC BY-SA 4.0).

elastic net, and SLOPE. And in Section 4, we introduce the concept of *screening rules*, which have been a game changer in the high-dimensional context.

3 Optimization

In the previous section we introduced the statistical models that this thesis will revolve around: the lasso, the elastic net, and SLOPE. They feature many interesting theoretical properties, which we have touched upon briefly, but it is actually not these properties that we will concern ourselves with in this thesis. Instead, we will be interested in the *numerical* aspects of these problems. That is: how do we actually solve them? And, moreover, how do we do this as efficiently as possible?

As we have seen, fitting these models to data is equivalent to solving optimization problems and so far we have assumed that we can solve any of these problems and recover their optima. This assumption is by no means wrong: methods for fitting the lasso, elastic net, and SLOPE are readily and freely available in many programming languages and can be installed and ran by running just a few lines of code. To fit the full lasso path to the diabetes data that we encountered previously (Figure 6), for instance, we need only to call the following `R` code.

```
1 library(lars)
```

```
2
3 data(diabetes)
4 fit <- lars(diabetes$x, diabetes$y, type = "lasso")
```

Behind the scenes, however, the method invoked through these commands actually involve a complicated optimization algorithm into which considerable effort has been put in order to ensure that what you get in `fit` is reliable—and that you get it *fast*.

Throughout the following sections, we will discuss various optimization methods that can be used to solve the lasso, the elastic net, and SLOPE. For simplicity, we will generally focus on the lasso and, in particular, the ordinary lasso (ℓ_1-regularized least-squares regression). In general, however, the methods that we introduce here can be used to solve all of these problems, including the case when the objective that is regularized is part of the family of generalized linear models, which include, for instance, logistic , multinomial, and Poisson regression.

3.1 Direct Methods

The first optimization problem that we encountered in this text was ordinary least-squares regression, which we formally defined in Problem (1). Naively speaking, the solution to this problem is actually relatively straightforward. Letting

$$f(\beta) = \frac{1}{2}\|y - X\beta\|_2^2$$

be the objective function that we want to minimize, we simply set the gradient of it to zero:

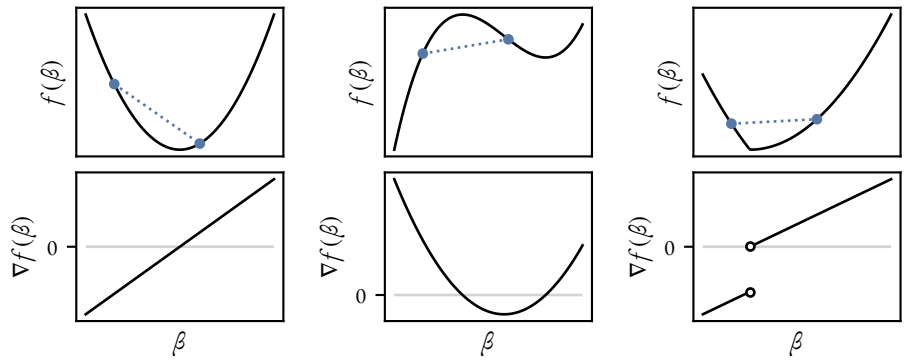
$$\begin{aligned} \nabla f(\beta) = X^\intercal(X\beta - y) = 0 \implies \\ X^\intercal X\beta = X^\intercal y. \end{aligned}$$

This system[9] is called the *normal equations*. Solving the system in β yields the ordinary least squares estimate, which, for a one-dimensional problem is equivalent to locating the "bottom" of the function $f(\beta)$ in Figure 10a. It might be tempting to want to simply invert $X^\intercal X$ here and premultiply by both sides to yield an explicit solution of the form

$$\beta = (X^\intercal X)^{-1} X^\intercal y$$

but this is typically a bad idea since the inverse need not be numerically stable or even exist. A better option is to use a method such as QR factorization and solve the resulting system through forward or backwards elimination, which is both more stable and efficient; all modern software use some variation on this approach.

[9] We have ignored the intercept β_0 here for simplicity, but it could be incorporated easily by prepending a vector of ones to X.



(a) A convex and smooth function. The minimum ocurs when $\nabla f(\beta) = 0$.

(b) A non-convex but smooth function. The derivative offers no information about the global minimum (at $-\infty$).

(c) A convex but non-smooth function. There is a global minimum, but the derivative does not exist at this point.

Figure 10: Three different kinds of functions. We show the objective value $f(\beta)$ and the gradient $\nabla f(\beta)$ for each. In each case, our objective is to find the minimum of the function. Only the first and last are convex (the complete line segment between two points on the function lies above the function) and have a global minimum.

Regardless, however, OLS regression can be solved directly and with accuracy at machine precision. This property is shared by ridge regression in which we can attain a solution simply by adding a diagonal matrix[10] to $\boldsymbol{X}^\intercal \boldsymbol{X}$ and solving as before. The key reason for why this is the case is that OLS regression is a differentiable and quadratic problem, which means that it is *convex* and hence has a global solution (Figure 10a), unlike, for instance, the problem in Figure 10b, which is non-convex (actually a third-degree polynomial) and hence has a local minimum.

All the problems that we have covered so far: ordinary least-squares regression, the lasso, the elastic net, and SLOPE are all convex, which is the class of problems this thesis focuses on. Being convex, however, does not necessarily mean that the problem is easy to solve. The lasso, Problem (2), for instance, is a convex problem, but the involvement of the inequality constraint means that a direct solution is not readily available.

Somewhat remarkably, however, it actually *is* possible to solve the lasso directly, as long as we also solve the full lasso path up to the t that we want. The class of methods that makes this possible are called *homotopy algorithms* since they can be used to solve

[10] This procedure refers to the *unconstrained* form of ridge regression, which have not yet—but will soon—introduce.

the problem for all values they are parameterized by (in this case t). The first homotopy method for the lasso was introduced by Osborne, Presnell, and Turlach (2000a,b) but it is the LARS algorithm (Efron et al. 2004), that we already saw in action at the beginning of the section, which popularized the method for the lasso.

In essence, homotopy methods for the lasso are based on the idea that the model can be solved directly if we know the support of the solution (the identity of the non-zero coefficients). Based on this idea, we start with the empty support at $t = 0$. From this point, it is possible to say which features will become active first and at what t this happens, which makes it possible to then solve the problem (directly) a this value of t. The process is repeated until the entire path has been computed. We give a rough, but slightly more formalized, description of the method in Algorithm 1.

Algorithm 1: A rough outline of the homotopy method for the lasso path. The steps in lines 3 and 4 represent the critical aspect of the algorithm, but are omitted here for brevity. They are not, however, particularly demanding computationally. Instead, the primary costs come from the linear systems that need to be solved at each step of the path.

Input: $\boldsymbol{\beta}^{(0)} \leftarrow \mathbf{0}$, $t \leftarrow 0$, $\mathcal{A} \leftarrow \emptyset$, $i \leftarrow 0$

1 **repeat**
2 $\quad i \leftarrow i + 1$;
3 $\quad t \leftarrow$ next value for which the support changes;
4 $\quad \mathcal{A} \leftarrow$ support at t;
5 $\quad \boldsymbol{\beta}_{\mathcal{A}}^{(i)} \leftarrow \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{|\mathcal{A}|}} f(\boldsymbol{\beta})$;
6 $\quad \boldsymbol{\beta}_{\mathcal{A}^C}^{(i)} \leftarrow \mathbf{0}$;
7 **until** $|\mathcal{A}| = p$;

At the time that these methods were introduced, they offered a remarkable boost in efficiency compared to the original algorithm used by Robert Tibshirani (1996), which consisted of an iterative method based on an algorithm by Lawson and Hanson (1995), which scales badly with p and is altogether inapplicable when $p > n$.

Since the elastic net can be recast as a lasso problem, it also means that the same homotopy methods can be used also in this case. In addition, there now also exists homotopy methods for SLOPE (Dupuis and P. J. C. Tardivel 2023; Nomura 2020), which comes from the SLOPE path sharing the piecewise-linear property of the lasso path (although the SLOPE path is typically more complicated and features more changes in support).

Even if these homotopy methods provide a much-wanted upgrade compared to the original method in the high-dimensional setting, it is nevertheless this domain that

they ultimately struggle to deal with. The root of this problem is that there are at least $\min(n, p)$ kinks (changes in support) along the full lasso path—and in the worst case as many as $(3^p + 1)/2$ such changes (Mairal and Yu 2012). The algorithm has to solve an equivalent number of OLS regression problems, albeit at a complexity much reduced from that of solving the full problem[11], which, in the end, means that the method has found itself outperformed by iterative optimization methods (Friedman, Hastie, and Robert Tibshirani 2010), which we will introduce in the next section.

3.2 Iterative Optimization

In iterative optimization, we start with an initial guess of the solution (usually $\boldsymbol{\beta} = \mathbf{0}$) and then update it step-by-step until we get "close enough" to the optimum[12]. This puts them apart from the direct methods we covered in the previous section, which typically involve solving a system of equations or a similar problem and yield a solution directly and at machine precision.

Gradient Descent

It is probably fair to say that *gradient descent* is the quintessential iterative optimization method. The basic idea is to update the optimization variable ($\boldsymbol{\beta}$) by moving in the direction of the negative gradient of the objective function ($\nabla f(\boldsymbol{\beta})$). For a convex objective, the negative gradient points in the direction of the global minimum, which means that we eventually reach it given appropriate choices of our step sizes (how far we move in the opposite direction of the gradient).

Technically, the idea in gradient descent is to form a second-order Taylor expansion of the objective around the current iterate $\boldsymbol{\beta}'$,

$$f(\boldsymbol{\beta}) \approx f(\boldsymbol{\beta}') + \nabla f(\boldsymbol{\beta}')^\intercal (\boldsymbol{\beta} - \boldsymbol{\beta}') + \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}')^\intercal \nabla^2 f(\boldsymbol{\beta}')(\boldsymbol{\beta} - \boldsymbol{\beta}'),$$

replace the Hessian matrix $\nabla^2 f(\boldsymbol{\beta}')$ with $\frac{1}{\tau}\boldsymbol{I}$ in this approximation, yielding

$$f'(\boldsymbol{\beta}; \boldsymbol{\beta}') = f(\boldsymbol{\beta}') + \nabla f(\boldsymbol{\beta}')^T (\boldsymbol{\beta} - \boldsymbol{\beta}') + \frac{1}{2\tau}\|\boldsymbol{\beta} - \boldsymbol{\beta}'\|_2^2,$$

and then, finally, solve for $\boldsymbol{\beta}$, which gives the gradient descent update

$$\boldsymbol{\beta}^+ = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min}\, f'(\boldsymbol{\beta}; \boldsymbol{\beta}') = \boldsymbol{\beta}' - \tau \nabla f(\boldsymbol{\beta}').$$

[11] LARS, for instance, incrementally update a Cholesky factorization of the Hessian matrix $\boldsymbol{X}^\intercal \boldsymbol{X}$.

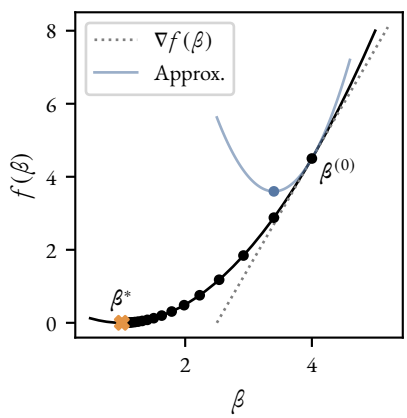
[12] As you might expect, defining "close enough" is not at all a trivial matter.

Observe that the τ we used when we replaced the Hessian with a diagonal matrix is the step size in the gradient descent algorithm. This is the key parameter in the algorithm, which controls how far we move in the direction of the negative gradient.

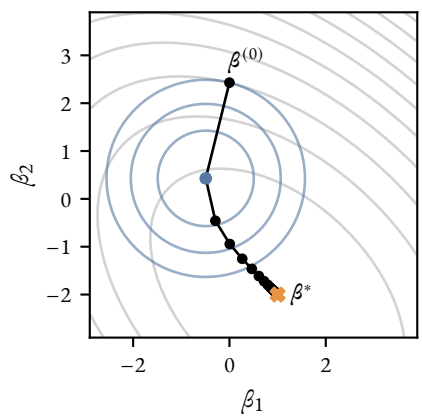
Algorithm 2: Gradient descent with fixed step size. An intercept can be added by either prepending a vector of ones to $\boldsymbol{X}$ or adding a separate update step where $\boldsymbol{\beta}$ is held fixed.

Input: $\boldsymbol{\beta} \leftarrow \mathbf{0}$, $\tau \leftarrow 1/\|\boldsymbol{X}\|_2^2$
repeat
  $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \tau \nabla f(\boldsymbol{\beta})$;
until *convergence*;

In Figure 11, we show how gradient descent works for simple problems in one and two features.



(a) A one-dimensional problem. The black curve shows the value of the objective $f(\beta)$.



(b) A two-dimensional problem. The grey curves are the level curves of the OLS regression objective $f(\beta)$.

Figure 11: Gradient descent in one and two dimensions. In each case the algorithm starts at $\boldsymbol{\beta}^{(0)}$ and then proceeds towards the optimum $\boldsymbol{\beta}^*$ (✖). The quadratic approximation of the first step is drawn in blue lines (and its optimum marked by a blue dot).

Projected Gradient Descent

The problem for us, however, is that the lasso, the elastic net, and SLOPE—the problems we are mainly concerned with—have inequality constraints, which gradient descent cannot handle directly. A natural alternative, however, exists in the form of *projected* gradient descent. The method consist of simply taking a gradient descent step, as in Algorithm 2, and then projecting the result onto the feasible region (if the update after the gradient step is infeasible). The update is then

$$\boldsymbol{\beta} \leftarrow \text{proj}_C\left(\boldsymbol{\beta} - \tau \nabla f(\boldsymbol{\beta})\right)$$

where

$$\text{proj}_C(\boldsymbol{u}) = \underset{v \in C}{\arg\min} \|\boldsymbol{v} - \boldsymbol{u}\|_2 \tag{5}$$

is the projection operator that projects $\boldsymbol{u}$ onto the feasible region C. For the lasso, for instance, C is the ℓ_1 norm ball (Figure 5), while for SLOPE it is the sorted ℓ_1 norm ball (Figure 8). The method is outlined in Algorithm 3 and is identical to the gradient descent algorithm (Algorithm 2), except that we have now wrapped a projection operator around the gradient descent update.

Algorithm 3: Projected gradient descent. The projection operator $\text{proj}_C(\cdot)$ is defined in Equation (5).

Input: $\boldsymbol{\beta} \leftarrow \mathbf{0}$, $\tau \leftarrow 1/\|\boldsymbol{X}\|_2^2$

1 **repeat**

2 $\quad \boldsymbol{\beta} \leftarrow \text{proj}_C\left(\boldsymbol{\beta} - \tau \nabla f(\boldsymbol{\beta})\right)$;

3 **until** *convergence*;

Figure 12 shows the method in action for a two-dimensional lasso problem. Note how the gradient step (dotted lines) takes the algorithm outside the feasible region and that the projection step (solid lines) then moves the solution back to the feasible region.

The efficiency of projected gradient descent hinges on the efficiency with which the projection can be computed. Thankfully, there exists efficient projections for both the ℓ_1 (Duchi et al. 2008) and sorted ℓ_1 norms (Davis 2015; Q. Li and X. Li 2021; Perez et al. 2022; Zeng and Figueiredo 2015), which means that the method is quite efficient for both the lasso and SLOPE.

The Subgradient Method

Projected gradient descent attacks the our optimization problems by tackling the inequality constraints directly, but the, by far, most popular approach for solving these

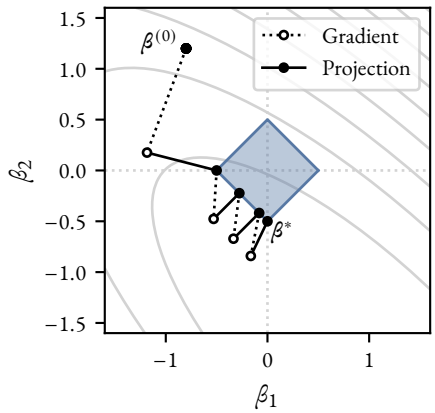


Figure 12: Projected gradient descent for a two-dimensional lasso problem. The gray curves are the level curves of the ordinary least-squares objective $f(\beta)$ and the blue diamond shape is the ℓ_1 norm constraint. The algorithm starts at β_0 and then proceeds towards the solution β^*. Each step consists of a gradient step (dashed lines) and a projection step (solid lines).

problems actually takes a different route by transforming the constrained problem into an unconstrained one. The idea is that we can achieve an equivalent regularization effect by, instead of constraining the solution directly, penalizing the coefficients via the objective function.

In other words, we turn the problem

$$\begin{aligned} &\text{minimize} && g(\beta), \\ &\text{subject to} && h(\beta) \leq t, \end{aligned}$$

into

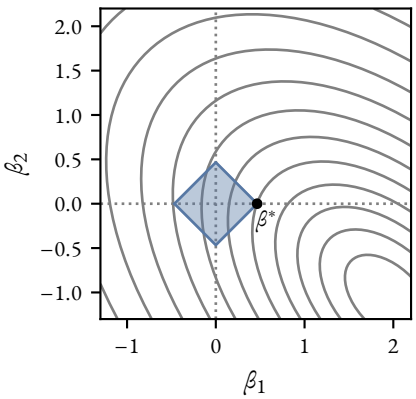
$$\text{minimize} \quad f(\beta) = g(\beta) + h(\beta).$$

In the case of SLOPE, for instance, the unconstrained version of Problem (4) is then

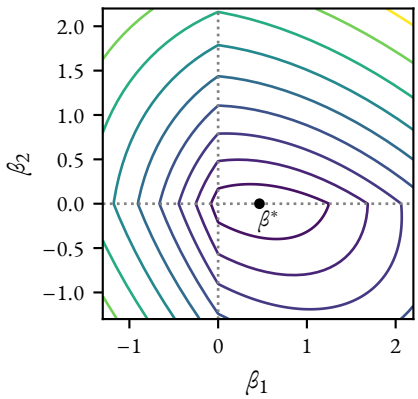
$$\text{minimize} \quad \frac{1}{2}\|y - X\beta\| + \sum_{j=1}^{p} \lambda_j |\beta_{(j)}|$$

You may wonder what good this did us, but the key is that the we have now gotten rid of the constraint, which means that we can focus on minimizing the objective

directly. We can see how the two optimization problems compare for an equivalent setting of λ (in the case of the unconstrained problem) and t (in the case of the constrained problem) for the lasso in Figure 13.



(a) The constrained form of the lasso. The level curves show the ordinary least-squares objective.



(b) The unconstrained form of the lasso. Here, the level curves show the unconstrained optimization objective.

Figure 13: Constrained and unconstrained versions of the lasso. The equivalence between the problems is obtained by setting $t = \|\beta\|_1$ after solving the unconstrained problem to convergence for a given λ.

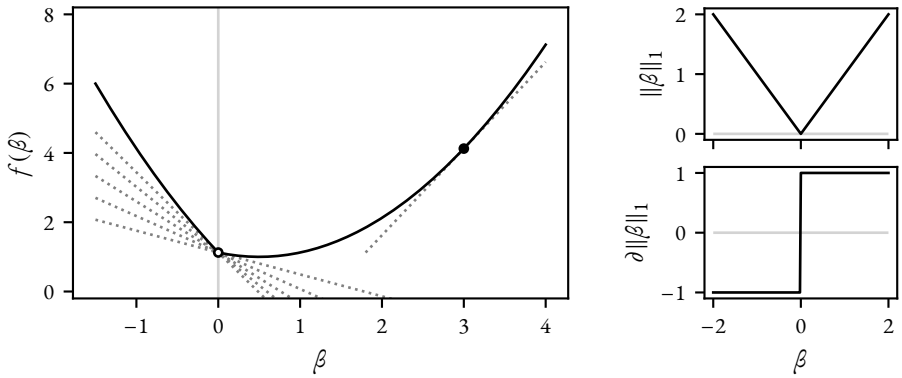
The bad news is that the new objective is no longer differentiable, which means that we cannot, for example, use gradient descent with the new formulation either. The good news, however, is that there is a generalization of the derivative that can be used in the non-differentiable case, namely the *subdifferential*.

The subdifferential of a function f at a point β is the set of all *subgradients* of f at that point. This means that the subdifferential is a set, rather than a single vector. A subgradient at a point β is a vector g such that

$$f(\beta') \geq f(\beta) + g^\intercal(\beta' - \beta).$$

In plain terms, this simply means that the linear approximation of the function at β always underestimates the function. If the function is differentiable, then the subdifferential is a singleton, containing only the gradient of the function. In Figure 14a, we show some of the subgradients of a one-dimensional lasso problem.

The existence of subgradients presents us with an intuitive solution to the problem of minimizing non-differentiable functions: use gradient descent, but replace the



(a) Subgradients of a one-dimensional lasso problem. At $\beta = 3$, there is just a single subgradient: the (ordinary) gradient, but at $\beta = 0$, there are multiple subgradients.

(b) The ℓ_1 norm (absolute value) of the coefficient β and its subgradient $\partial\|\beta\|_1$.

Figure 14: Subgradients of the lasso problem.

gradient with the subgradient, and take a step in its direction. We outline the algorithm is Algorithm 4.

Algorithm 4: The subgradient method. Note that g can be *any* subgradient of the subdifferential $\partial f(\beta)$. Special schemes for the step size τ exist but we omit them here since we are not interesting in studying the method in depth.

Input: $\beta \leftarrow \mathbf{0}$, $\tau > 0$
repeat
  $g \leftarrow \partial f(\beta)$;
  $\beta \leftarrow \beta - \tau g$;
until *convergence*;

The attractiveness of the subgradient method is that it is general and works for a large class of problems, including the lasso, the elastic net, and SLOPE. The problem is that it converges slowly and is therefore never used in practice for these problems.

Proximal Gradient Descent

Fortunately, we have structure in this problem that we have yet to explot fully, namely the fact that although h is not differentiable, h *is*. Based on this, one idea is to simply

leave h alone and minimize the quadratic expansion of g.

$$\arg\min_{\beta} \left(g(\beta') + \nabla g(\beta')^\intercal (\beta - \beta') + \frac{1}{2\tau}\|\beta - \beta'\|_2^2 + h(\beta) \right)$$
$$= \arg\min_{\beta} \left(\frac{1}{2\tau}\|\beta - \underbrace{(\beta' - \tau \nabla g(\beta'))}_{\text{Gradient update}}\|_2^2 + h(\beta) \right).$$

Let us call the function that solves this problem the *proximal operator* of h, denoted $\text{prox}_{h,\tau}$, and define it as

$$\text{prox}_{h,\tau}(\boldsymbol{u}) = \arg\min_{\boldsymbol{v}} \left(\frac{1}{2\tau}\|\boldsymbol{v} - \boldsymbol{u}\|_2^2 + h(\boldsymbol{v}) \right).$$

The proximal operator is a generalization of the projection operator that we introduced earlier, and we can obtain the latter by using the indicator function of the feasible region as the function h. We outline the basic version of the proximal gradient descent algorithm in Algorithm 5.

Algorithm 5: Proximal gradient descent. Note that the gradient is taken with respect to g and not f (for which it does not exist).

Input: $\beta \leftarrow \mathbf{0}$, $\tau \leftarrow 1/\|\boldsymbol{X}\|_2^2$

repeat

  $\beta \leftarrow \text{prox}_{h,\tau}\left(\beta - \tau \nabla g(\beta)\right)$;

until *convergence*;

The purpose of doing this is that the proximal operator has an explicit, and often efficient, form for all the problems we are interested in solving here. For the lasso, for instance, the proximal operator is

$$\text{prox}_{h,\tau}(\boldsymbol{u}) = \arg\min_{\boldsymbol{v}} \left(\frac{1}{2\tau}\|\boldsymbol{v} - \boldsymbol{u}\|_2^2 + \lambda\|\boldsymbol{v}\|_1 \right) = \text{S}_{t\lambda}(\boldsymbol{u}),$$

where

$$\text{S}_\lambda(\boldsymbol{u})_i = \text{sign}(u_i)\max(|u_i| - \lambda, 0),$$

which is called the *soft-thresholding operator* (Donoho and Johnstone 1995) and is shown in Figure 15. Using proximal gradient descent for the lasso is also known as the iterative shrinkage-thresholding algorithm (ISTA) (Beck and Teboulle 2009).

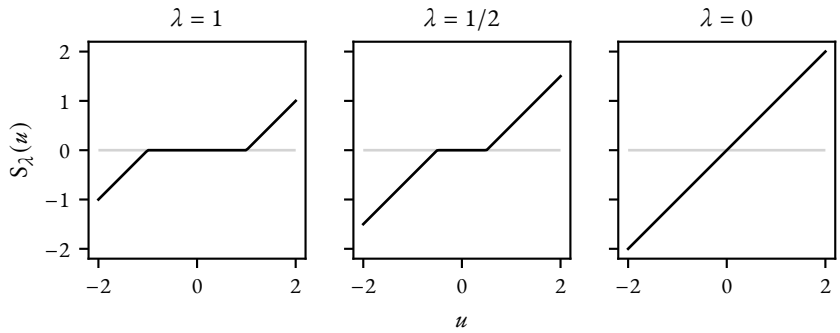


Figure 15: The soft-thresholding operator: the proximal operator in the case of the lasso. Here we show it for three different values of λ. For $\lambda = 0$, there is no thresholding (penalization), and the operator becomes the identity function. For all other values, the operator shrinks its input toward zero. For $|u| \leq \lambda$, the output is exactly zero.

Proximal gradient descent has a long history. Rockafellar (1970) was responsible for much of the early work. Nesterov (1983) introduced the first accelerated version of proximal gradient descent algorithms, which where later improved upon by, for instance, Beck and Teboulle (2009) in the form om the fast iterative shrinking-thresholding algorithm (FISTA). Other improvements include line searches and improved step size settings.

At this point you may wonder what we have gained compared to the projected gradient method. First, we have obtained an algorithm that is more general. Indeed, as Bogdan, Berg, Sabatti, et al. (2015) showed, there is an efficient algorithm for computing the proximal operator even in the case of the sorted ℓ_1 norm. Furthermore, we have also gained slightly in computational efficiency. In the lasso case, for instance, the soft-thresholding operator has $O(p)$ complexity whereas the projection onto the ℓ_1 ball has $O(p \log p)$ complexity. Please see Figure 16 for a small example on what this can amount to in practice.

In conclusion, proximal gradient descent is a simple algorithm that is easy to implement and has strong convergence guarantees, but as we shall see next, is typically outperformed by a more naive method: coordinate descent.

Proximal Coordinate Descent

Coordinate descent is a simple algorithm: update one coefficient (coordinate) at a time by finding the minimizer with respect to that coefficient. In other words, we solve a one-dimensional problem at each iteration of the algorithm. The algorithm extends

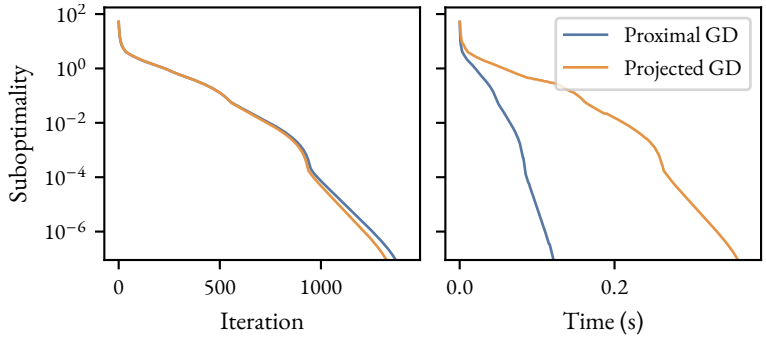


Figure 16: Comparison between projected and proximal gradient descent for an equivalent lasso problem. Per-iteration, the methods both perform almost on par with one another. In terms of wall-clock time, however, the proximal method is slightly more efficient.

naturally to the case when we have a composite objective of the form $f = h + g$, as in the case of proximal gradient descent, provided that h is differentiable and g is convex and *separable* in $\boldsymbol{\beta}$. The basic outline of the algorithm is given in Algorithm 6.

Algorithm 6: Proximal coordinate descent. Note that the implementation given here is designed for illustration; many improvements can be made that are critical to the practical performance of the algorithm.

Input: $\boldsymbol{\beta} \leftarrow \mathbf{0}$
1 **repeat**
2 Pick $j \in \{1, 2, \dots, p\}$;
3 $\tau \leftarrow 1/\|\boldsymbol{x}_j\|_2^2$;
4 $\beta_j \leftarrow \text{prox}_{h,\tau}\left(\beta_j - \tau \nabla g(\boldsymbol{\beta})_j\right)$;
5 **until** *convergence*;

For the lasso, the coordinate descent update in Line 4 (Algorithm 6) is particularly cheap and amounts only to soft-thresholding of the partial residual. In Figure 17, we show a simple example of how coordinate descent works for a two-dimensional lasso problem.

There are varies strategies for picking the coefficient to optimize over in step three. The simplest and most common alternative is to cycle for through the coefficients one by one, which is called *cyclic* coordinate descent. Another common alternative is to

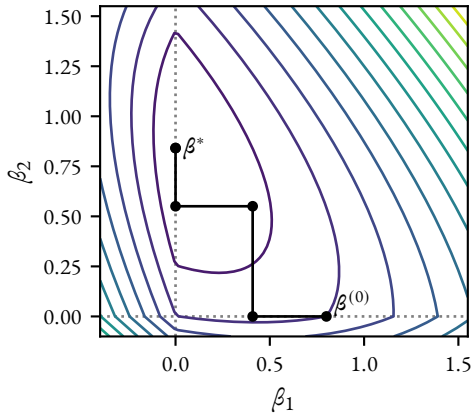


Figure 17: Cyclic coordinate descent for a lasso problem in two dimensions. At each step, we minimize only in the direction of one of he coefficients (along either the x or the y axis). Since g is the least-squares criterion, the step always takes us to the minimum in that direction.

pick the coefficient at random, which makes for an algorithm that's easier to analyze than cyclic coordinate descent and somewhat more robust, but which is also usually slightly slower.

At face-value it may seem somewhat surprising that this simple strategy should outperform proximal gradient descent, which considers all features simultaneously, particularly in light of the fact that proximal gradient descent sports better convergence rates (Wright 2015). Yet, barring a few exceptions, this is precisely the case (Figure 18).

The counterintuitive nature of this result may explain why coordinate descent algorithms have largely been ignored throughout the history of optimization. For the lasso, there were several papers on coordinate descent-like algorithms (Daubechies, Defrise, and De Mol 2004; Fu 1998; Shevade and S. S. Keerthi 2003), yet all of these initially received scant attention. The popularization of coordinate descent for the lasso took off with the work of Friedman, Hastie, Höfling, et al. (2007), which was later implemented into the software package `glmnet` (Friedman, Hastie, and Robert Tibshirani 2010).

The main reason for why coordinate descent works so well for the lasso, the elastic net, and many related problems is related to the size of the steps that the algorithms can take (while still guaranteeing convergence). Proximal gradient descent is limited by a step size of $1/\|\boldsymbol{X}\|_2^2$, whereas coordinate descent can take steps of size $1/\|\boldsymbol{x}_j\|_2^2$, which for much real data are decidedly larger for many of the features and hence promote

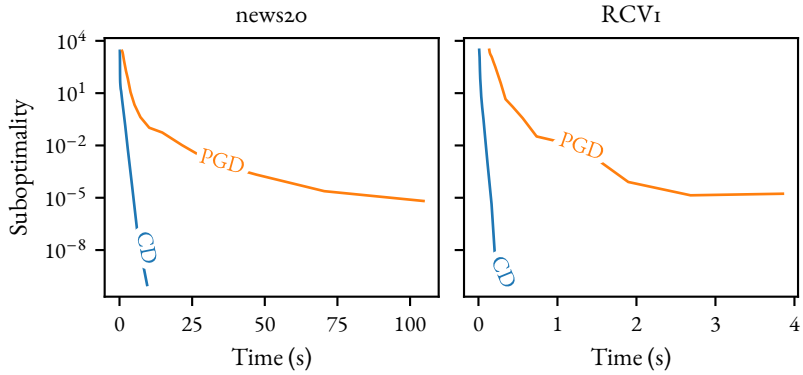


Figure 18: Proximal coordinate descent versus accelerated proximal gradient descent (FISTA) for two data sets: `news20` (S. Sathiya Keerthi and DeCoste 2005) and `RCV1` (Lewis et al. 2004). The dimensions of the data are $n = 199\,961$, $p = 355\,191$ for `news20` and $n = 20\,242$, $p = 47\,236$ for `RCV1`. The response in both cases is binary and we fit a standard lasso model In both cases we normalize with maximum–absolute value scaling. The comparisons were made using the `benchopt` package (Moreau et al. 2022).

faster convergence.

One problem with coordinate descent, however, is that it requires the objective to be separable in the optimization variable (β). This is not the case for SLOPE, which makes the method inapplicable for this problem.

We have introduced many optimization methods in this section, yet have only scratched the surface in the field. Several methods have been omitted, such as the alternating direction method of multipliers (ADMM) (Boyd et al. 2010), proximal Newton (Lee, Sun, and M. A. Saunders 2014), interior-point methods (Kim et al. 2007), and stochastic gradient descent (Bottou 2010; Robbins and Monro 1951) and its many derivatives.

We have also omitted several details regarding the implementation of the algorithms, such as the step size selection and convergence criteria. Yet, while these are important (and interesting) aspects of the algorithms, we omit them here for brevity and provide all the necessary details in the papers themselves.

4 Screening Rules

Screening rules are a remarkably efficient method for speeding up the optimization of sparse regression methods. They are based on the following reasoning:

1. We know that the solution is going to be sparse, especially if $p \gg n$[13].
2. We also know something about the importance of the features, even before fitting the model, since we for instance (as in Figure 1) can compute the correlation between the features and the response. In addition to this, we are also typically interested in solving for a range of regularization parameters, which means that the problem we are currently trying to solve is likely related to a problem we have already solved.
3. Therefore, we might be better of by only considering a subset of the features when solving the problem. And if this subset is small and selecting it is cheap, then we should be able to save a lot of time.

This intuition turns out to be correct, and screening rules have been turned out to be extremely effective in practice, particularly in the high-dimensional domain.

The first screening rule for the lasso, SAFE (SAfe Feature Elimination), was introduced by El Ghaoui, Viallon, and Rabbani (2010). In essence, the authors showed that it was possible to confine the solution for a given λ to a region of the feature space, and feature-by-feature ascertain whether it is possible for the feature be selected in the solution.

In practice, this means that if you have a data set of, say, one million features (p) and one thousand observations (n) and want to fit the lasso for some value of λ, then there's a good chance that you may just have to fit a problem with a few hundred features, while at the same time guaranteeing that you will reach the same solution as the one-million-feature problem would. In addition, the cost of running the screening rule is negligible next to solving the problem (even the reduced problem). The net result is a remarkable gain in computational performance. It is not in our experience rare to see speedups of several orders of magnitude, for instance reducing the time taking to fit this size of model from hours to minutes (or even less).

The screening rule introduced by El Ghaoui, Viallon, and Rabbani (2010) is a *safe* screening rule, meaning that all of the features discarded by the rule are guaranteed to be absent at the optimum. As it turns out, however, this requirement on safety in general leads to rules that are overly conservative. This was made clear in a paper by Robert Tibshirani, Bien, et al. (2012), in which they introduced the *strong screening rule for the lasso*. This rule uses an initial screening test that is *heuristic* (as opposed to

[13] Recall, for instance, that the lasso can only select $\min(n, p)$ features.

safe), which means that it may discard features that actually are part of the solution. This necessitates a check of the optimality conditions after solving the reduced problem and, in case any features were incorrectly discarded, a refitting of the model with these features icluded. As the authors showed, however, these *violations* of the screening rule are so rare in practice that there is often no need to refit. And if there is, the estimate from the reduced problem is often so close to the real problem that only a negligle number of extra iterations of the optimization algorithm are needed.

It is important to highlight that the nomenclature for screening rules is quite misleading[14]. Even if the screening performed by the strong rule is *not* safe, the actual screening rule method *is*, which stems directly from the fact that all of these methods include checks of the optimality conditions in order to safeguard against discarding features that are part of the solution[15]. In other words, even though we call some rules *safe* and some *heuristic*, they are in fact *all* safe as they are implemented in practice.

Robert Tibshirani, Bien, et al. (2012) also introduced the notion of *sequential* screening rules, meaning that screening is performed along the regularization (lasso) path with the solution at the current step on the path as the starting point for the screening rule, which significantly impoves the effectiveness of the rule. One problem with both SAFE and the strong rule, however, is that their effectiveness decrease when features in the design are heavily correlated—the problem is particularly severe in SAFE. This lead the authors to conclude, which they also empirically demonstrated, that the most efficient screening method (at least in the sequential case) is to begin with the set of features that have so far *ever* been active along the path: $\mathcal{E}$, and solve the problem for this set first. After reaching a solution for this set, the optimality checks are then performed on the features present in the strong rule set $\mathcal{S}$, but not in $\mathcal{E}$ ($\mathcal{S} \setminus \mathcal{E}$). If there are any violations, which is often the case since no screening was performed, these features are entered into $\mathcal{E}$. The problem is then solved, again, but this time for the updated $\mathcal{E}$. This procedure is then repeated until no violations are found among $\mathcal{S} \setminus \mathcal{E}$. Finally, when this happens, the optimality conditions are checked for the entire set of features, and if no violations are found, the solution is deemed to be found. If there are violations, the procedure repeats with these features added to $\mathcal{E}$.

Since these two papers were published, there has been a slew of research around screening rules for the lasso as well as other sparse regression problems.

[14] We stress this point since we have (repeatedly) found this fact to stump the reviewers of our papers.

[15] There *are* screening rules that actually are unsafe, but they are used for an altogether different purpose in which one is interested not just in screening, but also changing the model (for instance the lasso) itself.

4.1 Screening Rules and the Correlation Vector

To better understand how screening rules work, it is helpful to consider the optimality conditions of our problems. For the unconstrained version of our problems, they are

$$\mathbf{0} \in \nabla g(\boldsymbol{\beta}) + \lambda \boldsymbol{s} h(\boldsymbol{\beta}),$$

where $\boldsymbol{s}$ is a subgradient of the penalty term h. Letting $\boldsymbol{c} = -\nabla g(\boldsymbol{\beta})$, we can rewrite this as

$$\boldsymbol{c} \in \lambda \boldsymbol{s} h(\boldsymbol{\beta}).$$

We will call $\boldsymbol{c}$ the *correlation vector*, which is simply the negative gradient of $g(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$.

For illustrative purposes, we will focus on the lasso here, for which the $\partial h(\boldsymbol{\beta})$ is the subdifferential of the ℓ_1 norm, defined as

$$s_j \in \begin{cases} \{\text{sign}(\hat{\beta}_j)\} & \text{if } \hat{\beta}_j \neq 0, \\ [-1, 1] & \text{otherwise,} \end{cases}$$

which we previously visualized in Figure 14b. One consequence of this is that, if

$$|\nabla g(\boldsymbol{\beta})_j| < \lambda,$$

then it must also hold that $\beta_j^* = 0$. In other words, if the absolute value of the gradient with respect to feature j is smaller than our level of regularization, then the feature must be absent from the solution. Similarly, this also means that if $\beta_j^* \neq 0$, then its gradient will be $\pm\lambda$. We can observe this behavior in Figure 19, where we have plotted the lasso path for the diabetes data set that we have encountered before. Note that we start at the left. At this point only a single feature is active. Then, as we lower λ (move from right-to-left in the plot), the features become active one-by-one. And at exactly the point where they do so, they also join either of the lines $c = \pm\lambda$ and as long as they remain active, they will stay on these lines.

So, if we knew, before fitting, that $|c_j| < \lambda$, then we could safely discard feature j from the problem and still solve it as if it were always there. The problem, however, is that we do not have access to the correlation vector before fitting the model.

What we can do instead, however, is to estimate the correlation vector and use that in the place of the true value. Let us say that we are at step k on the path and have computed the solution at this step; then we also know $\boldsymbol{c}$ at this step. Next, we want to predict what $\boldsymbol{c}^{(k+1)}$ will be in order to screen features for this (upcoming) step. One rather conservative approach for this is to assume that gradient of the correlation vector is bounded by one, in other words, we introduce the approximation

$$\hat{c}_j^{(k+1)} = c_j^{(k)} + \text{sign}\left(c_j^{(k)}\right)(\lambda_k - \lambda_{k+1}).$$

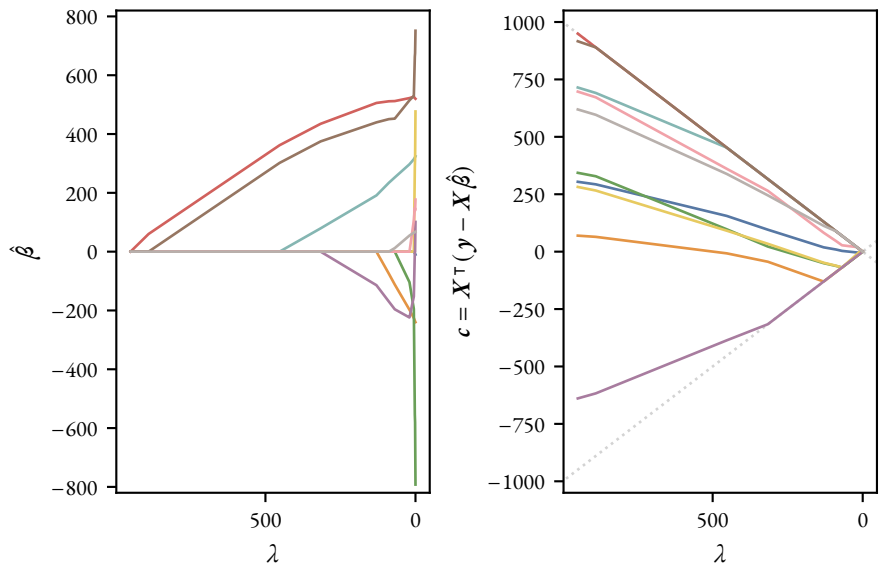


Figure 19: The lasso path for the diabetes data. This time, we have also plotted the correlation (negative gradients) along the path. In order for a feature to become active from one point on the path (one value of λ) to the next, its correlation c_j has to reach the $\lambda = \pm c$ line. Note that the x-axis is flipped to illustrate the fact that we typically proceed from a large λ towards a small one.

Using this approximation is in fact exactly the strong rule for the lasso. In Figure 20, we illustrate how this approximation looks like in practice and how it is applied in the case of the diabetes data.

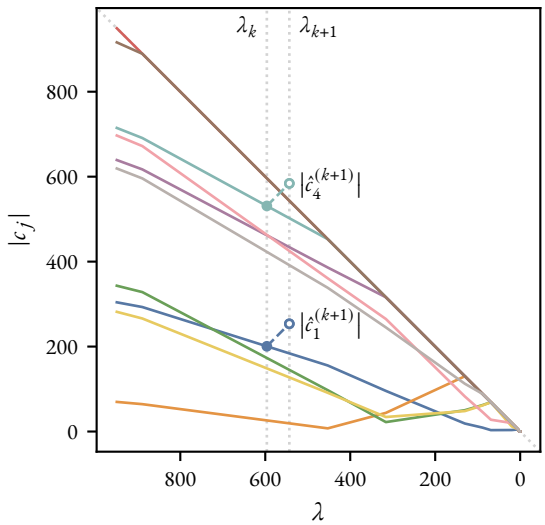


Figure 20: The sequential strong rule for the lasso. We are at step k and want to screen features for step $k+1$. We show the screening rule for two features here. The filled circles show $|c_j|$ for each feature and the open circles show the strong rule estimate $\hat{c}_j$. For feature 1, the approximation $|\hat{c}_1|$ does not reach λ in time for the step step, so the rule discards it. For the fourth feature, however, the estimate $|\hat{c}_4|$ is larger than λ, so we cannot discard feature 4.

5 Normalization

One important aspect of regularized models is that they are generally sensitive to the scale of the features in the problem. This is the case for the lasso, the elastic net, and SLOPE, since they penalize the magnitude of the coefficients. And it puts them apart from ordinary least squares and best-subset selection, which we introduced earlier.

For a simple example of this behavior, assume that we have a data set that is generated from a normal distribution in the following manner:

$$\boldsymbol{X} \sim \text{Normal}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}4 & 0\\0 & 1\end{bmatrix}\right), \qquad \boldsymbol{\beta} = \begin{bmatrix}1/2\\1\end{bmatrix}, \qquad \boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

with $\varepsilon \sim \text{Normal}(0, 1)$, identically and independently for each $i = 1, 2, \ldots, n$.

In other words, we have generated our features from two normally distributed variables, both with mean zero and standard deviation 2 and 4, respectively, and with no correlation between them. We have constructed our problem such that the effects on the response for the two features are equivalent, in the sense that a change of one standard deviation in either feature will result in a change of one in the response.

Since the errors are generated according to the assumptions of the ordinary least squares model, we will recover the true coefficients in expectance. And when we compute the standardized coefficients from our estimates, according to

$$\hat{\beta}_{\text{std}} = s_j \hat{\beta}_j,$$

where s_j is the standard deviation of the feature $\boldsymbol{x}_j$, the coefficients are both 1 (for ordinary least-squares regression).

If we fit the lasso to this data, however, we will find that the feature with the larger standard deviation will be penalized *less* than the other (Table 2), which we see from looking at the standardized coefficients. The same effect occurs in ridge regression.

Table 2: The effect of regularization on the regression coefficients and standardized versions thereof.

Model	$\hat{\boldsymbol{\beta}}$	$\hat{\boldsymbol{\beta}}_{\text{std}}$
OLS	$\begin{bmatrix}0.50 & 1.00\end{bmatrix}^\intercal$	$\begin{bmatrix}1.00 & 1.00\end{bmatrix}^\intercal$
Lasso	$\begin{bmatrix}0.38 & 0.50\end{bmatrix}^\intercal$	$\begin{bmatrix}0.75 & 0.50\end{bmatrix}^\intercal$
Ridge	$\begin{bmatrix}0.47 & 0.78\end{bmatrix}^\intercal$	$\begin{bmatrix}0.93 & 0.78\end{bmatrix}^\intercal$

To tackle this result, the standard approach is to *normalize* the design matrix $\boldsymbol{X}$ before fitting the model by centering (subtracting a value, for instance its mean) and scaling each feature (by dividing with a value, for instance its standard deviation).[16] The goal is to place the features on the same scale. For features that are normally distributed, this is relatively straightforward to do. You simply need to scale with the standard deviation. But when the distribution of the features is different, for instance when whey are binary $x_{ij} \in \{0, 1\}$, the situation is more complicated.

Unfortunately, there is little literature to fall back on in this case; in fact, this holds in general for the case of normalization. Typically, researchers and practioners instead rely on "standard practice", but this differs from field to field. Two common types

[16] Afterwards, the coefficients are typically returned to their original scale.

of normalization is *standardization*, which is shifting by the mean and scaling by the standard deviation[17] of each feature, and *max–abs* scaling, which does no shifting but scales with the maximum absolute value of each feature. The first type, standardization, is common in the statistical literature, whilst the second type, max–abs normalization, is common in machine learning, particularly when dealing with sparse data. As we can observe in Figure 21, however, the two methods generate starkly different results for the estimated coefficients in many datasets.

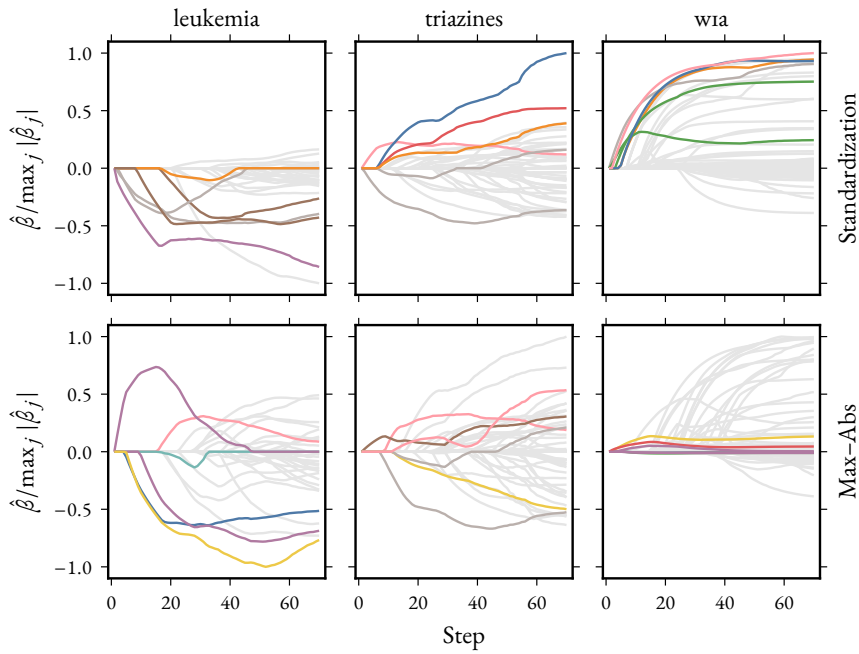


Figure 21: Lasso paths for some real data sets using two types of normalization (standardization and maximum–absolute value scaling (max–abs)). We have fit the lasso path to three different data sets: `leukemia` (Golub et al. 1999), `triazines` (R. King 2024), and `w1a` (Platt 1998). For each pair of plots per data set, we have colored the coefficients if they were among the first ten coefficients to become active in under either of the two types of normalization schemes. We see that the baths differ with regards to the size as well as the signs of the coefficients, and that, in addition, the coefficients to become active first differ between the normalization types.

[17] The formula without Bessel's correction is typically used.

6 Summary of the Papers

6.1 Paper I

In Section 4, we demonstrated the considerable effect that screening rules have had on sparse regression problems in the large-p setting. As a result, the papers by El Ghaoui, Viallon, and Rabbani (2010) and Robert Tibshirani, Bien, et al. (2012) led to a surge in interest for screening rules for the lasso and its derivatives, including the elastic net. For SLOPE, however, there existed no screening rule until our work in this paper, in which we introduced the *strong screening rule for SLOPE* (Larsson, Bogdan, and Wallin 2020).

A key contribution in our paper is the development of a practical characterization of the subdifferential for the sorted ℓ_1 norm, from which we derive an effective algorithm for implementing the screening rule in practice. Just as for the lasso, we demonstrate remarkable boosts in performance for fitting SLOPE (Table 3). The net result is a considerable extension in reach for SLOPE in the high-dimensional setting.

Table 3: Benchmarks measuring wall-clock time for four data sets fit with different models using either the strong screening rule or no rule.

				Time (s)	
Dataset	Model	n	p	No screening	Screening
dorothea	logistic	800	88 119	914	14
e2006-tfidf	least squares	3308	150 358	43 353	4944
news20	multinomial	1000	62 061	5485	517
physician	poisson	4406	25	34	34

6.2 Paper II

As we saw in Section 4, screening rules are particularly effective when they are sequential, that is, operate along the lasso, elastic net, or SLOPE path. But another possibility that had previously not been explored is the idea of screening not only for the next step on the path, but for all of the remaining steps as well. This is the idea behind *look-ahead screening rules*, which we introduce in this short paper (Larsson 2021). We employ this idea for the gap-safe screening rule, which, as the name suggests, is a safe screening rule. This means that if a feature is screened out, it is guaranteed to be zero in the solution.

In Figure 22, we illustrate the effectiveness of look-ahead screening for the lasso path at the first step for the `leukemia` data set. What the plot indicates is that many

features can be safely exempt from model fitting for a range of steps on the path, which means that we only need to re-screen them at the first grey square for each given feature.

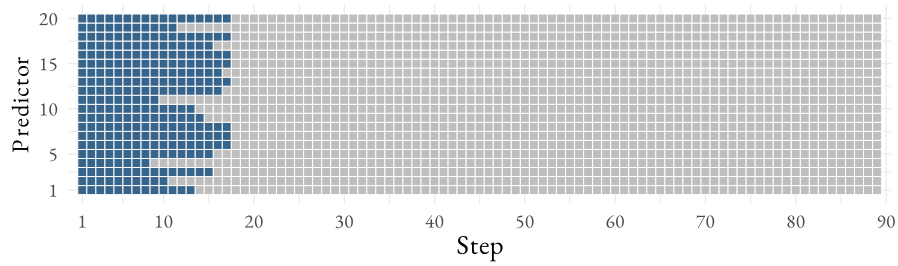


Figure 22: Look-ahead screening for the lasso at the first step on the path for the `leukemia` data set. The plot shows a random sample of 20 features. Blue squares indicate that the corresponding feature can be discarded for those steps on the lasso path.

In the paper, we show that this effect has sizeable consequences for the computation time requires for fitting the full path.

6.3 Paper III

Even though the strong rule for the lasso is highly effective in general, there is one area in which it struggles, namely, when features are highly correlated. Robert Tibshirani, Bien, et al. (2012) in fact noted this themselves and forwarded it as the main motivation for using the working-set strategy (where the model is initially fit using the ever-active set, rather than the strong set).

The reason for this is that the strong rule, and every other screening rules we know of, ignores information about the gradient of the correlation vector $\boldsymbol{c}$, even though it contains useful information about the structure of the path. Looking at Figure 20, for instance, we see that the strong rule bound is indifferent to the slopes of the correlation vectors. This is the motivation for the *Hessian screening rule* that we introduce in the third paper of the thesis (Larsson and Wallin 2022). The name stems from the fact that we use second-order information about the optimization problem, which involves the Hessian matrix $\boldsymbol{X}^\intercal \boldsymbol{X}$. The rule offers a better estimate of the correlation vector (Figure 23), which in practice leads to better screening performance.

The screening method manages to be effective, particularly when h is the least-squares objective, because of efficient updates of the Hessian matrix and its inverse. And as a bonus, the availability of these quantities also allow for adaptively tailoring the λ sequence in order to better approximate the true lasso path.

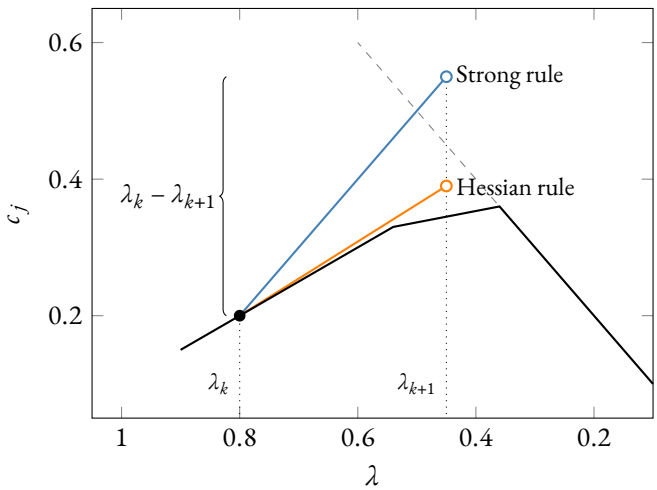


Figure 23: An illustration of the strong and Hessian screening rules for a lasso problem. We are at λ_k and are looking to screen features for λ_{k+1}. The black solid line shows the path of the correlation vector for the jth feature, which is inactive (its coefficient is zero) until the point where it joins the dashed line. In blue and orange colors, we show the predicted values for c_j between λ_k and λ_{k+1}. For the strong rule, the predicted value lies above the dashed line, so it is not discarded by the rule (even though it is inactive). For the Hessian rule, however, the prediction $\hat{c}_j$ does lie below the dashed line, and so the feature *is* discarded.

6.4 Paper IV

The few optimization methods that we have introduced so far amount to no more than a drop in the ocean that is the vast field of optimization. We have compared a couple of our methods on a few data sets, but are still far from any kind of comprehensive study of their comparative effectiveness. This is not just a problem for our limited overview in this thesis introduction but also research in general in the field of optimization.

The problem is that there are now so many optimization methods to examine and so many different models and data sets on which to compare them on, that it has become difficult to keep track of which methods it is that actually do best on a given problem. You can easily find a paper A that studies optimization methods X and Y on data sets I and II and conclude that X is better than Y but then find another paper, B, which studies methods X, Y, and Z on data sets I and III and conclude that, actually, Y is better than X and by the way, Z happens to be best of them all. Then, later, you find paper C, which actually demonstrates that Z actually is considerably worse than X and in fact always was. In addition, these papers can easily have used different criteria for convergence, programmed the methods using different software, as well as have run their experiments on different hardware.

In short, there is a need for a framework through which this process can be made simple, reproducible, and transparent. This is the motivation behind the `benchopt` package, which we present in the fifth of this thesis' papers (Moreau et al. 2022).

The goal of `benchopt` is to make life easier for both researchers in optimization and users of optimization software (Figure 24). For a researcher who has developed new optimization method for SLOPE, for instance, all you need to do is to write the optimization method for your algorithm and plug it into the existing `benchopt` benchmark for SLOPE and run it. The package will then automatically compare your method to all the other methods in the benchmark and output table and plots of the results. And if you instead are a user who is interested in using SLOPE for your applied work and want to know which algorithm to use, you can either browse the extensive database of results that other users have already uploaded or just download the benchmark and run it yourself on the data that you are interested in using it for.

6.5 Paper V

As we saw in Figure 18, proximal coordinate descent is an efficient optimization algorithm for fitting the lasso. But as we also noted, however, it cannot handle the case when the penalty term h is non-separable, which is the case in SLOPE. In practice, this has reduced the applicability of SLOPE to large data, which is unfortunate given the many appealing properties of the model.

In paper V (Larsson, Klopfenstein, et al. 2023), however, we present a way to cir-

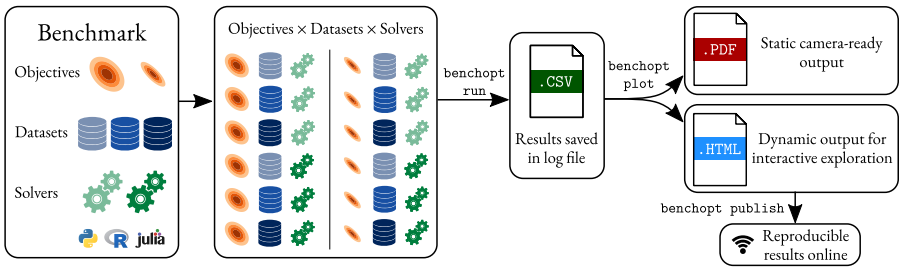


Figure 24: A schematic over how a benchmark is set up and run using `benchopt`. The benchmark consists of a set of files that define an objective, datasets, and solvers. When the user runs `benchopt run`, the package combines all of the possible combinations of objectives, datasets, and solvers and outputs a neatly formatted database of the results. Using `benchopt plot`, the user can then easily compare the different methods through interative visualizations or produce publication-ready plots to insert directly into a paper. Finally, to make the results available to the `benchopt` community, the user can run `benchopt publish`, which opens up a pull-request against the public repository of benchmark results.

cumvent this issue by using a hybrid of proximal coordinate and proximal gradient descent (Figure 25). Our main discovery is that if we fix the clusters and optimize over each cluster in turn, rather than each feature, the problem becomes separable, which means that coordinate descent can be used. And if we combine this with proximal gradient descent steps, which allow us to discover the clusters, then we can guarantee convergence and at the same time benefit from the quickly converging coordinate descent steps

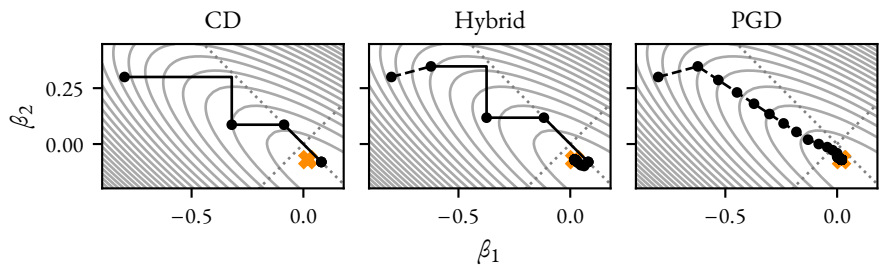


Figure 25: A illustration of the hybrid coordinate descent solver we developed for SLOPE.

6.6 Paper VI

In the final paper of this thesis, we tackle the issue of normalization of binary features, which we touched upon in Section 5. As we saw in that section, normalization is necessary in order to put the features on the "same scale". What this means, however, is less clear, yet has still been largely ignored in the literature, in spite of the fact that normalization is widely used in both statistics and machine learning and, as we saw in Figure 21, has large effects on the solution path.

In our paper, we begin to bridge this knowledge gap by studying normalization for the lasso and ridge regression when they are used on binary features (features that only contain values 0 or 1). What we find is that there is a large effect of normalization with respect to the class balance of the features: the proportion of ones to zeros (or vice versa). Both the lasso and the ridge estimators turn out to be sensitive to this class balance and, depending on the type of normalization used, have trouble recovering effects that are associated with binary features provided that their class balance is severe enough (Figure 26).

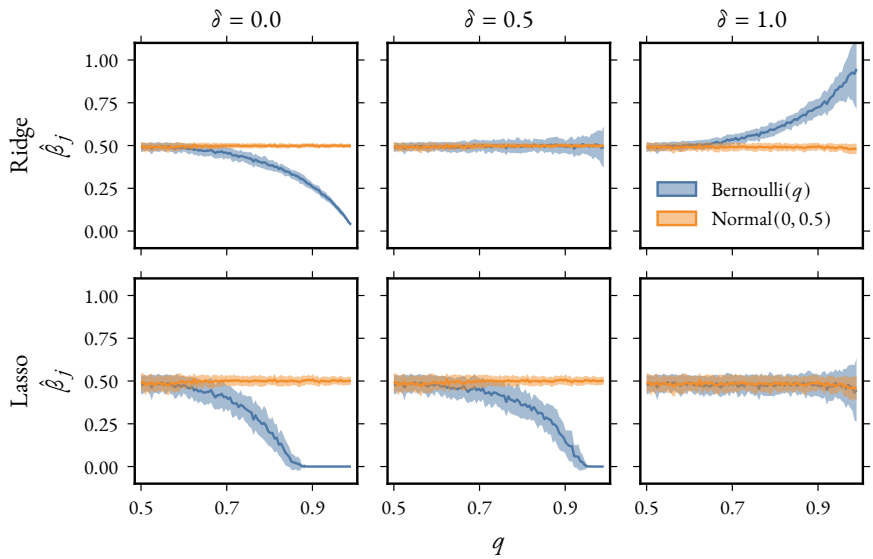


Figure 26: Estimated coefficients from the lasso and ridge for a two-feature problem where one of the features has a normal-like distribution, with standard deviation 1/2, and the other is a binary (pseduo-Bernoulli) feature with class-balance q. The normal feature is standardized in every case, whereas the binary feature is scaled with $(q - q^2)^\delta$—its variance to the power of δ. In other words, we have no scaling for $\delta = 0$, standard deviation scaling when $\delta = 1/2$, and variance-scaling when $\delta = 1$.

Bibliography

Beck, A. and M. Teboulle (Jan. 1, 2009). "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems". In: *SIAM Journal on Imaging Sciences* 2.1, pp. 183–202. DOI: 10.1137/080716542.

Belloni, Alexandre, Victor Chernozhukov, and Lie Wang (Dec. 2011). "Square-Root Lasso: Pivotal Recovery of Sparse Signals via Conic Programming". In: *Biometrika* 98.4, pp. 791–806. ISSN: 0006-3444. DOI: 10.1093/biomet/asr043.

Bertsimas, Dimitris, Angela King, and Rahul Mazumder (Apr. 1, 2016). "Best Subset Selection via a Modern Optimization Lens". In: *The Annals of Statistics* 44.2, pp. 813–852. ISSN: 0090-5364. DOI: 10.1214/15-AOS1388.

Bogdan, Małgorzata, Ewout van den Berg, Chiara Sabatti, et al. (Sept. 2015). "SLOPE – Adaptive Variable Selection via Convex Optimization". In: *The annals of applied statistics* 9.3, pp. 1103–1140. ISSN: 1932-6157. DOI: 10.1214/15-AOAS842. pmid: 26709357.

Bogdan, Małgorzata, Ewout van den Berg, Weijie Su, et al. (Oct. 29, 2013). *Statistical Estimation and Testing via the Sorted L1 Norm*. DOI: 10.48550/arXiv.1310.1969. arXiv: 1310.1969 [math, stat]. URL: http://arxiv.org/abs/1310.1969 (visited on 04/16/2020). preprint.

Bogdan, Małgorzata, Xavier Dupuis, et al. (Mar. 22, 2022). *Pattern Recovery by SLOPE*. DOI: 10.48550/arXiv.2203.12086. arXiv: 2203.12086 [math, stat]. URL: http://arxiv.org/abs/2203.12086 (visited on 06/03/2022). preprint.

Bondell, Howard D. and Brian J. Reich (Mar. 2008). "Simultaneous Regression Shrinkage, Variable Selection, and Supervised Clustering of Predictors with OSCAR". In: *Biometrics* 64.1, pp. 115–123. ISSN: 0006-341X. DOI: 10.1111/j.1541-0420.2007.00843.x. JSTOR: 25502027.

Bottou, Léon (Sept. 30, 2010). "Large-Scale Machine Learning with Stochastic Gradient Descent". In: *Proceedings of COMPSTAT'2010*. 19th International Conference on Computational Statistics. Ed. by Yves Lechevallier and Gilbert Saporta. Berlin,

Germany: Physica-Verlag, pp. 177–186. ISBN: 978-3-7908-2604-3. DOI: 10.1007/978-3-7908-2604-3_16.

Boyd, Stephen et al. (2010). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Foundations and Trends® in Machine Learning* 3.1, pp. 1–122. ISSN: 1935-8237. DOI: 10.1561/2200000016.

Daubechies, I., M. Defrise, and C. De Mol (Aug. 26, 2004). "An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint". In: *Communications on Pure and Applied Mathematics* 57.11, pp. 1413–1457. ISSN: 1097-0312. DOI: 10.1002/cpa.20042.

Davis, Damek (June 26, 2015). *An $O(N\log(n))$ Algorithm for Projecting onto the Ordered Weighted ℓ_1 Norm Ball*. DOI: 10.48550/arXiv.1505.00870. arXiv: 1505.00870 [cs, math]. URL: http://arxiv.org/abs/1505.00870 (visited on 04/23/2024). preprint.

Donoho, David L. and Iain M. Johnstone (Aug. 1994). "Ideal Spatial Adaptation by Wavelet Shrinkage". In: *Biometrika* 81.3, pp. 425–455. ISSN: 0006-3444. DOI: 10.2307/2337118. JSTOR: 2337118.

— (1995). "Adapting to Unknown Smoothness via Wavelet Shrinkage". In: *Journal of the American Statistical Association* 90.432, pp. 1200–1224. ISSN: 0162-1459. DOI: 10.2307/2291512. JSTOR: 2291512.

Duchi, John et al. (July 5–9, 2008). "Efficient Projections onto the L1-Ball for Learning in High Dimensions". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML 2008. Ed. by Andrew McCallum and Sam Roweis. Helsinki, Finland: Association for Computing Machinery, pp. 272–279. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390191.

Dupuis, Xavier and Patrick J C Tardivel (Oct. 28, 2023). *The Solution Path of SLOPE*. HAL: hal-04100441v2. URL: https://hal.science/hal-04100441v2 (visited on 01/17/2024). preprint.

Efron, Bradley et al. (Apr. 2004). "Least Angle Regression". In: *Annals of Statistics* 32.2, pp. 407–499. ISSN: 0090-5364. DOI: 10.1214/009053604000000067.

El Ghaoui, Laurent, Vivian Viallon, and Tarek Rabbani (Sept. 21, 2010). *Safe Feature Elimination in Sparse Supervised Learning*. Technical report UCB/EECS-2010-126. Berkeley: EECS Department, University of California.

Fan, Jianqing and Runze Li (Dec. 1, 2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties". In: *Journal of the American Statistical Association* 96.456, pp. 1348–1360. ISSN: 0162-1459. DOI: 10/fd7bfs.

Figueiredo, Mário A. T. and Robert D. Nowak (Sept. 13, 2014). *Sparse Estimation with Strongly Correlated Variables Using Ordered Weighted L1 Regularization*. DOI: 10.48550/arXiv.1409.4005. arXiv: 1409.4005. URL: http://arxiv.org/abs/1409.4005 (visited on 06/03/2022). preprint.

Friedman, Jerome, Trevor Hastie, Holger Höfling, et al. (Dec. 2007). "Pathwise Coordinate Optimization". In: *The Annals of Applied Statistics* 1.2, pp. 302–332. ISSN: 1932-6157. DOI: 10/d88g8c.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (July 2008). "Sparse Inverse Covariance Estimation with the Graphical Lasso". In: *Biostatistics* 9.3, pp. 432–441. ISSN: 1465-4644. DOI: 10.1093/biostatistics/kxm045.

— (Jan. 2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1, pp. 1–22. DOI: 10.18637/jss.v033.i01.

Fu, Wenjiang J. (1998). "Penalized Regressions: The Bridge versus the Lasso". In: *Journal of Computational and Graphical Statistics* 7.3, pp. 397–416. ISSN: 1061-8600. DOI: 10.2307/1390712. JSTOR: 1390712.

Golub, T. R. et al. (Oct. 15, 1999). "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring". In: *Science* 286.5439, pp. 531–537. ISSN: 0036-8075. DOI: 10.1126/science.286.5439.531. pmid: 10521349.

Guyon, Isabelle et al. (Dec. 13–18, 2004). "Result Analysis of the NIPS 2003 Feature Selection Challenge". In: *Advances in Neural Information Processing Systems 17*. Neural Information Processing Systems 2004. Ed. by Lawrence K. Saul, Yair Weiss, and Léon Bottou. Vancouver, BC, Canada: MIT Press, pp. 545–552. ISBN: 978-0-262-19534-8.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. 2nd ed. Springer Series in Statistics. New York: Springer-Verlag. ISBN: 978-0-387-84857-0.

Hastie, Trevor, Robert Tibshirani, and Ryan Tibshirani (Nov. 2020). "Best Subset, Forward Stepwise or Lasso? Analysis and Recommendations Based on Extensive Comparisons". In: *Statistical Science* 35.4, pp. 579–592. ISSN: 0883-4237. DOI: 10.1214/19-STS733.

Keerthi, S. Sathiya and Dennis DeCoste (Dec. 1, 2005). "A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs". In: *The Journal of Machine Learning Research* 6.12, pp. 341–361. ISSN: 1532-4435.

Kim, Seung-Jean et al. (Dec. 2007). "An Interior-Point Method for Large-Scale -Regularized Least Squares". In: *IEEE Journal of Selected Topics in Signal Processing* 1.4, pp. 606–617. ISSN: 1932-4553, 1941-0484. DOI: 10.1109/JSTSP.2007.910971.

King, Ross (2024). *Qualitative Structure Activity Relationships*. UCI Machine Learning Repository. DOI: 10.24432/C5TP54.

Larsson, Johan (Sept. 6, 2021). "Look-Ahead Screening Rules for the Lasso". In: *22nd European Young Statisticians Meeting - Proceedings*. 22nd European Young Statisticians Meeting. Ed. by Andreas Makridis et al. Athens, Greece: Panteion university of social and political sciences, pp. 61–65. ISBN: 978-960-7943-23-1.

Larsson, Johan, Małgorzata Bogdan, and Jonas Wallin (Dec. 6–12, 2020). "The Strong Screening Rule for SLOPE". In: *Advances in Neural Information Processing Systems 33*. 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Ed. by Hugo Larochelle et al. Vol. 33. Virtual: Curran Associates, Inc., pp. 14592–14603. ISBN: 978-1-71382-954-6.

Larsson, Johan, Quentin Klopfenstein, et al. (Apr. 25–27, 2023). "Coordinate Descent for SLOPE". In: *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*. AISTATS 2023. Ed. by Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent. Vol. 206. Proceedings of Machine Learning Research. Valencia, Spain: PMLR, pp. 4802–4821.

Larsson, Johan and Jonas Wallin (Nov. 28–Dec. 9, 2022). "The Hessian Screening Rule". In: *Advances in Neural Information Processing Systems 35*. 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Ed. by Sanmi Koyejo et al. Vol. 35. New Orleans, USA: Curran Associates, Inc., pp. 15823–15835. ISBN: 978-1-71387-108-8.

Lawson, Charles L. and Richard J. Hanson (1995). *Solving Least Squares Problems*. 2nd ed. Classics in Applied Mathematics. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. 351 pp. ISBN: 978-0-89871-356-5. DOI: 10.1137/1.9781611971217.

Lee, Jason D., Yuekai Sun, and Michael A. Saunders (Jan. 1, 2014). "Proximal Newton-type Methods for Minimizing Composite Functions". In: *SIAM Journal on Optimization* 24.3, pp. 1420–1443. ISSN: 1052-6234. DOI: 10.1137/130921428.

Lewis, David D. et al. (Dec. 1, 2004). "RCV1: A New Benchmark Collection for Text Categorization Research". In: *The Journal of Machine Learning Research* 5, pp. 361–397. ISSN: 1532-4435.

Li, Qinzhen and Xudong Li (July 29, 2021). "Fast Projection onto the Ordered Weighted L1 Norm Ball". In: *Science China Mathematics* 65.4, pp. 869–886. ISSN: 1869-1862. DOI: 10.1007/s11425-020-1743-9.

Mairal, Julien and Bin Yu (June 2012). "Complexity Analysis of the Lasso Regularization Path". In: *Proceedings of the 29th International Conference on Machine Learning*. International Conference on Machine Learning 2012. Edinburgh, United Kingdom, pp. 1835–1842.

Moreau, Thomas et al. (Nov. 28–Dec. 9, 2022). "Benchopt: Reproducible, Efficient and Collaborative Optimization Benchmarks". In: *Advances in Neural Information Processing Systems 35*. 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Ed. by S. Koyejo et al. Vol. 35. New Orleans, USA: Curran Associates, Inc., pp. 25404–25421. ISBN: 978-1-71387-108-8.

Nesterov, Yuri (1983). "A Method of Solving a Convex Programming Problem with Convergence Rate O(1/K^2)". In: *Doklady Akademii Nauk SSSR* 269.3, pp. 543–547.

Nomura, Shunichi (Oct. 29, 2020). *An Exact Solution Path Algorithm for SLOPE and Quasi-Spherical OSCAR*. DOI: 10.48550/arXiv.2010.15511. arXiv: 2010.15511. URL: http://arxiv.org/abs/2010.15511 (visited on 05/27/2021). preprint.

Osborne, Michael R., Brett Presnell, and Berwin A. Turlach (July 1, 2000a). "A New Approach to Variable Selection in Least Squares Problems". In: *IMA Journal of Numerical Analysis* 20.3, pp. 389–403. ISSN: 1464-3642. DOI: 10.1093/imanum/20.3.389.

— (2000b). "On the LASSO and Its Dual". In: *Journal of Computational and Graphical Statistics* 9.2, pp. 319–337. ISSN: 1061-8600. DOI: 10.2307/1390657. JSTOR: 1390657.

Perez, Guillaume et al. (May 2022). "Efficient Projection Algorithms onto the Weighted L1 Ball". In: *Artificial Intelligence* 306, pp. 1–14. ISSN: 00043702. DOI: 10.1016/j.artint.2022.103683.

Platt, John C. (Jan. 1998). "Fast Training of Support Vector Machines Using Sequential Minimal Optimization". In: *Advances in Kernel Methods: Support Vector Learning*. Ed. by Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola. 1st ed. Boston, MA, USA: MIT Press, pp. 185–208. ISBN: 978-0-262-28319-9. DOI: 10.7551/mitpress/1130.003.0016.

Robbins, Herbert and Sutton Monro (Sept. 1951). "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407. ISSN: 0003-4851. DOI: 10.1214/aoms/1177729586.

Rockafellar, R. Tyrrell (1970). *Convex Analysis*. Princeton Landmarks in Mathematics and Physics. Princeton University Press. 472 pp. ISBN: 978-0-691-01586-6. JSTOR: j.ctt14bs1ff.

Santosa, Fadil and William W. Symes (Oct. 1986). "Linear Inversion of Band-Limited Reflection Seismograms". In: *SIAM Journal on Scientific and Statistical Computing* 7.4, pp. 1307–1330. ISSN: 0196-5204. DOI: 10.1137/0907087.

Schneider, Ulrike and Patrick Tardivel (Oct. 1, 2022). "The Geometry of Uniqueness, Sparsity and Clustering in Penalized Estimation". In: *The Journal of Machine Learning Research* 23.331, pp. 1–36. ISSN: 1532-4435.

Shevade, S. K. and S. S. Keerthi (Nov. 22, 2003). "A Simple and Efficient Algorithm for Gene Selection Using Sparse Logistic Regression". In: *Bioinformatics* 19.17, pp. 2246–2253. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btg308. pmid: 14630653.

Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society: Series B* 58.1, pp. 267–288. ISSN: 0035-9246. DOI: 10.1111/j.2517-6161.1996.tb02080.x. JSTOR: 2346178.

Tibshirani, Robert, Jacob Bien, et al. (Mar. 2012). "Strong Rules for Discarding Predictors in Lasso-Type Problems". In: *Journal of the Royal Statistical Society. Series B: Statistical Methodology* 74.2, pp. 245–266. ISSN: 1369-7412. DOI: 10/c4bb85.

Tibshirani, Robert, Michael Saunders, et al. (Feb. 2005). "Sparsity and Smoothness via the Fused Lasso". In: *Journal of the Royal Statistical Society: Series B* 67.1, pp. 91–108. ISSN: 1467-9868. DOI: 10.1111/j.1467-9868.2005.00490.x.

Wright, Stephen (Mar. 25, 2015). "Coordinate Descent Algorithms". In: *Mathematical Programming: Series B* 151.1, pp. 3–34. ISSN: 00255610. DOI: 10.1007/s10107-015-0892-3.

Yuan, Ming and Yi Lin (Dec. 21, 2005). "Model Selection and Estimation in Regression with Grouped Variables". In: *Journal of the Royal Statistical Society: Series B* 68.1, pp. 49–67. ISSN: 1467-9868. DOI: 10.1111/j.1467-9868.2005.00532.x.

Zeng, Xiangrong and Mário A. T. Figueiredo (Oct. 2014). "Decreasing Weighted Sorted L1 Regularization". In: *IEEE Signal Processing Letters* 21.10, pp. 1240–1244. ISSN: 1070-9908, 1558-2361. DOI: 10.1109/LSP.2014.2331977.

— (Apr. 10, 2015). *The Ordered Weighted L1 Norm: Atomic Formulation, Projections, and Algorithms*. DOI: 10.48550/arXiv.1409.4271. arXiv: 1409.4271. URL: http://arxiv.org/abs/1409.4271 (visited on 11/22/2019). preprint.

Zhang, Cun-Hui (Apr. 2010). "Nearly Unbiased Variable Selection under Minimax Concave Penalty". In: *The Annals of Statistics* 38.2, pp. 894–942. ISSN: 0090-5364. DOI: 10/bp22zz.

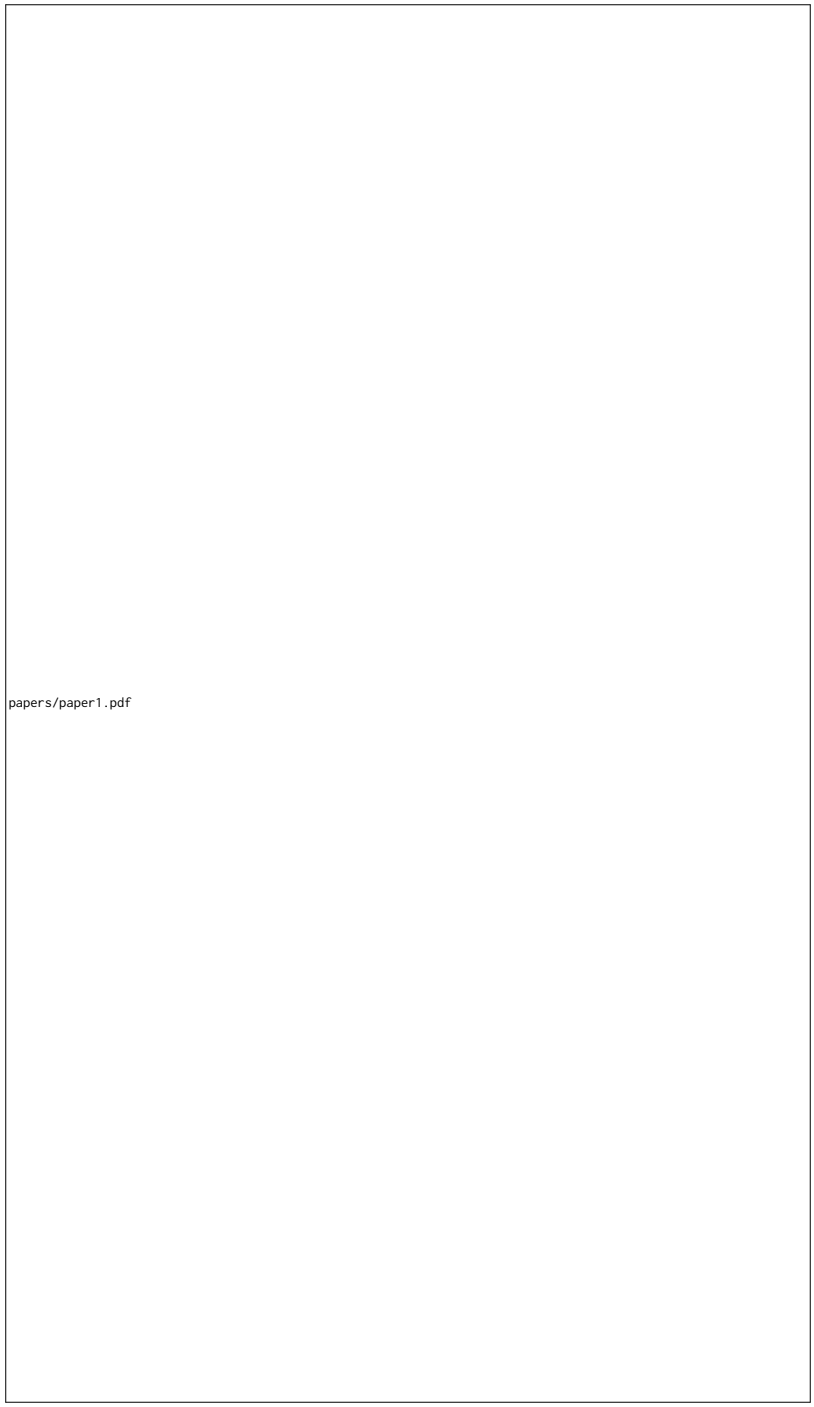
Zou, Hui (Dec. 1, 2006). "The Adaptive Lasso and Its Oracle Properties". In: *Journal of the American Statistical Association* 101.476, pp. 1418–1429. ISSN: 0162-1459. DOI: 10.1198/016214506000000735.

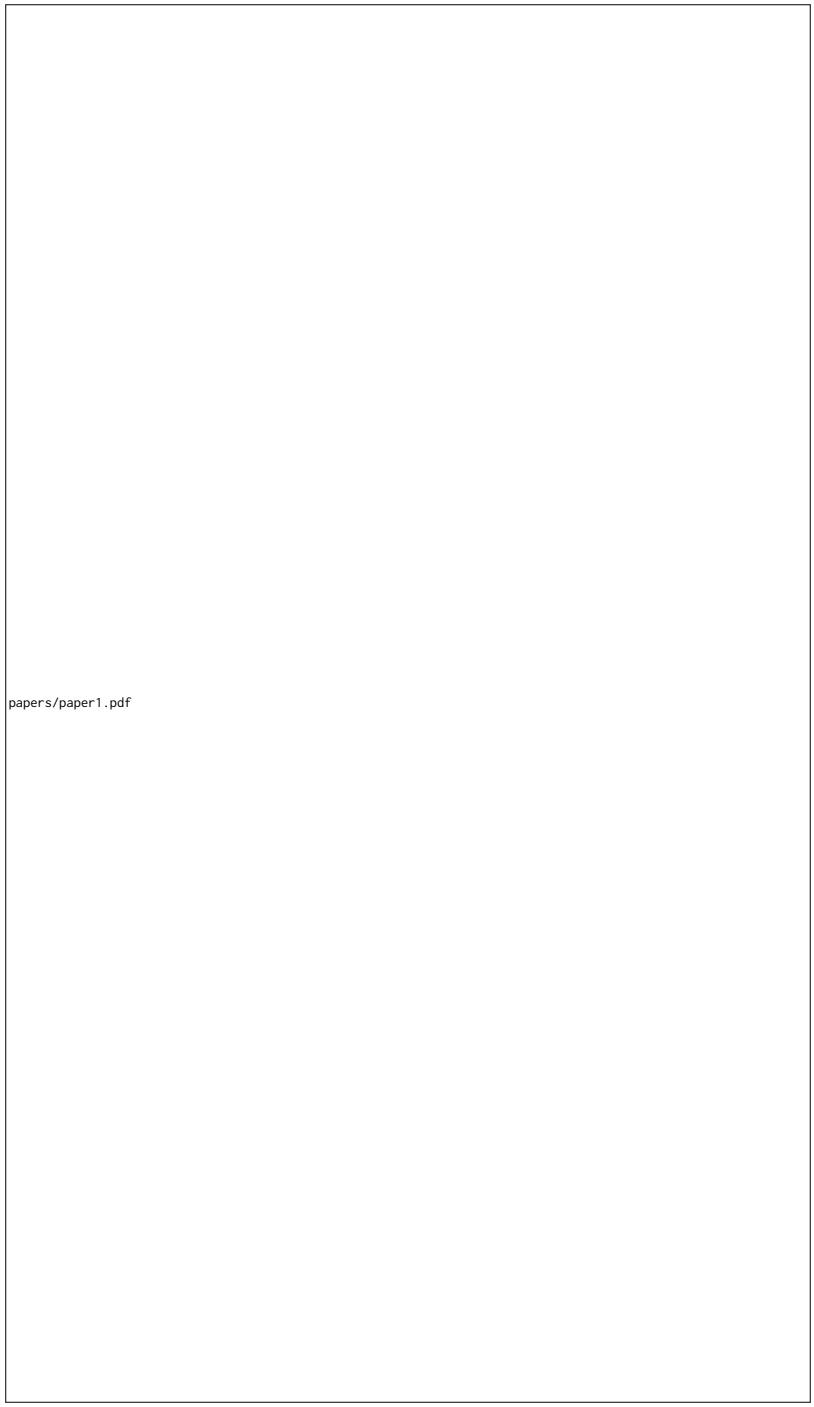
Zou, Hui and Trevor Hastie (2005). "Regularization and Variable Selection via the Elastic Net". In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 67.2, pp. 301–320. ISSN: 1369-7412.

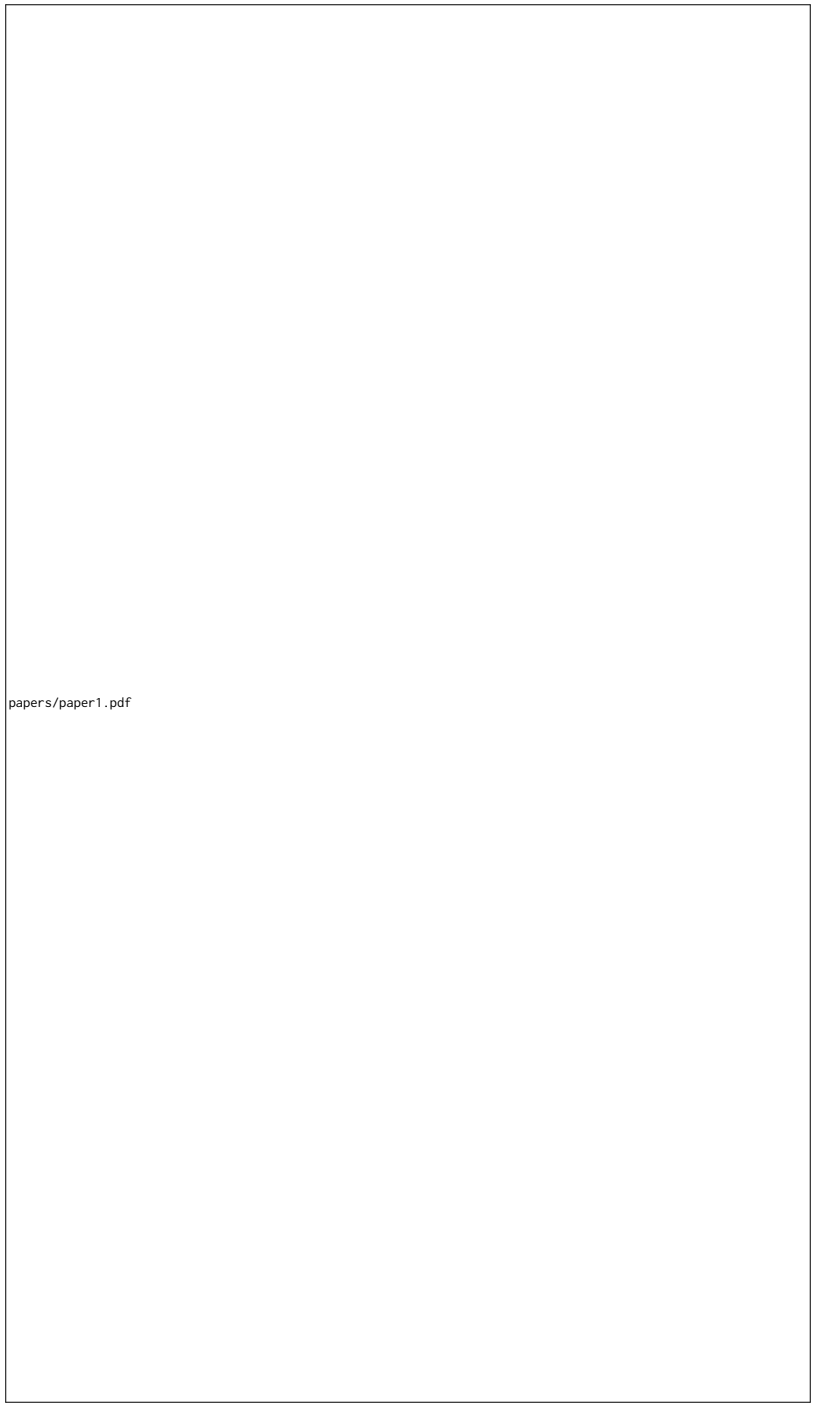
Papers

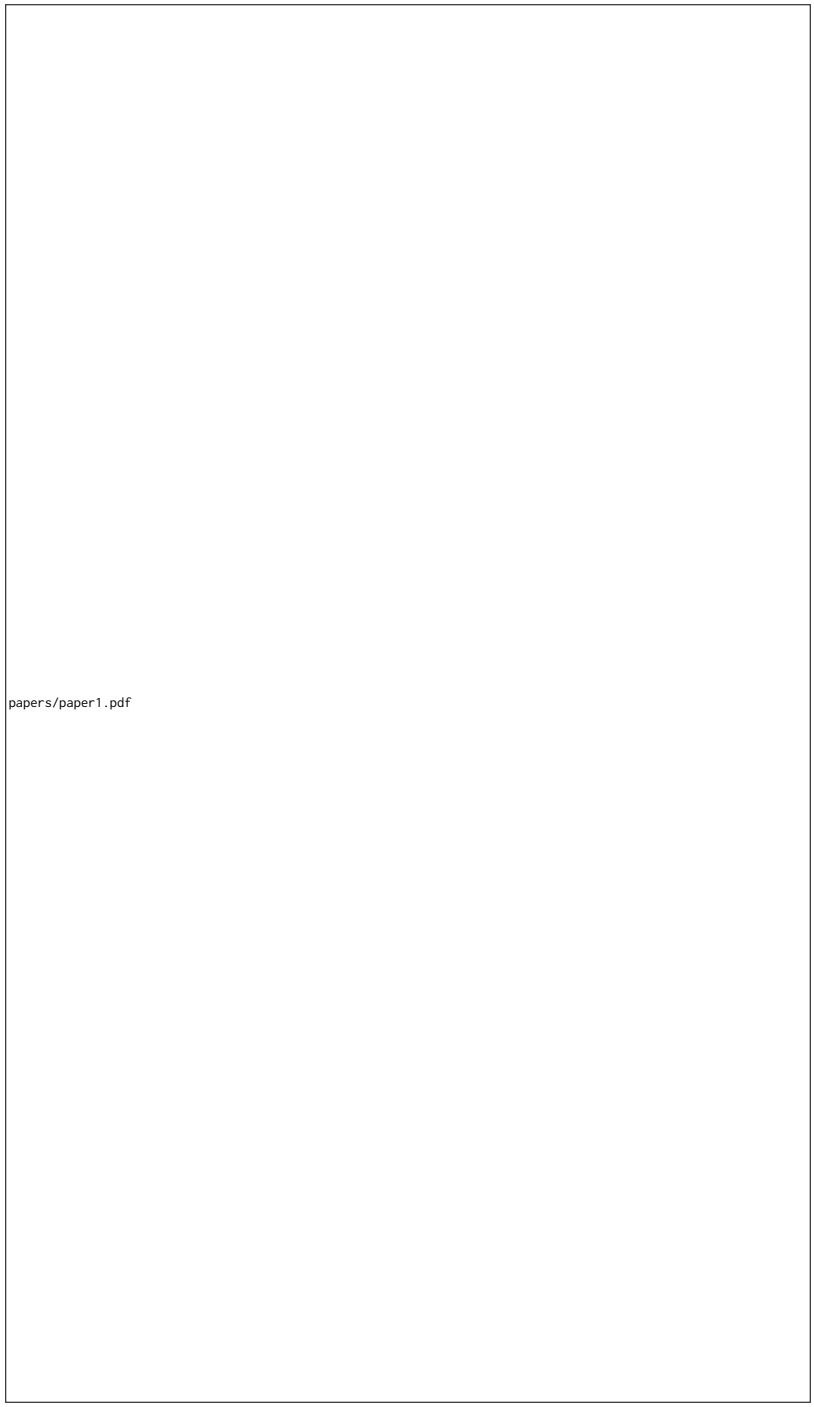
I

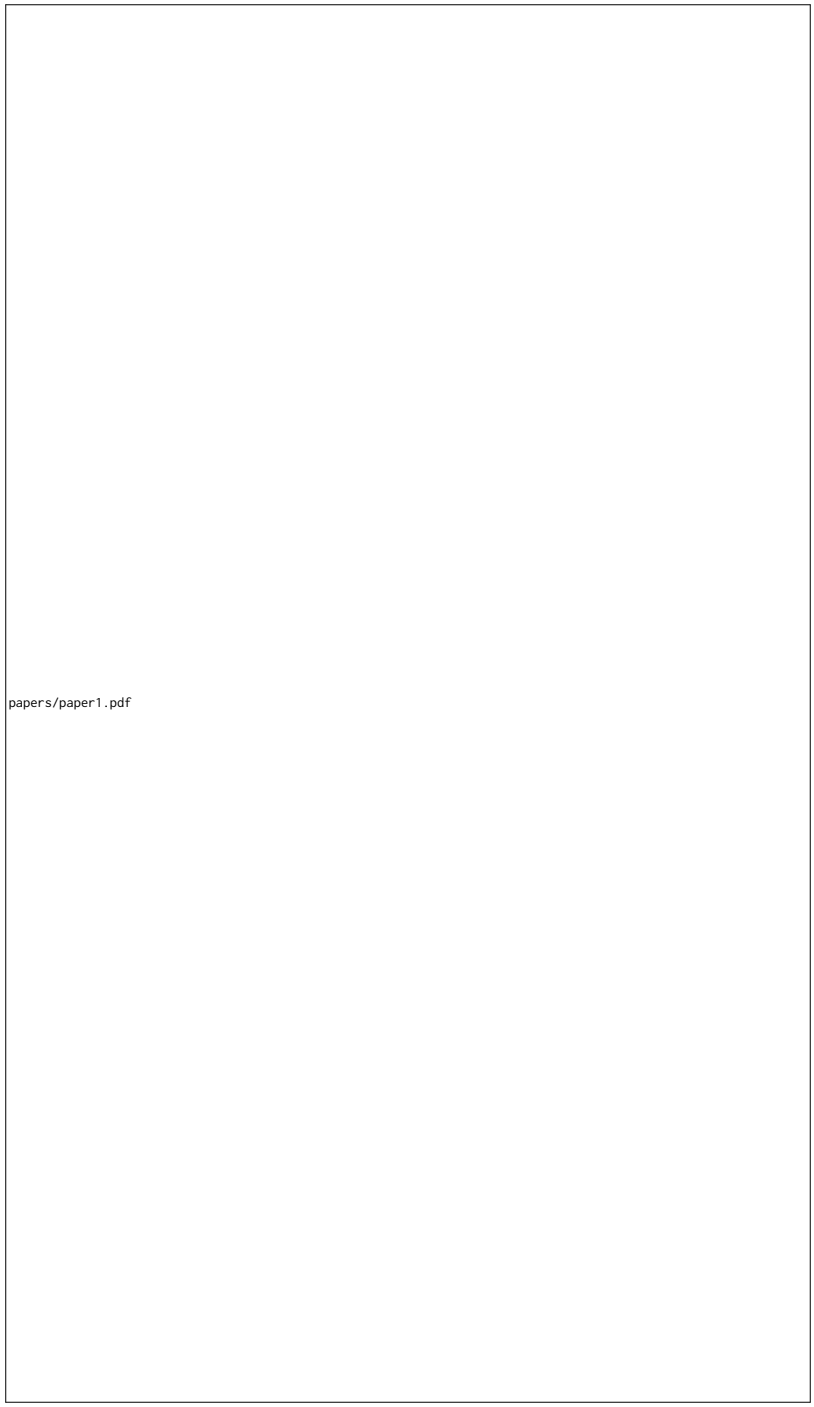
papers/paper1.pdf

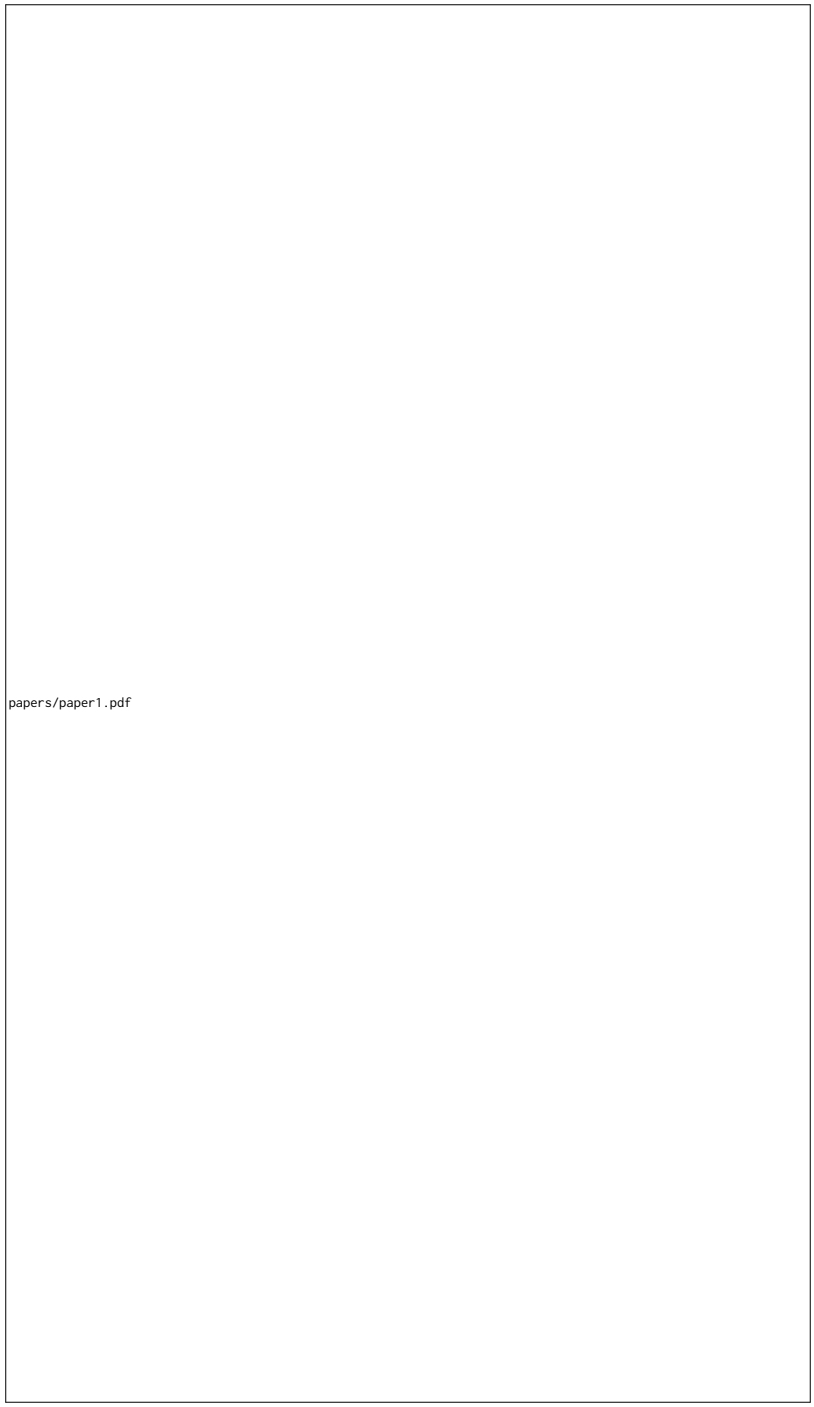

papers/paper1.pdf

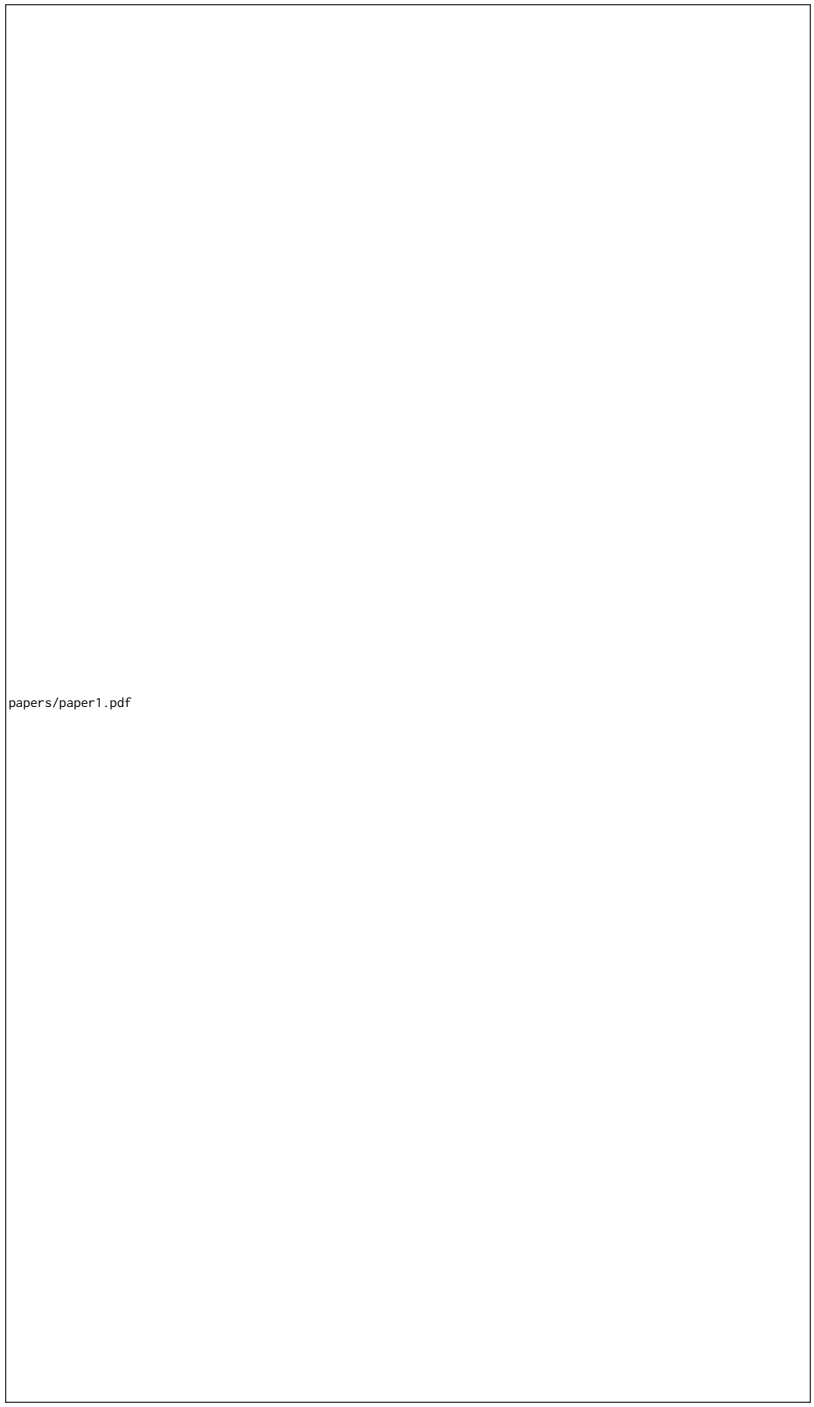

papers/paper1.pdf

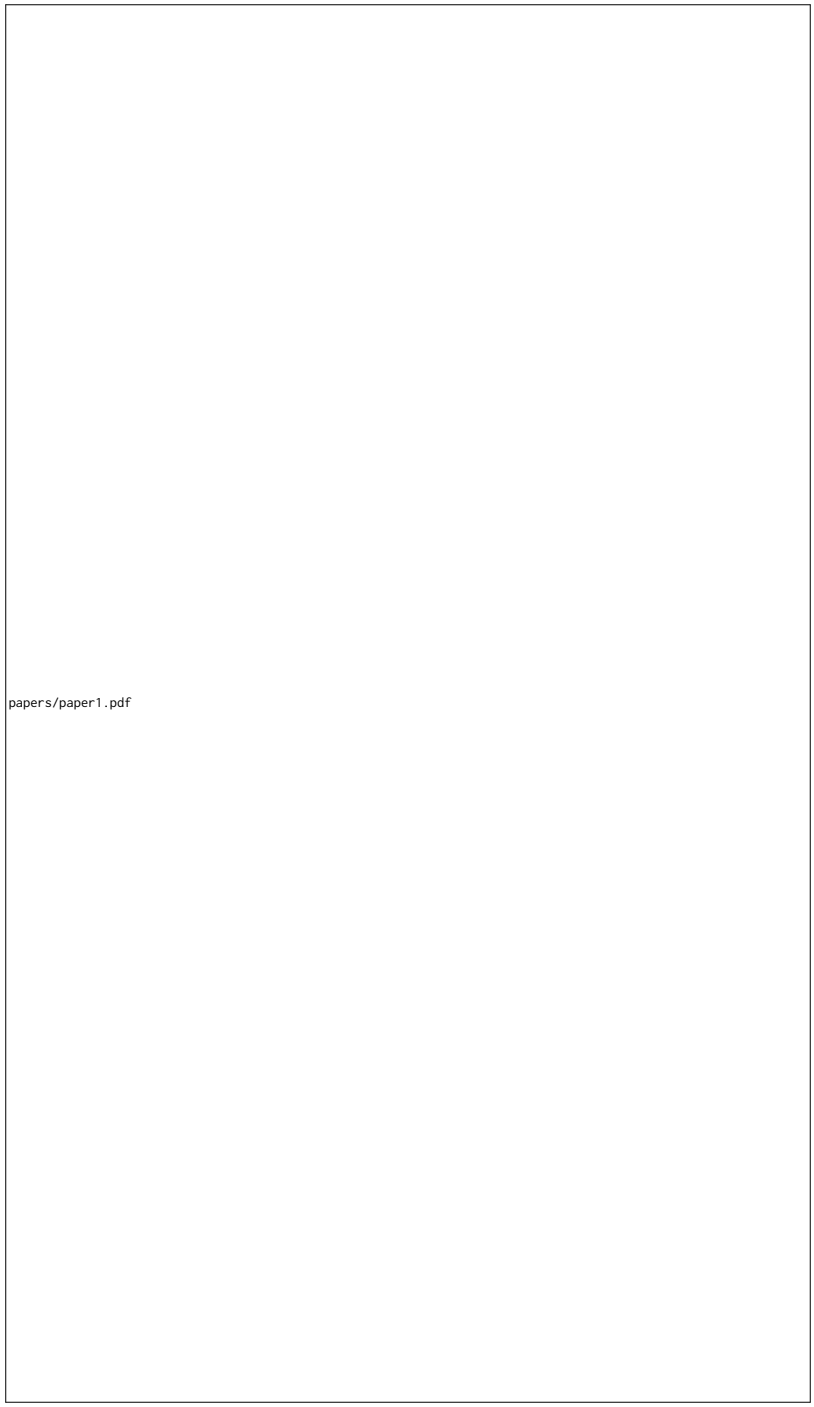

papers/paper1.pdf

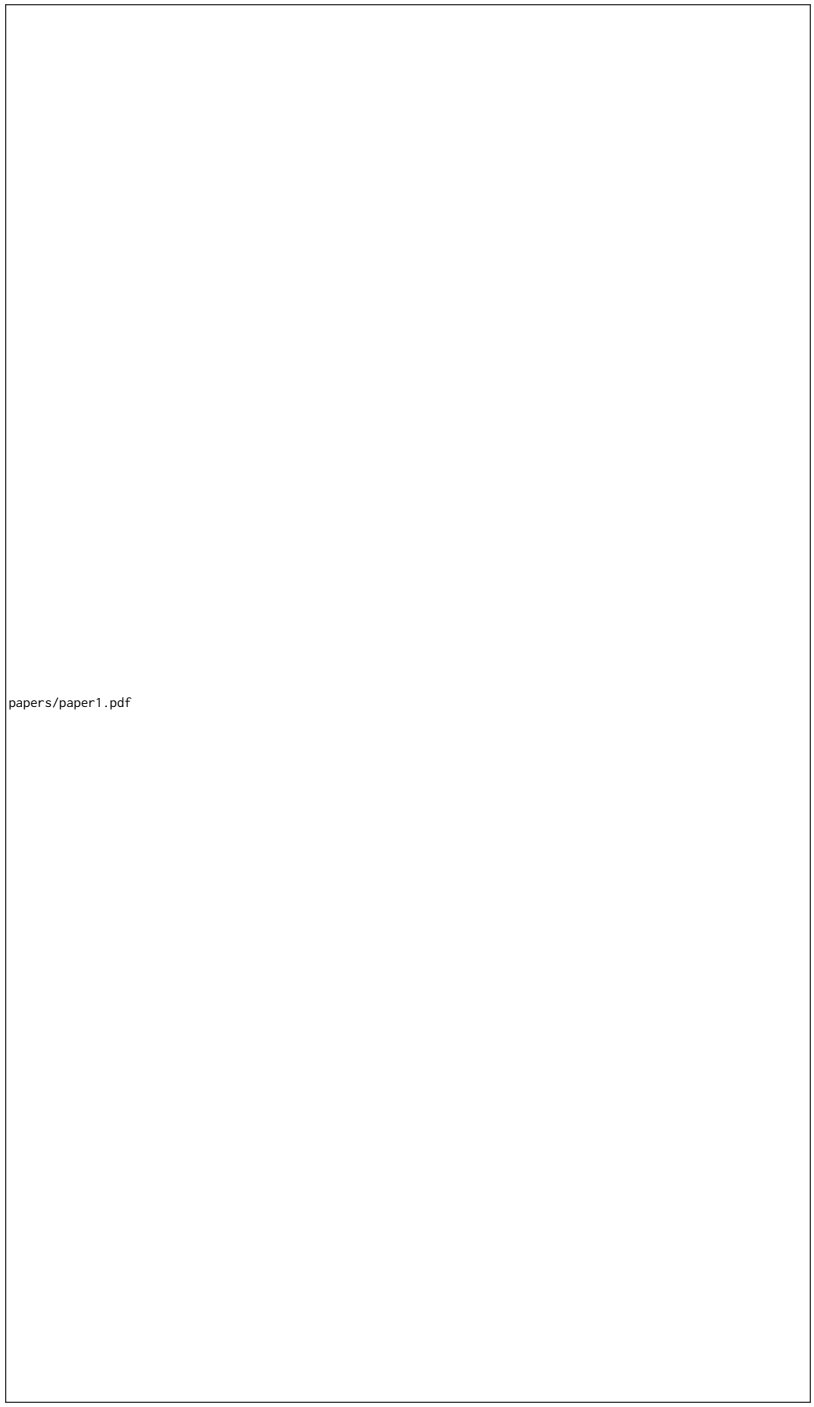

papers/paper1.pdf

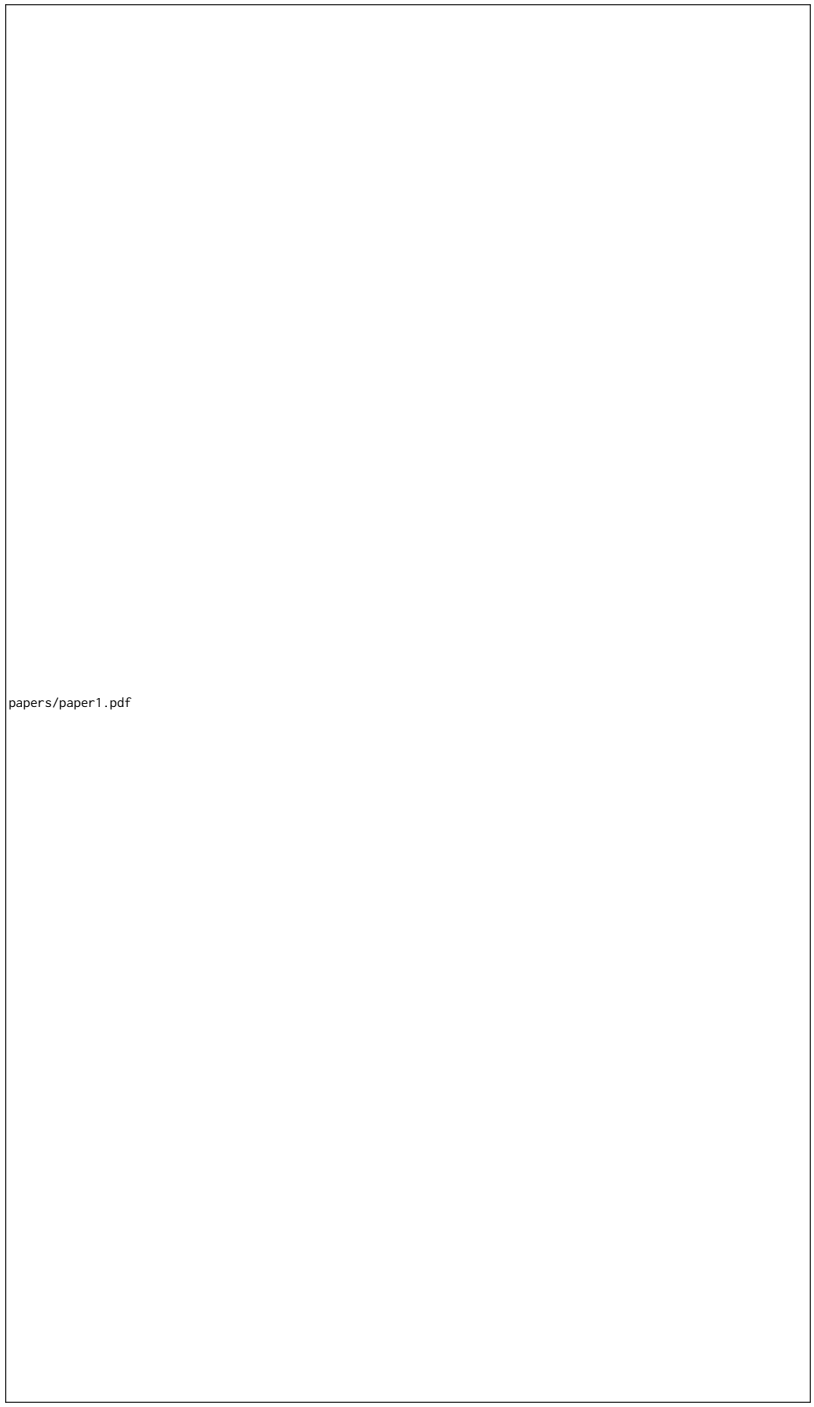

papers/paper1.pdf

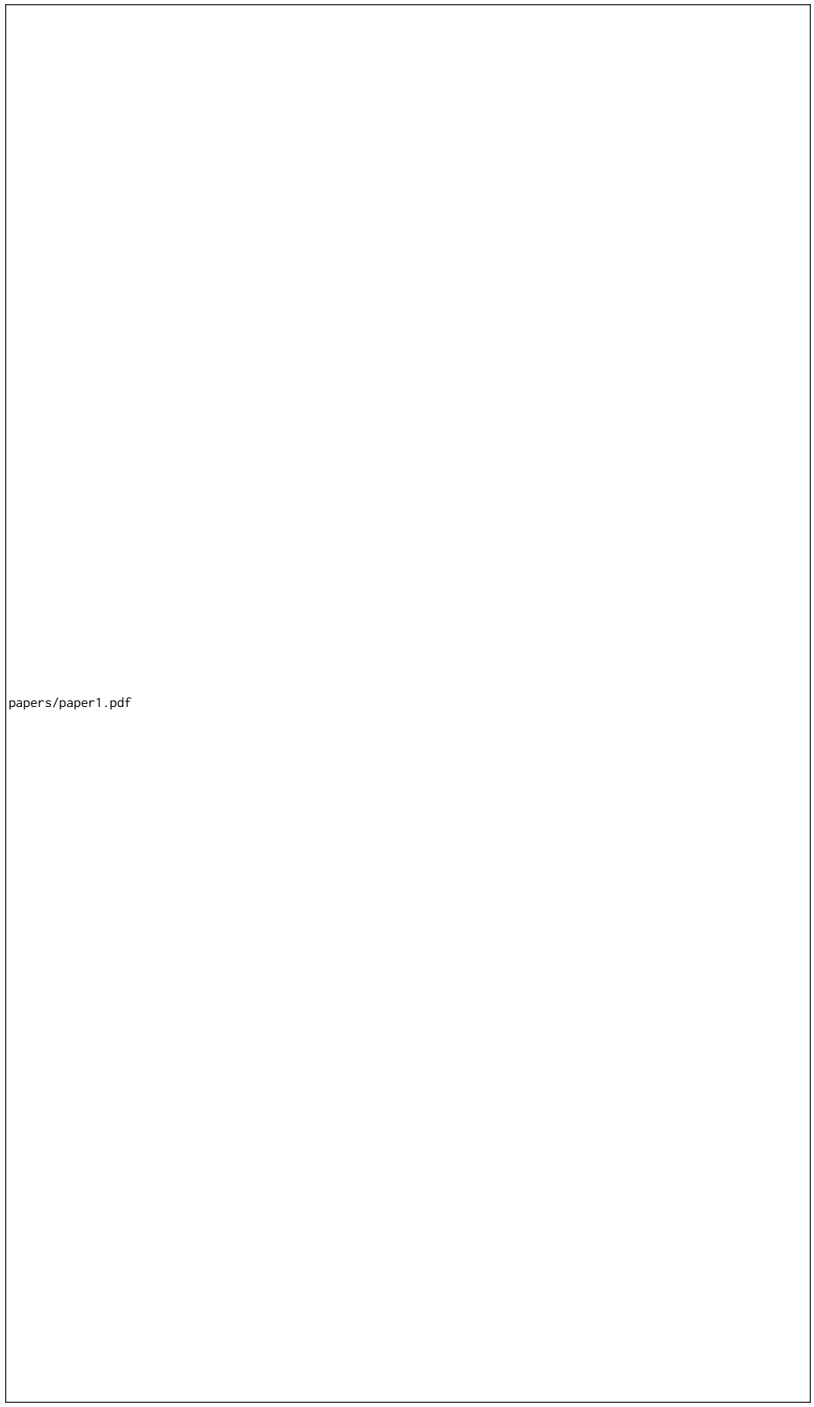

papers/paper1.pdf


papers/paper1.pdf


papers/paper1.pdf


papers/paper1.pdf


papers/paper1.pdf


papers/paper1.pdf

II

III

IV

V

VI